Tree Based Data Gathering from Sensors: Topology Management Sustaining QoS

Suchetana Chakraborty

TH-1251_11610108

Tree Based Data Gathering from Sensors: Topology Management Sustaining QoS

Thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

by

Suchetana Chakraborty

Under the supervision of

Dr. Sushanta Karmakar

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

Guwahati 781039, India

May 29, 2014

TH-1251_11610108

TH-1251_11610108

To

The Mother

"The powers of the mind are like rays of light dissipated; when they are concentrated, they illumine."

-Swami Vivekananda

TH-1251_11610108

TH-1251_11610108

Declaration

I certify that

a. The work contained in this thesis is original and has been done by myself and the general supervision of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

d. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT Guwahati
Date:

Suchetana Chakraborty
Research Scholar
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati, Assam, INDIA 781039

TH-1251_11610108

TH-1251_11610108

Certificate

This is to certify that the thesis entitled "**Tree Based Data Gathering from Sensors: Topology Management Sustaining QoS**" being submitted by **Suchetana Chakraborty** to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, is a record of bona fide research work under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

Place: IIT Guwahati
Date:

Dr. Sushanta Karmakar
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati, Assam, INDIA 781039

TH-1251_11610108

TH-1251_11610108

Acknowledgements

First and foremost I express my sincere gratitude to my supervisor Dr. Sushanta Karmakar, Assistant Professor, IIT Guwahati, for guiding me and sharing many of his insights. I wholeheartedly honor his profound knowledge, honesty, dedication and perseverance that strongly motivated me to choose a research career. I am highly indebted to him as he guided, advised, corrected and supported me at every step and retained constant trust and confidence on my potential. I feel gratified in extending my intense regards to Prof. Sukumar Nandi, Dept. of CSE, IIT Guwahati without whom it would have not been possible to make this journey. I consider myself as privileged and blessed to receive his valued guidance, sensible advice and endless support towards the realization of my dreams in pursuing my doctoral work. His extraordinary knowledge, positive outlook, unparalleled endurance, exceptional enthusiasm and perspicuous conception influenced me always in contributing with brighter ideas.

It is an extreme pleasure to have Prof. Diganta Goswami, Dr. Partha Sarathi Mandal and Dr. Sanasam Ranbir Singh as the honorable members of my thesis doctoral committee. I would like to express my earnest gratitude to all of them for their invaluable time in evaluating the work progress and fruitful advices towards improving the work quality. I am extremely grateful to my thesis reviewers, Prof. Indranil Sengupta from Indian Institute of Technology Kharagpur and Dr. Salil Kanhere from The University of New South Wales, Australia for expending their valuable time to review this thesis. I am also thankful to Dr. Santosh Biswas and Dr. Gautam K. Das for being the member of my thesis examination and viva-voce committee. I feel privileged and honored for being a student of IIT Guwahati. I take this opportunity to express my heartfelt thanks to all associated persons, whose constant effort made this such a nice place for research and study. I owe my special thanks to Prof. Purandar Bhaduri (former Head, Deptt. of CSE), Prof. Shivashankar B. Nair (present Head, Deptt. of CSE), Prof. Gautam Barua (former Director), Prof. Gautam Biswas (present Director), all the Deans and administrative staffs of the institute. I also express my sincere regards to all respected faculties and staffs of the department of CSE for their extended help and support.

TH-1251_11610108

It is of great honor to be selected for the prestigious TCS research fellowship for pursuing my doctoral work. In this regard, I would like to express my sincere gratitude to Tata Consultancy Services Private Limited for providing me the esteemed platform with all facilities to face the critical research challenges with utmost confidence and dedication. I would like to thank Mr. Rahul Pandey, the former program manager and Mr. Sachin Parkhi, the present program manager for coordinating the whole thing systematically and smoothly. I convey my special thanks to Dr. P. Balamuralidhar, Principal Scientist and Head, TCS Innovations Labs, Bangalore and my mentor in TCS, Dr. Manoj Nambiar, Chief Scientist, TCS Innovations Lab for their thoughtful inputs to my research. I also thank Microsoft Research, India for supporting me financially during my visit to the conference, IEEE/IFIP NOMS 2012, held in Maui, Hawaii, for presenting one of my research papers. Additionally, I extend my gratitude to National Internet Exchange of India (NIXI) for supporting my future research through the NIXI Fellowship Program for the year 2014-2015.

I wish to take this opportunity to express my thanks to all my respected teachers and good friends from my previous school and college life whose silent contribution made this journey possible. My special regards to Rama didibhai, Mathematics teacher in our secondary school who actually motivated my way to higher education. I am thankful to my friends and colleagues from IIT Guwahati- Soumyadip, Rahul, Tanushree, Girish, Ravi, Vishnu, Suddhasil-da, Ferdous-da, Niladri-da, Prithu, Mahasweta and Subhrangshu with whom I interacted closely. I am fortunate enough to have Sandip Chakraborty as my friend as well as colleague in the department of CSE, IIT Guwahati. In this occasion I would like to express my heartfelt gratitude to Sandip for his perpetual guidance, support, encouragement and inspiration. It was a wonderful experience and mind-boggling journey while working with him. I treasure all the moments and thoughts that I shared with him in this course of time.

I express my cordial thanks to my husband, Dip Sankar Banerjee, for his generous cooperation, constant inspiration and valuable guidance for the entire duration of my doctoral works. I am thankful to my parents-in-law for providing me a positive and supportive environment to continue my research work. I feel most blessed to have my lovely parents and brother in my family who have always supported me at every course of my life and offered me

TH-1251_11610108

constant encouragement and inspiration. My special regards to my parents for their intense trust and confidence over me which made my journey immensely happy, secure and comfortable. Lastly, I express my ardent gratitude and profound reverence for my loving Dida and in the memories of my Dadu and Thakuma, whose blessings are always there to protect me, and lead me to the right path.

Place: IIT Guwahati
Date: **Suchetana Chakraborty**

TH-1251_11610108

TH-1251_11610108

Abstract

Designing an efficient scheme for data gathering in wireless sensor network is a challenging problem due to the limitation in sensor resources and the inherent dynamics in the environment. Distributed nature of a sensor network requires localized solutions, in absence of any centralized control and complete knowledge of the communication network. Additionally, the design strategy demands to satisfy certain performance criteria like bounded delay, low cost and high reliability etc. as specified by different applications. Many applications in sensor network exploit the advantages of tree-based data gathering, where sensory data from all sensor nodes are collected at a sink or base station for statistical analysis. The sensors form a data gathering tree rooted at the sink, such that the data packets from every sensor node is forwarded towards the sink via its parent node in the tree. The existing works in the literature, that have studied the properties of tree based data gathering, lack in providing an efficient scheme that deals with the design parameters like connectivity, sensing coverage, fault tolerance, network lifetime and other application specific performance metrics, altogether. Considering all these aspects, the main objective of this thesis is to design and develop novel schemes for efficient data gathering in sensor network under various constraints.

The first contribution of the thesis proposes a distributed BFS tree construction scheme rooted at the sink node for crash-tolerant data gathering in a sensor network. Every node computes an alternate parent during the tree construction phase, such that on sudden failure of the parent node, the affected path can be repaired locally using a low-cost proactive approach. So application messages are delivered to the sink with minimum loss or redundancy even in presence of an arbitrary node crash. Multiple simultaneous node failures have also been handled through a reactive repairing technique. To extend the network lifetime by supporting arbitrary node failure, the primary objective is to

TH-1251_11610108

maintain both the connectivity and the sensing coverage in the network. While the first contribution focuses only on the connectivity aspect, the second contribution of the thesis considers both the connectivity and the coverage maintenance as the design objectives.

Considering irregular terrain property and optimal positioning of the sink, the energy depletion rate gradually decreases from the sink towards the terrain periphery. So, both the connectivity and the sensing coverage are affected as the nodes near the sink die out of energy sooner than the leaves of the tree rooted at the sink, thereby creating network holes. For an improved network lifetime, a gradient based node deployment strategy has been proposed that also satisfies the initial connectivity and the coverage criteria. The density of the deployed nodes follows a gradient, which is estimated as the amount of energy dissipation at any intermediate node to that of the leaf nodes in its rooted subtree. The proposed theory has been justified through the worst case analysis of the sensor network calculus. As unbalanced data gathering tree escalates the problem of uneven energy depletion in the network, a load-balanced distributed BFS tree construction scheme has been proposed. Further, to handle arbitrary node failure a localized and cost-effective tree maintenance scheme has been introduced. Finally, the characteristics and the design objectives of tree based data gathering with failure support have been explored considering two different applications, one for road traffic monitoring, and the other for critical infrastructure monitoring.

The inherent challenges in distribution and management of sensor network along the road require an application-specific protocol support for the network connectivity, sensing coverage, reliable data gathering and the network lifetime improvement. The next contribution of the thesis introduces the concept of *k-strip length coverage* along the road, that ensures a better sensing coverage for the detection of moving vehicles, compared to the conventional *barrier coverage* and *full area coverage*, in terms of the availability of sufficient information for statistical processing as well as the number of sensors required to be active. To extend the network lifetime, every sensor follows a sleep-wakeup schedule maintaining the network connectivity and the *k-strip length coverage*. This scheduling problem is modeled as a graph optimization, the NP-hardness of which motivates to design a centralized heuristic, providing an approximate solution. The properties of the proposed centralized heuristic are then explored to design a per-node solution based on local information.

Sensor network deployed for critical infrastructure monitoring requires high degree of reliability in sensory data gathering, in spite of arbitrary node or sink failures. The last contribution of this thesis proposes *RelBAS*, a robust data gathering scheme, specially

TH-1251_11610108

designed for the sensory applications on critical infrastructure monitoring. Redundancy in sensor network, in terms of both the number of deployed sensors and the sensory data, is explored to design an effective protocol that ensures reliable data delivery. A set of active sensors, that participate in data gathering, are selected from all sensors based on the connected-coverage criteria. The rest of the nodes go to the sleep state, and act as a replacement on failure of an active node. The proposed protocol aims to find out multiple node-disjoint paths to multiple sinks, so that the loss of connectivity in one path due to node failure does not disrupt the application services. The forwarding path selection at every node in *RelBAS* is based on the computation of a metric that is a function of two parameters - the hop-count distance to the sink and the node's residual energy. Moreover, *RelBAS* is capable of detecting an affected zone due to multiple node failures. Finally, the effectiveness of all the proposed schemes has been established through theoretical analysis and simulation results.

TH-1251_11610108

TH-1251_11610108

Contents

TH-1251_11610108

TH-1251_11610108

TH-1251_11610108

TH-1251_11610108

List of Figures

TH-1251_11610108

TH-1251_11610108

TH-1251_11610108

TH-1251_11610108

List of Tables

TH-1251_11610108

TH-1251_11610108

List of Algorithms

TH-1251_11610108

TH-1251_11610108

Nomenclature

ADC	Analog to Digital Converter
AMC	Application Message Controller
BFS	Breadth First Search
CBR	Constant Bit Rate
CI	Critical Infrastructure
CIIS	Critical Infrastructure Information System
CovIS	Coverage Interval Stabbing
CSLC	Connected Strip Length Coverage
CSMA	Carrier-Sense Multiple Access
DFS	Depth First Search
DMAC	Dynamic Media Access Control
EDF	Energy Dissipation Factor
GPS	Geo Positioning System
ITS	Intelligent Transport System
MAC	Media Access Control
PS	Periodic Sensing
QoS	Quality of Service
REDF	Rate dependent EDF
RSN	Road Sensor Network
RSSI	Received Signal Strength Index
SMAC	Static Media Access Control
SS	Steady Sensing
TMM	Tree Management Module
UDP	User Datagram protocol
WSN	Wireless Sensor Network

TH-1251_11610108

TH-1251_11610108

List of Symbols

$\mathbb{G}(V, E)$	Communication graph with the set of vertices and edges V and E respectively
$\mathbb{T}_u(V_{\mathbb{T}}, E_{\mathbb{T}})$	The tree rooted at node u with the vertex and edge set $V_{\mathbb{T}}$, $E_{\mathbb{T}}$, respectively
$\mathcal{N}$	The total number of deployed sensors in a target area
R_c	Communication radius of a sensor
R_s	Sensing radius of a sensor
$\mathbb{P}$	Path in a data forwarding tree
δ_u	The degree of node u
$\mathscr{P}\{.\}$	Probability of an event
$\langle \mathcal{C}_X(u), \mathcal{C}_Y(u) \rangle$	X and Y coordinate position of node u in Euclidean plane
$d_{u,v}$	Euclidean distance between two nodes u and v
d^{min}	The minimum Euclidean distance between two nodes
d^{MAX}	The maximum Euclidean distance between two nodes
$\mathcal{M}$	An application message
τ	Timer timeout interval
T_i	Time-slot i
Δ	Per node tree repairing delay
Ω	Per hop delay (average link delay + queuing delay)
$\mathfrak{T}$	The amount of delay the running application can tolerate
Θ	Maximum latency for forwarding data
$\Gamma(u)$	The disc that is communication covered by a node u
Λ_u	The disc that is sensing covered by a node u
$\Psi(u)$	Area of redundancy for a primary node u
$\mathbb{R}(u)$	Set of dedicated redundant nodes for a primary node u
$\mathbb{A}$	Total area of the target terrain

TH-1251_11610108

LIST OF SYMBOLS

ρ	the deployment density
$\eta(l)$	The average number of nodes in the subtree rooted at a node u in the level l
ξ	The average number of children of an intermediate node u
$\mathcal{SS}_d$	Received signal strength at distance d
$\mathcal{E}_T$	Sensor energy dissipation in transmit mode
$\mathcal{E}_R$	Sensor energy dissipation in receive mode
$\mathcal{E}_S$	Sensor energy dissipation in sleep mode
$\mathcal{E}_I$	Sensor energy dissipation in idle mode
$\mathcal{E}_{Res}(x)$	Residual energy of a node u
$\mathcal{E}_{Th}$	Threshold energy for the failure of a node due to power exhaustion
$\mathbb{L}_{s,s'}$	Lane segment between any two sinks s and s', with the length $\mathcal{L}$ and breadth $\mathcal{B}$
$\mathcal{D}_X(u), \mathcal{D}_Y(u)$	Diameter of the sensing disc for any node u, projected along the X-axis and the Y-axis, respectively
$\mathcal{P}(l_1, l_2)$	Perpendicular distance between two lines l_1 and l_2
ϑ	Maximum deviation in the clock drift per second among all the available sensor motes
$\Xi_u^{T_i}$	Back-off function computed at the node u for the time-slot T_i
$\mathfrak{S}_{s_i}$	Skeleton of a tree dedicated for the sink s_i

TH-1251_11610108

Chapter 1

Introduction

1.1 Background

Wireless sensor network (WSN) is a distributed system where a large number of low-cost small sensor nodes act cooperatively to form a communication network. A sensor node comprises of four basic components, as shown in Figure 1.1, a sensing unit, a processing unit, a transceiver unit, and a power unit [12]. The sensing unit contains one or multiple sensors and analog to digital converters (ADCs). Sensors are small micro-electronic devices that can sense various physical parameters like temperature, light, vibration, pressure, sound, magnetic fields, motion, speed, etc. ADC is responsible for converting the observed phenomena into digital signals, which are then fed into the processing unit of the sensor node. The processing unit contains a small memory and operates all procedures that the node should execute to perform the sensing task cooperatively with other nodes. The transceiver unit acts as the transmitter as well as the receiver to communicate with other sensor nodes within short distances. Sensor nodes, interconnected through wireless links forming a multi-hop communication network, communicate through radio, infrared or optical media. Power units are equipped with limited power source. Sometimes sensor nodes are supported with energy harvesting module like solar cell, global positioning system (GPS) or a mobility module depending on application requirement. However in such case, the installation cost of the sensor nodes is increased, and the extra power consumption for the operations of these modules decreases the node lifetime. Being micro-electronic devices, sensor nodes are limited in resources like battery power, storage and capacity of the processor. Therefore, the design of sensor network applications strictly requires an efficient utilization of sensor resources.

A large number of sensor nodes are deployed in the target area either manually or

TH-1251_11610108

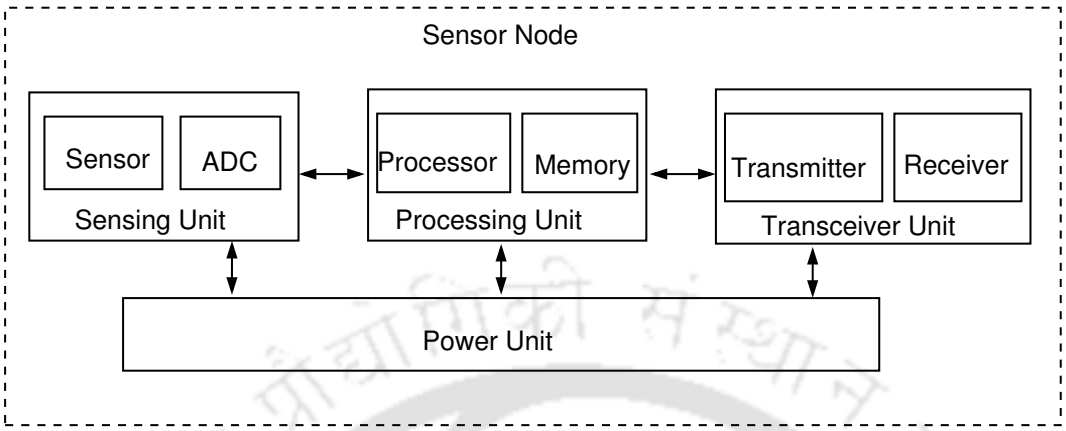

Figure 1.1: Schematic diagram of a sensor node

by some automated means like dropping the nodes from helicopter. Sensory data from the deployed sensor network is collected for monitoring critical infrastructure, battlefield surveillance, industrial automation and process control, health and traffic monitoring, and so on [12]. Proper distribution of sensor nodes and efficient collection of sensory data increases the overall network lifetime. A gateway or base station, that is accountable for taking all the management and control decisions for whole network may act as the source or the sink for all application data. A sensor network, deployed in an irregular terrain with the central base station positioning is shown in Figure 1.2. The communication and sensing area of every sensor node is assumed to be circular disc of radius R_c and R_s, respectively, as shown in the figure[1]. Usually, $R_c > R_s$. The main objective of sensor node deployment is to maintain the connectivity in the network such that every point in the area of interest is covered by at least one active sensor. The dotted lines in the figure represent the communication links between sensor nodes, which contribute to the formation of the communication graph. Depending on the purpose of monitoring and the flow of data with respect to the source or sink node, the running application can be of broadcast, convergecast or multicast. In a broadcast application, the data packets from the source node are forwarded to all sensor nodes in the network. On the contrary, in convergecast, the data packets from all sensor nodes are forwarded to the common sink node. In a query-response based application in sensor network, the query messages are broadcast from the base station; where as the response messages, as generated at every

[1]Though during actual communication and sensing, the communication and the sensing areas of a sensor node may not be perfectly circular due to anisotropic radio model as well as terrain geometry, this work considers the approximated inner circular region with maximum radius for theoretical modeling.

TH-1251_11610108

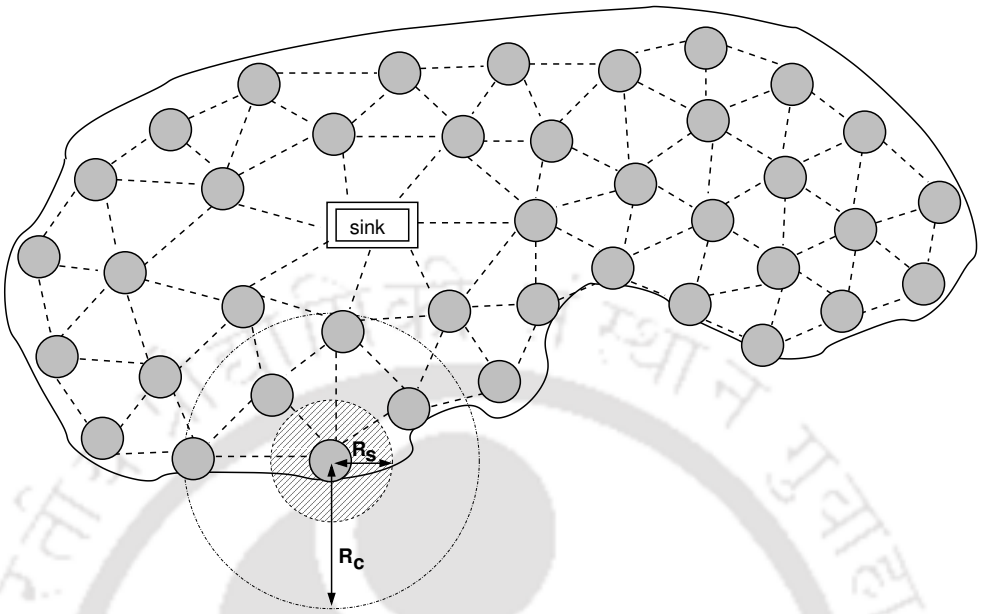

Figure 1.2: A sensor network and the corresponding communication graph

sensor node, are forwarded towards the base station. In a multicast application, the data packets are forwarded from a single source to a group of destinations. In this thesis, the design challenges of sensory data gathering application has been explored, which is based on the convergecast paradigm. The terms, base station and sink are used interchangeably in this thesis, unless specified otherwise.

1.1.1 Sensory Data Collection

Data gathering or convergecast [18, 123] is one of the most widely used applications in sensor network, where sensory data is forwarded from the sensor nodes to the base station for statistical analysis. The process of data collection can be either event-triggered or query-driven. In event-triggered data gathering, every sensor node generates sensory data packets, either periodically or on occurrence of an event, called an 'epoch'. For query-driven data gathering, sensor nodes generate the response data packets on reception of the query messages, broadcast from the base station. Whether it is event-triggered or query-driven, the generated sensory data is forwarded from every node to the base station through a multi-hop path. Sometimes, correlated data packets are aggregated at an intermediate sensor node before being forwarded towards the base station. This in-network data aggregation is a common practice to reduce the traffic load and power

TH-1251_11610108

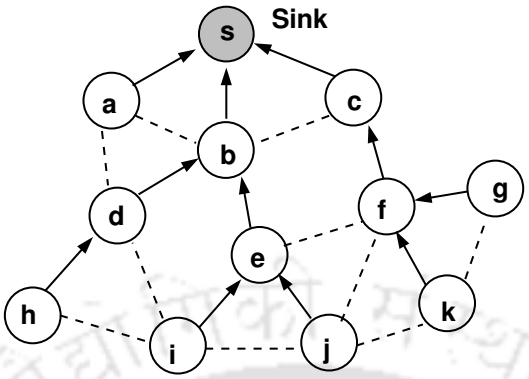

Figure 1.3: A BFS tree for data gathering rooted at the sink

consumption for resource constrained sensor devices. For example, data aggregation is possible for statistical queries like COUNT, SUM etc. However, in-network aggregation of sensory data is possible only when high degree of correlation exists among the source data. Applications like collecting video images for a battlefield surveillance do not support data-aggregation because of low correlation, and so, all data are required to be forwarded to the base station. Efficiency of data gathering depends on application specific performance criteria. Some applications require all data to be delivered to the sink without any loss. Another set of applications may focus on the strict bound on delay of the delivery of sensory data. Some other applications may require both the delay and the reliability constraints to be met for efficient data gathering. The delivery of redundant data might be a performance degrading factor for certain applications, whereas other applications would appreciate the redundancy in data delivery on reliability perspective. Therefore, depending on various application requirements, the definition of efficient data gathering varies. However, for every application there exists some minimum bound on delay, loss and redundancy on delivery of sensory data. Moreover, all applications require a certain level of transparency from the lower layer activities to provide an uninterrupted service to the end-users.

1.1.2 Data Gathering Tree

Inexpensive small sensor nodes are limited in resources. Non-replaceable batteries demand efficient power management for all sensors in the network. Energy dissipation is more for communicating through radio channels compared to sensing and local processing. A reduction in control message transmissions helps in reducing the power consumption at each sensor node. Energy savings at each sensor node increases the battery life for the node

TH-1251_11610108

while improving the overall network lifetime in effect. The traditional approach of data gathering exploits usual broadcast paradigm of communication, where data packets from every node is forwarded to all the nodes within its communication range. This flooding based data gathering incurs large overhead in terms of redundant message delivery and power consumption due to message explosion in the network. A spanning tree rooted at the sink is one of the best options to communicate among all nodes in the network. This is because tree based data collection offers maximum efficiency with minimum contention and forwarding delay. Forwarding data through a spanning tree towards the root ensures the ordered delivery of application data. Further, the communication overhead gets reduced as the data is forwarded with minimum duplicate delivery. The usefulness of different data gathering trees in different scenarios are mainly justified by various application specific requirements and physical parameters of the target terrain. A connected dominating set (CDS) based topology is used where the participating nodes are required to be interconnected through a common chain. The set of base stations, placed at a certain interval on the side of the roadways, form this kind of topology to maintain the backbone infrastructure. The depth first search (DFS) tree might be useful for certain geographic distribution of a sensor network. For delay-bound applications, the data gathering tree should support the shortest path distance from all nodes to the sink. A breadth first search (BFS) tree is a spanning tree where each node maintains the shortest path distance from the root [53]. Past researches [107], [41], have shown that the total number of packet relays as well as the delay will be minimum where as the the collection capacity will be maximum if data is forwarded through a BFS tree. A BFS tree rooted at the sink node for data gathering is shown in Figure 1.3 where the tree edges are represented by solid arrows and the non-tree edges by dotted lines. Every leaf node forwards only the self-generated sensory data to its parent node in the tree. Every intermediate node forwards the self-generated sensory data as well as all data from its rooted subtree to its parent node. In this way all data from the network are accumulated in the base station, which is the root of the tree. Depending on various application requirement different types of data gathering trees can be constructed with different objectives like low-latency and energy-efficient, fast data gathering, collision avoidance, maximum collection capacity, better aggregation, etc. as proposed in the literature [40,42,43,57,71,100,103,108,112,121,128,134,137,151,155,163,169,178,185,204].

TH-1251_11610108

1.2 Motivation

Efficient utilization of sensor resources through tree based data forwarding helps in improving the network lifetime. However, cheap sensor devices are failure-prone. Nodes may fail permanently due to either technical error or energy depletion. A sudden crash of an arbitrary sensor node may lead to the problem of incorrect tree structure, which in turn, affects the performance of the running application. Usually per node load increases along the walk from the leaves to the root of the tree. So, nodes near the sink are more prone to failure due to fast power exhaustion. The crash of all nodes in a particular level would make the tree disconnected and interrupt the data gathering process. Quality of Service (QoS) for the application is degraded because of data loss. An online repairing mechanism or building a new tree would not be a good option due to increase in control message overhead and delayed delivery of application messages. The increased rate of consumed energy per node would reduce the overall network lifetime in effect.

Redundancy in sensor deployment is used to ensure uninterrupted data delivery and improved network lifetime [162]. The failed node can be replaced by a suitable redundant node to continue data gathering. However, a proper estimation of redundancy based on the traffic load at different level of the data forwarding tree is required to ensure extended network lifetime. Further, in real life, the area of interest or terrain may be irregular as well as inaccessible in nature. Homogeneous deployment density would not be suitable for irregular terrain as the deployed redundant nodes might not be able to serve the faulty node in this case [209]. The amount of redundancy also depends on the nature of the data forwarding tree. If the data forwarding tree is unbalanced, even for the shortest path tree, the deviation in rate of energy dissipation between two nodes in neighborhood is more than that of a balanced tree. Uneven load distribution in the network leads to failure of highly loaded node due to fast energy exhaustion and thus reduced network lifetime.

Similar to arbitrary node failure, the periodic sleep schedule for sensor power management also affects the dynamics of underlying topology. An efficient scheduling scheme maintains proper sleep and wake up cycle for every node; and thus defers the first node die due to power exhaustion, which in turn extends the overall network life time. Any scheduling scheme partitions the time with either equal or weighted time slots. In every time slot a set of sensor nodes remains active, and performs data sensing and forwarding actively. The remaining nodes sleep during that slot, and saves energy. In the very next slot, the roles of wake up and sleep are exchanged among the set of sensor nodes such that the communication network remains connected all the time. As the nodes leave and join

TH-1251_11610108

the network periodically, the underlying data forwarding tree requires special maintenance activities to provide constant connectivity as well as the sensing coverage. Precisely, the performance of running application should not be affected due to changes in underlying topology.

Recent developments in wireless sensor network have proven it as an emerging as well as challenging research field to serve various critical delay-sensitive applications like military surveillance, plant automation and process control, etc [22, 105]. The QoS requirements, which vary from application to application, dictate the design of the topology management protocol. In border area sensor network, applications demand strong area coverage. If there is a node failure due to power outage or sudden crash, the node which replaces the faulty node should not produce any sensing hole. Reliable and secured delivery of sensory data to the sink are other important criteria that must be assured for data gathering from border area sensors. However, unlike, border area network, road sensor network do not impose the strict constraint on complete area coverage. For road sensor network, it is sufficient to maintain k-strip-length coverage. If m sensors are deployed along the length of the road, then assuring every passing vehicle is sensed by at least $k \leq m$ sensors is called the k-strip-length coverage. In this case, the security for sensory data delivery is not important, but the requirement of reliability in data delivery cannot be ignored. The data must be delivered to the sink within a delay bound. Overall, whether it is border area network or road sensor network, the basic objectives are as follows:

- Continuous sensing coverage should be assured based on the application demand. As discussed, border area network require strong area coverage, whereas k-strip-length coverage is sufficient for road sensor network.
- Network connectivity should be maintained all the time while ensuring the sensing coverage.
- Sensory data should be forwarded to the base station with minimum processing and forwarding delay. Most of the delay-sensitive applications require continuous sensing and forwarding of data and may not support data aggregation at intermediate nodes. Extra delay imposed over the data traffic due to repairing in the forwarding path should be minimum.
- Sensory data must be delivered to the base station with minimum loss.
- The network lifetime should be improved by efficient sensor resource utilization.

TH-1251_11610108

Existing works in the literature on tree based data gathering in sensor network [119, 177] mainly addressed the issues of energy-latency trade-off. Other notable works [58, 61, 71, 129, 137, 138, 158, 208] that dealt with the topology dynamics in tree based data forwarding are constrained with respect to various parameters like repairing delay, fault-tolerance, overhead, etc. The existing algorithms for distributed tree construction [25, 88] are particularly suitable for ideal scenario, and hence, are not applicable to serve the required purpose. Delay sensitive wireless sensor network has been studied in literature [9, 34, 135, 152, 166, 167] for offering improved end-to-end delay by tuning either MAC scheduling or routing path selection. The sensing coverage and node failure issues remained unaddressed. The sensor energy management through different sleep based schedule schemes [23, 99, 181, 194] has been an interesting research topic in literature. However, they are not suitable to serve various critical QoS constraints. Although a set of works exists on sensor deployment framework [24, 64, 76, 81, 109, 136, 157, 200], they suffer from different limitations like overhead, feasibility, uniform distribution of sensors, homogeneous environment, fault-tolerance, grid coverage, etc. Therefore, proposing an efficient topology management protocol with proper theoretical and practical justification, and considering sensing coverage and network connectivity, network lifetime and overhead, deployment pattern and fault-tolerance altogether were the key motivation to formulate the problems presented in this thesis.

1.3 Objectives

The main objective of this work is to design an efficient topology management protocol for tree based data gathering in sensor network sustaining any arbitrary node failure. The design paradigm should support extended network lifetime as well as improved QoS for the running application. The implementation overhead for the proposed scheme must be as low as possible. Further, the design should be flexible enough to customize the functionalities based on different application specific requirements. The lack of completeness in existing schemes in the literature and the necessity for a better framework lead to design a set of novel protocols with following key objectives:

- To design distributed tree construction scheme for data gathering from sensors considering application specific requirements for topology management.
- To propose a local tree maintenance scheme that can handle a set of arbitrary node failures, and is of low overhead.

TH-1251_11610108

- To handle the application message flow during topology maintenance and assure all QoS requirements for the running application.
- Considering irregular terrain nature, to propose an initial deployment framework based on redundancy that can handle potential node failures without affecting the connectivity and the sensing coverage in the network. Further, to provide an estimation of the required redundancy in sensor node deployment.
- To deal with the connectivity, sensing coverage, network lifetime, fault tolerance, and the low overhead while designing the tree management protocol.
- To customize the topology management scheme according to different application specific requirements, and improve the overall performance through efficient resource optimization.
- Finally, the verification and validation of the proposed schemes through mathematical analysis as well as simulation based results.

1.4 Solution Approach : An Overview

Considering the above objectives, a set of solutions have been proposed for the identified problems in this thesis. The theory of efficient data gathering in a sensor network has been developed with a design of fault-tolerant data gathering tree management protocol to satisfy the performance specific criteria like the network connectivity and sensing coverage, delay-sensitivity and low overhead, deployment issues and network lifetime, and the fault-tolerance and other application specific QoS. The proposed tree management protocol has been designed as a pair of modules that work cooperatively in the network layer. The Tree Management Module (TMM) is responsible for data gathering tree construction and the maintenance related activities on potential node failure. The Application Message Controller (AMC) is designed to handle the application message flow for assuring application specific QoS even in presence of node failure. An efficient coordination between TMM and AMC is the key steering factor to achieve the desired efficiency. As a part of the TMM activities, first, a distributed BFS tree construction scheme has been proposed through a set of distributed message-passing algorithms. According to the proposed scheme, initially, a spanning tree rooted at the sink node is constructed that is eventually transformed into a BFS tree through an incremental level improvement. The cost of tree construction in the proposed way is lower than that in existing schemes in the literature

TH-1251_11610108

in terms of control message overhead. During the tree construction, every node performs little preprocessing on neighborhood information locally to sustain potential node failure. The precomputation of an alternate parent on failure of the parent node in the tree helps in fixing the tree locally with minimum delay as well as low control overhead. To handle multiple node failures either in a series or in parallel, the activities of the proposed TMM has been extended using a maintenance module that acts reactively. The correctness of the proposed scheme has been justified through theoretical analysis. With performance analysis through simulation results, the proposed scheme of crash-tolerant data gathering tree management has been established as a first contribution of this thesis.

To introduce the effect of other design parameters like sensing coverage, sleep based energy saving, network lifetime and deployment pattern in the proposed solution of data gathering tree management, the initial node deployment has been emphasized. It has been observed that for tree based data collection, the nodes near the sink are more failure prone due to fast energy exhaustion as a result of higher load than the ones near the terrain periphery. This uneven energy depletion and consequent node failures degrades application QoS by affecting the connectivity and the coverage in the network. In tree based data gathering that does not support in-network data aggregation, traffic load of an intermediate node depends on the number of nodes in its rooted subtree. First, the number of children for any intermediate node has been estimated based on the connected-coverage criteria imposed by a BFS tree as well as the effect of potential node failure. Considering irregular nature of the terrain, nodes are deployed in high redundancy to sustain node failures. The density of deployment has been estimated that depends on the computed Energy Dissipation Factor (EDF), considering the gradient effect of energy depletion. The proposed EDF is defined as the ratio of energy dissipation of any intermediate node at a particular level to the one of a leaf node in its rooted subtree. The proposed theory has been developed considering both the *steady sensing* (where nodes remain active all the time) and the *periodic sensing* (where nodes follow sleep based energy saving) mode of energy dissipation. The estimated EDF is justified by theoretical analysis of sensor network calculus. In real-life the value of EDF depends on actual traffic load. The effectiveness of using the EDF for any node, that considers the total number of nodes in the rooted subtree, as a metric for the deployment density computation has been proved using the theoretical analysis and the simulation results.

Assuming an existence of the proposed gradient based initial deployment with redundancy, a fault-tolerant tree management scheme has been proposed next to satisfy the objectives. A set of redundant nodes is identified during the initial set-up of the

TH-1251_11610108

network for every individual primary node that participates in tree based data gathering. The redundancy based node deployment ensures that on failure of a primary node there exists sufficient number of redundant nodes such that the faulty node can be replaced by a redundant node from the dedicated set maintaining both the connectivity and the sensing coverage. The estimated deployment density exploiting the gradient based energy depletion of sensors assures that the network lifetime is extended by allowing the first level nodes to remain active until the leaf nodes die out of energy. Fast repairing also satisfies the required QoS for delay-sensitive applications. As uneven load distribution in the tree structure affects the network lifetime by triggering a fast failure of the highly loaded nodes, a distributed load-balanced BFS tree construction scheme has been introduced as a part of the proposed scheme. According to the theory, the proposed algorithms balance the uneven distribution of the children for any pair of adjacent nodes at the same level in any BFS tree. The tree management activities of the proposed TMM includes both the tree construction and the maintenance of the tree from arbitrary node failure. Two types of node failures, the *ageing*, that is the failure of node due to power outage and the *crash*, that is the failure of node due to tampering, has been considered while designing the TMM. AMC takes care of the application message flow during tree management, and thus assures reliable data delivery in spite of node failure. Performance of the proposed scheme has been compared with existing schemes in the literature along with the one proposed in the first contribution of this thesis.

At the end, to customize the proposed data gathering tree management scheme according to different application specific requirements, the problem of crash-tolerant data gathering has been explored under two special sensor network environments, separately. Both the Intelligent Transport System (ITS) and the Critical Infrastructure Information System (CIIS) demand strict QoS requirements like delay and reliability for the desired performance of the running application. Hence, these two different network environments have been considered to develop the theory of crash-tolerant data gathering tree management scheme by incorporating different application specific requirements in design criteria.

WSN have grown a significant attention among the researchers for providing a flexible and low-cost framework to design an architecture for ITS. Data gathering is one of the most popular applications of road sensor network where the sensory data, like the vehicle type, speed and direction etc., are accumulated in the road side gateways or sinks for traffic monitoring purpose. The inherent challenges in distribution and management of a sensor network along the road require an application-specific protocol support for

TH-1251_11610108

the network connectivity, sensing coverage, reliable data forwarding and the network lifetime improvement. *ADCROSS*, an efficient sensory data collection scheme for road surveillance applications has been proposed. The proposed scheme introduces the concept of *k-strip length coverage*, that ensures a better sensing coverage for the detection of the moving vehicles, compared to the conventional *barrier coverage* and *full area coverage*, in terms of the availability of sufficient information for statistical processing as well as the number of sensors required to be active. According to the proposed sensing coverage criteria, any passing vehicle to be detected by k active sensors along the length of the road segment. To extend the network lifetime, every sensor node follows a sleep-wakeup schedule maintaining the network connectivity and the *k-strip length coverage*. This scheduling problem is modeled as a graph optimization, the NP-hardness of which motivates to design a centralized heuristic, providing an approximate solution. As sensor network are inherently distributed in nature, the properties of the centralized heuristic are explored to design a per-node solution based on local information. Every node individually computes a back-off function to select its state for any particular time slot. The selection is globally consistent, and costs very low. For any particular time slot, the active set of nodes form the data gathering trees rooted at the nearest sinks, such that both the in-flow and out-flow traffic analysis are possible with respect to any individual sensor. The reliability and other QoS metrics like delay have been ensured for the delivery of sensory data. The proposed solution is not only distributed, but also localized in nature (the computations are based on local information only), that significantly improves the overall performance of the network.

Sensor network deployed for critical infrastructure monitoring requires high degree of reliability in sensory data gathering, in spite of arbitrary node or sink failures. *RelBAS*, a robust data gathering scheme, has been specially designed for the sensory applications on critical infrastructure monitoring, to provide a guaranteed delivery of sensory data. Redundancy in sensor network, in terms of both the number of deployed sensors as well as the amount of duplicate data delivery, is explored to design an effective protocol that ensures the reliable data delivery, while assuring the timeliness, connectivity and the sensing coverage. A set of active sensors are selected from all sensors deployed, based on the network connectivity and the sensing coverage criteria, that participate in the data forwarding process. The rest of the nodes go to the sleep state, and act as a replacement on failure of an active node. The proposed protocol finds multiple node-disjoint paths to multiple sinks, so that the loss of connectivity in one path due to node failure does not disrupt the application services. The forwarding path selection at every node in *RelBAS* is

TH-1251_11610108

based on the computation of a metric that is a function of two parameters - the hop-count distance to the sink and the node's residual energy. This helps in adapting the protocol to the application requirement depending on the forwarding delay and the energy efficiency. Moreover, *RelBAS* is capable of detecting an affected zone due to multiple node failures. The effectiveness of the proposed scheme has been analyzed using simulation results, and compared with other protocols proposed in the literature for reliable data delivery.

1.5 Organization of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 gives a brief overview of the literature survey related to the problems addressed in this thesis. A basic concept and the corresponding formal definitions for certain terms frequently used in sensor network environment are also defined in this Chapter.

Chapter 3 introduces a set of algorithms for distributed BFS tree construction. The proposed proactive and reactive tree repairing scheme for crash-tolerant data gathering is also given. Further, theoretical proofs of correctness and the simulation results have been provided for verification of the scheme.

Chapter 4 presents a gradient based node deployment framework. The mathematical analysis for gradient estimation and validation through sensor network calculus have been provided. The gradient is calculated considering both the cases, when nodes follow a sleep based schedule and when they do not. The simulation results show the correctness of the proposed theory.

Chapter 5 proposes a set of algorithms to construct a load-balanced BFS tree. A localized tree maintenance scheme has also been provided to support node failures considering the gradient based initial deployment with redundancy. The simulation results are presented to prove the effectiveness of the proposed scheme.

Chapter 6 presents *ADCROSS*, an efficient data gathering scheme specially designed for road sensor network. The concept of *k-strip length coverage* has been introduced. The scheduling problem for the active nodes selection is modeled as a graph optimization, the NP-hardness of which motivates to design a centralized heuristic, providing an approximate solution. The properties of the centralized heuristic are explored to design a per-node solution based on local information. The performance of the proposed scheme is analyzed through the simulation results.

Chapter 7 proposes *RelBAS*, a reliable data gathering scheme, specially designed for

TH-1251_11610108

the sensory applications on critical infrastructure monitoring. The distributed localized scheme for selecting the active nodes for tree construction has been provided. The proposed theory of multiple sink based data gathering, such that every node maintains multiple node-disjoint paths to many sinks, has been justified with lemmas and theorems. The activities of the proposed protocol for handling node failures have been described. Trade-off between the cost, in terms of redundancy, and the gain, in terms of reliability has been analyzed both theoretically and with the help of simulations results.

Finally, **Chapter 8** concludes the thesis with a summary of the contributions and an overview of the potential research works for future.

TH-1251_11610108

Chapter 2

Literature Survey

Wireless sensor network (WSN) has been identified as one of the key technologies of 21^{st} century [5,6] because of its ability to provide cost-effective, flexible and scalable solutions. After its first existence in the 1950s (Sound Surveillance System, SOSUS[1]), the formal research and development in the field of distributed wireless sensor network started in 1980 by United States Defense Advanced Research Projects Agency (DARPA) [50]. In a distributed sensor network environment large number of sensor nodes are deployed in the target terrain to form a communication network. Every *Sensor node* (also expressed as '*sensor*' or '*node*' in this thesis) is capable of physically sensing environmental phenomena and reporting the measurements through wireless communication as described in the previous chapter. A *phenomenon* is an entity of interest to an end user, such as temperature, pressure, light, speed, etc., that is being sensed and analyzed by the sensor network [172]. Sensing activities are either event-triggered or query based. In event-triggered sensing, the desired phenomena is sensed by the sensor either periodically or by detection of the occurrence of an event. In query based sensing, the end user indicates its interest by forwarding query packets in the network. Data packets with the required measurement (for event-triggered and query based sensing they are called '*epochs*' and '*response*', respectively) are generated at the corresponding sensor node on detection of the desired phenomena. If in-network aggregation is supported by the application, co-related data packets are processed at intermediate nodes for potential aggregation before being forwarded to the sink or base station. Otherwise, each individual data packet is independently forwarded towards the sink for further processing by the end user. Being micro-electronic devices, sensor nodes are limited in critical resources like battery power,

[1]SOSUS was developed by the United States Military to detect and track Soviet submarines.

TH-1251_11610108

memory, processing capacity, etc. The inherent challenges that can be identified for designing efficient data gathering protocols for sensor network are based on following characteristics [12, 27, 50, 172, 195].

- Sensor nodes are densely deployed with high number in total.
- Sensor nodes are failure-prone.
- Sensor nodes are resource-constrained.
- Sensor nodes mainly use broadcast communication paradigm.
- The topology of sensor network changes very frequently.

"A sensor network design is influenced by many factors, which include fault tolerance, scalability, production costs, operating environment, sensor network topology, hardware constraints, transmission media, and power consumption" [12]. The existing works on data gathering in sensor network are limited with respect to various constraints as discussed in following sections. Some of the works have addressed the problems of data gathering tree construction algorithms designed for distributed environment. The design paradigm is in abstract level and does not consider the feasibility issues in application perspective. Some works have concentrated on the development of general theory of data gathering by addressing the issues related to communication and routing path establishment. Most of these protocols lack in providing a complete solution covering both the theoretical and implementation aspects. Few works exist in the literature on tree based data gathering considering the dynamics in underlying topology as well as other performance metrics required by the running application. However, they suffer from various drawbacks to provide cost-effective, scalable and flexible solutions. In this chapter, the state-of-the-art works in the field of WSN and several issues of tree based data gathering have been discussed. To focus on the definite problems identified in this thesis, the discussions are mainly limited to the area of distributed algorithms for data gathering tree construction, data gathering and its challenges, fault-tolerance and topology control schemes, node deployment strategy, sleep based schedule techniques for energy saving, and various application specific performance criteria like delay, reliability, overhead, etc. For the sake of better readability and understanding of the problems, the state-of-the-art works have been described in following sections under four different categories. In first section the challenges and objectives for tree based data gathering in sensor network, that have been identified by the existing literature, are summarized. Second section discusses different fault-tolerant

TH-1251_11610108

issues as addressed by the existing works in the literature. Next, some notable works on both the theoretical and the practical aspects of wireless sensor network technology have been discussed, which include the area of connectivity and coverage, sensor power management, node deployment and various QoS requirements for the running application. Finally, the area of intelligent transport system and critical infrastructure information system have been explored for application specific customization of the proposed topology management protocols for data gathering from sensors. Several issues addressed by existing works in the literature along with their limitations have been discussed briefly in this section, followed by the summary of the chapter at the end.

2.1 Tree Based data Gathering in Wireless Sensor Network

Underlying topology of the communication network, formed by the active sensor nodes that participate in data gathering, can be of different types, such as star, bus, ring, mesh, etc. As discussed in the previous chapter, data gathering tree rooted at the sink offers maximum efficiency due to low contention and ordered delivery with minimum redundancy. So, the issues of tree based data gathering for sensor network have been thoroughly investigated in the literature [40, 42, 43, 57, 71, 100, 103, 108, 112, 121, 128, 134, 137, 151, 155, 158, 163, 169, 178, 185, 204]. In this section, different aspects of tree based data gathering, which include energy-latency trade-off analysis, data gathering tree construction technique and other topology control schemes, as addressed in the literature have been described briefly.

2.1.1 Energy-Latency Trade-off Analysis

The sink based data gathering is one of the most challenging fields of WSN that has drawn much research interests in recent years. Lindsey *et al.* [112] have investigated the energy-delay metric for data gathering, and proposed a near-optimal solution to increase the network life through power savings. The proposed protocol constructs a chain in greedy approach out of all sensor nodes such that each node communicates to a close neighbor, and in turn, only a designated node can forward the fused data to the base station. In another work [113], the authors have proposed a hierarchical scheme to reduce the $energy \times delay$ metric for data gathering. Yu *et al.* [199] have also studied the energy-latency trade-off for data aggregation. For a given aggregation tree, the proposed algorithm reduces the total power consumption under latency constraints. However, for all of the above works, the focus confined within the analysis of the data gathering performance in terms of energy and delay. Effect of underlying topologies on the performance of data gathering has not been

TH-1251_11610108

explored. Kesselman *et al.* [93] worked on the trade-off between the latency and energy of data gathering. Latency increases as the number of relay nodes is maximized while consuming minimum energy. There is a trade-off between reaching a node directly using high transmission power and reaching a node via multiple nodes using low transmission power. Their proposed algorithm has the expected running time of $O(\log n)$, and that is randomized in nature. Also, their proposed algorithm consumes at most $O(|n \log n|)$ times the minimum energy. Krishnamachari *et al.* [100] modeled a data-centric routing scheme and studied the energy-latency trade-off involved in the data-aggregation. The study includes an effect of the source-sink placements, the communication network topology, and the density of the network on convergecast. Depending on different routing models, a good amount of energy saving is possible adjusting those factors. They have shown that the latency of convergecast is proportional to the number of hops between the sink and the furthest source. They argued that the formation of an optimal data-gathering tree is an NP-hard problem. So, they proposed the sub-optimal data-aggregation tree generation heuristics that runs on a polynomial time for the special cases. The main objective of all these works is to reduce the cost of data transmissions in terms of delay, power consumption, data loss due to collision and so on.

2.1.2 Data Gathering and Topology Control

There exists a number of chain based, cluster based or tree based solutions for data gathering in WSN [154]. As pointed out in [151], most of them do not consider the coverage, connectivity and fault tolerance during the data forwarding. A connected dominating set (CDS) based topology control scheme has been proposed in [185] that focused mainly on the coverage issues. Here, nodes require the neighbor distance information. Additionally, no scheme for topology maintenance and the application data handling has been provided. A comparative study between different graph based topology control and the CDS based topology control has been reported in [125]. Most of them require a 2-hop neighborhood information. For highly dense sensor network, a topology control scheme has been proposed by Iyengar *et al.* [82] assuring the connected-coverage criteria and low-coordination among sensors. They proposed and evaluated different distributed random pattern based wakeup policies to construct multiple independent connected covered topologies. The proposed scheme is fault tolerant and offers low overhead in terms of message exchange. The work assumed the position information of every node, and did not consider the changes in the underlying topology. Moser *et al.* [132] have proposed a distributed topology construction algorithm based on the agreement on

TH-1251_11610108

minimal-weight clusters for creating a k-regular k-connected fault tolerant communication network. Their algorithm works in distributed asynchronous system and is able to adapt to the crashing or moving nodes and the changing communication cost. It also assures the sensing coverage. Moh *et al.* [130] have studied a comparison based evaluation between two dynamic tree based data gathering protocols, based on the distributed version of the shortest path tree(SPT) and the maximum leaf tree(MLT) algorithms. A localized tree reconstruction and flooding algorithm have been incorporated to take care of the joining and leaving events of sensor nodes. An accurate energy consumption and overhead in terms of control message exchange have been modeled through the simulation. The results showed that the performance of the SPT is better than that of the MLT. Upadhyaula *et al.* [177] proposed a heuristic algorithm for data gathering with minimum energy. The objective is to minimize the energy consumption during communication while ensuring low-latency through a faster data transfer and reliability through a collision-free transmission. The algorithm constructs a tree using a greedy approach where new nodes are added to the tree such that the weight on the branch to which it is added is less, and then allocates the DSSS or the FHSS codes. For data gathering, latency of a communication is defined as the time taken from the start of the transmission at leaf nodes until all data is received by the sink. The latency depends on the number of parallel transmission. Higher the number, lower the latency. The latency is improved by constructing a balanced tree such that the likelihood of multiple simultaneous transmissions in a given time-slot gets increased. In [42], the authors have designed a load-balanced data gathering protocol, where the traffic load is shared among the non-leaf nodes. When the energy of a non-leaf node is lower than a designated threshold, the tree structure is readjusted to prolong the lifetime of the tree. Annamalai *et al.* [178] proposed a heuristic algorithm for tree construction and channel allocation to provide a collision free convergecast. The algorithm constructs a tree and allocates a schedule for each node that specifies the time-slot(s) in which it can transmit data to other nodes. The tree is constructed as well as the schedule is allocated to the nodes at level by level. The main objective in this case is to achieve a greater efficiency in terms of latency and power consumption for both the convergecast and the broadcast. Tan *et al.* have proposed an energy efficient data gathering protocol, called PEDAP [169]. The protocol constructs a minimum cost data gathering tree rooted at the sink node based on Prim's algorithm. The algorithm runs in $O(n^2)$ complexity and performs in-network data aggregation. In distributed sensor network environment, the implementation of this centralized scheme is not feasible. Further, the scheme is limited in terms of satisfying connectivity-coverage criteria and other QoS related metrics.

TH-1251_11610108

Authors in [158] have introduced the concept of source based routing trees. A routing tree is constructed rooted at every source node on demand for efficient congestion control through resource control technique. This is different from sink based routing, where single routing tree rooted at the sink node is constructed, in the sense that in this case not all the nodes participate in every tree. There can be multiple trees rooted at different source with single final destination as the sink node, such that there exist multiple options for choosing a data forwarding path. However, this scheme requires the position information of sensor nodes and also does not deal with sensing coverage and fault-tolerance issues. Further, construction and maintenance of source based trees involves high overhead when the number of sources is high. The delay bound issues on minimum degree spanning tree problem have been investigated by the authors in [103]. They proposed a tree structure called DB-MDST (delay bounded minimum degree spanning tree) to achieve efficient data gathering in terms of both the per node fairness in energy consumption and the delay bound. The objective of the proposed tree construction is to reduce the height while increasing the degree. The proposed scheme is centralized in nature as the global graph knowledge is required and the application supports the collection of aggregated queries.

2.1.3 BFS Tree for Data Gathering

Li *et al.* [107] have analyzed the complexity of the data collection in terms of message transmission, power consumption and delay for WSN. They claimed that the total number of packet relays as well as the delay will be the minimum if forwarded through a BFS tree. In another paper [40], the authors proved that the data collection using a BFS tree offers maximum collection capacity. So, using a BFS tree would be a good option for data gathering as well as data aggregation, because the hop-count distance between the sink and each sensor node in the network is the minimum for a BFS tree [53]. Using the traditional Bellman-Ford algorithm a BFS tree can be constructed with the communication cost of $O(|V||E|)$, where V and E be the set of vertices and edges, respectively. However, the construction of a BFS tree in distributed environment usually involves an explosion of control messages due to the flooding, which incurs a significant overhead in terms of communication cost. So, the study of an efficient distributed BFS tree construction is also an interest of research since long [88], [25]. Though the above mentioned works attempted to gain a low latency and a reduced energy dissipation at sensor nodes, they are abstract in nature and particularly suitable for an ideal scenario, such as stable and static network. In a real-life, the environment is volatile. A node may crash at any point of time or may leave the network, which have not been considered in any of these works. Lu *et al.* [119]

TH-1251_11610108

have proposed an energy-efficient scheme for data gathering in a large scale system. A Breadth-First Search (BFS) tree is constructed level by level such that the leaf nodes are more than the intermediate nodes, and thus, the power is saved by turning off the radio of the leaf nodes, as well as minimizing the distance to the root. In [43], the authors have proposed a shortest path tree (SPT) construction algorithm, based on balanced BFS tree, to prolong the network lifetime through energy balancing in the network. The tree protocol dynamically constructs an approximate SPT based on the energy metric and hop distance. They have also designed an adjustment protocol that dynamically adjust the tree structure based on local information gathering during data communication. However, they have not considered the effect of sudden node crash over the data gathering procedure.

2.2 Fault-tolerance

Nodes near the sink, with high traffic forwarding load, are more prone to failure due to a fast power exhaustion. Node failure, either by power exhaustion or by a sudden crash would affect the connectivity and the sensing coverage of the network, and interrupt the data forwarding process. Alwan *et al.* [15] have studied the pros and cons of different fault-tolerant routing techniques in WSN. The survey classifies the routing into broadly three categories, the proactive, reactive and hybrid routing mechanism. They have also classified the fault tolerance mechanism in broadly two categories, re-transmission based and replication based. Their study have identified some of the key issues like the cross layer interaction with power management and the trade-off between different parameters like memory usage, power consumption, fault tolerance, delay and bandwidth utilization to be investigated for the performance benefit. Fault-tolerance support as proposed by different notable works in the literature are discussed in two subsections: first, the topology maintenance schemes for tree based data gathering and second, other strategies to support fault-tolerance.

2.2.1 Tree Maintenance from Node Failure

Few works exist in the literature that provide the robustness and reliability for the fault tolerant tree based data forwarding [58, 61, 78, 208]. In [208], the nodes having remaining energy below a certain threshold are considered as critical nodes. The tree for data aggregation is built in such a way so that the critical nodes become the leaf nodes of the tree. The main motivation of the paper is to reduce the data aggregation overhead at critical nodes, and prolong their lifetime before they fail. This scheme avoids the repairing

TH-1251_11610108

overhead as only the leaf nodes can fail. However, extra delay is introduced at the network as the nodes near the sink can act as the leaf nodes. The proposed work in [78] assumed the existing topology as a complete graph, and provided a self-stabilizing spanning tree construction algorithm considering the node join and leave events. This approach is not applicable for the general topologies of sensor network. Fei *et al.* [61] have proposed a proactive approach of tree recovery for multicast under the degree constraint. Here, every non-leaf node precalculates a spanning tree out of its parent and the children satisfying a degree constraint, and finds a *"parent-to-be"* node for each of its children. If a non-leaf node leaves, all of its children set their parents to the precalculated *"parent-to-be"* node. This approach is not suitable for the sensor network as nodes need to communicate with its grandfather or descendants in the tree for new path set up, which is definitely a power consuming strategy. Cases that deal with the failure of the leaf nodes are not considered in this scheme. Also, the scheme is specific to the multicast application, and works for a degree-constrained spanning tree only. Authors in [137] have proposed an autonomous tree maintenance scheme for WSN with the assumption that all nodes including the sink are homogeneous, and can leave the network. The paper lacks the theoretical complexity analysis and the coverage issues. Neighbor information is maintained at each node, and the maintenance scheme is not local also. England *et al.* [58] have proposed a scheme where a spanning tree is constructed based on the weighted average of hop-count and weight of the tree edges. The weight of the tree edges is a function of the remaining power of the end nodes. Therefore, this topology construction method is adaptive to a node failure. As the spanning tree construction is based on the traditional Bellman-Ford algorithm, the overhead due to the tree construction is too high compared to the maintenance overhead. Adjusting tree edges periodically, as proposed in [208] and [58], is costlier than the local tree repairing. All the above schemes do not consider the failure of the multiple nodes. Furthermore, all the past researches on fault tolerant wireless sensor network have concentrated on the faults that occurred during the routing path selection only [8]. Their study did not consider the effect of node crashes, whether in a series or in parallel, on the performance of the running application. Gnawali *et al.* have proposed highly reliable and efficient collection protocol for tree based data forwarding, called the collection tree protocol (CTP) [71]. A collection tree is constructed on the fly with minimum cost to each of the nodes that advertise themselves as the root of the tree. The collection is address-free, i.e. with multiple base stations, data is forwarded to the one offering minimum cost path without knowing its address. The CTP uses two mechanisms to make the routing protocol robust: it considers channel link dynamics through periodic

TH-1251_11610108

beaconing and avoids loop creation through data paths validation.

1. Adaptive beaconing: Collection layer updates the state routing information by sending control beacons.

2. Data path validation: Every node maintains an estimate of the cost of its route to a collection point.

In CTP, data is forwarded from the sensor nodes to the sink using a source based routing mechanism. When a fault is detected in the routing path, a new routing path is established on-the-fly using a source based path finding mechanism. Additionally, every sensor node uses the per-flow queuing to handle the data loss and redundancy during the path establishment and the path change. Though the CTP provides an efficient data forwarding and a fault tolerance mechanism, it is known to perform poorly in highly loaded network [129]. Furthermore, the per-flow queuing is hard to maintain at a low capacity sensor nodes. Moeller *et al.* have proposed a back-pressure routing protocol [129] for an efficient sensor network data forwarding. In case of the back-pressure routing, forwarding decision is made per packet basis. Both the CTP and the back-pressure routing protocol recomputes a new path when a fault is detected in the routing path. This increases the repairing delay in the network.

2.2.2 Supporting Failure in Hierarchical Network Framework

There exist few works in the literature that tried to support failure in forwarding paths of data gathering tree through designing hierarchical architecture of network with heterogeneous nodes into consideration. Radi *et al.* have presented the basic concept of the multi-path routing and its challenges with respect to WSN in their recent work [148]. They have also given a comprehensive taxonomy along with a comparative pros and cons of different existing multi-path routing protocols considering the network point of view. A disjoint multipath routing mechanism is proposed in [36], where data is forwarded through more than one paths, so that in spite of node failure, data can be forwarded to the sink within a guaranteed delay. However, the scheme is not energy efficient as same data is forwarded through multiple paths. Authors in [20] have proposed a two tier network for fault tolerance in sensor network, where the network is divided into the cluster nodes and the relay nodes. The cluster nodes gather sensing data, and the relay nodes forward the data to the sink. The objective of the paper is to select proper relay nodes such that the network lifetime is prolonged. The scheme is not suitable for delay sensitive network

TH-1251_11610108

as the switching between a cluster and a relay node introduces an extra network delay. An approximation algorithm to ensure the fault-tolerance and the sensing coverage have been proposed in [60] while maximizing the network lifetime. A complex sensing event is decomposed into smaller atomic events to ensure the correct delivery of the data while minimizing the energy requirement. However, the algorithm is not delay guaranteed. A relay based fault tolerant sensor network topology is proposed in [21] where the relay nodes are special nodes with a higher energy. This type of heterogeneous sensor deployment is costly in practice, and can not be useful for applications like military surveillance. Authors in [138] have proposed a virtual robust spanning tree based protocol for data gathering maintaining the point coverage in WSN. Here, the sensing and relaying activities are distributed among nodes. Every node is assumed to know the location of all the targets. The tree construction algorithm selects a tree edge based on the link weight and the hop-count distance.

All the above mentioned works on tree based convergecast and topology control for general WSN are not applicable directly to the applications of sensor network having specific performance criteria. The limitations of the above works can be summarized as follows:

- Most of the existing works do not ensure the coverage, connectivity and fault-tolerance simultaneously [151], which are essential to be considered for the applications with strict QoS requirements. The existing works that ensure the coverage during data forwarding require either a position information or are centralized in nature.
- Most of the existing works as reported in [125] use the reactive path repairing technique, which is costly for the delay sensitive applications in road sensor network or critical infrastructure information system.
- During the path establishment after a node fails, the reliability of application message delivery should be maintained. The existing data forwarding schemes do not consider the reliable delivery of application messages during the repairing.

2.3 Wireless Sensor Network: Theory and Practice

Wireless Sensor Network is a broad area to count a large number of past researches as well as the current on going works. To get a hold on the background of the problems addressed in the thesis, the study of this area is mainly limited to the discussion of a few notable

TH-1251_11610108

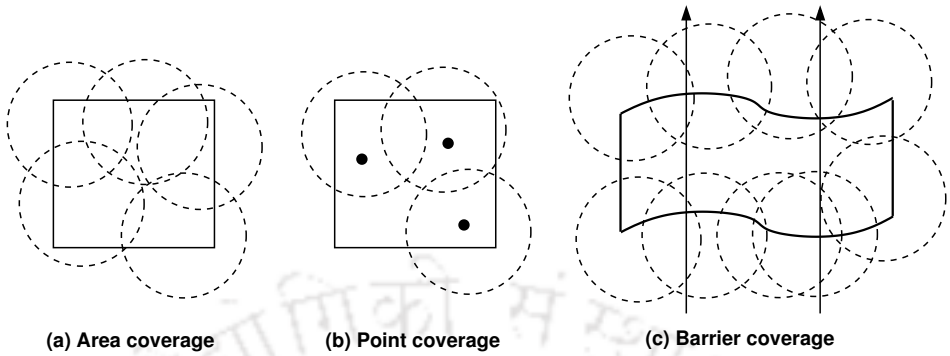

Figure 2.1: Different types of sensing coverage

works on connectivity-coverage issues, deployment strategies, sensor power management techniques and other problems satisfying different application specific QoS like delay.

2.3.1 Network Connectivity and Sensing Coverage

Network connectivity and sensing coverage are two important metrics that are required to be supported by every data gathering protocol designed for sensing network [56]. Sensor nodes are deployed in the target terrain satisfying the sensing coverage criteria as required by the running application [30, 31, 80, 84, 92, 202]. In *point coverage* [179, 191], a set of target points are required to be sensing covered by the sensor nodes, where as in *k-area coverage* [32] every point in the target area needs to be covered by at least k number of active sensors. On the other hand, *k-barrier coverage* [101] demands that the any movement along the width of target belt region must be detected by at least k active sensors. Based on the purpose of monitoring or tracking, the sensing coverage criteria varies as shown in Figure 2.1. While the first objective is to sensing cover all target points, the next important concern is to maintain the connectivity among these sensors such that they can communicate to each other for data gathering. To access the sensory data at the sink, it is required that the network remains connected at any point of time maintaining the sensing coverage [207]. Most of the cases in theory, the sensing and communication area of a sensor node in homogeneous environment are considered to be unit disc. In practice, the communication and sensing area of every sensor node are assumed to be circular disc of radius R_c and R_s, respectively. However in reality, based on the terrain geometry and the actual dimension of the deployed sensor nodes, the communication and sensing area of a sensor are not perfectly circular according to anisotropic radio model. Although, this does not affect the performance of the MAC protocols, the routing protocols are affected

TH-1251_11610108

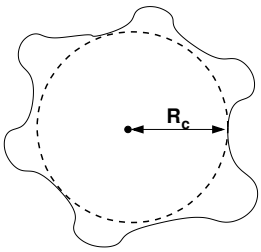

Figure 2.2: Approximation of the anisotropic radio model

as the set of communication and sensing neighbors for every node varies with time. From the received signal strength index or with the help of periodic beaconing [19, 206], the communication and sensing area for every node can be approximated as the inner circular region with maximum radius, as shown in Figure 2.2. The coverage and connectivity issues have been studied by Ostovari *et al.* [138]. They have introduced a point coverage and the connectivity mechanism that computes the waiting time to reduce the number of sensors. Virtual spanning tree and modified virtual spanning tree are proposed to maintain the connectivity. The data forwarding is done via a tree construction as edges are selected based on a combination of the number of hops and the distance metric. There exists a trade-off between the data loss and the energy consumption for their proposed scheme. In [17], the authors have provided a scheduling algorithm based on k-coverage. However, their proposed scheme does not consider the node failures. A random scheduling is proposed in [48] that assures the network coverage. In each round, k sensors are chosen to meet the desired sensing coverage. The selection is based on the geometric probability. Though the scheduling mechanism provides a guaranteed coverage, it incorporates an extra delay in the network, and also, does not support the node failure. The work in [26] provides a probabilistic scheduling by ensuring the connectivity and the coverage, but can not bound the network delay. Their scheme is also not fault tolerant.

2.3.2 Sensor Deployment Strategy

Initial deployment of sensor nodes plays a crucial role in prolonging the network lifetime while maintaining the connectivity and the coverage. A set of works have been proposed to deploy the sensor nodes in the network such that some predefined QoS requirements are satisfied [24, 64, 136]. Most of the applications require the area of interest to be sensing covered by enough number of sensors. However, finding the minimum number of sensors to cover an area is known to be NP-hard [68]. A two-level deployment strategy

TH-1251_11610108

is proposed in [81], where the low power sensor nodes forward data to the high power micro-servers, that in turn, forward data to the sink. This type of heterogeneous sensor deployment is difficult for rough and irregular terrain. Authors in [189] have designed an integer linear programming formulation to find the minimum cost deployment of sensors that provides a desired coverage. They have proposed a greedy algorithm to solve the integer linear programming. Their proposed algorithm can also provide a fault tolerance for k-coverage. That means the proposed deployment strategy is able to monitor all the points as long as $k-1$ sensors fail. However, the scheme supports the grid coverage only. They have not considered the gradient effect of energy dissipation for tree based data forwarding. The sensors are deployed uniformly in the terrain. Author in [157] have proposed a method to estimate the number of nodes to be deployed in a given area for a predetermined lifetime. In this method, the area is divided into an equal size strips. The density of the deployed sensors increases as the distance between a strip and the sink decreases. The method neither considers the coverage issues nor handles the node failure situations. A predetermined sensor deployment strategy is proposed in [76] to prolong the network lifetime. Their proposed scheme assumes the location information of the sensors which is hard to find out in an irregular and rough terrain. Though the scheme supports the coverage and the load based deployment of sensors, it is only suitable for a small terrain, where manual node placements are possible. Authors in [200] have studied different deployment patterns to achieve the full sensing coverage and k-connectivity under different ratios of R_c to R_s for homogeneous WSN. They proposed that there exists a universally elemental pattern that is hexagon based, which is able to generate all known optimal patterns with different connectivity. They designed a new deployment-polygon-based approach for proving the pattern optimality. The proposed scheme does not consider the traffic load pattern in the sensor network that is important for delay sensitive tree based data forwarding. The given theoretical analysis is abstract in nature that does not deal with implementation specific feasibility issues like network lifetime, fault-tolerance and QoS requirements. A gradient based sensor deployment scheme has been proposed in [109] to assure the complete coverage and the maximum network lifetime. The average angle between sensors is computed to find the number of sensors to be deployed in the monitoring area to ensure full coverage. A centralized deployment strategy for choosing minimum number of sensors maintaining full coverage has been proposed that is assumed to be executed by the sink node. The problem is formulated as the multiple knapsack problem that considers different energy level for different sensor nodes. Nevertheless, their scheme supports the regular terrain only and is not feasible to be implemented in

TH-1251_11610108

Table 2.1: Comparison of different sensor deployment strategies

Ref.	Terrain	Coverage	Node Distribution	Node Placement	Failure Model	Redundancy
[81]	irregular	yes	uniform	one-by-one	no	no
[189]	regular	yes	random	mass	yes	no
[157]	regular	no	gradient	mass	no	no
[200]	irregular	yes	uniform	mass	no	no
[76]	regular	yes	gradient	one-by-one	no	no
[109]	regular	yes	gradient	mass	no	no
[162]	regular	yes	uniform	mass	yes	yes

practical scenario that is distributed in nature. The requirement for redundant sensor deployment is analyzed in [162]. Their proposed deployment scheme with redundancy ensures k-coverage and m-connectivity in the network. However, it assumes an uniform distribution of sensors in the terrain, and does not consider the effect of node failure in the data forwarding tree. Different sensor deployment strategies have been summarized in Table 2.1. It can be seen from the table that the existing schemes do not consider all the aspects of deployment design for the sensor network. A deployment strategy is required to be designed considering the fine grained area coverage and the distribution of the traffic load from the border of the terrain towards the sink, exploring the redundant deployment of sensor nodes for failure handling. Further, the complexity for the estimating traffic load distribution increases for an irregular terrain.

2.3.3 Sensor Power Management

As the inter-node communication consumes maximum power [182], energy saving is a crucial requirement for resource-constrained sensors. Interference among neighbors, idle listening and overhearing are other major sources of energy wastage. One of the key issues in WSN is to prolong the network lifetime without handling node failure due to energy exhaustion. Past researches have contributed to the field of sensor energy management by introducing different sleep based schedule schemes [23, 99, 102, 181, 194]. Properly defined sleeping schedules enable every sensor node to switch its state between the active and the sleep periodically, and thus, extend the node lifetime by saving critical battery power. The traditional sleep based energy saving has been introduced by *Ye et al.* [193], where they proposed SMAC protocol. In SMAC protocol, every sensor sleeps for some time, and

TH-1251_11610108

then wakes up and listens to see if any other node wants to talk to it. During sleeping, the node turns off its radio, and sets a timer to awake itself later. On wakeup every node waits for a random amount of time. If it does not receive a schedule packet within that time, it generates its own schedule and broadcasts it through SYNC packet. On receiving a SYNC packet, every sensor follows it and rebroadcasts the schedule. Boundary nodes may receive more than one schedule and follow the wakeup duration for all schedules. SMAC implements the adaptive listening by overhearing a neighbor's transmission. Every node remains in the wakeup state for a small duration to check whether it is a potential receiver or not. However, the SMAC protocol does not consider the coverage and the connectivity criteria during scheduling. *Shen et al.* [161] have proposed the coverage aware sleep schedule, where a sensor node goes to sleep mode if its coverage area is already covered by some other sensor nodes. The protocol works on the calculation of a probability factor with the concept, that the greater the overlap of a sensor's coverage with its neighbor, the higher the probability that it can go to the sleep state. At each cycle the probability is calculated, and based on that, first β_s percentage of nodes is set to the sleep state, where β_s is a predefined parameter. The protocol follows a cluster based approach, where the cluster head determines the schedule for the sensors in a cluster. To utilize critical sensor resources like energy, a set of protocols like DMAC [118] and its variants [94] have been proposed in the literature. In DMAC protocol, the sensor nodes that neither transmit nor receive, go to sleep state to save energy. SMAC protocol suffers from a data forwarding interruption problem as all nodes on a multihop path to the sink can not be notified of data delivery in progress, which incurs significant sleep delay. DMAC solves this problem by giving the schedule of a node an offset based on its depth. This scheme allows continuous packet forwarding and adjusts node duty cycles adaptively according to the traffic load in the network by varying the number of active slots in an schedule interval. DMAC ensures high data reliability with significant energy saving while reducing the latency. The amount of sleep duration increases gradually from the sink to the leaf nodes of a data gathering tree where in-network data aggregation is not possible [205]. Thus, the energy dissipation of the leaf nodes is significantly less compared to the nodes near the sink. This results in an early die-out of sensors near the sink, and affects both the connectivity and the coverage of the network. However, most of these schemes require nodes to be synchronized globally, which is a costly affair considering the distributed nature of the sensor network. Secondly, maintaining the connected-coverage criteria in the network through an efficient schedule synchronization is costlier than the initial set up. Lastly, the sleep based schedules introduce an extra handshaking delay between the nodes during the

TH-1251_11610108

state transition period [167]. Many delay-sensitive applications do not allow this extra delay as it affects QoS for the running application.

2.3.4 Delay Sensitive Applications

Delay sensitive wireless sensor network has already been studied in the literature [9, 34, 135, 152, 166, 167] for offering an improved end-to-end delay by tuning either the MAC scheduling or the routing path selection. To enforce the delay sensitivity along with the energy efficiency, authors in [167] have used a separate signaling mechanism to forcefully wake up a node from the sleep state on availability of data. This forced wake-up technique requires a hardware interaction, and thus, is not power-effective. A topology control mechanism, proposed in [9], selects the next hop based on the wake-up schedule. The initial handshaking for the next hop selection consumes an extra power. Also the extra handshaking delay is introduced in the forwarding path. A centralized TDMA based MAC protocol for sleep-wakeup scheduling has been proposed in [166] that reduces the network delay. The DGRAM protocol proposed in [152] also uses a TDMA based scheduling and the scheduling mechanism is centralized in nature. The scheme proposed in [135] defines a channel monitoring interval (CMI) in which all nodes overhear the ongoing communications, and then, decide the sleep-wakeup schedule. The end-to-end delay in their scheme depends on the CMI. If the CMI is chosen too low to decrease the end-to-end delay, then the nodes may not capture the correct network information.

Recent works that addressed the issues of sensing coverage, fault-tolerance and network lifetime for delay sensitive WSN have been summarized in Table 2.2. The observations from the analysis of these related works can be outlined as follows:

(a) Sleep based schedule schemes are ineffective for delay sensitive wireless sensor network as they introduce an extra delay in the data forwarding due to schedule management. To design a delay sensitive network with sleep based schedule, a fine grained synchronization is required in the network that is practically impossible in a distributed environment.

(b) Online tree repairing during node failure imposes an extra delay in the network. Most of the works that deal with the challenges in fault-tolerant sensor network, remain silent about the coverage issue after a node fails.

(c) Maintaining the coverage after node failure is also costly as proposed in [60].

TH-1251_11610108

Table 2.2: Comparison of different related works

Ref.	Delay Sensitive	Coverage	Fault-tolerant	Network Lifetime	Mode
[208]	no	no	partial	partial	distributed
[167]	yes	no	no	no	distributed
[48]	no	yes	no	yes	distributed
[9]	partial	no	no	no	distributed
[61]	no	no	partial	partial	distributed multicast
[166]	partial	no	no	no	centralized
[152]	yes	no	no	no	centralized
[17]	partial	yes	no	yes	centralized
[26]	no	yes	no	yes	distributed
[36]	yes	no	yes	no	distributed multipath
[20]	no	no	yes	yes	distributed
[60]	no	yes	yes	yes	distributed
[135]	partial	no	no	no	distributed
[21]	yes	no	yes	yes	partially distributed

2.4 Application Specific Customization

The Libelium white paper [1] has reported the top 50 sensor applications, such as smart parking, traffic congestion control, waste management, forest fire detection, river floods monitoring, supply chain control, machine auto-diagnosis and assets control, medical fridges and many more, to build a smart city and to reach the goal of the *'Internet of Things'* (IoT). Such applications require developments of efficient sensors to detect and monitor application specific events, as well as designing of efficient protocols for sensory data collection and processing to extract necessary information. To review the effect of application specific requirements on the design of data gathering protocols, the thesis focuses on the Intelligent Transport System (ITS) and the critical infrastructure information system (CIIS). The increasing threats of vulnerabilities on the protection of critical infrastructure have motivated to contribute to this field with the design of robust data gathering schemes.

TH-1251_11610108

2.4.1 Intelligent Transport System

WSN has emerged to become an integral part for the design and development of ITS. Road sensor network for road and traffic monitoring, one major section of ITS, is the focus of the thesis. As the sensor nodes are deployed along the roadways for traffic monitoring purpose, sensory data are accumulated at the road side sinks. Designing an efficient data gathering protocol with failure support sustaining application QoS and required connectivity-coverage criteria involves several challenges. The state-of-the-art works in this area have been described briefly in following subsections with various steering factors.

Hardware Requirement:

Most of the works in the literature, based on sensor network for the development of ITS, have mainly focused on the design aspects of low cost and effective sensors that can detect road conditions and moving vehicles. Different types of sensors have been designed for vehicular traffic monitoring, such as, optical remote sensing [139], air-bone sensors [140], laser scanner sensors [65], magnetic sensors [2] etc., out of which magnetic sensors have been proved to be the most reliable and cost-effective solution [46, 203]. In [176], the authors have shown that magnetic impedance sensors provide better low-field sensitivity to detect the passing vehicles based on the Earth's magnetic flux. With the advances in the sensor hardware research for efficient road-surveillance, many algorithms have been designed for the development of ITS utilizing the advantages of magnetic sensors. The design objectives of these algorithms mainly include the detection of passing vehicles [73], the vehicle speed measurement [143], traffic information prediction [37, 141], collision warning [168], and the traffic light control [35, 174] etc.

Vehicular Sensor Network Vs. Road Sensor Network:

Though the design and use of sensors for ITS have been widely studied in the literature, its networking aspects are still under-explored. As discussed earlier, sensor network for road-surveillance demands special design issues in terms of deployment parameters for the network architecture, topology control, sensor scheduling based on the connectivity and the coverage and the design of data forwarding protocols based on road traffic characteristics. Two types of network architecture are explored in the literature - the vehicular sensor network [39, 75, 98, 111, 131, 198], where the sensors are placed inside the vehicles, and the road sensor network [86, 87, 106, 187, 201], where the sensor nodes are placed along the roads or the pavements.

TH-1251_11610108

In a vehicular sensor network, sensors are placed inside the vehicles, and sensory data are forwarded either to the neighboring nodes or to the roadside sinks, for further processing and information extraction. The nodes in a vehicular sensor network are mobile in nature, that imposes several research challenges for the design of data collection protocols. In [98], the authors have proposed a coordination mechanism among the vehicular sensors to design a data collection protocol that aims in minimizing the network congestion. They have used a hybrid model where the mobile vehicular sensors forward data to the static roadside sensors. For congestion free data collection, their proposed scheme uses a collaboration mechanism among the static and the mobile sensors, based on the position information obtained from Geographical Positioning System (GPS). However, their scheme requires an absolute position information, and GPS is costly enough to use with every sensor node as it consumes a significant amount of power. Momen *et al.* [131] have proposed a random structure vehicular sensor system, where the network coverage is guaranteed based on the vehicular mobility model. In [111], the authors have developed a multi-hop data dissemination protocol for vehicular sensor network based on the absolute positioning information obtained from the GPS server. Though their protocol reduces the transmission delay, it suffers from extra power consumption required for the GPS functioning. Haddadou *et al.* [75] have proposed another data dissemination protocol for vehicular sensor network using a diffusion based approach. Though the design of a vehicular sensor network have classified it as a cost effective solution for vehicle monitoring, it has some disadvantages for road surveillance. First, it is practically difficult to place the sensors on every vehicle, and the sensors may also be affected due to tampering. Next, in the context of network architecture, it is difficult to design a proactive mechanism for data collection from the mobile sensors. Further, mobile sensors may not guarantee the required connectivity and the sensing coverage always, and therefore, leaves the chances of false information prediction.

On the contrary, the road sensor network can provide a trustworthy architecture for road surveillance and vehicular traffic management. However, the design aspects of such a network are not being well-investigated in the existing literature. In [87], the authors have designed a passive localization algorithm for road sensor network. Xie *et al.* [187] have designed a topology management algorithm for road sensor network. The topology uses a linear placement strategy to reduce the load of the network traffic. However, their placement strategy does not guarantee the road coverage, and a vehicle may remain undetected. In [106], the authors have designed an optimization problem to reduce the communication cost in a road-sensor network. Being a non-convex joint optimization

TH-1251_11610108

problem, their solution is hard to implement in a distributed sensor network. Coleri *et al.* [51] have designed a traffic surveillance system based on road sensor network. They have used the *Power Efficient and Delay Aware Medium Access* (PEDAMACS) [59] protocol for the sensor scheduling to improve the network performance. In the PEDAMACS protocol, sensor nodes forward their one-hop neighbor information to the access points, and the access point determines the schedule based on the complete network information. In PEDAMACS, every node has three power levels $P_l > P_m > P_s$ corresponding to three transmission ranges $R_{cl} > R_{cm} > R_{cs}$. The largest transmission range R_{cl} is used by the base station to broadcast coordination packets to all sensor nodes. The smallest transmission range R_{cs} is used by sensor nodes to forward their data packets to the base station through multiple paths. Nodes transmit at medium transmission range to discover their latest topology. The PEDAMACS protocol works in four phases.

1. Topology learning phase: In this phase the topology learning coordination packets are forwarded from base station to sensors for synchronization. Thus, the construction of distributed tree rooted at the base station is performed by a flooding of tree construction packets in the network.

2. Topology collection phase: In this phase every node transmits 'local topology' packets, which include the list of node's parent, neighbors and interferers, to its parent. This information is propagated to the base station.

3. Scheduling phase: In this phase the base station generates an interference free scheduling and the information is broadcast to all nodes in the network.

4. Adjustment phase: In this phase the updated topology information is propagated to the base station from all sensor nodes for rescheduling purpose.

This protocol incurs high control overhead in terms of control message transmission and delay involved in scheduling.

Coverage Criteria:

Sensing coverage is an important issue for the designing of road sensor network. Most of the works in the literature have discussed about either the full area coverage [190] or the barrier coverage [38]. Mao *et al.* [126] have designed an efficient algorithm for finding k-road coverage, where every point in the road is covered by at least k sensors. In [69], the authors have modeled the k-area coverage problem smart roadways considering

TH-1251_11610108

cell geometries based on three-dimensional characteristics of road surfaces. Using basic trigonometric and geometric reasoning, the number of nodes required in each cell is estimated. They also proposed algorithms for computing the deployment positions of sensor nodes on roads. However, in a road sensor network, the full-area coverage is not required for detecting every passing vehicle. Further, the barrier coverage is insufficient for traffic analysis and signaling purposes. Another notion of coverage for road sensors has been introduced in [120], called the *sweep coverage*, where some specific points of the road are monitored by the sensors. Heterogeneous environment is considered where few sensors, that are equipped with actuators, are used for monitoring the target points at some user defined interval. Though sweep coverage is sometimes sufficient for several road-monitoring purposes, it is not acceptable for general purpose vehicle detection at a continuous rate. Further the connectivity among the sensors need to be assured for uninterrupted data collection. Based on the connectivity and the coverage requirements, the sensor scheduling need to be designed for the proper functioning of road sensor network.

2.4.2 Critical Infrastructure Information System

Critical Infrastructure (CI) like power generation and distribution centers, water supply plants, hospital, bank, bridge, international border area and transportation system, etc. are the indispensable assets for the socio-economical foundation of any country. The increasing threats of catastrophe or invasion on the security of the CI demand a robust and stable monitoring system to be deployed. Wireless sensor network has emerged to contribute to the imperative protection of the large-scale CI by providing flexible, cost-effective and automated solutions to achieve smooth services [50, 150, 171]. Most of the existing works in the literature, that have addressed the design challenges of CI monitoring system [96, 116, 124], have focused mainly on the hardware aspects of sensor nodes. Some of the works addressed the security issues related to the information vulnerabilities involved with the protection of CI [13, 150]. In [85], the authors have developed a framework for monitoring the critical infrastructures of pipelines carrying oil, gas, water, etc. They addresses the problems related to communication and routing of data by modeling a hierarchical routing technique considering heterogeneous role of deployed sensors. Additionally, their scheme supports the node failures by increasing the sensor's transmission power, which is a costly affair. Also, the proposed deployment architecture is not suitable for protection of the critical infrastructure other than the pipelines. The state-of-the-art works for this area are briefly discussed in following subsections, which motivates to use multiple paths for data gathering exploiting the redundancy. The principal objective

TH-1251_11610108

in this case is to ensure reliability in delivery of sensory data to the base station even if a single forwarding path is affected due to the failure of one or a set of nodes.

Traditional Approach to Achieve Reliability:

The characteristics and design goals of efficient data gathering protocols for sensor network are well-studied in the literature, such as [28, 49, 74, 79, 90, 164, 165, 180, 186, 197], and the references therein. These protocols mainly aim at finding the forwarding paths with minimum number of sensors involved, along with the possibility of in-network data aggregation. This reduces the average energy dissipation in the network, which results in an improved network lifetime. These protocols assure neither the reliability in data gathering, nor the sensing coverage and the network connectivity criteria during the node failures. A set of notable works in the literature [11,29,91,122,145,173,196] have focused on the problem of reliable data delivery through the classical single path forwarding technique. There are three basic approaches to provide the reliability over the traditional single path forwarding. The first approach is to derive the statistical correlation among the sensory data to aggregate them, so that the network load is reduced. A reduction in network load can also reduce the data losses due to the network contention and congestion. The second approach is to forward data through the most reliable links. The third approach is to use the packet retransmission policy to improve the reliability. All these schemes are not efficient as the maximum reliability bound depends on the path loss rate. So, the reliability can only be predicted probabilistically, but can not be guaranteed. The probabilistic reliability model is not suitable for the applications on CI monitoring, where a strict bound on the reliable delivery of the sensory data needs to be ensured. Further, the timeliness in data delivery is important for such applications, and therefore, the retransmission based approaches are not suitable too. A robust gradient based data delivery protocol, as proposed in [192] uses the concept of traditional single-sink data forwarding mechanism. The protocol builds and maintains a cost field for providing each sensor the direction to which the sensing data needs to be forwarded by propagating advertisement packets in the network. Cost at a node is the minimum energy overhead to forward a packet from the node to the sink along a path. The cost field gives the global direction towards the sink implicitly. When a node forwards a packet, it does not designate the nodes in the next hop, rather, it simply includes its own cost in the packet. Only the neighbors with smaller costs can continue the packet forwarding. In this way, messages roll towards the center of the stimulus for gradient formation, where the cost is minimum. All packets generated in future follow this gradient. The authors have shown

TH-1251_11610108

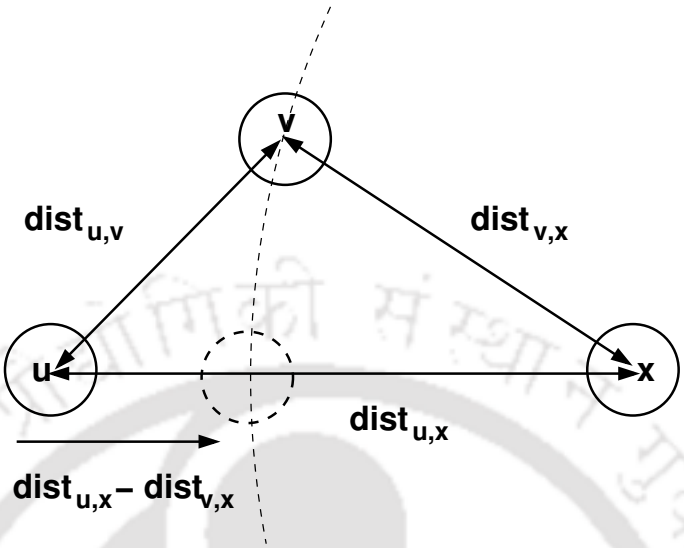

Figure 2.3: An example of progress speed estimation in MMSPEED protocol

that the protocol can deliver data with high degree of reliability in spite of multiple node failures that are not within close proximity. Due to the localization of node failures based on energy exhaustion the affected path is unable to provide the required reliability to the delivery of sensory data to the base station.

Redundancy based Reliability:

In [52,54,83,110], the authors have explored the advantages of redundancy, both in terms of the deployed redundant sensors and the redundant data delivery to satisfy the reliability and the sensing coverage. In [66], the authors have proposed to use multi-path data delivery in sensor network to improve the reliability in data delivery. Another multi-path data delivery protocol is proposed in [117], where the authors have designed a hybrid distributed data collection protocol that aims at finding multiple node disjoint paths from a sensor to the sink. In [55], the authors have proposed a protocol to find out multiple edge-disjoint paths between a source-sink pair using the concept of dynamic packet state in the context of sensor network, to control the number of paths required for the desired reliability. Even with the node-disjointness and the edge-disjointness, the paths tend to converge towards a common sink, resulting in a high probability of data loss when a node near the sink fails. The scheme MMSPEED, proposed in [62], builds a set of paths towards a single sink based on the computation of a path metric that minimizes the end-to-end data delivery time. Let a node x is the final destination as shown in Figure 2.3. Each node

TH-1251_11610108

u maintains delay estimation, expressed as $delay_{u,v}$ to each neighbor v, and calculates a metric, called $speed^x_{u,v}$ as follows:

$$speed^x_{u,v} = \frac{dist_{u,x} - dist_{v,x}}{delay_{u,v}} \tag{2.1}$$

According to this scheme, node u forwards a packet to a neighbor v whose progress speed $speed^x_{u,v}$ is higher than the 'set speed' (a predefined limit). The protocol is localized in nature, and provides good reliability assurance while ensuring the timeliness for the data delivery. However, path isolation is not guaranteed. Multicast tree based approaches [70, 153] are widely used for data forwarding from single source to multiple sinks. Multicast routing protocols build independent multicast trees from every individual source to multiple sinks, and thus, are ill-suited for data aggregation. Data aggregation is important where all the correlated data towards a common sink are aggregated at the intermediate nodes to reduce the network load. However, data towards different sinks should not be aggregated. Further, it is hard to find the node disjoint paths from all the sources to all the sinks if multicast tree based approaches are used [33].

Multi-sink based Data Gathering:

Very few works exist in the literature that explore the multi-sink data delivery in sensor network. Kim *et al.* [95] have designed a deployment scheme for multi-sink placement in sensor network. Based on the sink placements, they have designed a data dissemination protocol to collect the sensory data at multiple sinks. The authors in [114] have proposed a multi-source multi-sink anycast routing framework for sensor network. They have proposed a distributed and scalable potential field estimation algorithm, based on which a probabilistic forwarding scheme is designed to ensure low overhead and high resilience to the network dynamics. In [133], the authors have proposed a multi-sink routing protocol for wireless sensor network. The objective of the paper is to merge the paths from individual sources towards different sinks, so that the minimum number of sensors can be used for data forwarding. Individual collection trees rooted at two different sinks and the corresponding merged path from MUSTER protocol are shown in Figure 2.4 and Figure 2.5. The MUSTER protocol starts independently building trees by connecting sources to their sinks. The initial trees are mutated over time to optimize the routing topology. For every sink s and for every neighbor u, each node maintains a parameter, $Q(u,s) = R(u,s).T(u)$, where $R(u,s)$ denotes the routing quality and $T(u)$ is the estimate of expected lifetime of u. $R(u,s)$ is estimated based on three quantities, $reliability(u,s), paths(u)$ and $sinks(u)$. $reliability(u,s)$ is an indicator of how reliable is

TH-1251_11610108

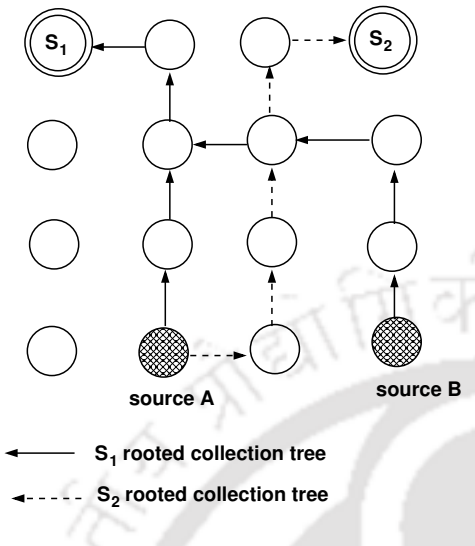

Figure 2.4: An example of MUSTER: multiple rooted collection trees

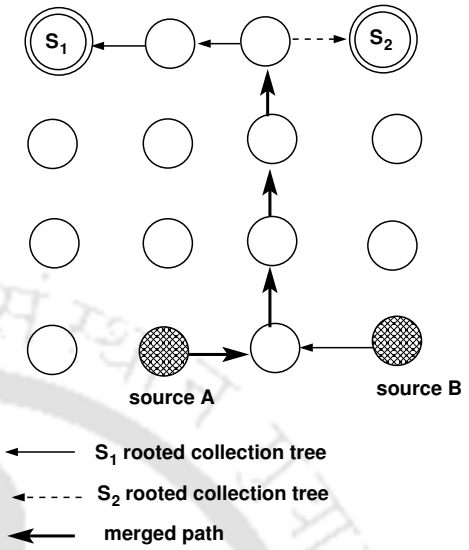

Figure 2.5: An example of MUSTER: path merging

the end-to-end communication from u to s. $paths(u)$ denotes the number of source-sink paths passing through u. $sinks(u)$ is the number of sinks u is currently sending data to. Therefore, $R(u, s)$ can be estimated as follows:

$$R(u, s) = \delta.reliability(u, s) + \alpha_1.paths(u) + \alpha_2.sinks(u) \tag{2.2}$$

The protocol works with the objective to choose the neighbor u for the sink s with maximum $Q(u, s)$. Though this approach saves the critical sensor energy by using a minimum number of sensors, it does not assure the reliability during node failures. Failure of a single node affects all the paths towards it.

As discussed, for the applications on CI monitoring, reliability in data delivery needs to be assured along with the network connectivity and the sensing coverage maintenance. While an uninterrupted application service is the main challenge, the timeliness in data delivery also need to be maintained for a proper detection and reporting of the inadmissible events. To the best of our knowledge, no prior works exist in the literature that consider all these properties simultaneously, while designing the data forwarding protocol for the sensor network applications on CI monitoring.

TH-1251_11610108

2.5 Summary

From the above discussions it can be observed that even though there exist many notable works in the literature related to the problems addressed in the thesis, they do not cover all the objectives. Various aspects of design challenges like connectivity, sensing coverage, energy efficiency and failure support along with the application specific performance metrics like, bounded delay and reliability, have not been reflected in any of the existing protocols in the literature for efficient data gathering in sensor network. Also, there is a strong requirement to bridge the gap between the pure theoretical concepts and the implementation based adaptation in feasibility point of view. Considering all these issues, the thesis work contributes a set of schemes for data gathering in WSN which aims to fulfill most of the requirements.

The first contribution proposes a fault-tolerant data gathering tree construction scheme where nodes perform neighborhood information preprocessing to support future node failure in cost-effective manner. The detail description of the scheme with theoretical analysis and simulation results has been provided in next chapter.

TH-1251_11610108

Chapter 3

Data Gathering Tree Management from Arbitrary Node Failure

Large number of sensor nodes are deployed uniformly in the target terrain to sense the desired phenomena and forward the sensory data towards a common base station or sink for end user access. As discussed in Chapter 1, sensor nodes are limited with power resource, and so, efficient power management through a reduction in control message transmission helps in improving the sensor life. QoS is another important performance metric that should be offered with an assurance for most of the applications. Most of the notable works in the literature [112, 113, 199] have focused on the analysis of data gathering performance in terms of energy and delay as discussed in Chapter 2. The effect of underlying topologies on the performance of data gathering remained unaddressed. Topology of sensor network, playing an effective role, can be modeled like a spanning tree rooted at the sink as routing through a spanning tree offers the ordered delivery of data with a minimum redundancy. Data collection using a BFS tree is the most efficient as it maximizes the collection capacity, and minimizes the total number of packet relays and the delay [40, 107]. So, using a BFS tree would be a good option for data gathering as the hop-count distance between the sink and each sensor node in the network is minimum for a BFS tree [53]. A sudden crash of an arbitrary sensor node may lead to the problem of incorrect tree structure. Usually, per node load increases along the walk from the leaves to the root of the tree. So, nodes near the sink are more prone to failure due to fast power exhaustion. The crash of all the nodes in a particular level would make the tree disconnected. This interrupts the data gathering process and the data loss degrades the desired QoS. Repairing the tree online or building a new tree from the scratch would not

TH-1251_11610108

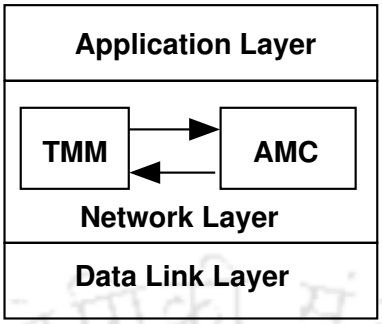

Figure 3.1: Network Protocol Stack

be efficient due to an increase in overhead, both in terms of delay and control message transmission. The increased rate of consumed energy per node would reduce the overall network lifetime in effect.

In this chapter, a novel crash-tolerant tree based data gathering scheme has been proposed that works as a module within the network layer at every node. The proposed module consists of two sub-modules, a Tree Management Module (TMM) for performing tree management activities on a node failure, and an Application Message Controller (AMC) to handle the application messages as shown in Figure 3.1[1]. Both the TMM and the AMC interact with each other through an exchange of signals, and act cooperatively below the application layer to assure the application transparency. The activities of the proposed TMM and AMC have been described by designing a set of distributed message-passing algorithms. The execution of the proposed TMM at each node constructs a spanning tree rooted at the sink. The tree eventually turns into a BFS tree maintaining the necessary properties. Although there exists a handful of papers to construct a distributed BFS tree in the literature [25, 88], they are required to be customized according to various application needs. Therefore, it is quite justified to build up a new system that supports all the application requirements with reduced implementation cost, rather than to perform an incremental improvement over the existing ones. The proposed distributed BFS tree construction is unique in a way that every node performs some local processing on neighborhood information during the tree construction period itself. Considering the possibility of parent node failure, every node precomputes its alternate parent such that the cost of local tree repairing on failure of the parent node is low, in terms of both the control message and the repairing time. Further, the algorithms for BFS tree construction converge through incremental level updating that avoids a large overhead

[1]It can be noted that all the schemes proposed in this thesis are based on this modular architecture.

TH-1251_11610108

of control message flooding. The TMM is also responsible for tree maintenance from an arbitrary node failure. The proposed maintenance scheme can handle a single node failure as well as the simultaneous multiple node failures. The AMC, that works along with the TMM, ensures the reliability of data gathering through efficient buffer management.

The rest of the chapter is organized as follows: Section 3.1 is a description of the system model assumptions followed by a description of the proposed BFS tree construction scheme in Section 3.2. First, the initialization of TMM through the design of a set of algorithms for tree construction is given. Then, the process of alternate parent precomputation is provided, followed by the proof of correctness through lemmas and theorems. Maintenance of the tree is performed in two phases, as provided in Section 3.3. First, the proactive way of repairing has been discussed, followed by the extended reactive way of repairing. In addition, the implementation of the AMC is provided in Section 3.4. Section 3.5 presents the simulation results for three different scenarios. First, for two extreme cases, one for a ring topology and another for a chain topology. Second, for a grid topology, and then finally, for a general arbitrary topology, with multiple node failures, both in parallel and in a series. Section 3.6 summarizes the contribution of this chapter.

3.1 System Model

Let there be $\mathcal{N}$ number of sensor nodes scattered uniformly in the environment to be monitored. The arbitrary distribution of sensors can be represented by the communication graph $\mathbb{G}(V, E)$, where V denotes the set of nodes, and E be the set of edges between any two nodes. Let Euclidean distance between any two nodes u and v be denoted by $d_{u,v}$. Two nodes u and v are said to be connected, and are called neighbors, if $d_{u,v} \leq R_c$, where R_c is the communication range of every sensor node. Each sensor node is assumed to be identical with respect to the processing and sensing capacity, transmitter power, or the memory limitation. There exists a special sink node in which all the sensory data are accumulated. A data gathering tree, out of all nodes in the network rooted at the sink, is constructed eventually. Each node in the network is assumed to keep a $Neighbor$ set storing the node identifiers (IDs) of all directly connected nodes. The mode of communication is assumed to be asynchronous. So, per hop message delay is finite but unbounded. Each sensor is assumed to participate in data sensing and forwarding until it fails due to power outage. The network is assumed to be static, but, any arbitrary node may crash suddenly due to technical fault, or leave the system because of power exhaustion. Moreover, a multiple

TH-1251_11610108

Table 3.1: State Variables at each node u

Variable Name	Type	Initial Value	Meaning
visited	Boolean	False	True if u selects its parent after receiving the first *Token*
block	Boolean	False	If True, stops forwarding data from Buffer
halt	Boolean	False	Controls the flow of application messages in Buffer
power	Boolean	False	True if $Neighbor(u) \setminus \{Child(u) \cup Sibling(u) \cup parent(u)\} \neq \phi$
rflag	Boolean	False	True if u has sent a *ReqParent* message to $altp(u)$
urgent	Boolean	False	True if both $parent(u)$ and $altp(u)$ have been crashed
$parent(u)$	Integer		Parent of u
$altp(u)$	Integer	Null	Alternate parent of u, ether powerParent or boostParent
$level(u)$	Integer		Level of u at the BFS tree
$bpLevel(u)$	Integer	∞	Temporary level required to find the node with minimum level for the selection of $boostParent$
$powerParent(u)$	Integer	Null	$altp(u)$ such that $power = True$ at u
$boostParent(u)$	Integer	Null	$altp(u)$ such that $power = False$ at u
$Child(u)$	Set		Children of u
$Neighbor(u)$	Set		Neighbor of u
$Sibling(u)$	Set	Null	IDs of those $w \in Neighbor(u) : parent(w) = parent(u)$
$altParentLevel(u)_w$	Map	∞	$level$ of $w, \forall w \in Neighbor(u)$
$parentSet(u)_w$	Map	Null	$parent(w), \forall w \in Neighbor(u)$
$serviceList(u)$	Integer List	Null	IDs of nodes $w \in Neighbor(u)$ from which u has received ReqParent message
$blockedList(u)$	Integer List	Null	IDs of those $w \in Neighbor(u) : parent(w) = parent(parent(u))$
$urgList(u)$	Integer List	Null	$Neighbor(u)$ sent with Urgent message

nodes may crash in parallel. Table 3.1 shows the system variables used in the algorithms.

3.2 Topology Management Module

The proposed TMM is designed as a set of distributed message-passing algorithms that interact with each other to construct a BFS tree. This section provides a detailed description of the tree construction and the alternate parent computation activities of the TMM with necessary examples and proof of correctness. Repairing activities of the TMM have been described in the following section. The term, broadcast of a message, whenever used in this chapter, indicates that the scope of broadcast is confined within 1-hop neighborhood only.

TH-1251_11610108

Algorithm 3.1 On receiving *Token(mLevel, pid)* from v by u

```
1.  parentSet[v] ← pid
2.  altParentLevel[v] ← mLevel
3.  if visited = False ∧ timer = False then
4.      visited ← True                                   /*Received Token for the first time*/
5.      parent(u) ← v
6.      level ← mLevel + 1
7.      Send Add(False) to parent(u)
8.      for all w ∈ Neighbor(u) ∧ w ≠ parent(u) do
9.          Send Token(level, parent(u)) to w
10.     end for
11. else
12.     if mLevel + 1 < level then
13.         halt ← True                                  /*Parent switching holds to clear data from Buffer*/
14.         switchParent(v)
15.         for all w ∈ Neighbor(u) ∧ w ≠ parent(u) do
16.             Send Token(level, parent(u)) to w
17.         end for
18.     end if
19. end if
20. if parent(u) ≠ Null then
21.     Reset UpdateTimer                                /*altp recomputation is required*/
22. end if
```

3.2.1 TMM Initialization : A Distributed Tree Construction

Initial tree construction and incremental level update are accomplished by an exchange of three types of control messages, *Token, Add*, and *Remove*, as described in Algorithm 3.1 to Algorithm 3.4. The sink node initiates tree construction by forwarding a *Token(0, Null)* to all its neighbors.

Every time a node u receives a *Token* message from a node v, it stores $parent(v)$ and *level* of v, as stated in lines 1-2 of Algorithm 3.1, required during the precomputation phase. Thus, for any node u, *parentSet* and *altParentLevel* holds the neighbors' parent information and the neighbors' level information, respectively.

Upon receiving a *Token(mLevel, pid)* for the first time from a node v, each node u performs lines 3-10 of Algorithm 3.1. It first updates the variables $parent(u)$, *visited* and *level*. Then, u sends an *Add(False)* message to its parent for confirmation. A False argument with the *Add* message forces the parent node to add u in its *Child* set immediately. u also sends a *Token* message with its updated information to all its neighbors.

Once u has set its parent, it may further receive the *Token* messages from its neighbors. It may happen that u receives a *Token* message from a node v that can

TH-1251_11610108

Algorithm 3.2 Procedure *switchParent(w)* triggered by u

```
1. while halt=True do
2.     Wait        /*Application Message Controller(AMC) will set halt to False eventually breaking the loop*/
3. end while
4. block ← True                                   /*Suspends application data flow in Buffer*/
5. Send Remove to parent(u)
6. parent(u) ← w
7. level ← mLevel + 1
8. Send Add(False) to parent(u)
9. block ← False                                  /*Resumes application data flow in Buffer*/
```

Algorithm 3.3 On receiving *Add(flag)* from v by u

```
1. if flag = False then
2.     if v ∉ Child(u) then
3.         Child(u) ← Child(u) ∪ {v}
4.     end if
5. else
6.     if block = True ∨ parent(u) = Null then
7.         Send Nak(True) to v
8.     else
9.         Send Nak(False) to v
10.        Child(u) ← Child(u) ∪ {v}
11.    end if
12. end if
```

Algorithm 3.4 On receiving *Remove* from v by u

```
1. halt ← True                                    /*Parent switching holds to clear data from Buffer*/
2. while halt = True do
3.     Wait                                       /*AMC will set halt to False eventually breaking the loop*/
4. end while
5. Child(u) ← Child(u) \ {v}
```

offer lower level than $parent(u)$, due to unbounded channel delay. On receiving a better *Token* (A *Token* message with such improved level information), u calls the *switchParent* procedure to change its parent for level improvement, and further broadcasts a *Token* with the improved information, according to lines 12-18 of Algorithm 3.1. This repeated event of parent changing and *Token* broadcasting may lead to the flooding of *Token* message in the network incurring a large overhead. This can be avoided by introducing the lazy improvement technique with minor modification at line 14 of Algorithm 3.1.

The lazy improvement technique does not allow an immediate level improvement of a node on receiving a better *Token* message, after it has set its parent. Rather, the parent changing event is delayed, and the number of repeated *Token* broadcasting is minimized based on a timer, called $LazyTimer$. According to this technique, on receiving

TH-1251_11610108

a better *Token* message, the *LazyTimer* is set. Once the timer expires, the *switchParent* procedure is triggered followed by an 1-hop *Token* broadcast. Within the timer period, if any better *Token* message is received, the information is stored, and considered only at the time of final parent changing event on timer expiry. The timer value plays a crucial role on effective outcome of this technique. If the timer value is very less, the number of 1-hop *Token* broadcast and parent changing event will be increased. If the timer value is very high, the tree might take a long time to reach the stability. Thus, there is a trade-off between the control message overhead and the convergence time. As the system is assumed to be asynchronous, there exists no bound on the number of times a node will receive the better *Token* message. However, it can be shown that a node will eventually receive the best *Token* message with the minimum level information, and set the parent correctly. The effect of the *LazyTimer* value on the performance of the proposed scheme has been shown in Section 3.5.

The *UpdateTimer* is another timer that updates the neighborhood information of a node periodically. These neighborhood information are required to precompute the alternate path for an emergency repairing due to the parent node crash. On receiving a *Token* each time, the *UpdateTimer* at a node u is reset, according to lines 20-22 of Algorithm 3.1. On expiry of *UpdateTimer*, the $updateInfo$ procedure is triggered to recompute the alternate parent. The value of the *UpdateTimer* depends on how frequent the neighborhood information is updated in a node and the necessity of alternate parent recomputation. Each time the *LazyTimer* is expired, a node changes its parent, and broadcasts fresh *Token* in 1-hop neighborhood. The alternate parent is required to be recomputed, and thus, the *UpdateTimer* is also required to be expired. Hence, the timer value for the *UpdateTimer* is set equal to the *LazyTimer* during the tree construction period. The detail of the $updateInfo$ procedure is discussed in later section.

On a timeout of the *LazyTimer*, the *switchParent* procedure of Algorithm 3.2 is triggered at u to change its parent to a node with a better level. A variable *halt* is set to True that clears the buffered data at u to its old parent to avoid the delivery of redundant application data. The AMC controls the data flow, and resets the *halt* variable eventually. Another Boolean variable *block* is set to True until the changing parent event is over to avoid application data loss. The detailed working of these variables have been discussed in later section. To change the parent, u sends a *Remove* message to its old parent and an *Add* message to the new one, as stated in lines 5-8 of Algorithm 3.2. In this case, the *Add* message carries a False argument to enforce an immediate child addition.

On receiving an *Add* message from a node v, node u adds v immediately in its *Child*

TH-1251_11610108

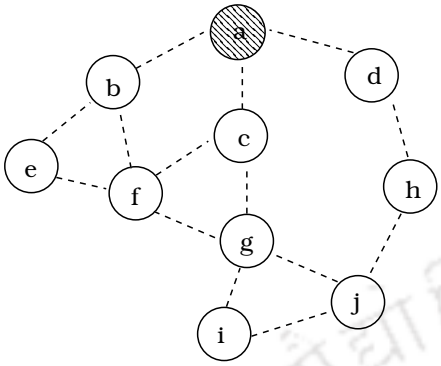

Figure 3.2: Initial graph $\mathbb{G}$

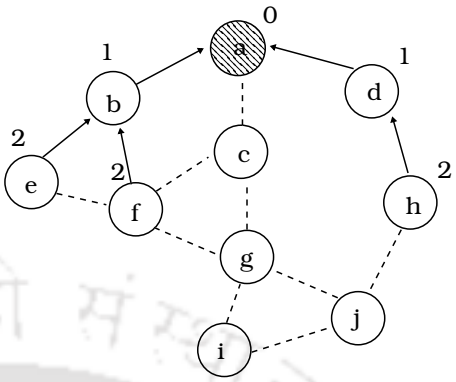

Figure 3.3: *a*, *b*, *d*, *e*, *f*, *h* assign correct level

set if it is not there already, and if the flag is set to False in the message, according to Algorithm 3.3. If the message flag is True, u adds v in its *Child* set only if u has a valid parent, and sends a *Nak(False)* message to v. Otherwise, u sends a *Nak(True)* to v that acts as a negative acknowledgment to node v. This acknowledgment based *Add* is introduced to avoid the forced addition of a child to a node whose parent is invalid due to crash of either the parent or some ancestor node.

On the other hand, on receiving a *Remove* message from v, node u removes it from its *Child* set, as in Algorithm 3.4. The removal of a node from the *Child* set is controlled by the *halt* variable of the AMC to guarantee a proper data delivery.

An Example of BFS Tree Construction :

The initial tree construction and the incremental level updating procedure have been described pictorially in Figure 3.2 to Figure 3.5, where each circle represents a sensor node, shaded circle being the sink. Each arrow represents the tree edge where as each dashed line between any two nodes represents the presence of a communication link. Initially, the sink node a starts tree construction by forwarding a *Token(0, Null)* to all its neighbors as in Figure 3.2. Figure 3.3 depicts that the nodes receive the *Token* message, and assign their levels correctly. In Figure 3.4, node c calls the procedure $TimeOut$ after receiving a *Token* from the root node, and thus, improves its level. Then, the updated level is advertised through *Token* messages forwarded to all neighbors except the parent. Node g and node j update their respective levels on timeout. In this way a BFS tree is constructed as shown in Figure 3.5, where each node eventually selects the correct parent having the shortest hop-count to the root node.

TH-1251_11610108

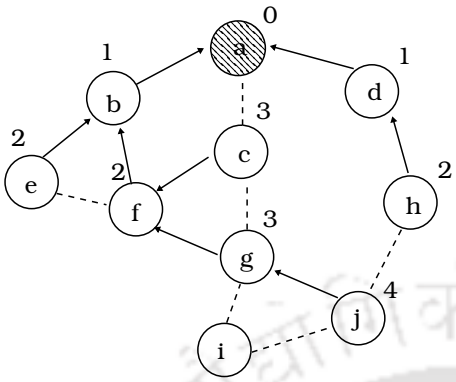

Figure 3.4: c and g select wrong parent; j selects right parent with incorrect level

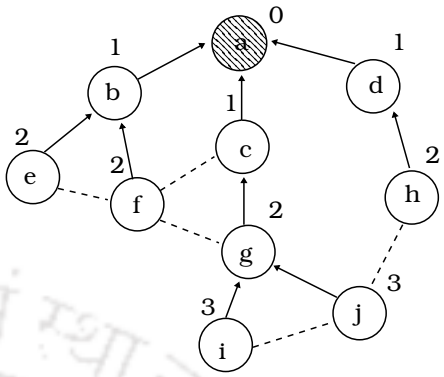

Figure 3.5: Final tree: all nodes select correct parent

3.2.2 Alternate Parent Precomputation

Once a node u has reached the stable state by setting its parent and child in the tree, on expiry of the $UpdateTimer$, it calls the $updateInfo$ procedure to compute the alternate parent. This precomputation of alternate path to the root through the alternate parent helps in fixing the tree through local adjustment only, after a parent node crashes in future. On further changes in underlying tree structure, each node recomputes the alternate parent based on an updated neighborhood information by calling the $updateInfo$ procedure. First, each node u calculates its $Sibling$ set that includes all the nodes in the neighborhood whose parents are the same with that of u, as stated in Algorithm 3.5. Node u is called a $Power$ node if there exists at least one node $w \in Neighbor(u)$, such that the edge between u and w in the graph $\mathbb{G}$ is a *cross edge*, and neither $w \in Sibling(u)$ nor $parent(w) \in Sibling(u)$. Then, w is called the $powerParent$ of u. If more than one such w exist, then, one with the minimum level becomes the $powerParent$, according to lines 5-11 of Algorithm 3.5. A $Power$ node u sets the $power$ variable to True, and forwards a $Booster$ message to all the neighbors. All the remaining nodes in $\mathbb{G}$, whose $power = False$ and has received such $Booster$ messages, are called $Booster$ nodes. If node u receives a $Booster$ message from a node v, then v is set as the $boostParent$ for the node u. If more than one such $Booster$ messages are received, then, the node offering the minimum level becomes the $boostParent(u)$, according to lines 6-12 of Algorithm 3.6. If $power$ is set to True, u sets $altp(u)$ to $powerParent$, otherwise to $boostParent$.

Let node u have selected a node v as its alternate parent, such that v is either the $powerParent$ or the $boostParent$ of u. If $parent(u)$ lies within the forwarding path of the node v to the root, selecting the node v as a parent on failure of $parent(u)$, would not be

TH-1251_11610108

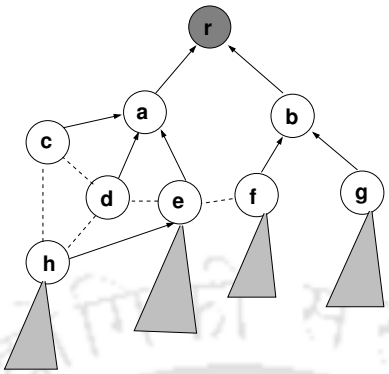

Figure 3.6: Neighborhood information pre-processing for alternate parent computation

correct. To ensure that the correctness of the alternate parent computation, each node locally computes a *blockedList*, according to lines 2-4 of Algorithm 3.5. For any node u, the *blockedList* includes all the nodes in $Neighbor(u)$ whose parents and $parent(parent(u))$ are the same. Every *Power* node forwards the computed *blockedList* along with the *Booster* message. The *blockedList* information, received with a *Booster* message, helps the node u in rejecting all such nodes as a potential candidate for being the *powerParent*, whose forwarding paths include the parent of the node u.

Let node u declare itself as a *Power* node on satisfying the conditions of line 5 and line 6 of Algorithm 3.5, and w be its *powerParent*. Now, w also independently computes the alternate path, and sets the node u as its *powerParent*. If $u \in blockedList$ computed by w, then, on receiving *Booster* message from the node w, u rejects w as its *powerParent*. Node u resets its *power* variable, according to lines 1-5 of Algorithm 3.6. As u is not a *Power* node anymore, the *powerParent* of u does not remain valid too. At this point, node u becomes a *Booster* node. However, if no valid *boostParent* is precomputed, node u can not compute any correct alternate parent. Let u set *power* to True at the time t_i, and reset *power* at the time t_j, where $t_j > t_i$. Let after the time t_j, u do not receive any *Booster* message from any of its neighbors. As both the *boostParent* and *powerParent* are invalid at the time $t_k > t_j$, u will not have any valid alternate parent precomputed. Thus, to handle the situation when a *Power* node u becomes a *Booster* node on receiving a *Booster* message from its *powerParent* v, such that $u \in blockedList(v)$, every node calculates the *boostParent* irrespective of its *power* status on receiving a *Booster* message. So, *altp* at every node is set to either the *powerParent* or the *boostParent*, correctly.

TH-1251_11610108

Algorithm 3.5 Procedure updateInfo triggered by u

1. $Sibling(u) \leftarrow Sibling(u) \cup \{v\},\ \forall v \in Neighbor(u) : parentSet(u)_v = parent(u)$
2. **for all** $w \in Neighbor(u)$ **do**
3. $\quad blockedList \leftarrow blockedList \cup \{w\} : parentSet(u)_{p(u)} = parentSet(u)_w$
4. **end for**
5. **if** $\exists w \in \{Neighbor(u) \setminus \{Sibling(u) \cup Child(u) \cup \{p(u)\}\}\}$ **then**
6. $\quad$ **if** $parentSet(u)_w \notin Sibling(u)$ **then**
7. $\quad\quad power \leftarrow True$
8. $\quad\quad powerParent \leftarrow x : altParentLevel(u)_x = min\{altParentLevel(u)_w\ \forall w \in parentSet(u)\}$
9. $\quad\quad altp(u) \leftarrow powerParent$
10. $\quad\quad$ Send Booster$((altParentLevel(u)_{altp} + 1), blockedList)$ to $Neighbor(u)$
11. $\quad$ **end if**
12. $\quad$ **if** $altParentLevel(u)_{altp} < altParentLevel(u)_{p(u)}$ **then**
13. $\quad\quad block \leftarrow True$ /*lines 12-15 are executed for level improvement during tree repairing*/
14. $\quad\quad$ Send Add(True) to $altp(u)$
15. $\quad$ **end if**
16. **end if**

Algorithm 3.6 On receiving Booster(mLevel, blist) from v by u

1. **if** $power = True \wedge u \in blist \wedge v = altp(u)$ **then**
2. $\quad power \leftarrow False$
3. $\quad altp(u) \leftarrow Null$
4. $\quad updateInfo$
5. **end if**
6. **if** $v \neq parent(u) \wedge mLvel < bpLevel \wedge u \notin blist$ **then**
7. $\quad boostParent \leftarrow v$
8. $\quad bpLevel \leftarrow mLevel$
9. $\quad$ **if** $power = False$ **then**
10. $\quad\quad altp(u) \leftarrow boostParent$
11. $\quad\quad$ Send Booster$(bpLevel + 1)$ to $Neighbor(u)$
12. $\quad$ **end if**
13. **end if**

An Example of Alternate Parent Computation :

In Figure 3.6, nodes c, d, and e are siblings. Node e is a *Power* node and node f is its $powerParent$ as there is a cross edge between e and f. Similarly, node d may select a node h as its $powerParent$ as $h \notin \{Sibling(d) \cup \{p(d)\} \cup Child(d)\}$. However, node h will calculate its $blockedList$, according to line 3 of Algorithm 3.5. As node $d \in blockedList(h)$, on receiving *Booster* message from the node h, node d will update its *power* status correctly as stated in Algorithm 3.6. Similarly, node c is a *Booster* node, and node d is its $boostParent$.

TH-1251_11610108

3.2.3 Proof of Correctness

The following Theorems and Lemmas are provided as a justification for the successful completion of the tree construction, the correctness of all the tree properties and an estimation of the message complexity.

Lemma 3.1. *Each node in $\mathbb{G}$ receives a Token message at least once.*

Proof. Let $x \in V$ be a node such that it have never received a *Token* message. Root node initiates the tree construction by sending a *Token* message to all its neighbors. On receiving the *Token* for the very first time, node u sets its parent, and forwards the *Token* to all nodes $v \in \{Neighbor(u) \setminus parent(u)\}$ as shown in lines 8-9 of Algorithm 3.1. Let for all nodes $w \in Neighbor(x)$, $x = parent(w)$, such that x has not received a *Token* from any of these nodes. However, a node can forward a *Token* message only upon receiving a *Token* message from its parent. This contradicts the assumption that there does not exist any node in $Neighbor(x)$ from which it has not received a *Token* message. As $\mathbb{G}$ is connected, there must exist at least one node $z \in Neighbor(x)$, such that there exists a path from root to the node z and $z = parent(x)$. Thus, every node in $\mathbb{G}$ eventually receives the *Token* message at least once. □

Lemma 3.2. *Every node u eventually receives the Token message from the right parent with the shortest path.*

Proof. Let u be a node such that there exists a node $w \in Neighbor(u)$, where w be the correct parent of u and it can offer the shortest path to the root. Let u receive the *Token* message from a node $x \in Neighbor(u)$, where $x \neq w$, and set its level at the time t_i. Due to unbounded channel delay, u might receive the *Token* message from w at the time t_j, where $t_j > t_i$, and improve its level. Now, in the worst case, u may receive several *Token* messages from all its neighbors except w, between the time t_i and t_j. The number of times a node calls the *switchParent* procedure and improves its level depends on the value of the $LazyTimer$. Every node in neighborhood of the root eventually sets the correct level on receiving the *Token* message from the root (as per Lemma 3.1). Therefore, as $\mathbb{G}$ is connected, every node u eventually receives the *Token* message from the right parent, and correctly sets its level. □

Theorem 3.1. *The algorithm for tree construction eventually terminates.*

Proof. A node is said to be reached to the stable state when it has set its parent correctly such that the shortest path distance from the root is being maintained. When all the

TH-1251_11610108

nodes in V reach the stable state, and do not exchange any more control messages with its neighbors, the tree construction algorithm terminates. Let a node $u \in V$ at level l, never reach the stable state. So, u always keeps itself involved in level improvement. From Lemma 3.2, there exists a bound on the number of times u calls the *switchParent* procedure for level improvement on receiving a *Token* message from node $w \in Neighbor(u)$. This contradicts the assumption that u never reaches the stable state. Once u reaches the stable state, it does not forward a *Token* message. Once all the nodes in V reach the stable state, no node further receives any of the *Token, Add*, or *Remove* message. Thus the tree construction algorithm eventually terminates. □

Lemma 3.3. *Let there be a path in the tree* $\mathbb{P} = \{x_1, x_2, ..., x_n\}$, *such that* $x_1 = parent(x_2), x_2 = parent(x_3), ..., x_{n-1} = parent(x_n)$. *Hence* $\nexists e \in E_{\mathbb{T}}$, *where* $E_{\mathbb{T}}$ *is the set of tree edges,* $E_{\mathbb{T}^c}$ *is the set of non-tree edges.* $E_{\mathbb{T}} \cup E_{\mathbb{T}^c} = E$, *such that* $e = \{x_n, x_1\}$ *and* $x_n = parent(x_1)$.

Proof. Let x_2 have selected x_1 as its parent on receiving a *Token* message from x_1 at the time t_j. That means x_1 has already set its parent to node w at the time $t_i < t_j$. Let at the time $t_k > t_j$, x_1 change its parent from node w to node x_n due to level improvement, according to Algorithm 3.3. This implies that the *level* of x_n must be less than the *level* of w. According to the assumption, x_n is a descendant of x_1, and thus, the *level* of x_n is greater than that of x_1. This contradicts the assumption that x_1 will set $parent(x_1) = x_n$ at some time t_k. So, x_1 can never choose x_n as its parent, and hence, no cycle is created during the parent selection. □

Lemma 3.4. *Let* $h(u)$ *denote the level of the node* u *in the tree. So,* $\forall u \in V$, $h(u) = min\{\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_n\}$, *where* $\{\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_n\}$ *is the set of all possible paths from the root to the node* u.

Proof. The proof follows from Lemma 3.2. □

Theorem 3.2. *The set of algorithms produces a correct BFS tree rooted at the sink node for data gathering.*

Proof. A BFS tree is said to be correct if the following conditions are true.

1. For every node in the tree, there exists a parent node that offers the shortest path to the root.

TH-1251_11610108

2. There exists no cycle in the tree.

As the sink node initiates the tree construction process, it eventually becomes the root of the tree. From Lemma 3.1, each node u in $\mathbb{G}$ selects its parent. From Lemma 3.1, each node receives the *Token* message at least once, and thus, all the edges incident on u become either a tree edge or a cross edge. Also, from Lemma 3.3, no cycle is formed due to an incorrect parent selection during the tree construction, and the distance from the root to any node eventually becomes the minimum as per Lemma 3.2, and Lemma 3.4. So, all the nodes in $\mathbb{G}$ correctly select their parent, such that distance to the root is the minimum. Thus, a BFS tree rooted at the sink node is eventually constructed. □

Theorem 3.3. *The worst case message complexity for the tree construction is* $O(|E|)$.

Proof. The total number of message transmissions depends on the trade-off between τ and i. τ is the $LazyTimer$ timeout interval. i is the number of times a node forwards a *Token* message due to improvement of its *level*, after it has selected its parent for the first time. i is a function of channel delay, which is unbounded. If τ is increased, i is automatically decreased. Now, for any node, if δ is the degree, then, for first time parent selection, it sends a $\delta - 1$ number of *Token* messages and one *Add* message. For each call of the procedure *switchParent*, a node sends a $\delta - 1$ number of *Token* messages, one *Add* message, and one *Remove* message. Not every node in $\mathbb{G}$ will call the *switchParent* procedure. On an average, if each node calls the *switchParent* once, then total number of message transmission for the tree construction is $2\delta + 1$ per node. Considering the precomputation of alternate parent into account, if each node calls the procedure $updateInfo$ once, then, additional message sent for precomputation is $\delta - 1$ per node. So, for all the nodes in the network, the total number of message transmissions, considering only one time level update, would be equal to $\sum_{v \in V}\{3\delta\} = 6|E|$, which is essentially $O(|E|)$. □

3.3 Tree Repairing from Arbitrary Node Crash

To start with the discussion of the tree repairing scheme from arbitrary node crash, it is assumed safely that the root node never fails. For the rest of the sensor nodes in the network, a node can fail any time due to disaster, technical error, power exhaustion, or any other critical reason. The failure of a node can be detected by all its neighbors through lower layer fault-detection techniques as proposed in [142, 160, 170, 184]. The failure of a parent node in the tree might lead to a degradation of the application performance due to a message loss or a redundant delivery in the data forwarding tree. One possible way

TH-1251_11610108

of repairing is the reconstruction of the data forwarding tree from the scratch. However, this indulges large amount of overhead in terms of the control message exchange. The better option is to repair the tree by fixing the parents of the victim nodes through a local adjustment. Considering only permanent failure of nodes, two cases are possible depending on the readiness of a node for leaving the network due to failure. First, due to a shortage of energy, a node may wish to leave the network gracefully after informing all its neighbors. Second, a node may crash suddenly without informing its neighbors. The tree repairing strategy is same for both the cases except the initial step. For the first case, the failed node itself initiates the repairing process by sending the necessary information to its neighbors. Where as for the second case, the neighbors of the failed node detect the crash and initiate the repairing process by performing the predefined tasks. As the remaining repairing scheme is similar for both the cases, only the second case of node crash and the tree repairing scheme for the same has been reported in this section.

After the successful completion of data gathering tree construction, the tree maintenance becomes the only responsibility of the proposed TMM. The proposed repairing scheme is able to handle both a single node failure as well as the multiple node failures. The TMM can repair the tree from any single node failure through a proactive repairing technique where precomputed alternate parent information is exploited for the new path establishment. Let, Δ denote the *repairing time for a node*, can be defined as the time taken between the parent node fails and the new parent is assigned. Δ includes an additional 1-hop channel delay to introduce an 1-hop neighbor updating delay after the repairing of the tree. In case of multiple node failures, the events of node failure can occur either in a series, or in parallel. For a series of node failures, if the time difference between occurrence of any two such events is more than Δ, each of such events of node failure can be treated as a single node failure, and handled accordingly. Two nodes are said to be failed in parallel or simultaneously, if the time difference between the events of the two node failures is less than Δ. For multiple node failures in parallel, if the distance between any two faulty nodes is more than 2-hop, each of them can be considered as an isolated single node failure. The proposed proactive tree repairing activities, performed for each of these nodes, would not interfere with each other. Further, let u and v be two simultaneously failed nodes, such that $u \in Neighbor(v)$. For all the cases, where if $u = parent(w)$, then $v \neq altp(w)$, and vice-versa, for node w, the simultaneous failure of nodes u and v can be treated as two different isolated single node failures, and thus can be handled separately by proactive repairing approach. For the sake of clarity, the proposed tree repairing scheme has been presented in two phases. The *first phase* describes the

TH-1251_11610108

proactive tree repairing activities from any arbitrary single node failure with the help of a precomputed alternate parent. In the *second phase*, the tree is repaired through a reactive approach to handle the special case scenario where both the parent and the precomputed alternate parent of a node fail in parallel. The transition from the first phase of the tree repairing to the second phase can be characterized by the probability of simultaneous node failures at the same locality, which is analyzed next.

Failure Probability Estimation : The failure of a wireless sensor node can be modeled as a specially localized random node failure [159]. The event of any arbitrary node failure can be considered as an independent and identically distributed process, where the probability of failure can be obtained from the Gaussian distribution. Let, $\mathscr{P}\{fail(u)\}$ be the probability of failure for any node u, then $\mathscr{P}\{fail(u)\}$ can be expressed as given in equation (3.1).

$$\mathscr{P}\{fail(u)\} = K_E e^{\left(-\frac{d_{u,s}^2}{2K_L^2}\right)} \tag{3.1}$$

$d_{u,s} = ||\{\mathcal{C}_X(u), \mathcal{C}_Y(u)\} - \{\mathcal{C}_X(s), \mathcal{C}_Y(s)\}||$, where $\{\mathcal{C}_X(u), \mathcal{C}_Y(u)\}$ denote the position coordinate of a node u in Euclidean plane and s is the sink. $d_{u,s}$ is the Euclidean distance between a node u and the sink s, and $K_E < 1$ is a positive constant that depends on the energy dissipation of a node. K_L is the localization factor that depends on the density of sensor node deployment. A large value of K_L signifies more number of nodes in the close vicinity. The failure of any node can be triggered either by the energy dissipation or by a random hardware crash, and hence, both of these parameters dictate the probability of an arbitrary node failure. Let, $\mathscr{P}\{fail(u) \wedge fail(v)\}$ be the probability that two nodes, u and v, fail simultaneously. Hence, $\mathscr{P}\{fail(u) \wedge fail(v)\} = \mathscr{P}\{fail(u)\} \times \mathscr{P}\{fail(v)\}$ as both the events are mutually exclusive. If both the nodes u and v are in the same locality, then $\mathscr{P}\{fail(u) \wedge fail(v)\} = \mathcal{F}^2$, where $\mathscr{P}\{fail(u)\} = \mathscr{P}\{fail(v)\} \simeq \mathcal{F}$. The probability for a single node failure and two simultaneous node failure have been shown in Figure 3.7, and Figure 3.8, with $K_L = 10$ nodes per unit area. From Figure 3.8, the value of two simultaneous node failure in a close vicinity is small. For example, with $K_E = 0.3$ and $d_{u,s} = 2$, the probability of two simultaneous node failure is 0.086. Therefore, only in 8.6% of the cases, the tree repairing activities would be the reactive in nature to handle multiple simultaneous node failures in a close vicinity. The proposed repairing scheme has been described in detail with the help of Algorithms 3.7 to 3.14 in the following subsections.

TH-1251_11610108

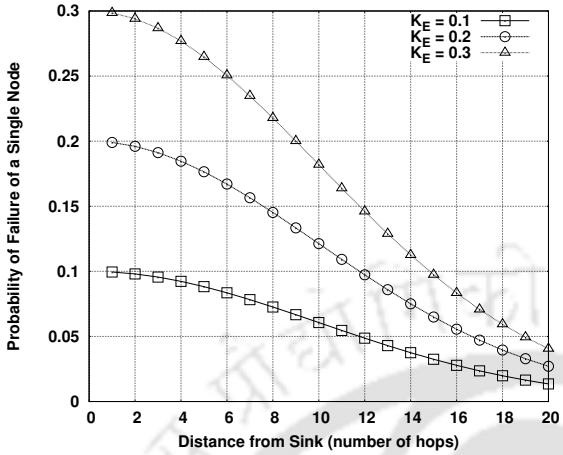

Figure 3.7: Probability of Single Node Failure

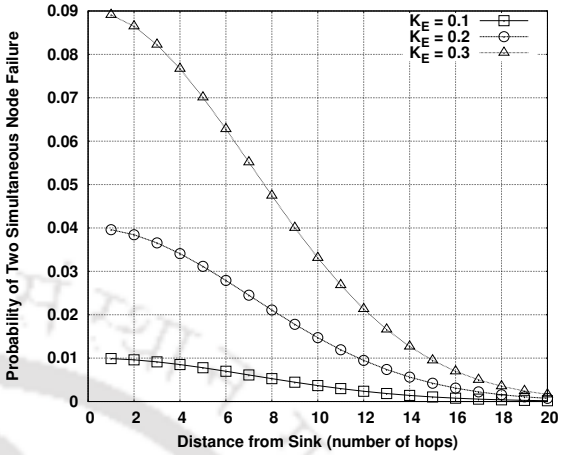

Figure 3.8: Probability of Two Simultaneous Failure

3.3.1 First Phase of Tree Repairing : Proactive Approach

The proposed proactive tree repairing scheme for handling any arbitrary single node failure has been described in Algorithms 3.7 to 3.12. However, the events of both the parent and the precomputed alternate parent failure simultaneously triggers the initiation of the second phase of tree repairing. Setting the Boolean variable *urgent* as True (default False), indicates the activities to be performed in the second phase of repairing, and hence, are discussed in the next subsection. In this subsection, only the proactive approach of repairing has been discussed to handle a single node failure. On detection of a neighbor failure through the lower layer fault detection technique, each node performs the necessary actions based on the precomputed information as stated in Algorithm 3.5.

Definition 3.1. *Let u be a node such that $parent(u) = v$. The node u is called a victim node if one of the following is true.*

1. *Node v has been crashed, and so, $parent(u)$ has been set to Null.*
2. *Node w has been crashed, where, there is a path $\mathbb{P} = \{w, x_1, x_2, ..., v, u\}$, such that $w = parent(x_1), x_1 = parent(x_2), ..., v = parent(u)$. Further, $parent(u)$ has been set to Null as u received ReqParent from v.*

The first phase of tree repairing is accomplished by exchanging four types of control messages, *ReqParent, ReqParentAck, Nak*, and *UpdateLevel* among the neighbors of the victim nodes. The cross-layer signal, *DetectCrash* is generated by the lower layer, on the event of detecting a node failure in the vicinity.

TH-1251_11610108

Algorithm 3.7 On receiving *DetectCrash(ID)* by u from the lower layer on detecting a node failure in neighborhood

```
1.  Neighbor(u) ← Neighbor(u) \ {ID}
2.  Remove entry from parentSet for ID
3.  Remove entry from altParentLevel for ID
4.  if ID ∈ Child(u) then
5.      halt ← True                                   /*Parent switching holds to clear data from Buffer*/
6.      while halt = True do
7.          Wait for AMC to return                    /*AMC will set halt to False eventually breaking the loop*/
8.      end while
9.      Child(u) ← Child(u) \ {ID}
10. else if ID = parent(u) then
11.     block ← True
12.     Send holdFlow to v, ∀v ∈ Child(u)
13.     parent(u) ← Null
14.     if altp(u) ≠ Null then
15.         rflag ← True
16.         Send ReqParent to altp(u)
17.     else
18.         urgent ← True                             /*Second phase of repairing initiates*/
19.         urgList ← urgList ∪ {u} ∪ {w}, ∀w ∈ Neighbor(u)
20.         for all w ∈ Neighbor(u) do
21.             Send Urgent(urgList) to w
22.         end for
23.     end if
24. else
25.     if ID = altp(u) then
26.         altp(u) ← Null
27.     end if
28.     Reset UpdateTimer                             /*altp recomputation is required*/
29. end if
```

On receiving a *DetectCrash(x)* signal from the lower layer, the tree maintenance module of a node first removes x from its neighborhood as given in line 1 of Algorithm 3.7. Then the following events occur.

1. If $x \in Child(u)$, it is removed from the *Child* set according to lines 4-9 of Algorithm 3.7. The *halt* variable, under the supervision of the AMC, controls the application message flow to assure a proper data delivery.

2. If $x = parent(u)$, then u first sets $block = True$ to assure no loss of application messages. u then forwards a *holdFlow* message towards the descendant subtree rooted at u, which in turn slows down the application data flow in all descendants of u. u sets its parent to Null, and sends a *ReqParent* message to the *altp*, if $altp(u)$ exists as stated in lines 10-16 of Algorithm 3.7. The $rflag$ is a Boolean variable that

TH-1251_11610108

Algorithm 3.8 On receiving *ReqParent* from v by u

```
1. if urgent = False then
2.     if v = p(u) then
3.         parent(u) ← Null
4.     end if
5.     if parent(u) ≠ Null then
6.         for all w ∈ Neighbor(u) \ {p(u)} do
7.             Send ReqParentAck(level, parent(u)) to w
8.         end for
9.         Child(u) ← Child(u) ∪ {v}
10.        if v = altp(u) then
11.            Reset UpdateTimer                    /*altp recomputation is required*/
12.        end if
13.    else
14.        if altp(u) = Null ∨ v = altp(u) then
15.            urgent ← True                        /*Second phase of repairing initiates*/
16.            block ← True
17.            urgList ← urgList ∪ {u} ∪ {w}, ∀w ∈ Neighbor(u)
18.            for all w ∈ Neighbor(u) do
19.                Send Urgent(urgList) to w
20.            end for
21.        else
22.            serviceList ← serviceList ∪ {v}
23.            if rflag = False then
24.                Send ReqParent to altp(u)
25.                rflag ← True
26.            end if
27.        end if
28.    end if
29. end if
```

keeps track of the event that a *ReqParent* message has been sent from u.

3. If $x = altp(u)$, then $altp(u)$ is set to Null.

If u is a node whose parent has not been failed, then on receiving *ReqParent* from v, u sends v a *ReqParentAck* message along with its *level* and parent information. u adds v in its *Child* set.

On receiving a *ReqParentAck* from a node v, node u updates its *altParentLevel* and *parentSet* set according to lines 1-5 of Algorithm 3.9. If $parent(u)$ has failed and $v = altp(u)$, then u works according to lines 6-21 of Algorithm 3.9. u first sets its parent to v, sets the *level*, and resets *block* and $rflag$. If $v \in Child(u)$, then u removes v from its *Child* set. For all the nodes from which u received a *ReqParent* once and are saved in the *serviceList*, are added to *Child* set. u then sends a *ReqParentAck* with its updated *level* and parent information to all the neighbors except the parent.

TH-1251_11610108

Algorithm 3.9 On receiving *ReqParentAck(mLevel, pid)* from v by u

1. $altParentLevel(u)_v \leftarrow mLevel$
2. $parentSet(u)_v \leftarrow pid$
3. **if** $pid \in Neighbor(u)$ **then**
4. $\quad altParentLevel(u)_{pid} \leftarrow mLevel - 1$
5. **end if**
6. **if** $urgent = False \wedge parent(u) = Null \wedge rflag = True \wedge altp(u) = v$ **then**
7. $\quad parent(u) \leftarrow v$
8. $\quad level \leftarrow mLevel + 1$
9. $\quad$ **if** $v \in Child(u)$ **then**
10. $\quad\quad Child(u) \leftarrow Child(u) \setminus \{v\}$
11. $\quad$ **end if**
12. $\quad block \leftarrow False$
13. $\quad rflag \leftarrow False$
14. $\quad$ **for all** $w \in Neighbor(u) \setminus \{p(u)\}$ **do**
15. $\quad\quad$ Send ReqParentAck$(level, parent(u))$ to w
16. $\quad$ **end for**
17. $\quad$ **for all** $w \in serviceList$ **do**
18. $\quad\quad Child(u) \leftarrow Child(u) \cup \{w\}$
19. $\quad$ **end for**
20. $\quad$ Clear $serviceList$ /*Required in Second phase of repairing*/
21. **end if**
22. **if** $urgent = False \wedge block = True \wedge v = parent(u) \vee v = altp(u)$ **then**
23. $\quad$ **if** $v = parent(u)$ **then**
24. $\quad\quad level \leftarrow mLevel + 1$
25. $\quad$ **end if**
26. $\quad$ **if** $parentSet(u)_{altp(u)} \in Neighbor(u) \wedge parentSet(u)_{altp(u)} \neq parent(u)$ **then**
27. $\quad\quad altp(u) \leftarrow parentSet(u)_{altp(u)}$
28. $\quad$ **end if**
29. $\quad$ Reset ImproveTimer /*Searches for better path option*/
30. **end if**
31. **if** $urgent = True$ **then**
32. $\quad$ **if** $parent(u) \neq Null$ **then**
33. $\quad\quad$ Send Remove to $parent(u)$ /*Lines 31-43 for Second phase of repairing*/
34. $\quad$ **end if**
35. $\quad parent(u) \leftarrow v$
36. $\quad level \leftarrow mLevel + 1$
37. $\quad block \leftarrow False$
38. $\quad urgent \leftarrow False$
39. $\quad$ Clear $urgList$
40. $\quad$ **for all** $w \in Neighbor(u)$ **do**
41. $\quad\quad$ Send UrgentAck$(level, parent(u))$ to w
42. $\quad$ **end for**
43. **end if**

If u is not a direct victim node but its *block* is set to True, and also, v is either $altp(u)$ or $parent(u)$, then on receiving *ReqParentAck* from v, u works as stated in lines 22-30 of Algorithm 3.9. If $v = parent(u)$ then the *level* is updated accordingly.

TH-1251_11610108

Algorithm 3.10 Procedure *improvePath()* triggered by u

```
1. if altParentLevel[altp(u)] + 1 < level then
2.     Send Add(True) to altp(u)
3. else
4.     for all w ∈ Neighbor(u) do
5.         Send UpdateLevel(level, parent(u)) to w
6.     end for
7.     Reset UpdateTimer                          /*altp recomputation is required*/
8. end if
```

Algorithm 3.11 On receiving *Nak(mFlag)* from v by u

```
1. if mFlag = False then
2.     Send Remove to parent(u)
3.     parent(u) ← altp(u)
4.     level ← altParentLevel(u)_altp(u) + 1
5. end if
6. block ← False
7. for all w ∈ Neighbor(u) do
8.     Send UpdateLevel(level, parent(u)) to w
9. end for
10. Reset UpdateTimer                             /*altp recomputation is required*/
```

If $parent(altp(u)) \in Neighbor(u)$ and $parent(altp(u)) \neq parent(u)$, then $altp(u)$ is set to $parent(altp(u))$, that of course, can offer a better *level* information. Next, the *improvePath* procedure is triggered on expiry of the *ImproveTimer*. Thus, u can change the parent to a neighbor offering the shortest level. Let $\delta_{(l-1)}$ denote the number of edges incident on u at the level l, and are connected to nodes at the level $l-1$. The *ImproveTimer* can be set to the value of $K \times \delta_{(l-1)}$, where K is a constant such that $0 < K < 1$. This is because the value of *ImproveTimer* depends on the maximum possible number of *ReqParentAck* messages that a node can receive. A node which is not directly a victim but *block* is set to True because of receiving a *holdFlow* from the parent (all the nodes in the subtree rooted at the victim node receive a *holdFlow*, which sets *block* to True), might receive a $\delta_{(l-1)}$ number of *ReqParentAck* messages at most.

On triggering of the *improvePath* procedure, node u acts according to Algorithm 3.10. If the level of $altp(u)$ is less than that of its current parent, then an *Add* message with a True argument is sent to the $altp(u)$. The addition of the node u to the *Child* set of v depends on whether v has a valid parent or not. If $block = True$ at the node v, or $parent(v)$ has failed, v sends a *Nak(True)* to u. Otherwise, v sends a *Nak(False)* to u, and immediately adds u in its *Child* set according to lines 5-12 of Algorithm 3.3. If the *level* of $altp(u)$ is not less than that of its parent, u sends an *UpdateLevel* message along with the updated parent and level information to all its neighbors.

TH-1251_11610108

Algorithm 3.12 On receiving *UpdateLevel(mLevel, pid)* from v by u

```
1: if urgent = False then
2: altParentLevel(u)_v ← mLevel
3: parentSet(u)_v ← pid
4: if pid ∈ Neighbor(u) then
5: altParentLevel(u)_pid ← mLevel − 1
6: end if
7: if v = parent(u) then
8: level ← mLevel + 1
9: Reset ImproveTimer                          /*Searches for better path option*/
10: end if
11: end if
```

The receipt of a *Nak* message with a False argument from a node v implies that u can change its parent immediately. So, u first sends a *Remove* message to its current parent, and then, sets its parent to $altp(u)$. It then updates the level accordingly as given in lines 1-5 of Algorithm 3.11. If the argument of *Nak* message is received as True, node u is not allowed to change its parent, and remains silent. Lines 6-10 of Algorithm 3.11 are executed irrespective of the status of the message argument received at the node u. u first resets *block* to resume the application data flow, and sends an *UpdateLevel* message along with the updated parent and level information to all its neighbors.

On receiving an *UpdateLevel* message from a node v, node u acts according to lines 2-10 of Algorithm 3.12. u first updates its neighborhood information according to lines 2-6. Then, if the message is received from the current parent, the level is updated, and u resets $ImproveTimer$ to search for a better path to the root. Moreover, each time any of the alternate parent and $Child$ information is updated for a node, the $UpdateTimer$ is reset. The $UpdateInfo$ procedure is triggered as a result to recompute the alternate parent for that node, based on the updated neighborhood information as stated in Algorithms 3.7,3.8,3.10, and 3.11.

An Example of Tree Repairing :

Figure 3.9 - Figure 3.11 give a pictorial description of the repairing mechanism from an arbitrary node crash. Let node c fail. So, its neighbors nodes a, f and g detect that, and remove the node c from their $Neighbor$ set. Node a removes a node c from its $Child$ set as well. Now, node g finds a node f as its $powerParent$, and sends a *ReqParent* message to f. Then, node g updates its $level$, and sends an *UpdateLevel* message to its children, the nodes i and j as shown in Figure 3.10. On receiving an *UpdateLevel*, node i and node j update their levels according to Algorithm 3.14. Node j changes its parent to a node h

TH-1251_11610108

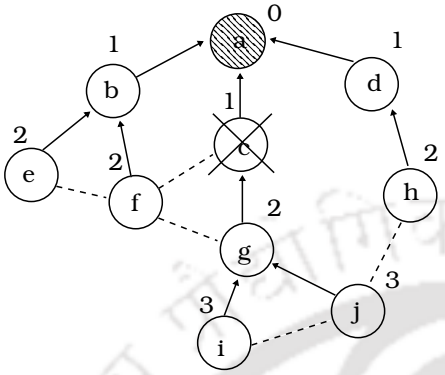

Figure 3.9: c fails. a, f and g detect and update neighbor/child accordingly

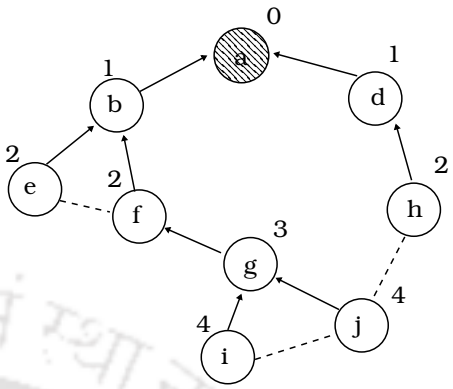

Figure 3.10: g sends *ReqParent* to its *powerParent*, f. On receiving *ReqParentACK*, g sends *UpdateLevel* to its children, i and j

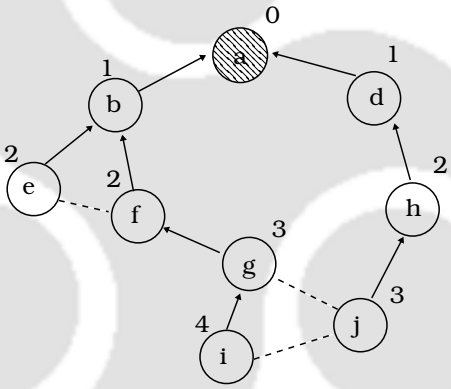

Figure 3.11: j changes parent from g to h due to level improvement. Final tree after repairing

as h offers lower level to the node j according to Figure 3.11.

Proof of Correctness :

The proof of correctness for the algorithms to repair the tree from a node crash as well as the computation cost of the tree repairing from any arbitrary single node failure have been provided with the following lemmas and theorems. Let the node x crash, and $\mathbb{T}_x$ be the subtree rooted at x. Let $V_{\mathbb{T}}$ be the set of all nodes in $\mathbb{T}_x$, V_P and V_C be the set of nodes, such that $V_P \cup V_C = V_{\mathbb{T}}$ and $V_P, V_C, V_{\mathbb{T}} \subset V$. All nodes in V_P change their parent due to either the parent failure or the level improvement, where as all nodes in V_C do not change their parent.

TH-1251_11610108

Lemma 3.5. *Let u be a node such that $power = True$ at u at the time t_i. On receiving a Booster message at the time $t_j > t_i$, u may change its status from Power to Booster by resetting the power variable.*

Proof. Let u receive a *Booster* message at the time $t_j > t_i$, from a node v where $v = altp(u)$. If $u \in blockedList(v)$ as sent by the message argument, it implies that $parent(parent(v)) = parent(u)$. Let u set v as its *altp*. On crash of $parent(u)$, u sends a *ReqParent* to v. However, the path from v to the root contains $parent(u)$ according to the assumption. Therefore, on crash of $parent(u)$, v also becomes a victim due to the grandparent crash. Thus, the selection of v as an alternate parent is a wrong decision at u. Therefore, u resets its *power*, and becomes a *Booster* node on receiving *Booster* message from v at the time $t_j > t_i$, according to lines 1-5 of Algorithm 3.6. □

Lemma 3.6. *Let u be a Booster node and w be its boostParent, then either of the following is true.*
i) $w \in Child(u)$
ii) $parent(w) = parent(u)$

Proof. According to lines 5-7 of Algorithm 3.5, the *power* variable at u is not set to True. This means either of the following is true.
Case 1. $[\{Neighbor(u) \setminus \{Sibling(u) \cup Child(u) \cup parent(u)\}\} = \phi]$ **:** Now, three cases are possible depending on $Neighbor(u)$.
Case (i) $[Child(u) = \phi$ and $Sibling(u) \neq \phi]$: Then, $w \in Sibling(u)$.
Case (ii) $[Child(u) \neq \phi$ and $Sibling(u) = \phi]$: Then, $w \in Child(u)$.
Case (iii) $[Child(u) \neq \phi$ and $Sibling(u) \neq \phi]$: Then, either $w \in Sibling(u)$ or $w \in Child(u)$. According to Algorithm 3.6, u selects only one w as its alternate parent, such that it offers the shortest path from the root.
Case 2. $[\exists z \in \{Neighbor(u) \setminus \{Sibling(u) \cup Child(u) \cup parent(u)\}\} \wedge parent(z) \in Sibling(u)\ ]$ **:**
u does not select z as its alternate parent according to lines 5-6 of Algorithm 3.5. If the above statement is true, then from lines 2-3 of Algorithm 3.5, $parent(parent(z)) = parent(u)$ is also true, and hence, $u \in blockedList(z)$. Thus, u never selects z as its alternate parent. □

Theorem 3.4. *Let U_p be a set of nodes such that $\forall u \in U_p, power = True$. U_b be another set of nodes such that $\forall v \in U_b, power = False$. Then, $U_p \cup U_b = V$ and $U_p \cap U_b = \phi$ are always true.*

TH-1251_11610108

Proof. Let there exists a node w such that $U_p \cap U_b = w$ is true. Now, if $w \in U_p$, then $power = True$ at w; again, if $w \in U_b$ then $power = False$ at w. As *power* is a Boolean variable, only one value persists at a time. So, $U_p \cap U_b$ must be empty.

Let $\exists w \in X$ such that $V \setminus \{U_p \cup U_b\} = X$. So, $\forall w \in X$, either $power = True$ or $power = Fasle$. If $power = True$, then $X \subseteq U_p$. If $power = False$, as $\mathbb{G}$ is connected, so, all such w must have received a *Booster* message, and thus, $X \subseteq U_b$.

Let $w \in U_p$ at the time t_i, and it reset its *power* at the time $t_j > t_i$ on receiving a *Booster* message. From Lemma 3.5, at the time t_j, $w \in U_b$ as *power* is a Boolean variable, and only one state persists at a time. So, in any of the cases, $X = \phi$, which follows $U_p \cup U_b = V$. □

Theorem 3.5. *Let there exist a path $\mathbb{P} = \{x_1, x_2, ..., x_n\}$ in the tree such that $x_1 = parent(x_2), x_2 = parent(x_3), ..., x_{n-1} = parent(x_n)$. Let x_1 fail, $S = \{x_3, x_4, ..., x_n\}$ and $w = parent(x_2)$ after repairing, one of the following must be true.*
Case 1 : $w \notin S$ such that $power = True$ at w
Case 2 : $w \in S$ such that $power = False$ at w. There must be at least one node $y \in S$ such that either $power = True$ at y or $altp(y) \notin S$.

Proof. Let $x_1 \in \mathbb{P}$ fail. So, x_2 has to choose node $w \in Neighbor(x_2)$ as its alternate parent. If $power = True$ at x_2, then from Lemma 3.6, $w \notin S$. If $power = False$ at w, then $w \in S$ from Lemma 3.6. Let $\forall v \in S \setminus \{w\}$, $power = False$. So, $\forall v \in S \setminus \{w\}$, $boostParent(v) \in Child(v)$ or $parent(boostParent(v)) = parent(v)$ has to be true. $Child(x_n) = \phi$, and as $\mathbb{G}$ remains connected even after x_1 fails, there must be a node $q \in Neighbor(x_n)$, such that either $parent(x_n) \neq parent(q)$ or $power = True$ at q. Hence, proved. □

Lemma 3.7. *The level of the alternate parent of a node is the second lowest in its neighborhood.*

Proof. From line 5 and line 8 of Algorithm 3.5, node u selects a node w as its $powerParent$, where w offers the minimum level, and $w \in \{Neighbor(u) \setminus \{Sibling(u) \cup Child(u) \cup parent(u)\}\}$. From Lemma 3.4, each node in $\mathbb{G}$ eventually selects a parent with the minimum level in neighborhood. As $parent(u)$ offers the lowest level to u in the neighborhood, the level of w is the second lowest in the neighborhood of u from the above.

Similarly, on receiving several *Booster* messages, a node selects w as its $boostParent$ if the *level* of w is the lowest among all the senders of all the *Booster* messages, according

TH-1251_11610108

to line 6 of Algorithm 3.6. So, the *level* of *boostParent* is the second lowest in the neighborhood of u.

From line 9 of Algorithm 3.5 and line 10 of Algorithm 3.6, the alternate parent of a node u, the $altp(u)$ is set to either *powerParent* or *boostParent* depending on the *power* status. So, the level of the alternate parent of a node is the second lowest in its neighborhood. □

Corollary 3.1. *On crash of the parent node each node resets its parent by exchanging a pair of messages (ReqParent, ReqParentAck) with its alternate parent, where the alternate parent does not fail during the repairing time.*

Proof. The proof is trivial. □

Theorem 3.6. *Once the first phase of the repairing (proactive repairing) is over, each node maintains the shortest distance to the root.*

Proof. Let V_I be the set of victim nodes such that $V_I \subseteq V_P$. From Algorithm 3.9, each node in V_I updates its level on receiving a *ReqParentAck* message. A node receives a *ReqParentAck* from a node v where v is its *powerParent* or *boostParent*. From Lemma 3.7, the level of v is the second lowest in the neighborhood. So, after the parent node dies, the *level* of v becomes the minimum in the neighborhood. Thus, on assigning v as a parent, each node $u \in V_I$ maintains the shortest distance to the root.

From lines 23-29 of Algorithm 3.9, on receiving a *ReqParentAck* from the parent, node u checks for a better available path than its existing alternate path to the root. If the updated level of $parent(altp(u))$ is lower than that of $altp(u)$, the *altp* is updated, and the $ImproveTimer$ is reset. As the *improvePath* procedure is triggered on expiry of the $ImproveTimer$, u changes its parent to its alternate parent that offers the lowest level in the neighborhood of u.

From Algorithm 3.12, all nodes in V_C update their levels on receiving a *UpdateLevel* message, but do not change the parent due to an absence of any better path to the root. From line 9 of Algorithm 3.12, on expiry of the $ImproveTimer$, the *improvePath* procedure is triggered that searches for a path improvement option according to Algorithm 3.12 and Algorithm 3.10. All nodes $u \in V_P \setminus V_I$ change their parent from p to p' if *level* of p' is less than the updated *level* of p. Thus, selecting p' as a parent, each $u \in V_P \setminus V_I$ maintains the shortest distance to the root. All nodes in $\{V - V_{\mathbb{T}}\}$ maintain the shortest distance according to Lemma 3.4 as they do not get affected due to the crash of x. Hence, after the local repairing, all nodes in the tree maintain the shortest distance to the root. □

TH-1251_11610108

Theorem 3.7. *Let any arbitrary node x crash and $\mathbb{T}_x$ be the subtree rooted at node x. The tree can be repaired locally with the worst case message complexity of $O(|E_{\mathbb{T}}|)$, where $E_{\mathbb{T}} \in E$ be the set of edges in $\mathbb{T}_x$.*

Proof. Let u be a node such that $x = parent(u)$. Let $V_I \in \mathbb{T}_x$ be the set of victim nodes in $\mathbb{T}_x$ due to the crash of x. Now, every node $u \in V_P$ changes the parent either due to the parent crash or due to the level improvement. For parent crash, every node $u \in V_I$ sets its parent through exchanging a pair of messages with the alternate parent from Corollary 3.1. For level improvement, every $u \in V_P \setminus V_I$ changes the parent through a pair of *Add* and *Remove* message exchange. So, every u transmits two messages for changing the parent $\forall u \in V_P$. Every node $v \in V_{\mathbb{T}} \setminus V_I$ receives an *UpdateLevel* message from all the neighbors for updating the neighborhood information, which results in total $\frac{1}{2}\sum_v \delta_v, \forall v \in V_{\mathbb{T}} \setminus V_I$ of *UpdateLevel* message transmissions, where δ_v is the degree of a node v. This eventually estimates to $O(|E_{\mathbb{T}}|)$, where $E_{\mathbb{T}} \in E$ be the set of edges in $\mathbb{T}_x$. In the worst case, if all nodes in $\mathbb{T}_x$ change the parent due to the crash of x, the total number of control message transmissions would be equal to $2 \times |V_P| + \frac{1}{2}\sum_v \delta_v, \forall v \in V_{\mathbb{T}} \setminus V_I$, which is effectively $O(|E_{\mathbb{T}}|)$. Hence, proved. □

Theorem 3.8. *Let node x fail where $x = parent(u)$ at the time t_i, and $w = altp(u)$. Let at the time t_j, where $t_j > t_i$, a set of nodes $F = \{z_1, z_2, ..., z_n\} \in Neighbor(u)$ fail. u can still set an alternate path to the root at the time $t_k > t_j$, provided that both the conditions hold true.*

1. *$w \neq z_i,\ \forall z_i \in F$*
2. *The failure of such $z_i \in F$ does not make the graph disconnected*

Proof. Let u be a node such that $u \in Child(x)$. So, node $w \in Neighbor(u)$ is either a *powerParent* or a *boostParent*, from Theorem 3.4. So, $\forall v \in Neighbor(u) \setminus \{w\} \cup \{x\}$, either $v \in Child(u)$ or $v \in Sibling(u)$, or the edge (u, v) is a non-tree edge where $v = z_i, \forall z_i \in F$.
Case 1 : Let $z_i \in Child(u)$; u performs the actions as given in lines 4-9 of Algorithm 3.7, which is disjoint with the actions performed due to failure of x.
Case 2 : Let either $z_i \in Sibling(u)$ or $z_i \in Neighbor \setminus \{parent(u) \cup Child(u) \cup Sibling(u)\}$; u calls the $updateInfo$ for updating the neighborhood information, which is again disjoint with the actions performed for the repairing of the failure of node x. If the failure of $z_i \in F$ does not make the graph disconnected, and if $z_i \neq altp(u)$, then the crash of the set $F \in Neighbor(u)$ within the time t_i and t_k can be tolerated for any cardinality of the

TH-1251_11610108

set F. The tree repairing process initiated for each individual node in F does not interfere with the repairing process due to failure of x. Hence, proved. □

3.3.2 Second Phase of Tree Repairing : Reactive Approach

The first phase of repairing is sufficient to repair the tree from any isolated single node failure based on the precomputed alternate parent information. The proactive way of repairing is fast and cost-effective. However, as mentioned earlier, the proactive way of repairing fails when both the parent and the alternate parent of a node fail simultaneously. In this case, the victim nodes perform the repairing activities in a reactive way as discussed in this subsection. Let, for any node u, $\mathscr{E}_u^p(t_i)$ denote the event of parent node failure at the time t_i, and $\mathscr{E}_u^a(t_j)$ denote the event of alternate parent node failure at the time t_j. Δ be the repairing time for any node as mentioned earlier. Therefore, the second phase of repairing technique is triggered at the time t, called the decision point, if any of the following statements is true.

1. $\mathscr{E}_u^p(t_i) \wedge \mathscr{E}_u^a(t_j)$ where $t = t_i \quad$ and $\quad t_j < t_i < t_j + \Delta$
2. $\mathscr{E}_u^p(t_i) \wedge \mathscr{E}_u^a(t_j)$ where $t = t_j \quad$ and $\quad t_i < t_j < t_i + \Delta$
3. $\mathscr{E}_u^p(t_i) \wedge \mathscr{E}_u^a(t_j)$ where $t = t_i = t_j$

Thus, the precomputed alternate path information at a node becomes invalid on simultaneous failure of both the parent and the alternate parent in the data gathering tree. Therefore, the victim node searches for an alternate path to the root as the failure events occur. The search procedure relies on the precomputed information at other neighbors of the victim node. The second phase of repairing has been implemented by introducing two more control messages, *Urgent* and *UrgentAck*, as given in Algorithm 3.13 and Algorithm 3.14.

Let u be a node such that both of its parent and alternate parent have been crashed. Therefore, u enters the second phase of repairing by executing lines 17-23 of Algorithm 3.7. u sets the *urgent* flag to True, and sends an *Urgent* message to all its neighbors. All victim nodes, participating in the second phase of tree repairing, set the *urgent* flag to True, either on detecting the failure of both the parent and the alternate parent, or, on receiving an *Urgent* message. The *Urgent* message carries an *urgList* as an argument which contains $Neighbor(u) \cup \{u\}$.

Every *Urgent* message is broadcast in 1-hop neighborhood. On receiving an *Urgent(urgList)* message from a node v for the first time, node w checks the message

TH-1251_11610108

Algorithm 3.13 On receiving *Urgent(ulist)* from v by u

```
1.  if urgent = False then
2.      if parent(u) ∈ ulist ∨ parent(u) = Null then
3.          urgent ← True
4.          block ← True
5.          if altp(u) ∉ ulist ∧ parentSet(u)_altp(u) ∉ ulist then
6.              Send ReqParent to altp(u)
7.          else
8.              urgList ← urgList ∪ {u} ∪ {w}, ∀w ∈ Neighbor(u)
9.              for all w ∈ Neighbor(u) ∧ w ∉ ulist do
10.                 Send Urgent(urgList) to w
11.             end for
12.         end if
13.     else
14.         if parentSet(u)_p(u) ∉ ulist then
15.             for all w ∈ Neighbor(u) do
16.                 Send UrgentAck(level, parent(u)) to w
17.             end for
18.         else
19.             urgent ← True
20.             block ← True
21.             urgList ← urgList ∪ {u} ∪ {w}, ∀w ∈ Neighbor(u)
22.             for all w ∈ Neighbor(u) ∧ w ∉ ulist do
23.                 Send Urgent(urgList) to w
24.             end for
25.         end if
26.     end if
27. end if
```

argument $urgList$, that contains all the victim node IDs. A node w becomes a victim if its parent is also a victim, according to lines 2-4 of Algorithm 3.13. If neither the $altp(w)$ nor the $parent(altp(w))$ is in the $urgList$, then the $altp(w)$ has a valid path to the root. So, w sends a *ReqParent* to the $altp(w)$. Otherwise, w becomes a victim, and broadcasts an *Urgent* message in 1-hop neighborhood with its updated $urgList$, according to lines 5-12 of Algorithm 3.13. If neither the $parent(w)$ nor the $parent(parent(w))$ is in the $urgList$, then w itself has a valid path to the root. So, it replies back with an *UrgentAck* message to v. Otherwise, w broadcasts an *Urgent* message in 1-hop neighborhood with the updated $urgList$, according to lines 18-26 of Algorithm 3.13.

The *UrgentAck* message is 1-hop broadcast message that carries the level and parent information of the sending node. On receiving an *UrgentAck*, every node updates its neighborhood information according to lines 1-5 of Algorithm 3.14. Let u receive an *UrgentAck* from a node v. If the *urgent* is set to True at u, then u is a victim node. Either its parent has been crashed or its parent became invalid on receiving an *Urgent*

TH-1251_11610108

Algorithm 3.14 On receiving *UrgentAck(mLevel, pid)* from v by u

```
1.  altParentLevel(u)_v ← mLevel
2.  parentSet(u)_v ← pid
3.  if pid ∈ Neighbor(u) then
4.      altParentLevel(u)_pid ← mLevel − 1
5.  end if
6.  if pid = u then
7.      Child(u) ← Child(u) ∪ {v}
8.  end if
9.  if urgent = True then
10.     if parent(u) ≠ Null then
11.         Send Remove to parent(u)
12.     end if
13.     parent(u) ← v
14.     level ← mLevel + 1
15.     urgent ← False
16.     block ← False
17.     Clear urgList
18.     for all w ∈ Neighbor(u) do
19.         Send UrgentAck(level, parent(u)) to w
20.     end for
21. end if
```

message from the parent. If $parent(u)$ has not been crashed, u first sends a *Remove* message to the parent to delete the edge with its existing parent as stated in lines 10-12 of Algorithm 3.14. In any case, on receiving an *UrgentAck*, a victim node sets its parent and level, resets all variables, and further, broadcasts the *UrgentAck* message with its updated level and parent information, according to lines 13-20 of Algorithm 3.14. If the parent information of the message matches with its own ID, u adds it in its *Child* set according to lines 6-8 of Algorithm 3.14.

A victim node whose *urgent* is set to True may receive an *ReqParentAck* message as an acknowledgment from its valid alternate parent. In that case, if $parent(u)$ is not crashed, u sends a *Remove* message to its old parent. Irrespective of the status of the parent node, on receiving a *ReqParentAck* message from the *altp*, a node u sets its parent and level, resets all variables and broadcasts *UrgentAck* message in 1-hop neighborhood according to lines 35-42 of Algorithm 3.9.

Every node that becomes a victim due to crash of both the parent and the alternate parent in the tree, eventually establishes a new path to the root through this reactive way of repairing. Once the node updates its parent, level, and other state variables correctly, and reach the stable state, it recomputes its alternate parent with the help of the updated neighborhood information. Thus, multiple node crashes even in parallel can be tolerated,

TH-1251_11610108

and the data gathering tree is repaired through a local adjustment using this scheme. However, there is a trade-off between the repairing time and the optimal path selection during the repairing. According to this scheme, a victim node u selects a node v as its parent if u receives an *UrgentAck* or a *ReqParentAck* message from v, where v is the first node in $Neighbor(u)$ from which it has received that message. On receiving multiple such messages from other neighbors, u ignores the rest. This ensures the repairing would be fast, and the tree will eventually converge to the state where every node maintains the shortest distance to the root. The proposed scheme is designed to offer the fast repairing, and avoids an optimal path selection due to an extra processing delay required for searching. For the delay tolerant network, this scheme can be extended accordingly for an optimal path selection.

Proof of Correctness

The correctness of the repairing scheme along with the total message and the per node space overhead have been established by the following lemmas and theorems.

Theorem 3.9. *The tree can be repaired from any arbitrary multiple node crashes either in parallel or in a series, including the simultaneous crash of the parent and the alternate parent.*

Proof. The correctness of the repairing scheme from multiple node failures can be justified by the analysis of the following cases. Let a set of nodes $S = \{x_1, x_2, ..., x_n\}$ fail either in parallel or in a series.

Case 1 : Every node in S is located in more than 2-hop distance from another node in the set. In this case, the failure of one node does not affect the local repairing process from failure of another node. Thus, every node failure can be treated as an isolated single node failure, and the tree can be repaired as stated in Subsection 3.3.1.

Case 2 : Nodes in S are in 1-hop neighborhood such that for any node u, if $parent(u) \in S$ then $altp(u) \notin S$ or vice-versa. In this case, the tree can be repaired using the technique stated in Subsection 3.3.1.

Case 3 : Nodes in S are in 1-hop neighborhood such that for any node u both $parent(u), altp(u) \in S$. Let at the time t_i, nodes x_1 and x_2 crash. Also, let $u \in Child(x_1)$ and $altp(u) = x_2$. $\mathbb{T}_{x_1}$ and $\mathbb{T}_{x_2}$ be the subtree rooted at node x_1 and node x_2 respectively. From Algorithm 3.7, node u will broadcast an *Urgent* message. From Algorithm 3.13, the *Urgent* message is forwarded to all nodes except those belong to $urgList$. $urgList(u) = u \cup Neighbor(u)$. So, all the nodes $w \in \{\mathbb{T}_{x_1} \cup \mathbb{T}_{x_2}\}$ become a victim

TH-1251_11610108

on receiving an *Urgent* message. Now, as $\mathbb{G}$ is connected, $\exists e \in E$ such that $(r, s) = e$, and $r \in \{\mathbb{T}_{x_1} \cup \mathbb{T}_{x_2}\}$ and $s \in V \setminus \{\mathbb{T}_{x_1} \cup \mathbb{T}_{x_2}\}$. As the *Urgent* is a broadcast message, s will receive it eventually from a node r. Now, from Algorithm 3.14, as $parent(s) \notin \{\mathbb{T}_{x_1} \cup \mathbb{T}_{x_2}\}$, and also, $parent(parent(s)) \notin \{\mathbb{T}_{x_1} \cup \mathbb{T}_{x_2}\}$, s has a valid path to the root of the tree. So, s replies with an *UrgentAck* that is eventually received by all victim nodes in $\{\mathbb{T}_{x_1} \cup \mathbb{T}_{x_2}\}$. Thus, every victim node eventually sets its parent correctly even after multiple node crashes in a close vicinity. If $x_1 \in Neighbor(x_2)$, the vicinity of failure is 1-hop. If $x_1 \notin Neighbor(x_2)$, the vicinity of failure is 2-hop or more than 2-hop.
Case 4 : If the failure of nodes in S makes the graph partitioned, the tree has to be reconstructed from the scratch. □

Theorem 3.10. *Let the nodes x and y fail in close vicinity, such that $x = parent(u)$ and $y = altp(u)$. Let $\mathbb{T}_x$ and $\mathbb{T}_y$ be the subtree rooted at the nodes x and y respectively. Therefore, the control message overhead for repairing the tree from simultaneous failure of both the nodes x and y is $O(|E_{\mathbb{T}}|)$, where $E_{\mathbb{T}} \in E$ be the set of edges in $\mathbb{T}_x \cup \mathbb{T}_y$.*

Proof. From Theorem 3.9, all victim nodes set their parent correctly after simultaneous crash of multiple nodes, in close vicinity. From Algorithm 3.13, every edge $e \in \{\mathbb{T}_x \cup \mathbb{T}_y\}$ transmits one *Urgent* message. From Theorem 3.9, one pair of *Urgent* and *UrgentAck* messages are exchanged with a node $w \notin \{\mathbb{T}_x \cup \mathbb{T}_y\}$, such that w has a valid path to the root of the tree. The *UrgentAck* is a broadcast message. So, every edge $e \in \{\mathbb{T}_x \cup \mathbb{T}_y\}$ transmits an *UrgentAck* message twice. The total number of edges in $\{\mathbb{T}_x \cup \mathbb{T}_y\}$ is $\frac{1}{2}(\sum_{v \in \{\mathbb{T}_x \cup \mathbb{T}_y\}} \delta_v)$, where δ_v denotes the degree of a node $v \in \{\mathbb{T}_x \cup \mathbb{T}_y\}$. Therefore, the total number of control messages required for repairing the tree from multiple simultaneous node crashes is $3 \times \frac{1}{2}(\sum_{v \in \{\mathbb{T}_x \cup \mathbb{T}_y\}} \delta_v) + 2$, which is essentially $O(|E_{\mathbb{T}}|)$, where $E_{\mathbb{T}} \in E$ is the set of edges in $\mathbb{T}_x \cup \mathbb{T}_y$. □

Theorem 3.11. *Per node space complexity for the proposed scheme is constant multiple of δ_u in the worst case, where δ_u denotes the degree of a node u.*

Proof. From Table 3.1, every node stores five Boolean variables and six Integer variables. Considering each Boolean and each Integer variable consumes one unit of memory space, the total memory usage for storing these variables is 11 units. The $Neighbor(u)$ contains δ_u number of node IDs, where each node ID is assumed to take 1 unit of memory. Thus, every node stores three Set and three List variables, *Neighbor*, *Child*, *Sibling*, *serviceList*, *blockedList* and *urgList*; each of which can be as large as the *Neighbor* set in the worst case. So, all these variables contribute to the memory usage of $6 \times \delta_u$. Additionally,

TH-1251_11610108

Algorithm 3.15 *Application Message Controller*: On receiving Application message $\mathcal{M}$

1. Put $\mathcal{M}$ in Buffer
2. **if** $halt = True$ **then**
3. Insert # in Buffer
4. **end if**
5. **while** $Buffer \neq Full \wedge block = False \wedge *Buffer \neq \#$ **do**
6. Send *Buffer to $parent(u)$
7. **end while**
8. **if** *Buffer = # **then**
9. Adjust Buffer index
10. $halt \leftarrow False$
11. **end if**

every node keeps two Map variables, $altParentLevel$ that stores the *level* of each node in $Neighbor(u)$, and $parentSet$ that stores the parent of each node in $Neighbor(u)$. Therefore, these two variables add $2 \times \delta_u$ amount of space to the total usage. Hence, the total space required for a node u to maintain all the state variables in the worst case is $8 \times \delta_u + 11$. □

3.4 Application Message Controller

Application data sensed at each node is forwarded to its parent such that all data are eventually delivered to the root of the tree. Each received data packet is enqueued in a local Buffer. The application data may get lost or delivered to the root as a redundant due to the changes in the forwarding path during the local repairing process from any arbitrary node failure. On failure of the parent node x, the application data, that are already forwarded to x, or is stored in the local buffer of x, can not be recovered further. However, the loss or the redundant delivery of data due to the parent change can be avoided by a proper design of the Application Message Controller (AMC). The flow of application data to and from the Buffer is controlled by the AMC as given in Algorithm 3.15. The TMM and the AMC cooperatively access two Boolean variables, *block* and *halt*, to assure the correct delivery of application data. During the parent changing, the TMM sets the *block* to True, and resets only after the successful completion of the parent change event. Thus, during the parent change, based on the status of the *block*, the AMC stops out-flow of data from the Buffer as stated in lines 5-6 of Algorithm 3.15 to avoid any data loss. The data packets, that are yet to be forwarded to x and buffered at all the children of x, are rerouted to the sink after the parent is reassigned correctly. The *halt* variable, set by the TMM at every victim node, is responsible for assuring no redundancy

TH-1251_11610108

in the delivered data. Once $halt = True$, the AMC inserts a marker # in the Buffer that distinguishes the data packets to be forwarded to the old parent from the ones to be forwarded to the new parent, such that no packet is forwarded as a duplicate. At the time of tree repairing, each node u for which $parent(u)$ became Null, forwards a *holdFlow* message to its children. This suspends the outgoing data flow by setting $block = True$ at each child of u. The *holdFlow* messages are forwarded towards the leaves through the paths of the subtree rooted at x. Thus, the upward application data flow is suspended at each node in those paths. This reduces the overhead of a redundant path traversal for the application data due to a node crash. The AMC at every node, thus controls the local buffer to ensure the forwarding of application data without any loss or redundancy. This in turn, assures the correct delivery of all application data in the sink without any loss or redundancy even in presence of a node failure.

3.5 Simulation Results

The proposed tree repairing scheme for data gathering has been simulated through NS-2.35 network simulator [4]. The simulation parameters have been given in Table 3.2. The application message forwarding rate has been synchronized with the MAC layer service rate such that the network be never saturated. The simulation has been done in three phases to analyze the behavior of the proposed scheme under all possible scenarios. In the first phase, to analyze the behavior of the proposed scheme under extreme conditions, two different scenarios have been considered, one for a ring topology, and other, for a chain topology. In the second phase, a regular grid topology has been considered for the simulation. In the third phase, the simulation has been performed for a random topology. The proposed scheme is compared in terms of repairing delay with the collection tree protocol (CTP) [71] that offers fault tolerant data gathering from sensor nodes. The source tree based forwarding protocol [158] offers reliable congestion-free data delivery from sensor nodes to the sink, but, does not consider any node failure. The proposed scheme is also compared with the source tree based forwarding in terms of the number of packets received at the sink.

3.5.1 Phase I : Extreme Case Scenario

In the worst case scenario, with a ring topology only two extreme leaf nodes are *Power* nodes; the remaining nodes are *Booster* nodes. So, for a single node failure, a series of nodes become victim as their respective parent nodes become invalid. The repairing

TH-1251_11610108

Table 3.2: Parameters for Simulation Environment

Parameter	**Value**
Channel bandwidth	2 Mbps
Transport traffic type	UDP
MAC layer protocol	CSMA
Application traffic type	CBR
CBR packet length	12 bytes
Mean packet duration	100 ms
Control buffer timeout	50 ms
Data buffer timeout	100 ms
UpdateTimer timeout	400 ms
ImproveTimer timeout	400 ms
Communication range for sensor nodes	200 m
Interference range for sensor nodes	450 m

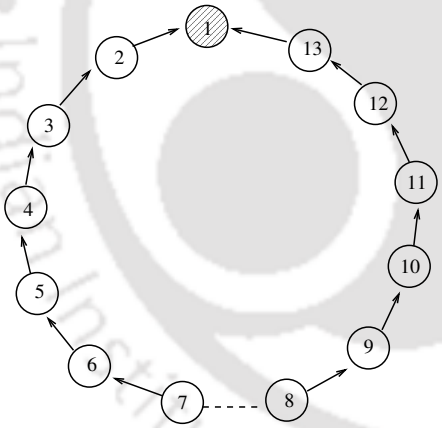

Figure 3.12: BFS tree for a ring

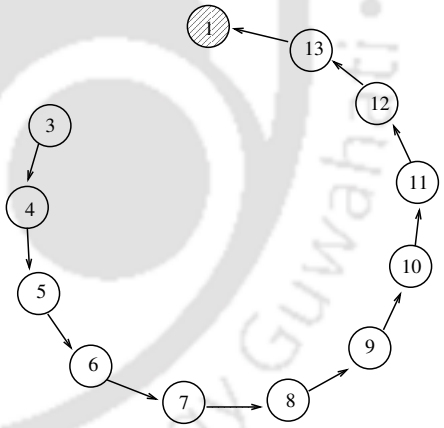

Figure 3.13: The tree after node *2* fails

involves many nodes with an increase in both the repairing time and the control message overhead. The best case scenario is when the subtree rooted at the node u, where u is a *Power* node, form a chain structure after $parent(u)$ fails. Then, u sets its new parent in a constant time. Also, the descendants of u do not need to change the parent if the level of u remains the same. The repairing cost that has to bear by all these nodes is only the blocking time in forwarding the application messages.

TH-1251_11610108

Table 3.3: Repairing time(in sec) for the victim nodes in a ring topology

Node Id	Start-time	End-time	Δ
3	250.0435	250.3566	0.3131
4	250.0537	250.3285	0.2758
5	250.0773	250.2944	0.2171
6	250.1130	250.2727	0.1097
7	250.1346	250.2409	0.1063

For a Ring Topology :

Figure 3.12 shows a BFS tree for a ring topology of 13 nodes, where node *1* is the sink, and thus, the root of the tree. Node *7* and node *8* are the leaf nodes for the initial data gathering tree. All application data are accumulated at the sink node *1*. Now, let a node *2* suddenly fail. So, the parent variable for the node *3* becomes invalid. Likewise, the parent for all the nodes in a series, from the node *4* to the node *7*, become invalid as all of them are the descendants of the node *3*, and are the *Booster* nodes. Each node fixes its parent through exchanging a pair of messages with its alternate parent according to Lemma 3.1. The repairing time, denoted by Δ, at each node for the series of victim nodes have been given in Table 3.3. Δ is the difference between Start-time (when $block = True$ after receiving a *holdFlow*) and End-time (when $block = False$, after receiving an *UpdateLevel*). It can be noted that Δ decreases through the walk from the original victim, node *3*, towards the leaf node. The tree is repaired as the nodes exchange control messages with their respective alternate parent at the next higher level, starting from the node *3* at the level 2 and ending to the node *7* at the level 6. So, the total number of control message exchange for the repairing is 18, which also satisfies the theoretical proposition given in Theorem 3.7. It has been observed from the analysis of the simulation trace that only one message is lost as it was in the Buffer of the parent of the node *3* during its failure. The remaining messages are delivered to the sink without any loss or redundancy. However, depending on the application data rate, some messages might be delivered to the sink out of order as the parent-child relationship gets reverse for a series of nodes in this scenario.

For a Chain Topology :

Figure 3.14 shows the BFS tree for a chain topology of 8 nodes. Let a node *3* fail. This is the simplest case as node *4* is a *Power* node, and it sets its parent to the node *2*,

TH-1251_11610108

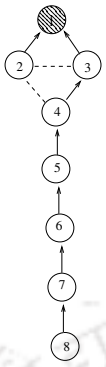

Figure 3.14: BFS tree for a chain

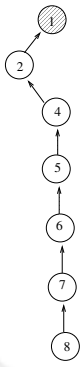

Figure 3.15: The tree after node *2* fails

Table 3.4: Repairing time (in sec) for the victim nodes in a chain topology

Node Id	Start-time	End-time	Δ
4	250.026	250.045	0.019
5	250.035	250.052	0.017
6	250.043	250.060	0.017
7	250.047	250.067	0.020
8	250.054	250.074	0.020

after node *3* fails, through only a pair of message exchange from Lemma 3.1. Each of the nodes, from the node *5* to the node *8*, does not change its parent; neither due to the parent invalidation, nor due to the level improvement. Each of them receives one *holdFlow* message and one *UpdateLevel* message. So, 10 control messages are exchanged in total to repair the tree, which also validates the theoretical proposition stated in Theorem 3.7. The repairing time for each node in the chain is given in the Table 3.4. Depending on the depth of the leaf node in the path (starting from node *3*), and the link delay (for receiving the first *holdFlow* followed by the *UpdateLevel* message), Δ increases as observed from Table 3.4.

3.5.2 Phase II : Regular Grid Topology

In this phase, the scalability of the proposed scheme in terms of the tree repairing time and the control message overhead has been analyzed by simulating it for a grid topology with the diagonal connectivity. Starting from 5×5, the grid size has been increased up to 20×20. Two different sets of nodes in more than 2-hop neighborhood have been made

TH-1251_11610108

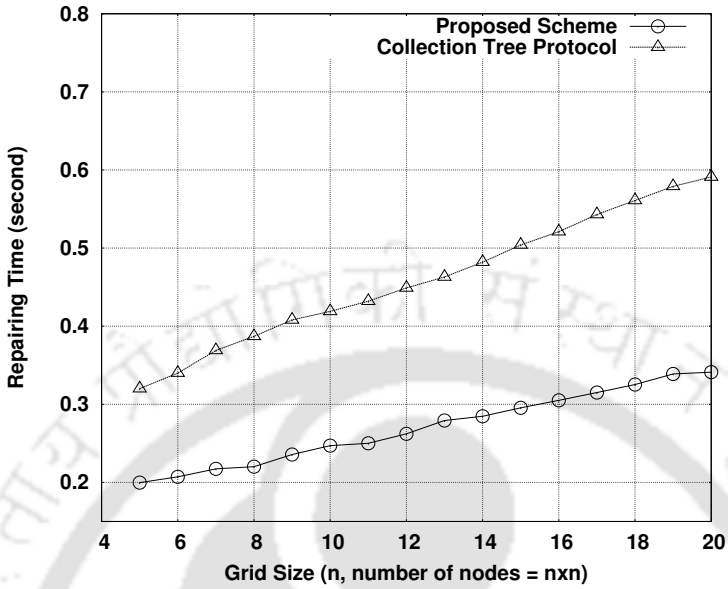

Figure 3.16: Scalability in terms of tree repairing time

to be failed where each set consists of two nodes, the parent and the alternate parent of a particular node. From equation (3.1), the probability of failure is more near the sink, as shown in Figure 3.7 and Figure 3.8. Hence, both the sets of nodes are considered to be positioned at near the sink such that the simultaneous failure of all these nodes do not partition the network. The tree repairing time has been plotted along the Y-axis as shown in Figure 3.16. The tree repairing time can be defined by the maximum time required to reach the stable state after the repairing among all the affected nodes in the subtree rooted at any failed node. The proposed tree repairing time has been compared with that of the collection tree protocol (CTP) [71]. It can be observed that the tree repairing time is linear with respect to an increase in the grid size, and considerably less than that of the CTP. The CTP uses a source based routing policy to find out the optimum path from sensor nodes to the sink, and a node failure is repaired on-the-fly as a new path is discovered. After a fault is detected, the complete path from the sensor node to the sink needs to be recomputed, which requires considerable amount of time. According to the proposed scheme, the repairing is done only at the neighborhood of the faulty node. That is why, the repairing delay is less compared to the CTP.

Percentage Overhead is the metric, defined as $(N_c/N_a) \times 100$, where N_c and N_a denote the total number of control messages for the repairing received at the sink node, and the total number of application messages (with 12 bps data rate) received at the sink,

TH-1251_11610108

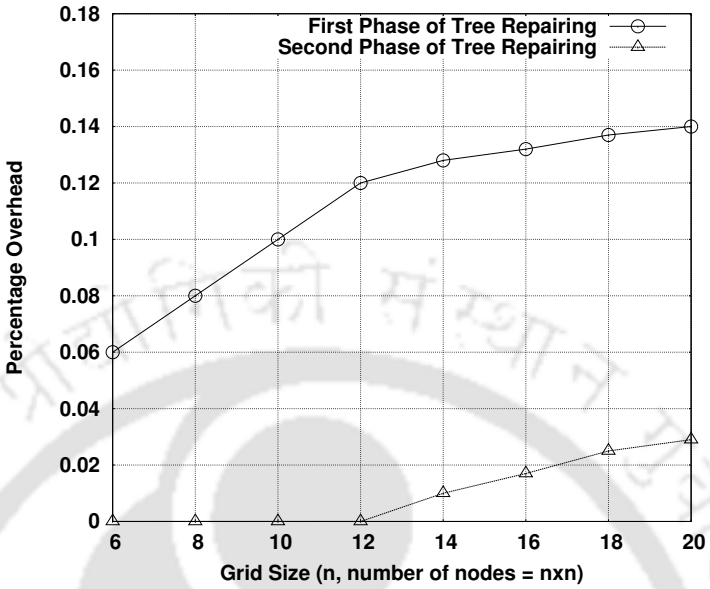

Figure 3.17: Scalability of the proposed scheme in terms of control message overhead

respectively. *Percentage Overhead* for the network has been plotted with respect to the grid size in the range of 6×6 to 20×20. It can be noted from Figure 3.17 that the *Percentage Overhead* is significantly low, and it increases linearly with an increase in the grid size. Furthermore, for small size of network, the second phase of repairing has no effect on the amount of *Percentage Overhead*.

Another metric, the percentage of packets received at the sink, can be defined to determine the % of data loss for the running application. The proposed scheme has been compared with the source based data forwarding technique. As the grid size increases, the percentage of packets received at the sink decreases linearly as shown in Figure 3.18. However, it can be noted that the % of data loss for the proposed scheme is about 6.7% less on average than that of the source based data forwarding [158]. Moreover, the slope of the data loss curve is less in the proposed scheme compared to the source based data forwarding.

The effect of the *LazyTimer* and the *UpdateTimer* value on the tree convergence time, and the number of *Token* broadcast have been shown in Figure 3.19, and Figure 3.20. As discussed earlier, the value of the *LazyTimer* and the *UpdateTimer* ideally should be the same, and the timer value is increased from 0.05 sec (average per hop delay in the network which is equivalent to the link delay + average queuing delay) to 1.3 sec. From Figure 3.19 and Figure 3.20, when the timer value is small, the convergence time is large

TH-1251_11610108

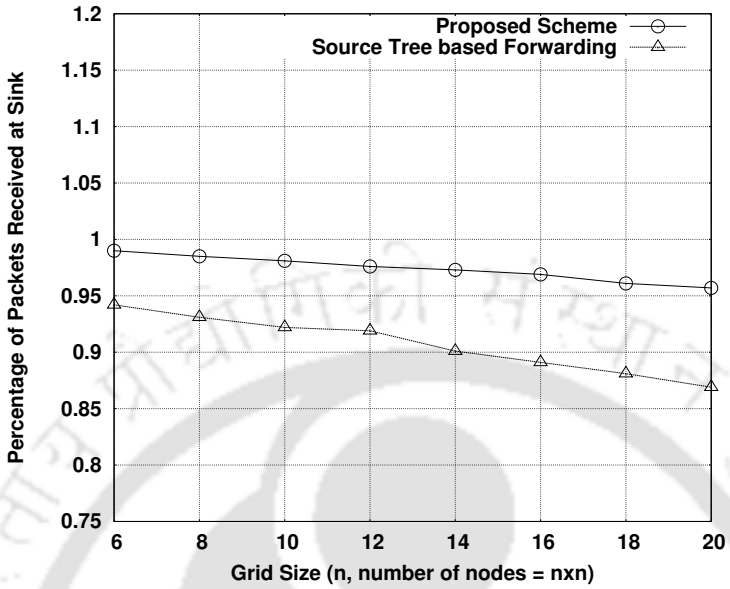

Figure 3.18: Percentage Packets Received at Sink

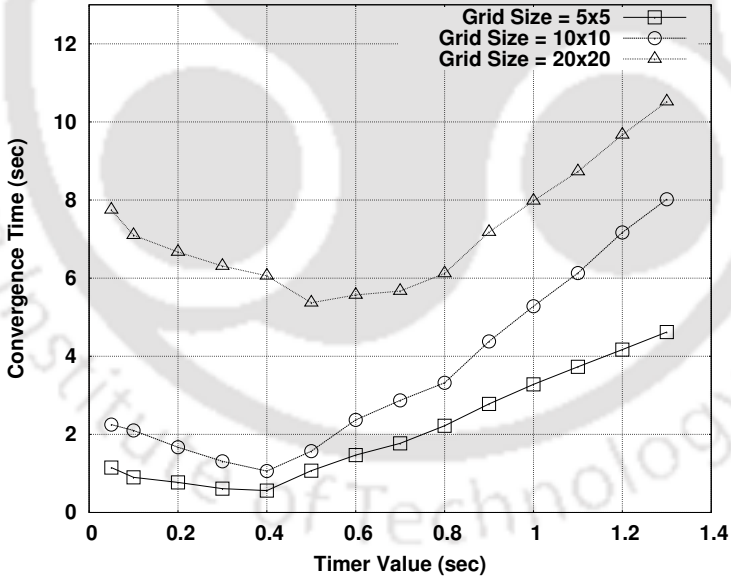

Figure 3.19: The proposed tree convergence time with respect to timer value

because more number of *Token* broadcast is required to converge the tree to a BFS tree. Similarly, large timer value minimizes the number of *Token* broadcast, but increases the convergence time due to a late expiry of the timer value. From extensive simulation, it has been observed that if the per hop delay (link delay + average queuing delay) is Ω

TH-1251_11610108

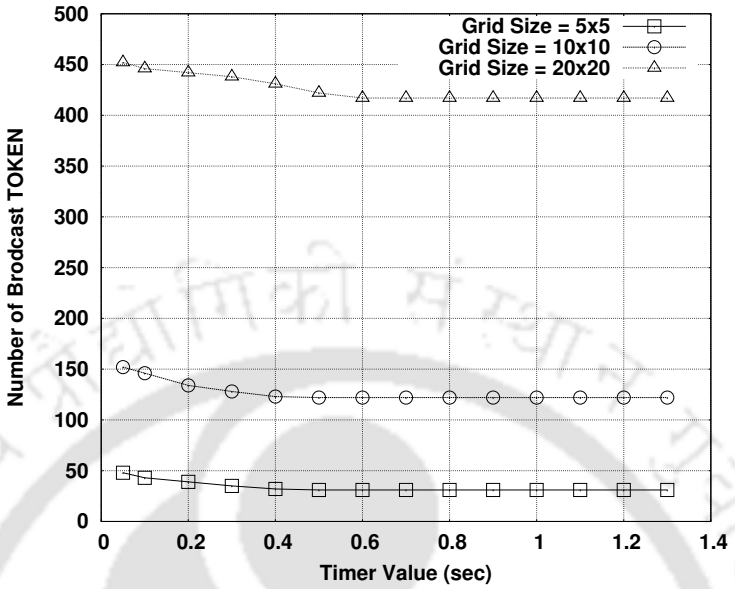

Figure 3.20: Amount of Token broadcast for the proposed scheme with respect to timer value

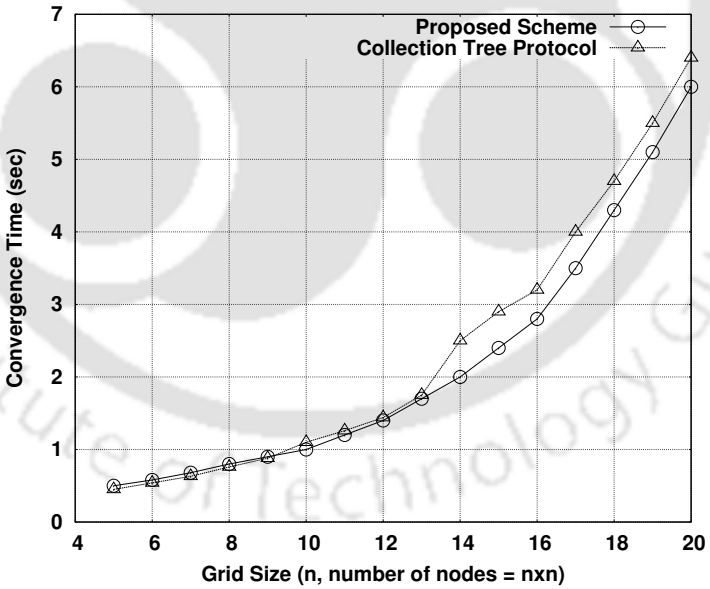

Figure 3.21: Convergence Time

and the average degree of all nodes in the network is δ, then the optimum timer value is $1.5 \times \delta \times \Omega$.

Figure 3.21 and Figure 3.22 compare the proposed scheme with respect to the

TH-1251_11610108

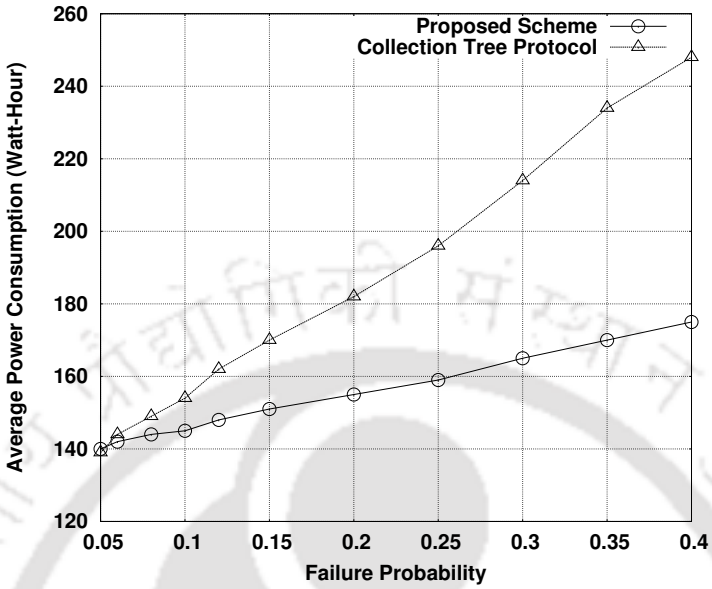

Figure 3.22: Average Power Consumption

collection tree protocol in terms of the convergence in tree construction and the average power consumption during the repairing. The CTP uses a source based tree construction mechanism using an adaptive beaconing where the beacon timer is adjusted based on the link characteristics. The convergence of CTP depends on the beacon timer. In Figure 3.21, the timer values have been taken as 0.4 sec. The figure shows that the convergence time of the proposed scheme is almost similar to that of the CTP when the grid size is small (less than 13). When the grid size increases, the proposed scheme performs better than the CTP in terms of the convergence time. For a large network, the adaptive beaconing mechanism in the CTP updates the network slowly to avoid the outdated link information by recalculating the link parameter (Expected Transmission Count). With an appropriate settings of the timer values, the convergence time for the tree construction can be made faster compared to the CTP, as shown in the figure.

Figure 3.22 compares the average power consumption in the network using the proposed scheme with that of the CTP. For this experiment, the average transmit, receive, and idle power for a sensor node is considered as 53 mW, 50 mW and 47 mW, similar to the standard micaZ sensor settings. Whenever a node fails, the CTP repairs the complete path from the source to the sink. This repairing requires involvement of all the nodes in the path, and therefore, more power is consumed, as reflected in Figure 3.22. As the failure probability increases, the adaptive beaconing of CTP broadcasts more number of beacons

TH-1251_11610108

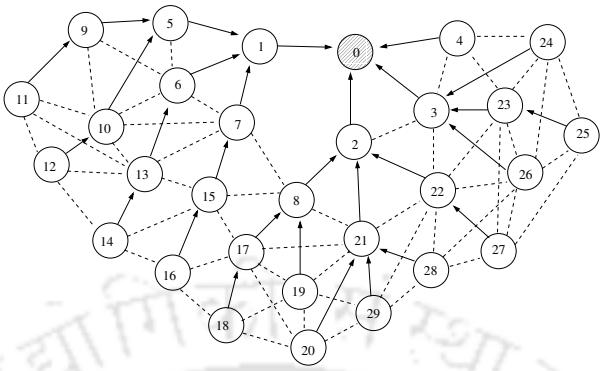

Figure 3.23: Initial BFS tree

for the tree repairing. On the other hand, the proposed scheme uses the precomputation method to switch the path towards the alternate parent. Therefore, the average power consumption in the network is considerably less for the proposed scheme compared to the CTP.

3.5.3 Phase III : Random Topology

The third phase of simulation has been done for a random topology with necessary connectivity. 30 nodes have been placed according to the Poisson distribution with mean 4-connectivity in an arena of 3415×100. The initial communication graph has been given in Figure 3.23 where node *0* is the root of the BFS tree. To analyze the effect of multiple node failures, set of nodes are made to be crashed simultaneously either in isolation or in a close vicinity. Total four sets of nodes are made to be crashed in a series until $\mathbb{G}$ becomes bi-connected. The experiment also includes the case where both the parent and alternate parent of a node fail simultaneously. Results have been given to show both the correctness of the tree repairing and its effect on data gathering.

Set I. At the time 250.005 sec, nodes *1*, *21* and *24* crash. This is an event of independent node crash in more than 2-hop neighborhood distance. Node *24* is a leaf node. So, it does not affect any other nodes. Where as, due to failure of the nodes *1* and *21*, all nodes in the subtree rooted at these nodes are affected. Nodes *5*, *6* and *7* lose their parent, and so, communicate to their corresponding alternate parent through a *ReqParent* message. All other nodes in the subtree rooted at the node *1* suspend outgoing application data flow on receiving a *holdFlow* message from the parent node. *altp(5) = 6*, *altp(6) = 7* and *altp(7) = 8*. So, finally node *7* receives a *ReqParentAck* message from a node *8*, and sets its parent to *8*. Now, nodes *15, 13, 10* and *9* also receive a *ReqParentAck* as nodes

TH-1251_11610108

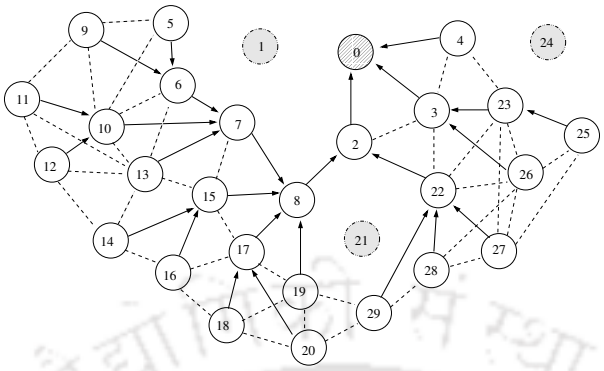

Figure 3.24: BFS tree after nodes *1, 21* and *24* crashed

7, 6 and *5* broadcast the same eventually. Node *15* changes its parent from the node *7* to the node *8* due to the level improvement. Nodes *9, 10* and *13* also change their parent from the nodes *5* and *6* to the nodes *6* and *7*, respectively. As nodes *9, 10, 13* and *15* set their parent correctly, they forward an *UpdateLevel* message towards their respective children. Finally, nodes *11* and *14* change their parent due to level improvement that make the local repairing complete. Each node then recomputes its alternate parent based on the updated neighborhood information. Similarly, due to failure of the node *21*, nodes *20, 29* and *28* become victim and reset their parent accordingly. As all of them are leaf nodes, no other nodes get affected due to failure of the node *21*. Final BFS tree after repairing has been given in Figure 3.24. The repairing time for all nodes due to failure of the first set of nodes have been given in Table 3.5.

Set II. At the time 350.005 sec nodes *10*, *17* and *22* crash. This is also an event of isolated node crash in more than 2-hop neighborhood distance, similar to the previous case. However, the failure of the nodes *21* and *22* occur in the same neighborhood, but in a series. Due to crash of the node *10*, its children node *11* and node *12* get victim, and set parent to the node *13* by exchanging a *ReqParent* and a *ReqParentAck* with their alternate parent. Node *18* and node *20* lose parent due to failure of the node *17*. Both of them set parent to the node *19* which was their alternate parent. Similarly, due to failure of the node *22*, its children node *27*, node *28* and node *29* get victim. Node *27* sets the parent to the node *23*, node *28* to the node *26*, and node *29* to the node *19*, based on the precomputed alternate path information. The final tree after repairing is shown in Figure 3.25. The repairing time for all nodes due to failure of the second set of nodes have been given in Table 3.6.

Set III. At the time 450.005 sec nodes *7* and *15* crash. The simultaneous failure of

TH-1251_11610108

Table 3.5: Repairing time(in sec) for the victim nodes due to failure of the Set I nodes

Node Id	Start-time	End-time	Δ
5	250.148	250.602	0.454
6	250.338	250.541	0.203
7	250.275	250.479	0.204
9	250.219	251.872	1.653
10	250.283	251.874	1.591
13	250.412	251.938	1.526
15	250.477	253.019	2.542
11	250.291	254.229	3.938
12	250.357	254.214	3.857
14	250.490	254.557	4.067
16	250.557	251.688	1.111
20	250.443	250.815	0.372
29	250.506	251.270	0.764
28	250.570	250.953	0.383

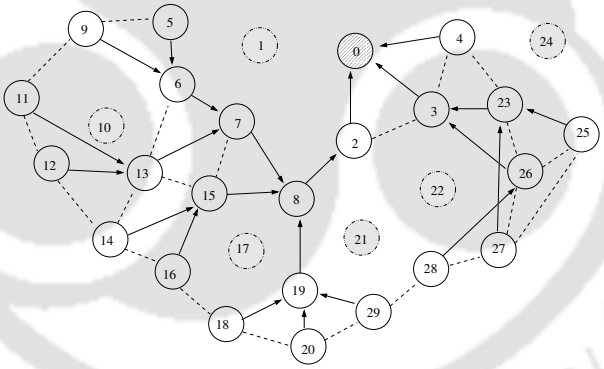

Figure 3.25: BFS tree after nodes *10, 17* and *22* crashed

these two nodes in 1-hop vicinity triggers the second phase of repairing which is reactive in nature. Both the parent and the alternate parent fail for node *13*. This in turn also affects the node *6*, as the node *13* was its alternate parent. Therefore, all nodes belong to the subtree rooted at the nodes *7* and *15* become victim. Node *13* initiates the repairing process by broadcasting an *Urgent* message in neighborhood. The *Urgent* message is forwarded through all victim nodes until is received by node *18*, whose none of the parent, alternate parent and grandparent belong to the set of victim nodes. Node *18* replies back with an *UrgentAck* message, and adds a node *16* in its *Child* set. *UrgentAck* message is

TH-1251_11610108

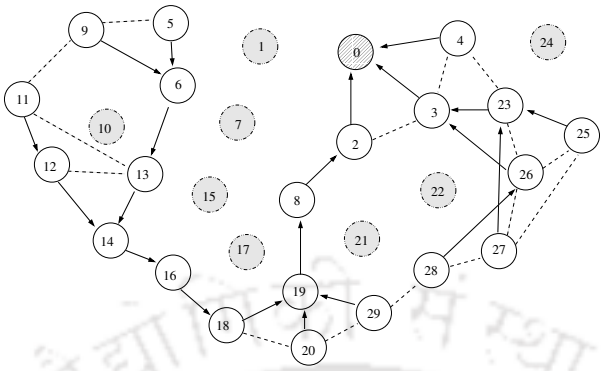

Figure 3.26: BFS tree after nodes 7 and 15 crashed

Table 3.6: Repairing time(in sec) for the victim nodes due to failure of the Set II nodes

Node Id	Start-time	End-time	Δ
11	350.350	350.610	0.26
12	350.414	350.674	0.26
18	350.301	350.496	0.195
20	350.364	350.559	0.195
27	350.502	350.885	0.383
28	350.375	350.827	0.452
29	350.438	350.623	0.185

forwarded through the victim nodes in reverse direction of the *Urgent* message. As the nodes receive the *UrgentAck*, they set their parents accordingly on first come first serve basis. After all victim nodes have set their parents correctly, the final tree looks like the one shown in Figure 3.26. The repairing time for each victim node has been reported in Table 3.7.

Set IV. Nodes *19* and *3*, that are separated by more than 2-hop distance, crash at the time 550.005 sec. However, the second phase of repairing is induced as an effect of previous node crashes into the graph. As node *3* fails, its children, nodes *23* and *26* set the parent to their respective alternate parent, the nodes *4* and *23*, respectively. Nodes *25*, *27* and *28* update their levels on receiving an *UpdateLevel* from the respective parent. Due to the failure of node *19*, nodes *20* and *29* set the parent to their respective alternate parent. However, *altp(18) = 16* is not any more correct due to the series of node failure. Node *16* has only one path to the root through the node *18*. Therefore, as node *18* sends

TH-1251_11610108

Table 3.7: Repairing time(in sec) for the victim nodes due to failure of the Set III nodes

Node Id	Start-time	End-time	Δ
6	450.084	450.985	0.901
13	450.211	450.923	0.712
14	450.291	450.673	0.382
16	450.354	450.551	0.197
5	450.145	451.034	0.889
9	450.209	451.097	0.888
11	450.286	451.046	0.760
12	450.350	450.860	0.510

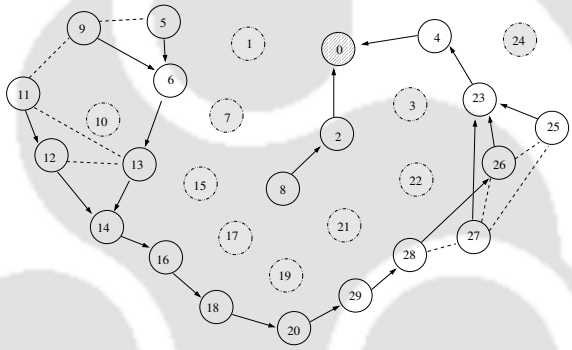

Figure 3.27: BFS tree after nodes *19* and *3* crashed

a *ReqParent* to the node *16*, node *16* broadcasts an *Urgent* message to search for an alternate path to the root according to lines 14-20 of Algorithm 3.8. As all nodes in the subtree rooted at the node *16* receives an *Urgent*, it becomes a victim as a consequence. Finally, when node *20* replies back with an *UrgentAck* message after setting its parent to the node *29*, node *18* sets its parent to the node *20*. All victim nodes eventually update their levels as *UrgentAck* is received. Figure 3.27 shows the final tree after repairing. The repairing time for all victim nodes has been presented in Table 3.8.

At this stage, the crash of any of the nodes *14, 16, 18, 20, 29, 28* would make the network partitioned. No local repairing scheme would be able to repair the tree in that case. It can be noted that the repairing time for victim nodes increases along a walk in the subtree rooted at the crashed node, from the root to the leaves. The *block* variable at each victim node suspends the outgoing application data flow during the repairing time. Once the nodes become stable, the *block* is reset to resume the outgoing application data

TH-1251_11610108

Table 3.8: Repairing time(in sec) for the victim nodes due to failure of the Set IV nodes

Node Id	Start-time	End-time	Δ
23	550.387	550.706	0.319
26	550.323	550.872	0.549
25	550.491	552.888	3.397
28	550.434	552.068	1.634
27	550.554	553.015	2.461
29	550.292	550.628	0.336
20	55.229	550.757	0.528
18	550.165	550.929	0.764
16	550.290	551.052	0.762
14	550.350	551.175	0.825
13	550.473	551.361	0.888
12	550.409	551.298	0.889
6	550.534	551.550	1.016
11	550.532	551.421	0.889
5	550.583	551.662	1.079
9	550.647	551.546	0.947

flow. The effect of node crash and the local tree repairing on application data has been shown in Figure 3.28. Result given in Figure 3.28 shows the count of application packets received at the sink node *0* that were sent from node *14* with respect to the the simulation time. Four different graphs have been plotted considering four sets of failures, as discussed before. It can be noted that no packet was delivered to the sink in redundant. The total number of packet loss is also very nominal. Only few packets, that were in the buffer of the failed node, are lost, and could not be recovered. In each of the cases, except the one in Figure 3.28(b), one such transit packet has been lost. The delay introduced in all the cases are due to the application data flow suspension during the repairing. Once the *block* is reset, all buffered application data are forwarded to the parent without any delay.

3.6 Summary

In this chapter, a set of distributed algorithms has been proposed to construct a BFS tree for data gathering. Each node computes its alternate path to the sink, based on the neighborhood information collected during tree construction. So when a node fails, all its

TH-1251_11610108

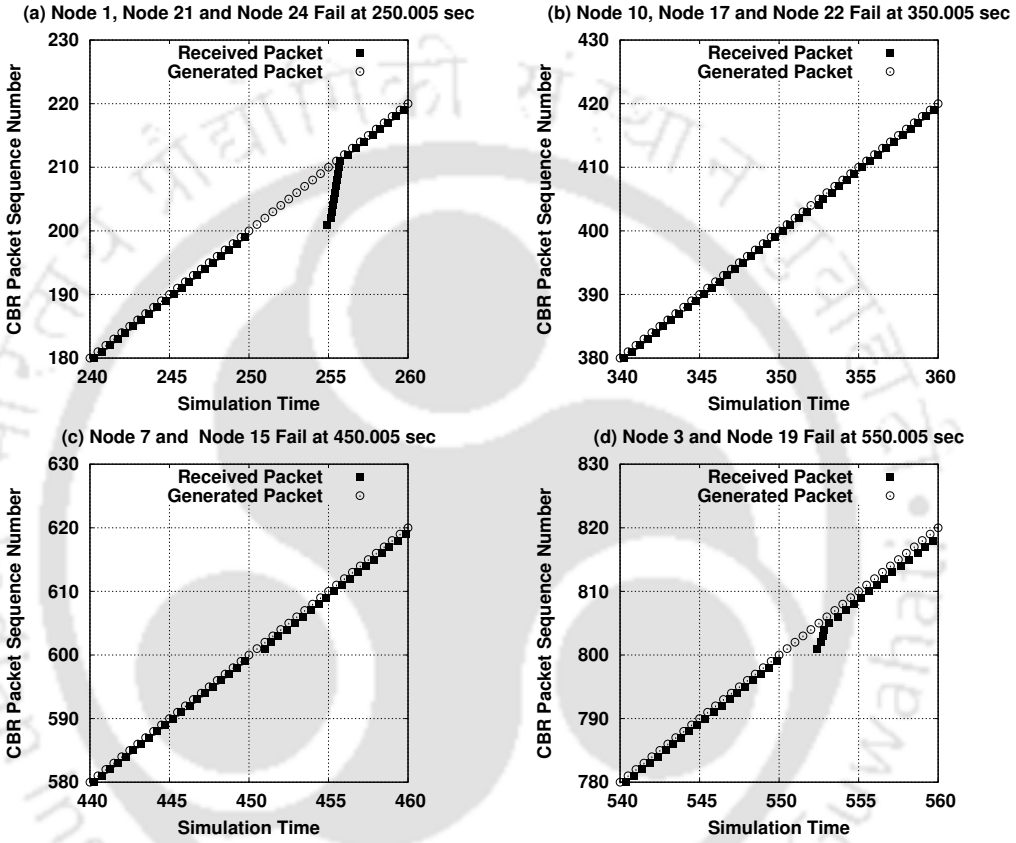

Figure 3.28: Application data received by the sink sent from node *14*

neighbors can take actions in a constant time to repair the tree locally. If both the parent and the alternate parent fail at a time, the proactive approach of repairing does not work. An extended repairing scheme has been proposed that works reactively to find the alternate path to the root in this adverse scenario. The proposed repairing scheme is fast and incurs low overhead in terms of the control message exchange. Low communication cost improves the QoS for the application and extends the effective network lifetime by saving the battery power of sensor nodes. Simulation results confirm the theoretical propositions with assurance of the correct delivery of the application messages without any redundancy.

TH-1251_11610108

The delay in delivery of application messages, introduced due to the local repairing, is also nominal. The scheme works for both the single and multiple node crashes as a series of events or the parallel events.

However, considering various application specific QoS requirements, as well as the limitation and overhead of data gathering tree maintenance, it is important to focus on the design of initial deployment strategy to ensure efficient data forwarding even in presence of arbitrary node failures. Therefore, to deal with all the design metrics like network connectivity, sensing coverage, fault-tolerance and network lifetime, a theory of efficient data gathering tree management scheme through redundancy based node deployment has been developed in following two chapters. A gradient based node deployment strategy has been proposed in next chapter. The framework of the data gathering tree management protocol based on the proposed deployment technique has been provided in subsequent chapter.

TH-1251_11610108

Chapter 4

Gradient Based Sensor Deployment

Data gathering and processing in a distributed environment is one of the most widely used applications for WSN. In-network data aggregation is a common practice to reduce traffic load and power consumption for the resource constrained sensor devices. However, in-network aggregation of sensory data is possible only when a high degree of correlation exists among the source data. Applications like collecting video images for a battlefield surveillance do not support data-aggregation because of low correlation, and so, all data packets are required to be forwarded to the sink. In data gathering applications that do not support data aggregation, the intermediate sensor nodes have to forward both its own sensory data and the relayed data from all of its children. In this case, the application data load increases gradually from the leaf nodes to the root of the tree as discussed in Chapter 1. For many delay-sensitive applications, the specific QoS constraints demand the sensor nodes to remain active throughout the lifetime. However, when applications permit, nodes are made to follow a periodic sleep schedule to save the critical battery power. A set of protocols like DMAC [118] and its variants [94] have been proposed in the literature on sleep based schedule management. In DMAC protocol, sensor nodes that neither transmit nor receive, go to sleep state to save energy as discussed in Chapter 2. The amount of sleep duration increases gradually from the sink to the leaf nodes of a data gathering tree where in-network data aggregation is not possible [205]. Thus, energy dissipation of the leaf nodes is significantly less compared to the nodes near the sink. This results in an early die-out of sensors near the sink, and affects both the connectivity and the sensing coverage in the network. On failure of a sensor node, the maintenance of the data gathering tree by

TH-1251_11610108

reconstructing the tree from the scratch is not at all cost-effective [16, 175]. Additionally, an involvement of all nodes in the maintenance activities introduces a global freeze, which in effect, degrades QoS for the application. Tree maintenance in reactive approach charges a significant cost in terms of control message communication as well as repairing delay. Proactive repairing with a low cost serves better as discussed in the previous chapter. However, considering an irregular topology, the proactive repairing also fails to perform well. Moreover, multiple simultaneous node failures in a close vicinity would be difficult to incorporate as this type of maintenance scheme increases per node load after repairing the tree on every node failure.

Redundancy in sensor deployment is an efficient method to ensure uninterrupted data delivery and improved network lifetime [162]. However, a proper estimation of redundancy is required based on the traffic load at different level of data gathering tree to ensure the complete network lifetime. Further, in real life, the area of interest or the terrain may be irregular as well as inaccessible in nature. Homogeneous deployment density would not be suitable for an irregular terrain as the deployed redundant nodes might not be able to serve the faulty node in this case [209]. Therefore, the initial deployment of sensor nodes plays a crucial role in prolonging the network lifetime while maintaining the connectivity and the coverage. The existing schemes in the literature [68, 76, 81, 109, 157, 200], do not consider all aspects of deployment design like the potential irregularity in terrain, gradient in traffic load distribution for tree based data gathering and the effect of node failures on connectivity and coverage of the network as discussed in Chapter 2.

In this chapter, a gradient based node deployment framework has been introduced considering both the cases of sensor energy dissipation model. In the *steady sensing* model, nodes remain active throughout the lifetime, and participate in data gathering activities all the time, until they die-out of energy or crash. In the *periodic sensing* model, nodes follow a periodic sleep-wakeup schedule for power saving and participate in data gathering activities only in an active state. Considering irregular terrain, the proposed deployment framework assumes that a sufficient number of redundant nodes are available for any failed node. Also, the gradient is calculated based on the energy dissipation metric considering both the cases of energy dissipation model. The amount of energy dissipation of a sensor depends on the number of nodes in its rooted subtree, when in-network data aggregation is not supported. This chapter proposes a model to estimate the number of nodes in the rooted subtree of an intermediate node in the data gathering tree, assuring the network connectivity and the sensing coverage during a node failure. Based on the estimation, the number of redundant nodes required to be placed is calculated. Let the boundary region is k-sensing covered,

TH-1251_11610108

out of which one node is in active state and the remaining are redundant. The proposed framework ensures the condition that the network can support at most $(k-1)$-failures at every point of the boundary region. To ensure this condition, an intermediate node in the data gathering tree should have at least $\rho \times k$ number of redundant nodes where $\rho > 0$ and ρ increases from the leaf nodes towards the sink. This chapter proposes a mathematical model to estimate the value of ρ based on the connectivity, coverage and the frequency of node failures due to energy dissipation. Applicability of the proposed mathematical model and the trade-off among the connectivity, coverage, fault-tolerance and the redundancy are justified through analysis using sensor network calculus [156]. Finally, the performance of the proposed gradient based sensor deployment framework is compared with the deployment frameworks proposed in [109] and [200] through the simulation results.

The rest of the chapter is organized as follows: Section 4.1 gives a brief description of the system model and assumptions for the proposed framework. Few concepts and definitions required to establish the proposed theory are provided in Section 4.2. Section 4.3 describes the mathematical analysis for the estimation of the proposed gradient based deployment density. The theoretical analysis through sensor network calculus is provided in Section 4.4. Finally, Section 4.5 presents the simulation results followed by a summary of the contribution in Section 4.6.

4.1 System Model and Assumptions

Let a large number of sensor nodes are deployed in the field of interest or terrain. Each sensor node is assumed to be identical in terms of transceiver power, memory and processing capacity. Let the communication range and the data sensing range of each node be denoted by the radius of R_c and R_s, respectively. The range of R_c can vary from R_s to $2R_s$. Initially it is assumed that $R_c = R_s$. Deployed nodes can be categorized into two types, a set of *primary nodes* and a set of *redundant nodes*. The primary nodes are assumed to construct a BFS tree, denoted by $\mathbb{T}$, rooted at the sink and then perform the tree based data gathering. For data gathering application, every intermediate node in the data gathering tree forwards the self-generated sensory data along with all received data from nodes in its rooted subtree to its parent. $\mathbb{T}_V$ and $\mathbb{T}_E$ denote the set of primary nodes and the set of edges between any two primary nodes, respectively. The set of primary nodes remain in active state all the time according to the *steady sensing* model of sensor energy dissipation. According to the *periodic sensing* model of sensor energy dissipation,

TH-1251_11610108

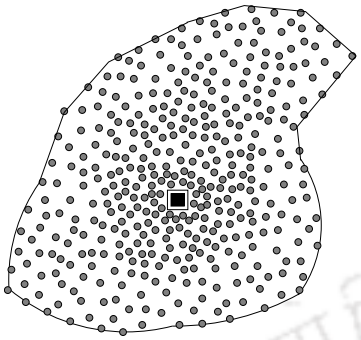

Figure 4.1: The terrain : sink and sensors

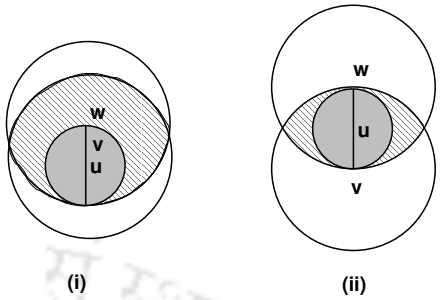

Figure 4.2: Proof for Theorem 4.1

the primary nodes follow a sleep-wakeup based schedule similar to DMAC as proposed in [118] for energy saving. Initially, the redundant nodes are set to sleep mode for saving energy. A node is said to be failed if it either dies out of energy or crashes suddenly due to hardware fault, disaster or some other reasons. On failure of a node, another node from the set of redundant nodes is set active on behalf of the failed node for maintaining the connectivity and the sensing coverage.

Definition 4.1. *The neighbor of a node u, denoted by $Neighbor(u)$, can be defined as the set of nodes such that for each node $v \in Neighbor(u)$, the condition $d_{u,v} \leq R_c(u)$ is satisfied, where $d_{u,v}$ is the euclidean distance between u and v.*

4.2 Deployment Strategy : Background

The area of interest for environment sensing (or terrain) can form any irregular polygon as shown in Figure 4.1. It is assumed that the sink is placed at an optimal position according to [144]. The *network lifetime* can be defined as the duration of time from the initial setup of the network till the network becomes partitioned or a network hole is created due to failure of a single node or a set of nodes. To achieve an improved lifetime, sensor nodes are deployed in high redundancy. On the completion of the deployment, the primary nodes are selected to construct a BFS tree[1].

Let the minimum and the maximum distance between two primary nodes be d^{min} and d^{MAX}, respectively, such that $0 < d^{min} < R_s/2$ and $d^{min} < d^{MAX} < R_c$. The value of d^{min} and d^{MAX} are chosen based on the application specific connectivity and coverage requirement. d^{min} and d^{MAX} are used to tune the trade-off among the connectivity,

[1]The primary node selection and tree construction procedure have been described in following chapter in detail.

TH-1251_11610108

coverage, fault-tolerance and the redundancy for application specific requirements, which would be discussed later through numeric analysis.

Definition 4.2. *The area of redundancy of a node u, denoted by $\Psi(u)$, can be defined as the area such that u is the only active node within $\Psi(u)$, and all other nodes inside the $\Psi(u)$ works as the dedicated redundant set for that area. If one active node inside the $\Psi(u)$ fails, another node from the set of redundant nodes replaces that faulty node.*

The area of redundancy for every $u \in \mathbb{T}_V$ is set to $\Psi(u) = \pi \left(d^{min}\right)^2$.

Definition 4.3. *Let u be a primary node. The set of dedicated redundant nodes for u, denoted by $\mathbb{R}(u)$, can be defined as a set of nodes such that $\mathbb{R}(u) = \{w | d_{u,w} \leq d^{min}\}$.*

To ensure that the area $\Psi(u)$ is always sensing-covered, and the network connectivity is maintained on failure of the primary node u and for all subsequent active nodes $v \in \mathbb{R}(u)$ that replaces the faulty node, $\mathbb{R}(v)$ is set to $\mathbb{R}(u)$.

Definition 4.4. *The area Λ_u is called sensing covered by a node u, where $\Lambda_u = \pi R_s^2$, and u is at the center of Λ_u.*

Initially, $\mathbb{A} = \bigcup_{\forall u \in \mathbb{T}_V} \Psi(u)$, where $\mathbb{A}$ denotes the total area of the terrain. If for any node u, $\mathbb{R}(u)$ is within the area of redundancy, $\Psi(u)$, then the following theorem holds true,

Theorem 4.1. *Let u be a primary node. Then for any two nodes w,$v \in \mathbb{R}(u)$, $\Psi(u) \subset \Lambda_w \cap \Lambda_v$.*

Proof. According to the initialization procedure, $d^{min} < R_s/2$. So,

$$w \in \mathbb{R}(u) \Rightarrow d_{u,w} < R_s/2$$
$$v \in \mathbb{R}(u) \Rightarrow d_{u,v} < R_s/2$$
$$w, v \in \mathbb{R}(u) \Rightarrow d_{w,v} < R_s$$

Now, $\Psi(u) < \pi(R_s/2)^2$ and $\Lambda_w = \Lambda_v = \pi R_s^2$. The different possible positioning for u, v and w are shown in Figure 4.2. As $d_{w,v} < R_s$, the circle centered at w will always be inscribed within the lens bounded by the circles centered at u and v. So, $\Psi(u) \subset \Lambda_w \cap \Lambda_v$ □

Theorem 4.2. *Let the cardinality of $\mathbb{R}(u)$ for a node u is i. Then on failure of node u, the network will remain connected as well as fully covered if $\mathbb{A} = \bigcup_{\forall u \in \mathbb{T}_V} \Psi(u)$ and $\mathbb{R}(u) \subseteq \Psi(u)$. This statement is true until i becomes 0.*

TH-1251_11610108

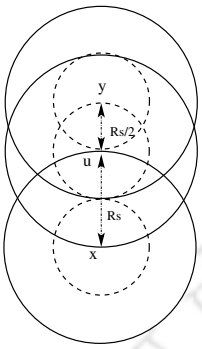

Figure 4.3: Coverage and connectivity under no node failure

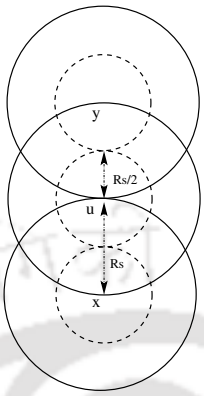

Figure 4.4: Parent and child of node u before failure

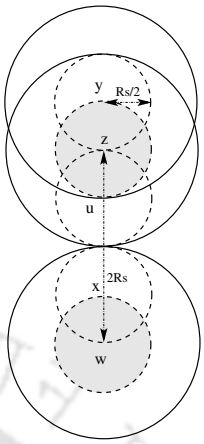

Figure 4.5: Node u and node x fails. Connectivity after recovery

Proof. Let $\mathbb{A} = \bigcup_{\forall u \in \mathbb{T}_V} \Psi(u)$, and $u \in \mathbb{T}_V$ such that $y = parent(u)$ and $x \in Child(u)$ as shown in Figure 4.3. Let $\mathbb{R}(u) \subseteq \Psi(u)$. Therefore, on failure of a node u, node $z \in \mathbb{R}(u)$ becomes active. In the worst case, from Theorem 4.1, z will be on the periphery of $\Psi(u)$, and so, $d_{z,u}^{MAX} = R_s/2$. If $d_{z,u} = R_s/2$ and $diam_{\Psi(u)} = R_s$, then, $\Psi(u) \cap \Lambda_z = \Psi(u)$, where $diam$ denotes the diameter of a circle. For $d_{z,u} < R_s/2$, $\Psi(u) \cap \Lambda_z = \Psi(u)$ is always true. Hence, $\Psi(u)$ will be sensing covered on failure of u for all $z \in \mathbb{R}(u)$ as shown in Figure 4.4.

Now, consider both the nodes u and x fail, and $z \in \mathbb{R}(u)$, $w \in \mathbb{R}(x)$ become active as shown in Figure 4.5. In the worst case, $d_{u,z}^{MAX} = R_s/2$ and $d_{x,w}^{MAX} = R_s/2$. Therefore,

$$\begin{aligned} d_{z,w}^{MAX} &= d_{u,x}^{MAX} + d_{u,z}^{MAX} + d_{x,w}^{MAX} \\ &= 2 \times R_s \end{aligned}$$

The connectivity between z and w is maintained if R_c is set to $2R_s$. Similarly, if the connectivity and sensing coverage are maintained even after a node failure, then it can be directly proved that the above condition holds true. Therefore, until $\mathbb{R}(u)$ becomes ϕ, the tree can be repaired/recovered maintaining both the connectivity and the sensing coverage. □

4.3 Estimation of Deployment Density

The density of deployment ρ, defined in Definition 4.5, dictates the level of tolerance for node failures.

TH-1251_11610108

Definition 4.5. *The density of deployment, denoted by ρ, is defined to be the number of nodes in the network (including both the redundant and primary), deployed per unit area of interest.*

However, the value of ρ is not homogeneous over the network. Sensor nodes are deployed in the terrain following a gradient based density distribution. As the nodes form a BFS tree rooted at the sink, all the leaf nodes are assumed to be placed at the terrain periphery. For a broadcast application, the traffic load is not dependent on the height of the tree as an intermediate node forwards the same copy of the received message to all its children. However, considering no in-network data aggregation, traffic load depends on the height of the data gathering tree. Each leaf node senses and forwards data to its parent independently, where as every intermediate node forwards the self generated data as well as all data received from its children. If an intermediate node u has two children, then as each of its children transmits one message to u, u receives two messages in total and forwards three messages (including the self generated) to the $parent(u)$. Thus, the traffic load increases from the leaves towards the root of the tree. The per node energy depletion is directly proportional to the per node traffic load. Let $\eta(l)$ be the average number of nodes in the subtree rooted at a node u at the level l of the BFS tree. The traffic load at the node u is directly proportional to $\eta(l)$. Let ξ be the average number of children of an intermediate node u. Then from the BFS tree properties [40], $\eta(l)$ can be recursively estimated as given in equation (4.1) where h is the maximum height of the tree. Solving equation (4.1), $\eta(l)$ can be expressed as shown in equation (4.2).

$$\eta(l) = \xi(\eta(l+1)+1) \tag{4.1}$$

$$\eta(l) = \frac{\xi(\xi^{h-l}-1)}{\xi-1} \tag{4.2}$$

4.3.1 Estimating Average Number of Children

The value of ξ needs to be estimated based on the connectivity and the coverage criteria. As shown in Figure 4.6, a primary node u is placed in position A. $\Psi(u)$ and Λ_u have been represented by two cocentric circles with radius d^{min} and R_s respectively. As $u \in \mathbb{T}_V$, its parent node v can be placed either in E (assuming $AE = d^{min}$) or in B (assuming $AB = d^{MAX}$) position. Let w be a child of node u. To maintain the BFS tree properties, $d_{v,w} > R_c$ must be true. Also, to maintain the connectivity, $d_{u,w} < R_c$ must be true.

Suppose $d_{u,v} = d^{min}$. Therefore, v is placed in E position. From Figure 4.6, $AH = ER = R_c = 2R_s$. Hence, w can be placed at any point within the region bounded by

TH-1251_11610108

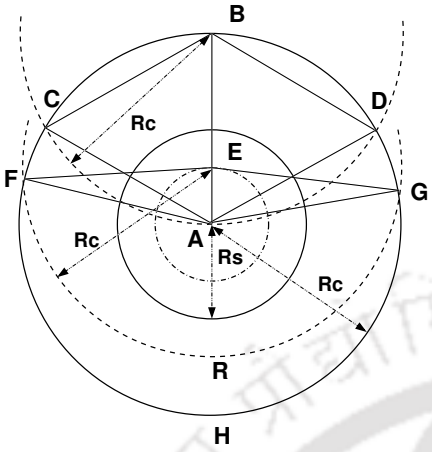

Figure 4.6: Avg no. of children

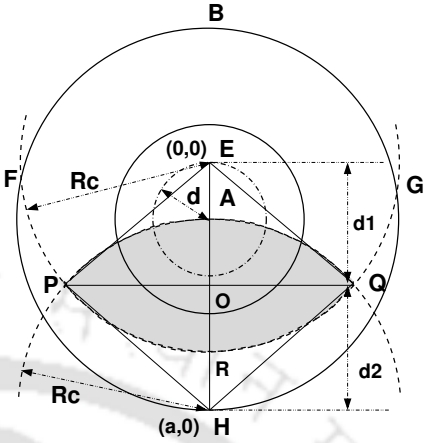

Figure 4.7: Avg no. of possible parents

$\widehat{F,R,G}$ and $\widehat{F,H,G}$ to maintain the connectivity. Let $\Upsilon_{\hat{x}}$ denote the fraction of perimeter is covered by the arc $\hat{x}$. From Figure 4.6, $AF = EF = 2R_s$ and $AE = d^{min}$. Therefore, from $\triangle AEF$,

$$\frac{AE}{\sin F} = \frac{EF}{\sin A} = \frac{AF}{\sin E} \tag{4.3}$$

$$\frac{\sin F}{\sin A} = \frac{d^{min}}{2R_s} \quad \text{;from (4.3)} \tag{4.4}$$

$$\sin A = \sin E \implies \angle A = \angle E \tag{4.5}$$

Now from $\triangle AEF$,

$$\angle A + \angle E + \angle F = 2\pi \tag{4.6}$$

$$\angle F = 2\pi - 2\angle A \quad \text{; from (4.5) and (4.6)}$$

$$\sin F = \sin(2\pi - 2A) = \sin 2A = 2\sin A \cos A \tag{4.7}$$

$$\frac{2\sin A\cos A}{\sin A} = \frac{d^{min}}{2R_s} \quad \text{; from (4.4) and (4.7)}$$

$$\angle A = \arccos\frac{d^{min}}{4R_s} \tag{4.8}$$

So, $\Upsilon_{\widehat{F,H,G}} = (2\pi - 2\angle EAF)/2\pi = (\pi - \arccos\frac{d^{min}}{4R_s})/\pi = \Upsilon_{min}$, say (as $\angle EAF = \angle EAG$ and $FAG = 2\angle EAF$).

Again, suppose $d_{u,v} = d^{MAX}$. Therefore, v is placed in B position. From Figure 4.6, $AH = AB = R_c = 2R_s$. Hence, w can be placed at any point within the

TH-1251_11610108

region bounded by $\widehat{C,A,D}$ and $\widehat{C,H,D}$ to maintain the connectivity. Similar to earlier, $\Upsilon_{\widehat{C,H,D}} = (2\pi - 2\angle CAB)/2\pi = (\pi - \arccos\frac{d^{MAX}}{4R_s})/\pi = \Upsilon_{MAX}$, say.

B and E are the two extreme positions of v. Node v can be placed at any point on the line segment BE, bounded by d^{min} to d^{MAX}. Let the variable μ denote the radius of an imaginary circle $\mathfrak{C}_{img}$ centered at u. v can be placed on the periphery of $\mathfrak{C}_{img}$. μ can be varied from d^{min} to d^{MAX}. Let c be a variable that denote the percentage of the perimeter of the circle $\mathfrak{C}_{img}$, where w can be placed. The value of c varies from Υ_{min} to Υ_{MAX}. As the minimum distance between two primary nodes is d^{min}, the number of maximum possible children that can be placed on the periphery of the circle $\mathfrak{C}_{img}$ is $c.2\pi\mu/d^{min}$. Let ξ_{MAX} denote the maximum number of children on average for an intermediate node. Then ξ_{MAX} can be estimated as follows:

$$\xi_{MAX} = \frac{1}{\Upsilon_{diff}} \int_{\Upsilon_{min}}^{\Upsilon_{MAX}} \frac{1}{D_{diff}} \int_{d^{min}}^{d^{MAX}} c\frac{2\pi\nu}{d^{min}} \, \mathrm{d}\nu \, \mathrm{d}c. \tag{4.9}$$

where $\Upsilon_{diff} = \Upsilon_{MAX} - \Upsilon_{min}$ and $D_{diff} = d^{MAX} - d^{min}$. Solving equation (4.9),

$$\xi_{MAX} = \frac{\{\pi(d^{MAX} + d^{min})(\Upsilon_{MAX} + \Upsilon_{min})\}}{2d^{min}} \tag{4.10}$$

Let $\Gamma(u)$ denote the communication disc of a node u. Above estimation assumes that all nodes that fall within the area $\Gamma(u) - \Gamma(v)$ are the children of u. So, ξ_{MAX} is the maximum number of children possible for an intermediate node. Now, as shown in Figure 4.7, for node u at position A, the parent of u is placed at position $E(0,0)$, where $AE = \iota$ and $d^{min} \leq \iota \leq d^{MAX}$. The child of node u, can be placed at any position within the region bounded by $\widehat{F,R,G}$ and $\widehat{F,H,G}$. Let w be a child of node u positioned at $H(a,0)$, where $a = 2R_s + \iota$. Therefore, all possible parents of the node w can be positioned within the lens bounded by the arcs $\widehat{P,A,Q}$ and $\widehat{P,R,Q}$. Let, $\mathcal{K}$ be a variable that denote the number of possible parent nodes for any node. From Figure 4.7, the equation of the circles, centered at $(0,0)$ and $(a,0)$, can be written as,

$$x^2 + y^2 = 4R_s^2 \tag{4.11}$$

$$(x-a)^2 + y^2 = 4R_s^2 \tag{4.12}$$

Solving equation (4.11) and equation (4.12),

TH-1251_11610108

$$x = \frac{1}{2} \times (2R_s + \iota) \tag{4.13}$$

$$y = \frac{1}{2} \times (2R_s - \iota) \tag{4.14}$$

The area of the lens bounded by the arcs $\widehat{P, A, Q}$ and $\widehat{P, R, Q}$ can be computed as

$$A(\iota) = 2 \times \left((2R_s)^2 \arccos\left(\frac{x}{2R_s}\right) - \frac{1}{2}xy \right) \tag{4.15}$$

Solving equation (4.15),

$$A(\iota) = 8R_s^2 \arccos\left(\frac{1}{2} + \frac{\iota}{4R_s}\right) - \frac{1}{4}(4R_s^2 - \iota^2) \tag{4.16}$$

Therefore, $\mathcal{K}$ can be computed as,

$$\mathcal{K} = \frac{1}{d^{MAX} - d^{min}} \int_{d^{min}}^{d^{MAX}} A(\iota) \mathrm{d}\iota \tag{4.17}$$

As the child node w can be uniformly placed in the region bounded by the arcs $\widehat{F, R, G}$ and $\widehat{F, H, G}$ for the parent of node u at position E, ξ can be computed as,

$$\xi = \xi_{MAX}/\mathcal{K} \tag{4.18}$$

4.3.2 Estimating Average Energy Dissipation Factor

The energy depletion of a node at level l is a function of $\eta(l)$, and can be defined by the Energy Dissipation Factor (EDF) as given in Definition 4.6.

Definition 4.6. *Let u be a node at level l of the BFS tree and $\mathbb{T}_u$ be the subtree rooted at u. w be a leaf node in $\mathbb{T}_u$. Energy Dissipation Factor (EDF) of node u measures the amount of energy dissipated by node u with respect to the node w in θ time interval, where $\theta \to 0$.*

Let, $\mathcal{E}_T$, $\mathcal{E}_R$, $\mathcal{E}_S$ and $\mathcal{E}_I$ denote the amount of energy dissipation by transmitting a single message, receiving a single message, staying in sleep state and staying in idle state respectively. EDF_{SS} denote the estimated EDF of a node according to the *steady sensing* energy dissipation model and EDF_{PS} denote the estimated EDF of a node according to the *periodic sensing* energy dissipation model. Considering different energy dissipation

TH-1251_11610108

model EDF at a node in level l can be expressed as shown in equation (4.19) and equation (4.20). However, for the sake of simplicity, EDF notation is used as a generic purpose.

$$EDF_{SS} = \frac{(\eta(l)+1)\mathcal{E}_T + \eta(l)\mathcal{E}_R}{2\eta(l)\mathcal{E}_I + \mathcal{E}_T} \tag{4.19}$$

$$EDF_{PS} = \frac{(\eta(l)+1)\mathcal{E}_T + \eta(l)\mathcal{E}_R}{2\eta(l)\mathcal{E}_S + \mathcal{E}_T} \tag{4.20}$$

4.3.3 Estimating the Gradient of Node Density

Let ρ_l denote the density of sensor nodes at the level l of the data gathering tree. An irregular terrain can be approximated as a cone rooted at the position of the sink node using a similar procedure as described in [67]. Let ς be the height of the cone. Then l can be approximated as $\frac{\varsigma - d_{\text{sink},o}}{(d^{MAX}+d^{min})/2}$, where o is a point in the terrain. The value of ρ_l can be characterized by the following Theorem,

Theorem 4.3. *If a point at the boundary of a terrain is k-covered, the value of ρ_l at the level l must be bounded by $EDF \times k/\{\pi \left(d^{min}\right)^2\}$ to maintain the connectivity and coverage.*

Proof. Let u be a primary node at the level l of the data gathering tree. Based on Theorem 4.1, to maintain the connectivity and the coverage, $EDF \times k$ number of nodes needs to be deployed inside the area $\Psi(u)$. So, the density of deployment at the level l is bounded by $EDF \times k/\Psi(u)$, which follows the Theorem. □

4.4 Theoretical Analysis

The basic idea behind the deployment of sensor nodes in an irregular terrain lies in the fact that the nodes closer to the sink are required to forward more traffic compared to the ones near the terrain periphery. As a result, considering tree based data gathering roted at the sink, nodes closer to the sink die out faster than the leaf nodes due to energy exhaustion. To improve the network lifetime satisfying the connected coverage criteria, a gradient based deployment framework has been proposed in this chapter. In the proposed framework, the EDF is taken as a metric for calculating the node density in a region. The EDF of a node at the level l is computed based on the estimation of the number of nodes in the subtree rooted at that node. However, the actual energy dissipation of a node depends directly on the network traffic condition. For example, if the network is highly

TH-1251_11610108

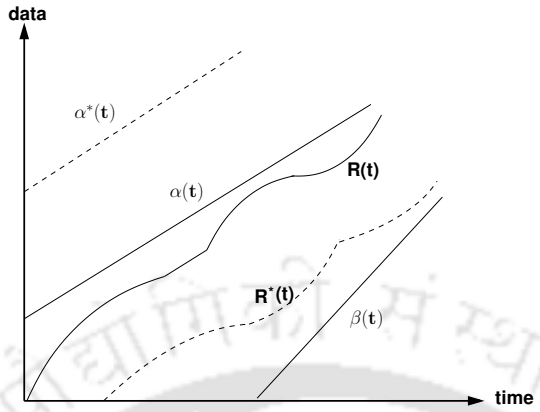

Figure 4.8: Example of Input Function, Arrival Curve, Output Function and Service Curve

loaded such that the traffic generated from the leaf nodes can saturate the network alone, the energy dissipation of all nodes in the network would be the same. Hence, all nodes, irrespective of the leaf nodes or the intermediate nodes, may die out within a short period of time. The worst case scenario occurs when the traffic generation rate at an individual node is moderately low. This is also the practical scenario. In this case, the nodes closer to the sink may die out significantly earlier compared to the leaf nodes, which may create the network holes, or may partition the network. One of the major objectives of this work is to deploy the nodes in such a way so that the network is not partitioned at any instance of time, and the leaf nodes would be able to forward data till they die out. By the worst case analysis of the sensor network calculus [89] over the proposed tree based forwarding methodology, it can be shown that the energy dissipation of a node compared to the leaf node is theoretically bounded by EDF of that node. Thus EDF can be effectively used as a metric for calculation of the gradient in the proposed sensor deployment framework.

The transmission links are assumed to be FIFO in the proposed mechanism. Let Q be a system. For a data flow through Q, $R(t)$ represents the cumulative number of bits that have arrived to system Q in the time interval $(0, t)$, and $R^*(t)$ represents the number of bits that have left the system Q in the same interval $(0, t)$. Both $R(t)$ and $R^*(t)$ are wide sense increasing, that is for $t_1 \geq t_2$, $R(t_1) \geq R(t_2)$, and $R^*(t_1) \geq R^*(t_2)$. $R(t)$ is called the input function and $R^*(t)$ is called the output function. Following definitions of basic network calculus terminologies have been provided for the sake of completeness.

Definition 4.7. *Arrival Curve $\alpha(t)$: Let $\alpha(t)$ be a wide sense increasing function for $t \geq 0$. Then for an incoming flow with input function $R(t)$, $\alpha(t)$ upper bounds $R(t)$ iff $\forall t_1, 0 \leq t_1 \leq t_2, R(t_2) - R(t_1) \leq \alpha(t_2 - t_1)$.*

TH-1251_11610108

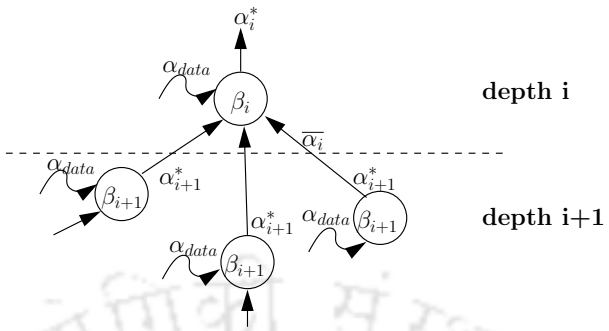

Figure 4.9: General data flow model with corresponding Arrival Curve, Service Curve and Output Bound

Definition 4.8. *Service Curve $\beta(t)$: Consider a flow through Q with input function as $R(t)$ and output function as $R^*(t)$. Then Q offers the traversing curve a service flow $\beta(t)$ iff $\beta(t)$ is a wide sense increasing function with $\beta(0) = 0$, and $\forall t, \exists t_0 \leq t$, such that $R^*(t) - R^*(t_0) \geq \beta(t - t_0)$.*

Definition 4.9. *Output Bound $\alpha^*(t)$: Assume that a flow with input function $R(t)$ and arrival curve $\alpha(t)$, traverse the system Q that offers the service curve $\beta(t)$. Then the output function $R^*(t)$ is upper bounded by the output bound $\alpha^*(t)$ as follows:*

$$\alpha^*(t) \geq (\alpha \odot \beta) \geq \alpha(t)$$

Where $\odot$ is the min-plus deconvolution that can be expressed as follows: for $f, g \in \mathbb{F}$ where $\mathbb{F}$ is the set of wide-sense increasing functions,

$$(f \odot g)(t) = \sup_{t_0 \geq 0} \{f(t + t_0) - g(t_0)\}$$

The arrival curve, the service curve and the output bound is shown in Figure 4.8. In case of the sensor network analysis, the arrival curve is assumed to be an affine arrival curve, which is defined as $\alpha(t) = b + r.t$, where b is the maximum number of bits that can arrive simultaneously at a given time to the sensor, and r is the data generation rate at every sensor. The service curve is assumed to be a rate-latency service curve, which is defined as $\beta(t) = R(t - \Theta)^+$, where $R \geq r$ is the channel bandwidth, Θ is the maximum latency of forwarding data, and $(x)^+$ denotes $\max\{0, x\}$. For an affine arrival curve and a rate-latency service curve, the output bound $\alpha^*(t)$ can be expressed as follows [89];

$$\alpha^*(t) = \alpha(t) \odot \beta(t) = \alpha(t) + r.\Theta \tag{4.21}$$

TH-1251_11610108

As described earlier, ξ denotes the average number of children for a sensor node. The general data flow model of a sensor network is shown in Figure 4.9 where $\xi = 3$. Let ξ_{leaf} denote the number of children that are the leaf nodes. Every node needs to forward its own sensory data, which is denoted as α_{data}. $\overline{\alpha_i}$ denotes the input curve, β_i denotes the service curve, and α_i^* denotes the output bound for a sensor at the level i. The arrival curves and the output bound is evaluated next using the network calculus methodology, in a hop-by-hop basis starting from the level h, where h is the maximum level of a sensor in the data gathering tree.

1. *At level h:*

   At the level h, all nodes are the leaf nodes. So, the output flow of each senor node at the level h is constrained by the output bound α^*_{data}, which can be computed using equation (4.21), as follows:

   $$\alpha^*_{data} = \alpha_{data} + r_{data}.\Theta_{leaf} \tag{4.22}$$

   Where r_{data} is the average data generation rate, and Θ_{leaf} is the maximum latency at the leaf nodes.

2. *At level h − 1:*

   At the level $h-1$, all the children are the leaf nodes. So the input curve $\overline{\alpha}_{h-1}$ can be computed as follows:

   $$\overline{\alpha}_{h-1} = \alpha_{data} + \xi.\alpha^*_{data} \tag{4.23}$$

   As a result, using equation (4.22),

   $$\overline{\alpha}_{h-1} = (1+\xi)\alpha_{data} + \xi.r_{data}.\Theta_{leaf} \tag{4.24}$$

   The total input flow, upper bounded by $\overline{\alpha}_{h-1}$, is forwarded by a sensor node at the level h to its parent at the level $h-1$, where the service rate is bounded by β_{h-1}. Hence, the output bound for a sensor at the level $h-1$ can be expressed as $\alpha^*_{h-1} = \overline{\alpha}_{h-1} \odot \beta_{h-1}$. Applying equation (4.21), this can be expressed as,

   $$\alpha^*_{h-1} = \overline{\alpha}_{h-1} + r_{data}.\Theta_{h-1} \tag{4.25}$$

3. At level $h-2$:

TH-1251_11610108

The total input flow of a sensor at the level $h-2$ comprises the output of the non-leaf nodes at the level $h-1$ plus the leaf nodes at the level $h-1$. So, the arrival curve can be expressed as,

$$\overline{\alpha}_{h-2} = \alpha_{data} + (\xi - \xi_{leaf}).\alpha^*_{h-1} + \xi_{leaf}.\alpha^*_{data} \tag{4.26}$$

Using equation (4.22), equation (4.24) and equation (4.25), this can be simplified as,

$$\overline{\alpha}_{h-2} = 1+\xi(1+\xi-\xi_{leaf})\alpha_{data}+r_{data}.\xi(\xi-\xi_{leaf})\Theta_{leaf}+(\xi-\xi_{leaf})\Theta_{h-1}+\xi_{leaf}.\Theta_{leaf} \tag{4.27}$$

The output bound constraining the output flow from a sensor at the level $h-2$ can be expressed as $\alpha^*_{h-2} = \overline{\alpha}_{h-2} \odot \beta_{h-2}$. As a result, using equation (4.21), this can be expressed as,

$$\alpha^*_{h-2} = \overline{\alpha}_{h-2} + r_{data}.\Theta_{h-2} \tag{4.28}$$

For the tree based data gathering, most of the children of an intermediate sensor would be the non-leaf nodes, and the leaf nodes can exist only at the boundary of the terrain. Therefore, for the simplicity, let for an intermediate node $\xi_{leaf} = 0$ and $\Theta_{leaf} = \Theta_h$. For such a node the input curve can be represented as,

$$\overline{\alpha}_{h-2} = (1 + \xi + \xi^2)\alpha_{data} + \xi.r_{data}(\xi.\Theta_h + \Theta_{h-1}) \tag{4.29}$$

Similarly, the output bound can be represented as,

$$\alpha^*_{h-2} = (1 + \xi + \xi^2)\alpha_{data} + r_{data}.(\xi^2.\Theta_h + \xi.\Theta_{h-1} + \Theta_{h-2}) \tag{4.30}$$

For all further analysis, only the intermediate nodes are considered, and the boundary effects are avoided to make the analysis simple.

4. *At level* $h-3$:

The total input flow for an intermediate sensor at the level $h-3$ can be represented as,

$$\overline{\alpha}_{h-3} = \alpha_{data} + \xi.\alpha^*_{h-2} \tag{4.31}$$

TH-1251_11610108

Using equation (4.30), this can be expressed as,

$$\overline{\alpha}_{h-3} = (1+\xi+\xi^2+\xi^3)\alpha_{data} + \xi.r_{data}.(\xi^2.\Theta_h + \xi.\Theta_{h-1} + \Theta_{h-2}) \tag{4.32}$$

Similarly, the output bound of an intermediate sensor node at the level $h-3$ can be represented as,

$$\alpha^*_{h-3} = (1+\xi+\xi^2+\xi^3)\alpha_{data} + r_{data}.(\xi^3.\Theta_h + \xi^2.\Theta_{h-1} + \xi.\Theta_{h-2} + \Theta_{h-3}) \tag{4.33}$$

5. *At level l*:

In general, the total input flow for an intermediate sensor at the level l can be represented as,

$$\overline{\alpha}_l = \left(\sum_{j=0}^{h-l} \xi^j\right) \alpha_{data} + \xi.r_{data}.\left(\sum_{j=0}^{h-l-1} \xi^j.\Theta_{(l+j+1)}\right) \tag{4.34}$$

Similarly, the output bound can be represented as,

$$\alpha^*_l = \left(\sum_{j=0}^{h-l} \xi^j\right) \alpha_{data} + r_{data}.\left(\sum_{j=0}^{h-l} \xi^j.\Theta_{l+j}\right) \tag{4.35}$$

The above analysis calculates the amount of input data forwarded to an intermediate sensor node at the level l as $(\overline{\alpha}_l)$ and amount of data forwarded from the same sensor node at the level $l+1$ as (α^*_l). Let $TxTime_l(r,t)$, $RxTime_l(r,t)$, $SleepTime_l(r,t)$ and $IdleTime_l(r,t)$ denote the total transmit time, total receive time, total sleep time and total idle time of a sensor node at the level l in the time duration $(0,t)$ with data generation rate r. Then, these values can be calculated as,

$$TxTime_l(r,t) = \frac{\alpha^*_l(t)}{R} \tag{4.36}$$

$$RxTime_l(r,t) = \frac{\overline{\alpha}_l(t)}{R} \tag{4.37}$$

$$SleepTime_l(r,t) = \begin{cases} t - TxTime_l(r,t) - RxTime_l(r,t), & \text{if } t > TxTime_l(r,t) + RxTime_l(r,t) \\ 0, & \text{Otherwise} \end{cases} \tag{4.38}$$

TH-1251_11610108

$$REDF_{SS} = \frac{TxTime_l(r,t).\mathcal{E}_T + RxTime_l(r,t).\mathcal{E}_R + IdleTime_l(r,t).\mathcal{E}_I}{TxTime_{leaf}(r,t).\mathcal{E}_T + (t - TxTime_{leaf}(r,t)).\mathcal{E}_I} \tag{4.41}$$

$$REDF_{PS} = \frac{TxTime_l(r,t).\mathcal{E}_T + RxTime_l(r,t).\mathcal{E}_R + SleepTime_l(r,t).\mathcal{E}_S}{TxTime_{leaf}(r,t).\mathcal{E}_T + (t - TxTime_{leaf}(r,t)).\mathcal{E}_S} \tag{4.42}$$

$$IdleTime_l(r,t) = \begin{cases} t - TxTime_l(r,t) - RxTime_l(r,t), & \text{if } t > TxTime_l(r,t) + RxTime_l(r,t) \\ 0, & \text{Otherwise} \end{cases} \tag{4.39}$$

Similarly, the transmit time for a leaf node can be calculated as,

$$TxTime_{leaf}(r,t) = \frac{\alpha^*_{data}(t)}{R} \tag{4.40}$$

Out of the total time t, a leaf node uses only $TxTime_{leaf}(r,t)$ time to forward the sensory data. So, the leaf node can go to sleep state for the time $(t - TxTime_{leaf}(r,t))$ to save critical battery power the *periodic sensing* energy dissipation model. However, considering the *steady sensing* energy dissipation model, the leaf node remains idle for the $(t - TxTime_{leaf}(r,t))$ time duration.

As discussed earlier, theoretically estimated EDF is a function of level, and hence, can be expressed as $EDF(l)$ as a generic term. However, in practice, the EDF value also depends on the actual data rate. So, EDF calculated from simulation is denoted by $REDF(l,r)$ (rate dependent EDF) as a generic term, wherever required. Let $REDF_{SS}$ and $REDF_{PS}$ denote the energy dissipation factor of a sensor node at the level l with data rate r, considering the *steady sensing* energy dissipation model and the *periodic sensing* energy dissipation model, respectively. Then $REDF_{SS}$ and $REDF_{PS}$ can be calculated using equation (4.41) and equation (4.42), respectively.

Theorem 4.4. *$REDF(l,r)$ is bounded above by $EDF(l)$.*

Proof. It can be noted that;

$$\sum_{j=0}^{h-l} \xi^j = \eta(l) - 1$$

where $\eta(l)$ is the average number of nodes in the subtree rooted at a sensor node at the level l. Replacing the value of $\overline{\alpha}_l(t)$, $\alpha^*_l(t)$ and $\alpha_{data}(t)$ in equation (4.36), equation (4.37),

TH-1251_11610108

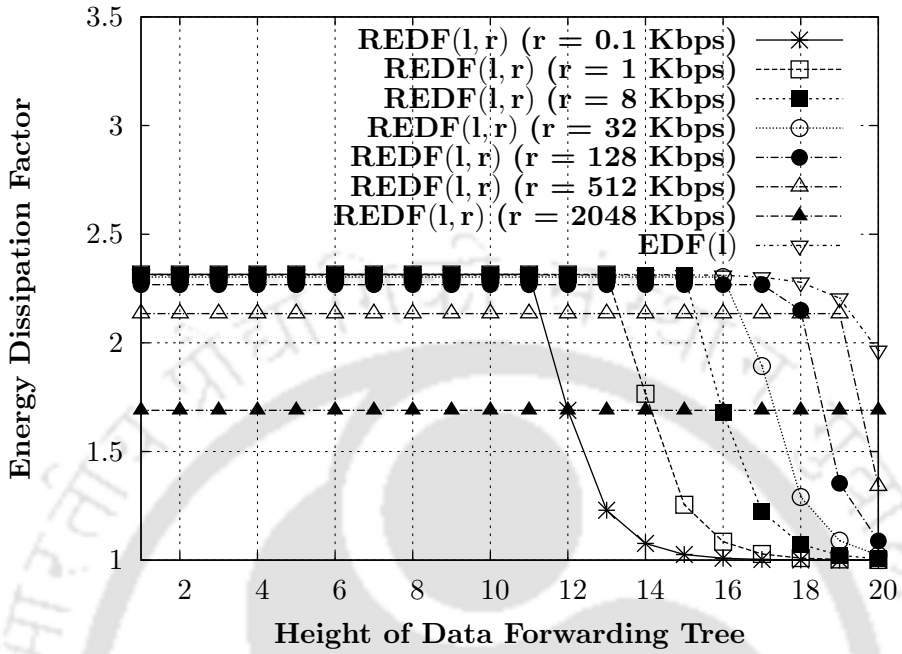

Figure 4.10: Comparison between $EDF(l)$ and $REDF(l,r)$ with different data generation rates

equation (4.40) and equation (4.41) (or equation (4.42), it can be shown directly that $EDF(l) \geq REDF(l,r); \forall r \geq 0$. □

In Figure 4.10, the values of $EDF(l)$ and $REDF(l,r)$ is plotted with respect to the levels of data gathering tree. The power consumption is taken according to MicaZ sensor mote specifications. For Willow MicaZ MPR2400 sensor motes [3], the power consumption for receive, transmit, sleep and idle mode are 0.053 watt, 0.047 watt, 0.001 watt and 0.0216 watt respectively. The $REDF(l,r)$ value is plotted for different data generation rates at the sensor nodes. Channel bandwidth is taken as 11 Mbps, and per-hop delay is assumed to be 0.01 seconds. It can be observed from the curve that the gradient effect becomes prominent when the data generation rate is low. With data generation rate as 2048 kbps, the leaf nodes alone saturate the network, and thus all nodes from the leaf nodes to the sink die out at the same time. At low data rates, the $REDF(l,r)$ follows the gradient pattern from the leaf nodes towards the sink. The data generation rates for the real life wireless sensor network is very low, and the practical sensor network rarely run at a saturation condition [115]. So, considering the gradient effect is necessary for wireless sensor network. It can be observed from Figure 4.10 that irrespective of data rates, the

TH-1251_11610108

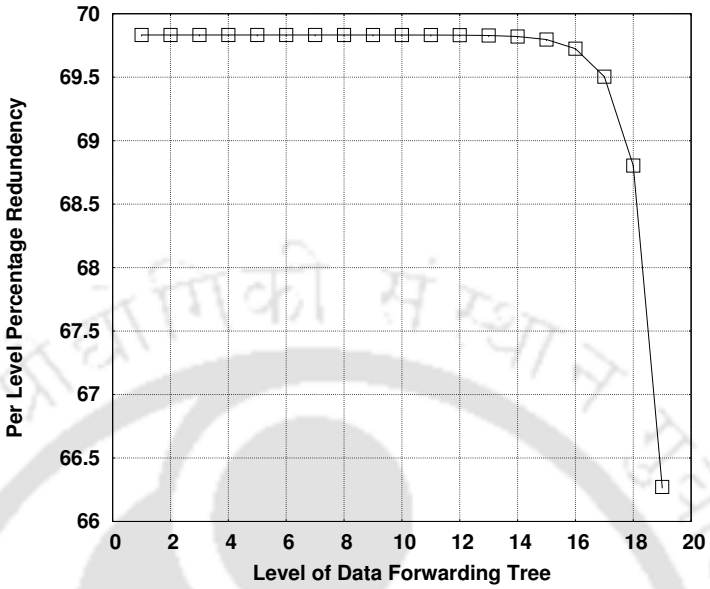

Figure 4.11: Per level percentage redundancy

$EDF(l)$ value upper bounds the $REDF(l, r)$ value. Thus EDF can be used as a valid metric for the the gradient based deployment of the sensor nodes.

4.4.1 Redundancy Analysis

Assuming a regular terrain that generates a balanced data gathering tree, the total number of nodes in the tree excluding the sink can be computed as, $\mathcal{N}_{\mathbb{T}} = \sum_{i=1}^{h} \xi^i$.

Total number of deployed nodes including redundancy $\mathcal{N} = \sum_{i=1}^{h} (\xi^i.EDF(i))$.

Per level percentage redundancy can be calculated as,

$$\Re_i = \frac{\xi^i.EDF(i)}{\xi^i.EDF(i) + \xi^i} \tag{4.43}$$

So, percentage redundancy in a terrain where maximum tree level is h, can be calculated as,

$$\overline{\Re} = \frac{\sum_{i=1}^{h} (\xi^i.EDF(i))}{\sum_{i=1}^{h} \xi^i + \sum_{i=1}^{h} (\xi^i.EDF(i))} \tag{4.44}$$

TH-1251_11610108

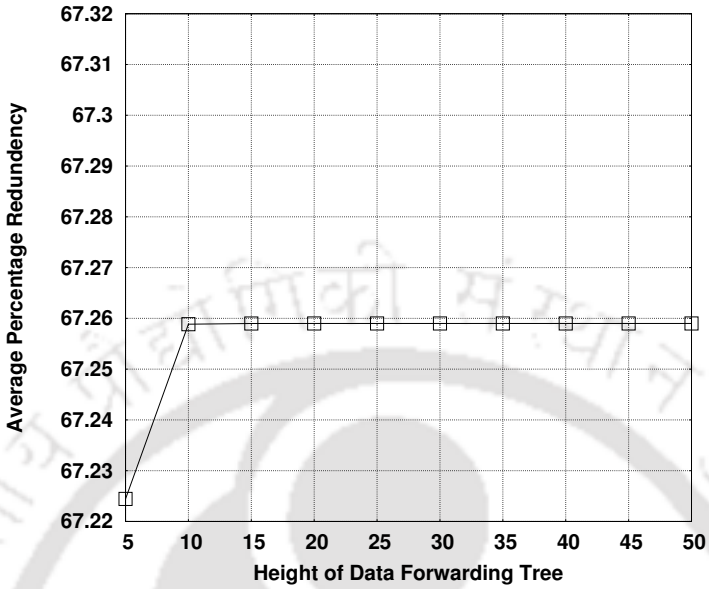

Figure 4.12: Average redundancy in the terrain

Estimated EDF: In Figure 4.11, the percentage redundancy is shown with respect to the level in the data gathering tree. Figure 4.12 shows the percentage redundancy in a regular terrain with respect to the maximum level of the data gathering tree. It can be seen from the figures, that though the percentage redundancy varies with respect to levels, irrespective of terrain size, the average percentage redundancy in a terrain is almost constant to 67.258%.

4.4.2 Probability of Sensing Coverage based on Deployment Strategy

Random deployment of sensor nodes can be considered as a Poisson process [127]. For such deployment, inter-sensor distance is exponentially distributed. According to the proposed deployment framework, the distance between two active sensors should be less than R_s. So, the probability of full coverage in a region with sensor density ρ can be represented as,

$$\mathscr{P}\{isCover(\mathbb{A})\} = 1 - e^{-R_s^2 \cdot \rho} \tag{4.45}$$

Where ρ can be computed according to Theorem 4.3. Let, X denote the random variable that indicates the number of redundant sensors deployed within the area of redundancy of a node. The probability that n number of redundant sensors will be present within the area of redundancy of any node can be given as,

TH-1251_11610108

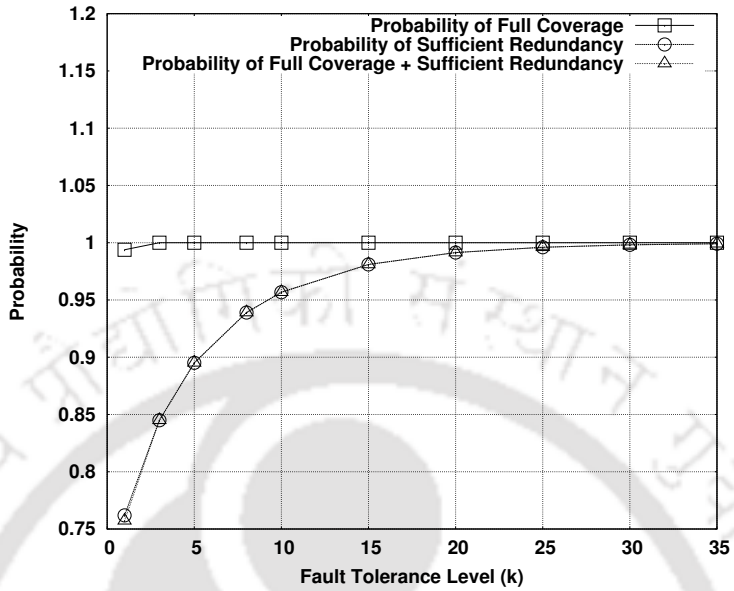

Figure 4.13: Probability of Full Coverage and Sufficient Redundancy

$$\mathscr{P}\{(X=n)\} = \frac{\left(\rho\pi\left(\frac{R_s}{2}\right)^2\right)^n e^{-\rho\pi\left(\frac{R_s}{2}\right)^2}}{n!} \tag{4.46}$$

Let the probability that every primary node has sufficient redundant nodes to serve (which is a function of EDF at that node) be represented by $\mathscr{P}\{suffRed(EDF)\}$. Combining equation (4.45) and equation (4.46), the probability that the desired area is covered and every sensor has at least $m \times EDF$ number of redundant sensors (where $m > 0$ is a constant), can be computed as,

$$\begin{aligned}\mathscr{P}\{isCover(\mathbb{A}) \wedge suffRed(EDF)\} &= \mathscr{P}\{isCover(\mathbb{A})\} \cap \mathscr{P}\{(X \geq m \times EDF)\} \\ &= \left(1 - e^{-R_s^2 \cdot \rho}\right)\left(1 - \sum_{i=0}^{m\times EDF-1} \frac{\left(\rho\pi\left(\frac{R_s}{2}\right)^2\right)^i e^{-\rho\pi\left(\frac{R_s}{2}\right)^2}}{i!}\right)\end{aligned} \tag{4.47}$$

Figure 4.13 shows the probability of full coverage and sufficient redundancy for the proposed gradient based deployment framework for different fault tolerance levels. EDF value is taken as 3. Sensing radius (R_s) is taken as 200 m. It can be seen from the figure that the probability of full coverage is very high. The overall probability is sufficiently large and for $m > 5$, the probability is more than 90%.

TH-1251_11610108

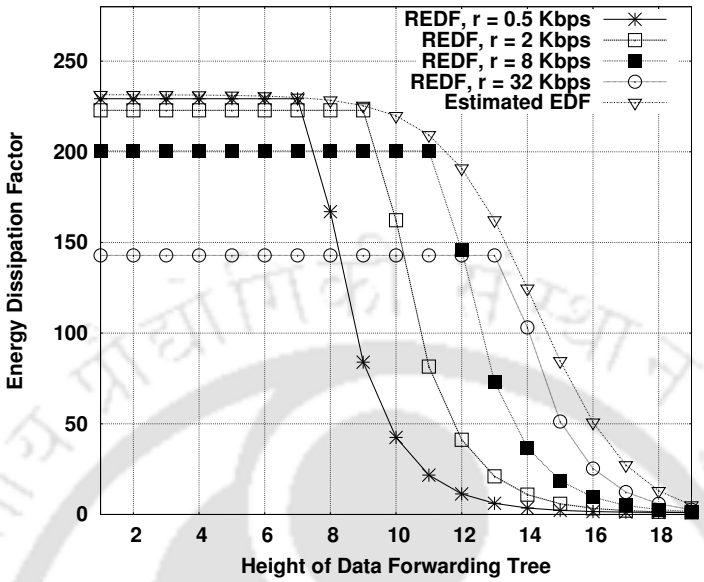

Figure 4.14: $d_{min} = R_s/4$ and $d_{MAX} = R_s/2$

From the theoretical analysis it can be derived that the estimated EDF, based on the number of nodes in the rooted subtree, can be considered to calculate the deployment density safely. The proposed gradient based deployment is also capable of ensuring full sensing coverage and sufficient redundancy to serve the failed nodes in most of the times. The performance of the proposed deployment framework has been compared with that of other deployment frameworks existing in the literature with respect to network lifetime, forwarding efficiency, coverage and connectivity support, etc. in following section.

4.5 Simulation Results

Standard MicaZ [3] sensor motes specifications are used for numeric and simulation analysis. The power consumption for receive, transmit, sleep and idle mode are 0.053 watt, 0.047 watt, 0.001 watt and 0.0216 watt respectively. The channel bandwidth and per-hop latency are assumed as 1 Mbps and 0.01 sec. respectively. For simulation analysis, the proposed deployment framework is implemented in Qualnet-5.0.1 [146] network simulation framework. Sensor nodes are deployed in a terrain of 1500×800 keeping the sink at the center. Deployment is done based on both the proposed gradient based deployment framework (with $d_{min} = R_s/2$ and $d_{MAX} = R_s$) as well as the deployment frameworks proposed by Liao *et al.* [109] and Yun *et al.* [200]. Sensing and communication radius are

TH-1251_11610108

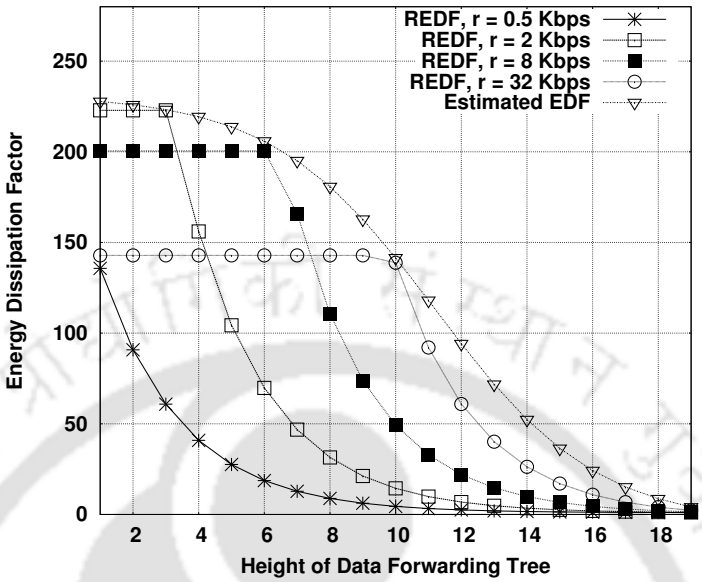

Figure 4.15: $d_{min} = R_s/2$ and $d_{MAX} = R_s$

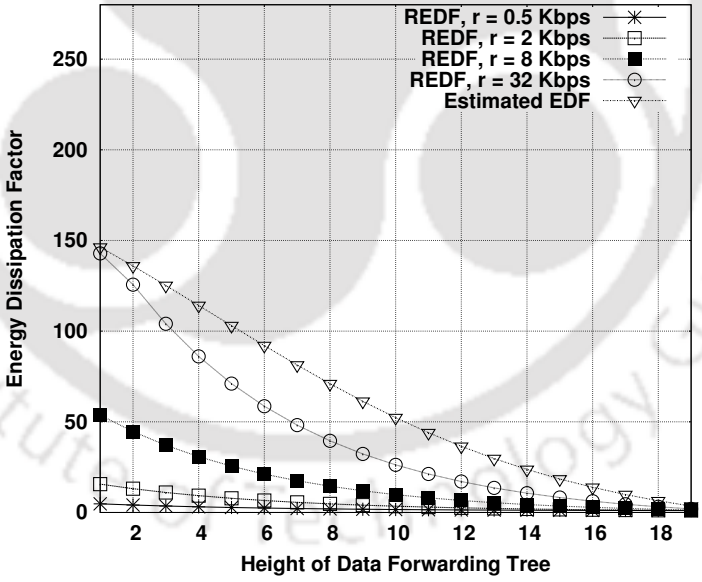

Figure 4.16: $d_{min} = R_s/2$ and $d_{MAX} = 3R_s/2$

taken as 125 m and 250 m respectively. To avoid the lengthy simulation process, the initial battery power of the sensor nodes is assumed to be 5 Joule. Sensory data is generated at every sensor node based on Weibull distribution. Considering *periodic sensing* model of energy dissipation, every sensor node supports DMAC protocol with *"more Data"* flag set

TH-1251_11610108

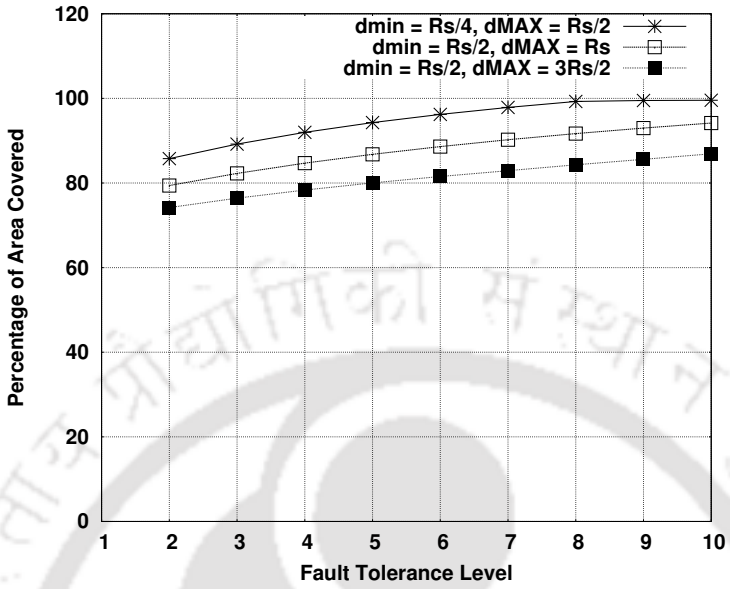

Figure 4.17: Percentage Coverage

to true, such that a node can continue in wakeup state if any one of its children has data to send.

Figure 4.14, Figure 4.15 and Figure 4.16 show the value of EDF from the theoretical analysis ($REDF$ in equation (4.42)) as well as based on the estimation (EDF in equation (4.20)). Three cases of d^{min} and d^{MAX} have been considered. First, it can be seen that for all the three cases, estimated EDF bounds the calculated $REDF$ based on the data rates. Second, as the values of d^{min}, d^{MAX}, and the difference between d^{MAX} and d^{min} increases, the gradient becomes more prominent. As the average distance between two active nodes increases, the number of children in the rooted sub-tree of a node decreases. So, the probability of saturation for the nodes near the sink decreases. Consequently, the difference between the sleep time of an intermediate node and that of a leaf node increases. The value of EDF depends on the difference of the sleep time between an intermediate node and a leaf node. As this difference increases, the gradient becomes more prominent.

Figure 4.17 shows the percentage coverage with respect to the fault-tolerance level k for different values of d^{min} and d^{MAX}. When d^{min} and d^{MAX} is low, the percentage coverage is high. Figure 4.18 shows the number of redundant nodes per unit active node with respect to the maximum height of the data gathering tree. When the height of the tree is more than 20, more number of redundant nodes are required for $d^{min} = R_s/4$ and

TH-1251_11610108

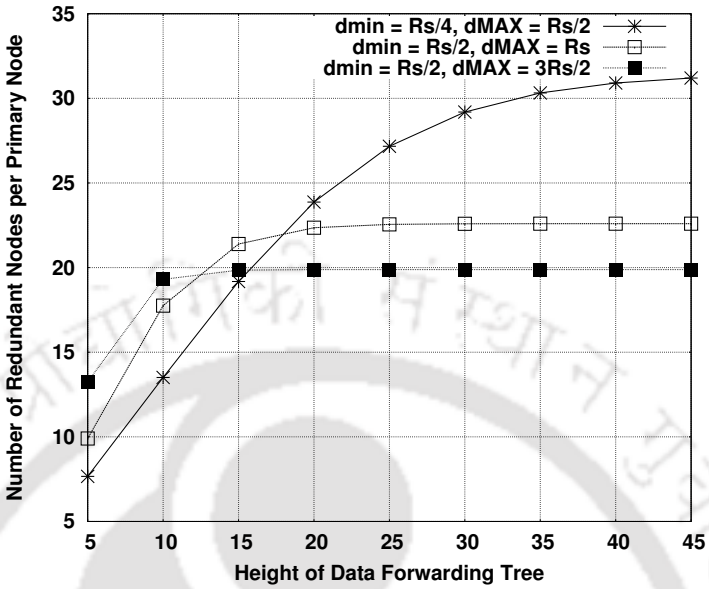

Figure 4.18: Redundancy Calculation

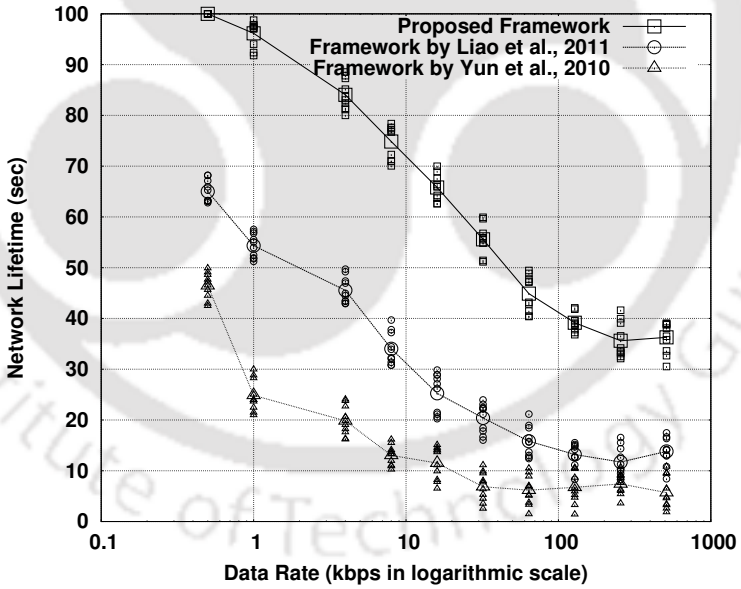

Figure 4.19: Network Lifetime Comparison

$d^{MAX} = R_s/2$. This shows the trade-off between the coverage and the redundancy. To support high coverage, more number of redundant nodes are required.

Figure 4.19 to Figure 4.22 compare the gradient based deployment framework with the framework proposed by Liao *et al.* for gradient based sensor deployment and Yun

TH-1251_11610108

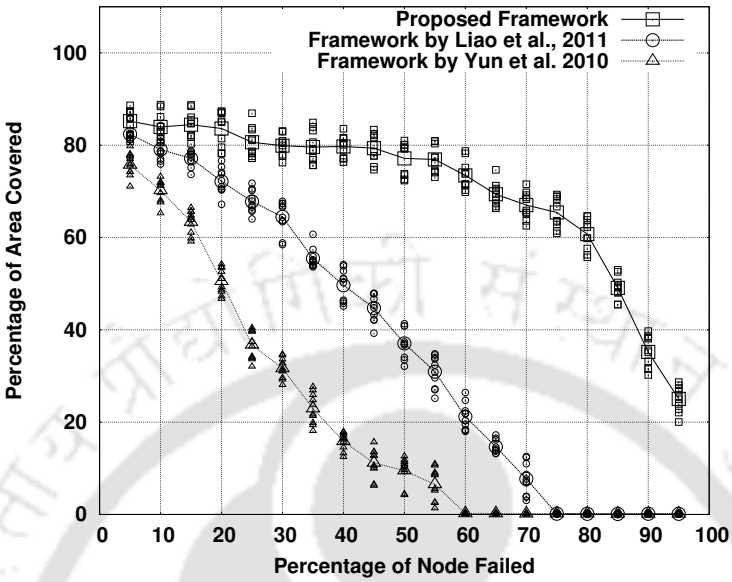

Figure 4.20: Coverage during Failure

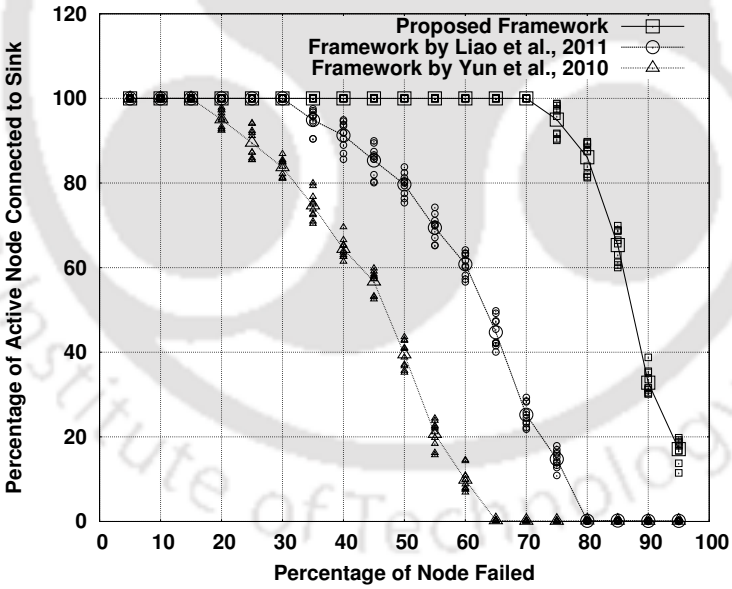

Figure 4.21: Connectivity during Failure

et al. for uniform sensor deployment. For these experiments, d^{min} and d^{MAX} for the proposed deployment framework is considered as $R_s/2$ and R_s respectively. Figure 4.19 compares the three frameworks with respect to network lifetime. The average sensor lifetime for the proposed framework is more than that of Liao *et al.* and Yun *et al.*.

TH-1251_11610108

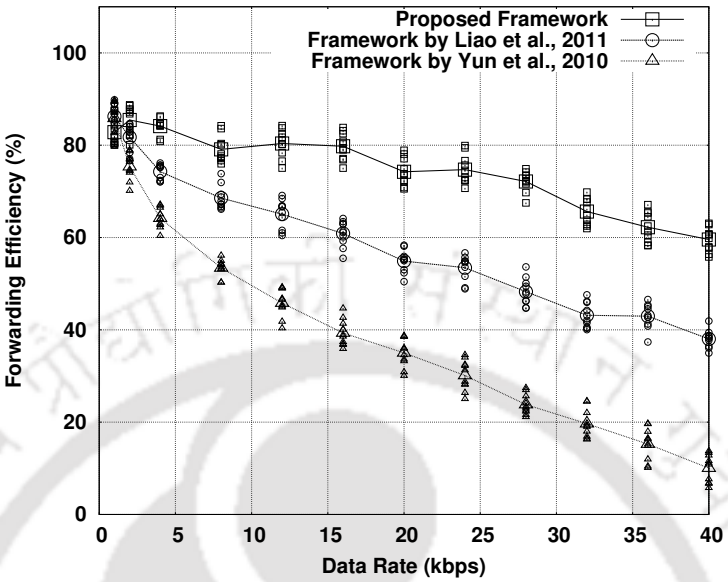

Figure 4.22: Forwarding Efficiency

Figure 4.20 and Figure 4.21 compare the two frameworks with respect to the connectivity and the coverage during node failure. The proposed framework can support up to 80% coverage when 50% of the deployed nodes (both primary and redundant) fail. Similarly, for the connectivity, full network connectivity is supported up to 70% of the node (both primary and redundant) failure. The proposed framework performs significantly better compared to the framework proposed by Liao *et al.* and Yun *et al.* with respect to the connectivity and the coverage. Figure 4.22 shows the forwarding efficiency (percentage of packets forwarded to the sink) for both the frameworks. The forwarding efficiency for uniform sensor deployment (Yun *et al.*) is less than the gradient based sensor deployment frameworks. However, the forwarding efficiency for the framework by Liao *et al.* is less compared to the proposed framework in spite of the gradient based deployment. This is because all the nodes remain in active state, and thus, the channel contention is more in the framework proposed by Liao *et al.*. This shows the requirement for keeping the extra nodes as redundant for future use.

4.6 Summary

In this chapter, a gradient based node deployment framework with redundancy has been proposed to assure the connected-coverage criteria on potential node failure. Considering

TH-1251_11610108

tree based data gathering in sensor network, according to the proposed framework, the deployment density decreases from the root towards the leaf nodes of the tree. The density of nodes at a particular level of the tree depends on the number of nodes in the subtree rooted at a node in that particular level. The estimated deployment density is based on the energy dissipation factor for a given node, that has been analyzed mathematically considering both the *steady sensing* and *periodic sensing* energy dissipation model of sensor nodes. The proposed theory is validated through an analysis of the sensor network calculus. The justification of the proposed estimation is also verified through the simulation results. A tree management scheme for fault-tolerant delay sensitive sensor network has been developed in the following chapter, where the initial node deployment is based on the theory proposed in this chapter.

TH-1251_11610108

Chapter 5

Data Gathering Tree Management through Gradient Based Deployment with Redundancy

Tree-based data gathering that does not support in-network data aggregation requires efficient management of data gathering tree. As stated in Chapter 4, for data collection using a spanning tree rooted at a sink, nodes near the sink are more failure prone than the leaf nodes. This is because, the intermediate sensor nodes are highly loaded as they have to forward the self-generated sensory data as well as the relayed data from all nodes in the rooted subtree. So, the energy depletion rate gradually decreases from the root to the leaf nodes. Any failure of an intermediate node in the data gathering tree affects both the connectivity and the sensing coverage in the network. To maintain the desired QoS for the running application, a fault-tolerant data gathering tree management scheme has been proposed in Chapter 3. The proposed scheme can repair the tree locally from any arbitrary node failure, single or multiple, either in a series or in parallel. However, the tree maintenance in reactive approach charges a significant cost in terms of control message communication as well as repairing delay, and the proactive repairing fails to perform properly considering an irregular terrain. For delay-sensitive applications in sensor network, it is strictly required that application data should be delivered to the sink without any loss or redundancy within a certain time bound even in presence of any change in the underlying topology due to arbitrary node failure. Moreover, multiple simultaneous node failures in vicinity would be difficult to incorporate as these maintenance schemes increase the per node load after repairing the tree on every node failure. A redundancy based node

TH-1251_11610108

deployment strategy has been proposed in Chapter 4, where the density of deployment decreases from the root of the tree towards the leaf nodes depending on the gradient effect of energy depletion in tree based data gathering. As stated in Chapter 4, the amount of required redundancy for any intermediate node is estimated by the number of nodes in its rooted subtree. Further, the amount of redundancy also depends on the nature of the data gathering tree. If the data gathering tree is unbalanced, even for the shortest path tree, the deviation in rate of energy dissipation between two nodes in neighborhood is more than that of a balanced tree. Uneven load distribution in a network leads to the failure of highly loaded nodes due to fast energy exhaustion, and naturally, a reduced network lifetime. This shows the requirement of designing a balanced data gathering tree for an efficient measurement of redundancy required during the initial node deployment.

In this work, a fault-tolerant tree-based data gathering scheme for delay-sensitive applications of sensor network has been proposed. For efficient utilization of sensor battery power, a load-balanced BFS tree rooted at the sink has been constructed. Considering the requirement of raw data collection for several sensor network applications, the gradient-based node deployment strategy, proposed in Chapter 4, has been used. The failure of nodes have been handled by incorporating a local tree maintenance scheme that assures both the connectivity and the sensing-coverage even after the repairing. QoS for the data gathering application has also been assured. The key contributions of the work have been summarized below.

1. Considering the irregular terrain, a gradient based node deployment strategy, proposed in Chapter 4, has been used for initial deployment of the sensors.

2. Initial network is set up on the selection of primary set of nodes. The proposed selection algorithm aims to offer full sensing coverage in the terrain.

3. A data gathering tree construction scheme has been proposed that converges to a balanced BFS tree based on neighborhood information processing and incremental level improvement.

4. A tree maintenance module has been designed that can repair the tree from node failure assuring both the connectivity and the sensing-coverage in the network after the repairing.

5. An application messages handler has been designed that controls the application message flow, and thus, offers a reliable delivery of the application messages to the sink.

TH-1251_11610108

6. Finally, the performance of the proposed scheme has been analyzed through the simulation results, and compared with other existing schemes related to this.

The rest of the Chapter is organized as follows:

Section 5.1, that includes the description of the system model and assumptions, provides the initial set-up requirements. The proposed data gathering tree management scheme has been described in two phases. Section 5.2 presents the first phase of tree management, where the process of primary nodes selection, a load-balanced BFS tree construction and the post-deployment activities have been described. The second phase of tree management includes the proposed tree maintenance scheme from arbitrary node failure as well as the workings of AMC, which are provided in Section 5.3 in detail. Section 5.4 presents the simulation results for the performance analysis of the proposed scheme. Finally, Section 5.5 concludes the Chapter.

5.1 System Model and Assumptions

Before proposing the data gathering tree construction and maintenance scheme in detail, the general system model and few assumptions on which the scheme is developed are required to be stated. Let a large number of sensor nodes are deployed in the field of interest or terrain. Each sensor node is assumed to be identical in terms of transceiver power, memory and processing capacity. Let the communication range and the sensing range of each node be denoted by the radius of R_c and R_s respectively. The range of R_c can vary from R_s to $2R_s$. Initially, it is assumed that $R_c = R_s$. Considering the failure-prone nature of sensor nodes, a set of nodes are kept as redundant that remains in sleep state and saves critical battery power. The primary set of nodes forms a data gathering tree such that both the connectivity and the sensing coverage can be assured. Two sensor nodes u and v can communicate to each other and called neighbors if $d_{u,v} \leq R_c$, where $d_{u,v}$ is the Euclidean distance between the nodes u and v. V represents the set of all nodes in the network. It is assumed that there exists a sink for collecting the sensory data from all nodes in the network. For efficient routing of data, a BFS tree, denoted by $\mathbb{T}$, and rooted at the sink, is used. $V_{\mathbb{T}}$ and $E_{\mathbb{T}}$ denote the set of primary nodes and the set of edges between any two primary nodes respectively. Each node is assumed to have a sufficiently large FIFO buffer for storing and forwarding application messages. The network is assumed to be static, but, any arbitrary node may leave the network due to failure. A node is said to be failed if it either dies out of energy or crashes suddenly due to hardware fault, disaster or some other reasons. Once a primary node u in the tree fails, a redundant node w from

TH-1251_11610108

Table 5.1: State variables at each node u

Variable Name	Type	Initial Value	Description
$rflag$	Boolean	False	set to True at all redundant nodes
$tflag$	Boolean	False	set to True on receiving *Token* message
$addflag$	Boolean	False	set to True on sending *AddMe* message
$posflag$	Boolean	False	set to True on receiving *PosAdd* message
$block$	Boolean	False	set to True to stop data forwarding from buffer
$level(u)$	Integer	-1	level of node u in the BFS tree
$bFactor$	Integer	0	half of the $\|Child(parent(u))\|$ that are not allowed to change parent
$tempLevel$	Integer	0	level of the node to be set as parent on expiry of $ImproveTimer$
$counter$	Integer	0	keeps the track of nodes from which received *AddMe* and can be accommodated in Child set
$parent(u)$	Node variable	-1	parent of node u in the BFS tree
$tempId$	Node variable	-1	ID of the node to set as parent on expiry of $ImproveTimer$
$posId$	Node variable	-1	ID of the node to set as parent on receiving *RemAck*
$desId$	Node variable	-1	ID of the designated node to serve that has failed
$Child(u)$	Node set	Null	children of node u in the BFS tree
$serveList$	Node set	Null	all designated nodes in neighbor to serve on failure
$altpSet(u)$	Node set	Null	$\forall v \in Neighbor(u)$ such that $level(v) = level(parent(u))$
$altpLevel(u)_w$	Map	Null	the pair (w,$level(w)$ $\forall w \in altpSet(u)$)
$desLevel$	Map	Null	the pair (w,$level(w)$ $\forall w \in serveList(u)$)
$desParent$	Map	Null	the pair (w,$parent(w)$ $\forall w \in serveList(u)$)

$\mathbb{R}(u)$ is set to active for maintaining the connectivity and the sensing coverage. The nodes are assumed to follow the *steady sensing* model of energy depletion, and hence, remain in active state until they fail. The area of interest for environment sensing (or terrain) can form any irregular polygon as shown in Figure 4.1. It is assumed that the sink is placed to an optimal position as stated in [144]. Initial sensor deployment is performed according

TH-1251_11610108

to the framework proposed in Chapter 4 with the corresponding estimated redundancy. On the completion of deployment, the primary set of nodes are selected to construct a load-balanced BFS tree. The primary node selection and tree construction procedures have been described in the following section in detail.

The proposed scheme of fault-tolerant data gathering tree management can be modeled as a module that works within the network layer. It has two submodules, a Tree Management Module (TMM) and an Application Message Controller (AMC) as shown in Fig. 3.1. The TMM is responsible for load-balanced BFS tree construction and maintenance where as the purpose of the AMC is to handle application messages. The TMM and the AMC, at every node, interact with each other so that the cooperative interaction among all the sensor nodes may lead to a successful implementation of the proposed scheme. The proposed fault-tolerant tree management scheme has been discussed in two phases. In the first phase, the primary node selection procedure and the detail of the distributed load-balanced BFS tree construction algorithms have been described. The second phase, where the tree maintenance scheme and working of AMC have been discussed, is given in the next section. Along with the initial system set-up including the sink and node deployment, the data gathering tree construction is also performed offline. Data gathering activities are initiated after the load-balanced BFS tree is fully constructed. Therefore, the tree maintenance activities of the TMM and the activities of the AMC begin only after the successful completion of the first phase of tree management. In a distributed environment, the activities of the TMM and the AMC have been designed by a set of message-passing algorithms as given in Algorithms 5.1 to 5.13. Different variables used in the algorithms have been summarized in Table 5.1. It is assumed that the intra-node communication (between the TMM and the AMC) is performed by signal exchange where as inter-node communication (between two nodes) is performed by an exchange of control messages. The term, broadcast of a message, whenever used in this chapter, indicates that the scope of the broadcast is confined within a 1-hop neighborhood only.

5.2 First Phase of Tree Management

On completion of the initial deployment as described in Chapter 4, all nodes in the network including the redundant ones remain in active state. First, the set of primary nodes is selected, which then participates in load-balanced BFS tree construction.

TH-1251_11610108

Algorithm 5.1 On receiving *Bcast* from v by u

1. **if** $RSSI \leq \mathcal{SS}_{R_s/2}$ **then**
2. $\quad rflag \leftarrow True$
3. $\quad serveList \leftarrow serveList \cup v$
4. **else if** $\mathcal{SS}_{3R_s/4} < RSSI \leq \mathcal{SS}_{R_s}$ **then**
5. $\quad$ wait for τ
6. **end if**
7. **if** τ expires and $rflag = False$ **then**
8. $\quad$ broadcast *Bcast*
9. **end if**

5.2.1 Active Nodes Selection Procedure

The primary nodes selection procedure is distributed in nature, and initiated by the sink. First, the sink forwards a *Bcast* message to all nodes in $Neighbor(sink)$. On receiving a *Bcast* message, a node decides locally whether to participate in the tree construction or to become a designated redundant node, depending on the received signal strength index (RSSI). The *Bcast* messages are propagated from the sink towards the periphery of the terrain such that every node in the network becomes either a primary tree member or a redundant. On receiving a *Bcast* message from a node v, node u compares the $RSSI$ with the $\mathcal{SS}_d$, where $\mathcal{SS}_d$ denotes the strength of the received signal from d distance. If $RSSI < \mathcal{SS}_{R_s/2}$, u becomes a redundant node designated to replace the node v on failure as stated in lines 1-3 of Algorithm 5.1. If u receives a *Bcast* message with $RSSI < \mathcal{SS}_{R_s/2}$ from another node w, u inserts w in its *serveList*. Thus, a redundant node can be designated to serve multiple nodes. If $RSSI > \mathcal{SS}_{(R_s/2+R_s/4)}$, but $RSSI < \mathcal{SS}_{R_s}$, u waits a random amount of time τ. If no more *Bcast* messages with $RSSI < \mathcal{SS}_{R_s/2}$ are received within τ, u declares itself as a primary node and forwards the *Bcast* in turn to all its neighbors. A tolerance of $R_s/4$ has been introduced for computing the $\mathbb{R}(u)$. All redundant nodes with $rflag$ set to True do not participate in the tree construction procedure. Algorithm 5.1 describes the steps a node performs on receiving a *Bcast* message. To handle the signal fluctuations, a moving average of the measured $RSSI$ value is considered that provides a satisfactory approximation of the measured signal strength, as shown in [14]. It can be noted that although the density of deployed nodes (all nodes in the terrain including the redundant ones) follows a gradient based on node's energy dissipation, the density of selected primary nodes is uniform throughout the terrain.

TH-1251_11610108

5.2.2 Load-Balanced BFS Tree Construction

Several algorithms for distributed BFS tree construction are available in the literature [25, 88]. Also, one scheme is provided in Chapter 3. However, none of these works can be directly fitted into the proposed scheme due to the absence of cost-effective load-balancing behavior of the data gathering tree. To the best of our knowledge, no distributed application specific load-balanced BFS tree construction scheme exists in the literature. The proposed tree construction procedure has two objectives, first, to maintain the BFS properties, and second, to satisfy the load-balancing criteria. Once the set of primary nodes and their corresponding set of redundant nodes are selected, the tree construction procedure is initiated by the sink. The tree is constructed by an incremental development as the control message wave propagates from the root to the leaf nodes of the tree.

Initialization :

Only the primary set of nodes participate in the tree construction procedure. First, the sink forwards a *Token* message with *level* 0 to all its neighbors. All primary nodes that receive the *Token*, set their *level* to 1 and *parent* to the sink. Each node in *level* 1 forwards the *Token* message to all its neighbor, in turn. The *Token* message carries the parent variable of the sending node along with its level. On receiving a *Token* message for the first time, node u sets its parent and level accordingly, and forwards the *Token* in its neighborhood as stated in Algorithm 5.2. On receiving a *Token* message from a neighbor v, such that v has set the node u as its parent, u includes v in its *Child* set by an implicit acknowledgment. Moreover, each node performs neighborhood information processing locally for future decision making as stated in the next paragraph.

Local Computation: Node w can be an alternate parent for a node u, if $level(w) = level(v')$ where $v' = parent(u)$. If required, choosing w as a parent would not increase the level of u. So, on receiving a *Token* from node w with the level equal to the $level(parent(u))$, node u stores the node ID and the level of w in its *altpSet* and *altpLevel* set respectively as a reference to its alternate parent.

Along with the initialization of the parent variable and the *Child* set, a node may perform some other activities to fulfill two objectives as mentioned before. To satisfy the BFS properties, every node in the tree must maintain the shortest distance in hop-count to the root. Similarly, to satisfy the load-balancing criteria, the number of children should be equally distributed between every two neighboring nodes at the same level. In a distributed

TH-1251_11610108

Algorithm 5.2 On receiving *Token(l, pid)* from v by u

```
1.  if tflag = False then
2.      tflag ← True                                   /*Token received for the first time*/
3.      level(u) ← l + 1
4.      parent(u) ← v
5.      broadcast Token(level(u), parent(u))
6.  else
7.      if l + 1 = level(u) then
8.          altpSet(u) ← altpSet(u) ∪ {v}              /*add entry for alternate parent*/
9.          altpLevel(u) ← (v, l)
10.     else if l + 1 < level(u) then
11.         tempLevel ← l + 1          /*parent change required due to level improvement*/
12.         tempId ← v
13.         reset ImproveTimer
14.     end if
15.     if u = pid then
16.         Child(u) ← Child(u) ∪ {v}
17.         if |Child(u)| > 1 then
18.             broadcast ChldUpdate(|Child(u)|)       /*for load-balancing*/
19.         end if
20.     end if
21. end if
```

Algorithm 5.3 On timeout of *ImproveTimer* by u

```
1. level ← tempLevel
2. parent(u) ← tempId
3. broadcast Update(level, parent(u), |Child(u)|)
```

and asynchronous message-passing environment, the initial parent and *Child* might be set up in such a way that the constructed tree satisfies neither the BFS properties, nor the load-balancing criteria. Thus, according to the proposed incremental tree construction scheme, a node is allowed to change its initially set up parent variable to fulfill the stated objectives. On receiving a better *Token* message (A *Token* message is called *better Token* if it carries the level information less than that of its parent variable), if there is a scope of level improvement, a node changes its parent to satisfy the BFS properties. Again, on receiving a *Token* from a child, a node triggers the parent changing events of its children to satisfy the load-balancing criteria. As both of these events of parent changing can be triggered on receiving a *Token* message, both the events may occur in parallel. Therefore,

TH-1251_11610108

Algorithm 5.4 On receiving *Update(l,pid,csize)* from v by u

```
1.  if l + 1 < level(u) then
2.      level(u) ← l + 1
3.      if ∃w ∈ altpSet(u)|altpLevel(u)_w > level(u) − 1 then
4.          remove entry for w from altpSet and altpLevel      /*remove stale alternate parent
            entry*/
5.      end if
6.      if v ≠ parent(u) then
7.          parent(u) ← v
8.          broadcast Update(level(u), parent(u), |Child(u)|)
9.      end if
10. else
11.     if pid = u and v ∉ Child(u) then
12.         Child(u) ← Child(u) ∪ {v}
13.         if |Child(u)| > 1 then
14.             broadcast ChldUpdate(|Child(u)|)                /*for load-balancing*/
15.         end if
16.     else if v ∈ Child(u) and pid ≠ u then
17.         Child(u) ← Child(u) \ {v}      /*received from a child whose parent has changed*/
18.     end if
19.     if v ≠ parent(u) and l + 1 = level(u) then
20.         altpSet(u) ← altpSet(u) ∪ v                         /*add entry for alternate parent*/
21.         altpLevel(u) ← (v, l)
22.         if addflag = False and bFactor > csize then
23.             send AddMe to v                                 /*for load-balancing*/
24.             addflag ← True                                  /*no more AddMe can be sent*/
25.         end if
26.     end if
27. end if
```

simultaneous occurrence of the parent changing events, for both the level improvement and the load-balancing, in close neighborhood may interfere with each other.

Level Improvement:

On receiving a better *Token*, node u does not update its *level* and *parent* information spontaneously. If the node is allowed to update its *parent* and *level* each time it receives a better *Token* message, the overall message overhead will increase due to a cascade effect in its rooted subtree. So, on receiving a better *Token* message, node u waits for a random

TH-1251_11610108

amount of time set by the *ImproveTimer*. On expiry of the timer, u updates its *parent* and *level* information accordingly, and forwards an *Update* message with the updated information to all its neighbors. On receiving an *Update* message from the parent node with an improved level, node u updates its level value as stated in lines 1-2 of Algorithm 5.4. As the $level(u)$ gets decreased, it may happen that there exists a node $w \in altpSet(u)$ that does not satisfy the condition $level(w) = level(parent(u))$, anymore. Therefore, u removes the entry of w from its *altpSet* and *altpLevel*. If the source of the received *Update* message with an improved level is not the $parent(u)$, u changes its parent, and then, broadcasts the updated information through an *Update* message. On receiving an *Update*, if the received level is not the improved one, u executes lines 10-27 of Algorithm 5.4 depending on the conditions. If the source of the message, node v had set its parent to u such that $pid = u$, and $v \notin Child(u)$, then u adds v in its *Child* set. If $v \in Child(u)$, but $parent(v) \neq u$, u removes v from its *Child* set as stated in lines 16-18 of Algorithm 5.4. If $v \neq parent(u)$ and $level(v) = level(parent(u))$, u adds an entry for v in both the *altpSet* and the *altLevel* according to lines 19-21 of Algorithm 5.4.

Proof of Correctness : Properties of the constructed tree have been established with the following lemmas and theorems.

Lemma 5.1. *For every primary node u, if $parent(u) = v$, then $d_{v,s} = min\{d_{w,s}, \forall w \in Neighbor(u)\}$, where s denotes the sink.*

Proof. Let for a primary node u, $parent(u) = v$. Therefore, u has received a *Token* message from the node v according to Algorithm 5.2. Let $\exists w \in Neighbor(u)$ such that $level(w) < level(v)$. Now, as node w has established a path to the sink via its parent, it has updated the information to all its neighbors. During the tree construction, a node updates its neighborhood by either broadcasting a *Token* (on receiving a *Token* for the first time) or broadcasting an *Update* message (on receiving a *Token* with an improved level) according to Algorithms 5.2 and 5.3. In any case, node $u \in Neighbor(w)$ receives either a *Token* or an *Update* message from node w. If u receives a *Token* from w, it changes the parent from v to w according to lines 10-13 of Algorithm 5.2 and Algorithm 5.3. Similarly, if u receives an *Update* from w, it changes its parent from v to w according to lines 1-9 of Algorithm 5.4. So, in any case u changes its parent from v to w. Hence, if $w = parent(u)$, $level(w)$ is the minimum among all the nodes in $Neighbor(u)$. □

Lemma 5.2. *For any pair of nodes u, v in the tree, the path $\mathbb{P}_{(u,v)} = \{u, x_1, x_2, ..., x_i, ..., x_n, v\}$ from u to the node v will not make any cycle.*

TH-1251_11610108

Proof. Let there exists a path $\mathbb{P}_{(u,v)} = \{u, x_1, x_2, ..., x_n, v\}$ in the tree from u to v such that $u = parent(x_1), x_1 = parent(x_2), ..., x_n = parent(v)$. Now to form a cycle, following two conditions must be satisfied.

1. $u \in Neighbor(v)$
2. $v = parent(u)$

From the assumption, node v is a descendant of node u in its rooted subtree. So, $level(u) < level(v)$, and also, $level(u) < level(x_n)$. Let condition 1 is true, as $level(u) < level(parent(v))$, node v will set $parent(v) = u$ according to Lemma 5.1. So condition 2 fails. Hence, no cycle can be formed in the tree. □

Theorem 5.1. *The proposed scheme produces a correct BFS tree eventually.*

Proof. The proof follows directly from the Lemmas 5.1 and 5.2. □

Theorem 5.2. *The cost of the proposed BFS tree construction algorithm is* $O(|E_{\mathbb{T}}|)$.

Proof. On receiving a *Token* for the first time, every node broadcast the *Token* message at least once from Algorithm 5.2. Due to the level improvement a node may change its parent, which requires the broadcast of an *Update* message. In the worst case, every node in the tree changes its parent once. Therefore, every node in the tree broadcast two messages. Let, δ_v denote the degree of a node v. One broadcast from a single node v costs δ_v units of message transmissions. Thus, for the whole network, considering every node broadcast two messages in the worst case, the communication cost for the BFS tree construction is $\sum_{v \in V_{\mathbb{T}}} 2 \times \delta_v = 4 \times E_{\mathbb{T}}$, which is essentially $O(E_{\mathbb{T}}|)$. □

Load Balancing :

A node adds another node in its *Child* set on receiving either a *Token* message or an *Update* message as stated in Algorithm 5.2 and Algorithm 5.4. Each time while adding a node in Child set, node u checks the cardinality of $Child(u)$. If the number of children for node u becomes greater than 1, u forwards a *ChldUpdate* message announcing its child size to all its neighbors. On receiving a *ChldUpdate* message, depending on certain conditions, a node decides whether to change its parent for the sake of load balancing or not. Let there exist two nodes u and v at the same level of the tree. Also, let *pChild* be a set as defined in Definition 5.1.

TH-1251_11610108

Algorithm 5.5 On receiving *ChldUpdate(csize)* from v by u

1. **if** $v = parent(u)$ **then**
2. **if** $csize > bFactor$ **then**
3. $bFactor \leftarrow csize$ /*updates bFactor*/
4. **end if**
5. **if** $\exists w \in altpSet(u)$ and $addflag = False$ **then**
6. send AddMe to w, $\forall w \in altpSet(u)$ /*received from parent, alternate parent exists*/
7. $addflag \leftarrow True$
8. **end if**
9. **else if** $v \in altpSet(u)$ and $addflag = False$ and $bFactor + 1 > csize$ **then**
10. send AddMe to v /*received from alternate parent*/
11. $addflag \leftarrow True$ /*no more AddMe can be sent*/
12. **end if**

Algorithm 5.6 On receiving *AddMe* from v by u

1. **if** $|Child(u)| + counter < \xi$ **then**
2. $counter \leftarrow counter + 1$
3. send $PosAdd(|Child(u)| + counter)$ to v /*adding child is permissible*/
4. **end if**

Algorithm 5.7 On receiving *PosAdd(csize)* from v by u

1. **if** $addflag = True$ and $(csize - \lceil (bFactor/2) \rceil) < 1$ and $posflag = False$ **then**
2. $posflag \leftarrow True$
3. $posId \leftarrow v$
4. send *Rem(csize)* to $parent(u)$ /*request parent to leave the Child set for load balancing*/
5. **end if**

Definition 5.1. *Let u be a primary node at the level l of the BFS tree. The set of all nodes in $Neighbor(u)$ are said to be the potential children of u, denoted as $pChild(u)$, such that $\forall w \in pChild(u)$, $level(w) = level(u) + 1$.*

Now, if following conditions are satisfied then there is a requirement of load-balancing such that the overall load is equally distributed between u and v.

Condition A $pChild(u) \cap pChild(v) \neq \phi$

Condition B $MOD\,(|Child(u)| - |Child(v)|) > 1$, where $MOD(a) = a$ if $a \geq 0$, or $MOD(a) = -a$ if $a < 0$

TH-1251_11610108

Algorithm 5.8 On receiving *Rem(csize)* from v by u

1. **if** $|Child(u)| > csize$ **then**
2. $Child(u) \leftarrow Child(u) \setminus \{v\}$
3. send *RemAck* to v /*leaving of the child is admissible*/
4. **if** $|Child(u)| < \xi$ **then**
5. broadcasts *ChldUpdate*($|Child(u)|$) /*announces the availability in Child set*/
6. **end if**
7. **end if**

Algorithm 5.9 On receiving *RemAck* from v by u

1. $parent(u) \leftarrow posId$
2. $level(u) \leftarrow altpLevel(u)_{posId} + 1$
3. remove entry for *posId* from *altpSet* and *altpLevel*
4. broadcasts *Update(level(u),parent(u),|Child(u)|)*

Condition C If $|Child(u)| \geq \xi$, then $|Child(v)| < \xi$ or vice-versa, where ξ is the average number of children for any intermediate node in the BFS tree from equation (4.18).

The basic idea of load-balancing is based on following propositions.

- Condition A and Condition C are necessary to reach the balanced state. That means if Condition A or Condition C fails, Condition B alone does not suffice to trigger the initiation of the load-balancing event.
- As this phase of tree construction is performed offline, once a node decides to change its parent for load-balancing, it is not allowed to trigger the occurrence of further parent change events to avoid the ping-pong effect in the neighborhood. Once the tree is load-balanced, it remains so till the end. The second phase does not affect the stability of the tree.
- A node u is called to be in balanced state if $\forall w \in Child(u)$ is in balanced state.
- The first level nodes in the tree can never trigger the occurrence of an parent-changing event as the sink is the only possible parent for those nodes.

The tree is load-balanced if every individual node reaches the balanced state according to the above propositions. However, the sink or the root of the tree is assumed to be in a balanced state. Now, a node that is initially in the unbalanced state may reach the balanced one through four stages.

TH-1251_11610108

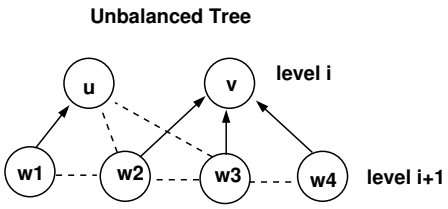

Figure 5.1: Unbalanced tree

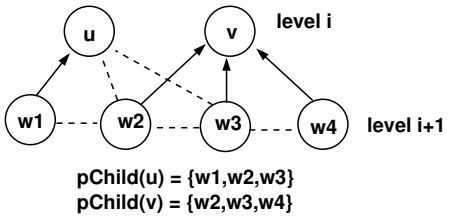

Figure 5.2: The contents of $pChild$ for nodes u and v

Stage 1. An intermediate node initiates the load-balancing by announcing its child size through an *ChldUpdate* message broadcast in the neighborhood. It may forward a *ChldUpdate* in any of the following two cases. *First*, after adding a node in its $Child$ set, and so, the child size exceeds the value of $\xi_{min} = 1$ (The minimum number of children an intermediate node can have is 1) as given in lines 16-18 of Algorithm 5.2 and in lines 13-15 of Algorithm 5.4. *Second*, after removing a node from the $Child$ set, and then, the child size reduces to less than ξ as stated in lines 4-6 of Algorithm 5.8. Thus, a node may participate in the load-balancing on receiving a *ChldUpdate* from either the parent or the alternate parent. On receiving a *ChldUpdate* from the parent node, every node u decides whether to participate in load-balancing activities or not depending on its $altpSet$. If $altpSet(u)$ is not empty, u participates in the load-balancing by sending an *AddMe* message to all $w \in altpSet(u)$ according to lines 5-8 of Algorithm 5.5. Otherwise, u remains silent. On receiving a *ChldUpdate* message from the alternate parent w, node u sends an *AddMe* message only to w (event A) if following three conditions are satisfied. Again, due to the level improvement, the $altpSet(u)$ may get updated by including a node w. If following three conditions are satisfied, u sends an *AddMe* message to w according to lines 22-25 of Algorithm 5.4 (event B).

Condition (a) $|Child(w)| < \xi$

Condition (b) $MOD\left(|Child(parent(u))| - |Child(w)|\right) > 1$

Condition (c) u has not yet sent an *AddMe* message ($addflag$ is False)

According to lines 2-4 of Algorithm 5.5, the variable $bFactor$ stores the value of $|Child(parent(u))|$ for a node u. Now, as both the events A and B can occur in parallel, to avoid multiple occurrence of the same event, an *AddMe* is forwarded only once to the alternate parent. This is controlled by the $addflag$ variable. For the unbalanced tree shown in Figure 5.1, node u has one child, node $w1$, where as node v has three children,

TH-1251_11610108

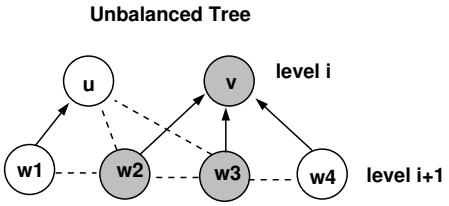

Figure 5.3: Node $w2$ and node $w3$ receives *ChldUpdate* message from node v

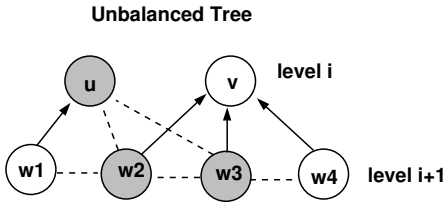

Figure 5.4: Nodes $w2$ and $w3$ communiate with alternate parent

nodes $w2$, $w3$ and $w4$. The contents of *pChild* for nodes u and v have been shown in Figure 5.2. On receiving *ChldUpdate* message from node v, both nodes $w2$ and $w3$ send *AddMe* message to the alternate parent node u according to Figure 5.3.

Stage 2. In this stage, the alternate parent takes the decision whether to participate in load-balancing or not. On receiving an *AddMe* from a node u, node w checks whether Condition (a) is true or not, including the requesting node in its $Child$ set. The variable *counter* is incremented each time an *AddMe* message is received, and the sending node can be added to the $Child$ set without violating Condition (a). The counter variable keeps the track of the number of such children that are actually not added to the $Child$ set yet, however, are allowed to be accommodated as per Algorithm 5.6. If allowed, w sends back a *PosAdd* message to node u, otherwise remains silent. Let on receiving *AddMe* message from nodes $w2$ and $w3$, node u finds Condition (a) to be true for node $w2$, and thus Condition (a) is evaluated as false for node $w3$, as shown in Figure 5.4. Node u sends a *PosAdd* message to node $w2$.

Stage 3. The $addflag$ at node u is set to True if u has sent one or more *AddMe* messages according to Stage 1. Thus, u may receive one or more *PosAdd* messages to continue the load-balancing process. The threshold of the child size of the node $v = parent(u)$, denoting the minimum number of children that the node v should keep to balance the load, is assumed to be half of the $|Child(v)|$. The *PosAdd* message carries the child size of the alternate parent w including node u as the argument *csize*. On receiving a *PosAdd*, node u sets the $posflag$ to True, and saves the source of the message as $posId$ before sending the *Rem* message to its parent according to lines 2-3 of Algorithm 5.7. Once $posflag$ is True, receiving further *PosAdd* can not change the value of $posId$, as restricted by line 1 of Algorithm 5.7. It is possible that every child of the node v individually decides to change its parent from v to some other nodes, on receiving a *PosAdd* from the respective alternate parents. This is because the parallel occurrence of the parent leaving events can not update the $bFactor$ variable at every child node due to the asynchronous

TH-1251_11610108

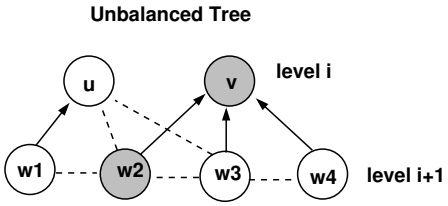

Figure 5.5: Node $w2$ validates its decision of parent changing from parent node v

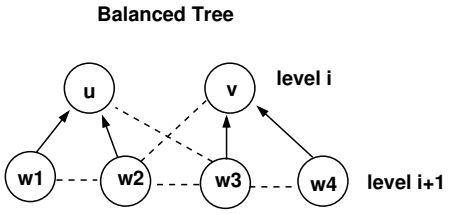

Figure 5.6: Final load-balanced tree

channel property. So, the aggregation of all individual decision of parent changing may lead to a wrong decision where all the children of v have left $Child(v)$, and thus, nodes reached the unbalanced state. In a distributed environment, it is not possible to take the correct decision based on only partial topology information that each node has. Also, the information at a node gets updated time to time at every node. Even if a node decides to change the parent for load-balancing on receiving the *PosAdd* from its alternate parent, only the parent can confirm whether that decision would lead to the balanced state or not, based on its present child size. This is why node u sends a *Rem* message to its parent so that to validate the parent changing event. As shown in Figure 5.5, node $w2$ sends a *Rem* message to its parent node v, on receiving *PosAdd* message from its alternate parent, node u.

Stage 4. In the last stage of load-balancing, the parent node v receives the *Rem* message from the child u with argument $csize$, that carries the child size of the alternate parent of u, including node u. Node v allows node u to leave its *Child* set, and join the $Child(w)$ only if $|Child(v)| > csize$ as per the condition in line 1 of Algorithm 5.8. In that case, v removes u from its *Child* set, and sends it a *RemAck* message as a positive acknowledgment. It also broadcasts a *ChldUpdate* message if its child size becomes less than ξ as per lines 4-6 of Algorithm 5.8. On receiving a *RemAck* node u finally changes its parent, and updates all neighbors according to Algorithm 5.9. This completes the load balancing process for u leading it to the balanced state. In Figure 5.5, node $w2$ receives *RemAck* message from node v and changes it parent from node v to node u. Final load-balanced tree is shown in Figure 5.6.

Proof of Correctness : The correctness of the algorithms for load-balancing has been established with the help of the following theorems.

Theorem 5.3. *For any pair of nodes u and v, if $level(u) = level(v)$ and $pChild(u) \cap pChild(v) \neq \phi$, then in load-balanced BFS tree $MOD\,(|Child(u)| - |Child(v)|) \leq 1$.*

TH-1251_11610108

Proof. For the sake of simplicity, the theorem has been proved for $\xi = 3$. Similar logic can be used for any ξ value. Let $\xi = 3$ and $|Child(u)| - |Child(v)| = 2$ for two nodes u and v, where $level(u) = level(v)$ and $pChild(u) \cap pChild(v) \neq \phi$. Load balancing is required in this case. Therefore, $|Child(u)| > 1$ for the node u from the assumption. So, node u will broadcast a *ChldUpdate* message according to Algorithm 5.2 or Algorithm 5.4. Let $w \in Child(u)$ be a node such that it receive a *ChldUpdate* from the node u, and also, $w \in \{pChild(u) \cap pChild(v)\}$. Therefore, according to Algorithm 5.8, node w sends an *AddMe* message to v (where $v \in altpSet(w)$ from the assumption). From Condition C of the load-balancing criteria, $|Child(v)| < \xi$ according to the assumption. On receiving an *AddMe* from the node w, node v sends it a *PosAdd* as the condition in line 1 of Algorithm 5.5 is satisfied. Now, node w calculates the threshold for child size of the node u as $\lceil(|Child(u)|/2)\rceil$. The alternate parent child size as received with the *PosAdd* message, is estimated to be $csize = (|Child(u)| - 2) + 1$. Therefore, for $2 \leq |Child(u)| \leq 4$,

$$(csize - \lceil(bFactor/2)\rceil) = \{(|Child(u)| - 1) - \lceil(|Child(u)|/2)\rceil\} < 1$$

If $|Child(u)| > 4$, $|Child(v)| \geq 3$. From the assumption ($\xi = 3$), load-balancing will not be possible according to Condition C of the load-balancing criteria. So, node w sends a *Rem* message to its parent u for final confirmation. Node u sends a positive acknowledgment through a *RemAck* message to node w as $|Child(u)| > csize$. Therefore, node w changes its parent from the node u to the node v making $|Child(u)| = |Child(v)|$. Hence, proved. □

Theorem 5.4. *The proposed set of distributed algorithms for tree construction eventually terminates.*

Proof. In distributed message-passing environment, the termination of a set of algorithms is determined on the verification that every node has reached the stable state. A node is said to be reached the stable state at the time t if it does not transmit any control messages after the time t. Let the node u be called in *pre-stable state* at the time t, if it does not send any control messages after t. Proving every node in the tree will reach the stable state eventually suffices to prove that the proposed set of tree construction algorithms eventually terminates. Let u be a node such that it receive a *Token* message, and set its parent. If u has chosen the correct parent, and also, in the balanced state, then it does not send any more control messages, and reaches the *pre-stable state*. However, it can receive some control messages from its neighbors. Otherwise, u may wish to change its parent due to either an level improvement or the load-balancing requirement.

TH-1251_11610108

Case 1. Let node u change the parent due to an level improvement. Thus, it receives either a better *Token* or an *Update* message with the improved level. u changes the parent, and broadcasts an *Update* message in either of the cases. So, node u reaches the *pre-stable state* at this point.

Case 2. Let node u change parent due to the load-balancing requirement. On receiving a *ChldUpdate* message from the parent, node u sends an *AddMe* message to its alternate parent. If there is a provision for load-balancing from the alternate parent point of view, u receives an *AddMe*, which in turn triggers the event of sending a *Rem* message to the $parent(u)$. Finally, if the parent node also acknowledges the parent changing event of the node u, u receives a *RemAck* message from the parent node. Now, u changes the parent, and finally, updates its neighbors by an *Update* message broadcast. At this point of time, node u reaches the *pre-stable state*. $addflag$ and $posflag$ are set to True, and never reset again such that a node can change its parent only once to fulfill the load-balancing criteria.

Let at the time t_i, all nodes in the tree reach the *pre-stable state*. Therefore, at the time $t_j > t_i$, all nodes in the tree will reach the stable state. Hence, proved. □

Theorem 5.5. *The cost of load-balancing in terms of control messages is estimated to be* $O(|V_{\mathbb{T}}|)$.

Proof. Let u be a node such that it change its parent due to the load-balancing requirement. Now, u receives a $\xi - 1$ number of *ChldUpdate* messages according to Algorithm 5.2 and Algorithm 5.4, where ξ is the average number of children for any intermediate node. On receiving a *ChldUpdate*, u sends maximum r number of *AddMe* messages, where $r = |altpSet(u)| \leq |\{Neighbor(u) \setminus Child(u)\}| - 1$. u also receives r number of *PosAdd* messages in response, in the worst case. However, u sends only one *Rem* to the parent, and receives only one *RemAck* as an acknowledgment. Finally, u broadcasts an *Update* message. Therefore, the transmission of total messages at every node that changes the parent due to the load-balancing requirement is estimated to be $(\xi - 1)_{ChldUpdate} + r_{AddMe} + r_{PosAdd} + 1_{Rem} + 1_{RemAck} + \delta_{Update}$ or $(\xi - 1) + 2 + 2\delta$ (δ is degree of any node), assuming $2r = \delta$. If it is assumed that W number of nodes change their parents due to the load-balancing requirement, where $W < \frac{1}{2}V_{\mathbb{T}}$, then the total cost of load-balancing can be estimated as $W \times (\xi + 1 + 2\delta)$, which is essentially $O(|V_{\mathbb{T}}|)$. □

TH-1251_11610108

5.2.3 Post Tree Construction Activities

For every primary node x in the network the set of designated redundant nodes, $\mathbb{R}(x)$ is identified during the initial set-up as discussed in Section 5.1. After the successful completion of the load-balanced BFS tree construction, x broadcast a sleeping schedule for all nodes in $\mathbb{R}(x)$. According to the schedule, one node $w \in \mathbb{R}(x)$ wakes up, and probes to check whether its designated node $x \in serveList(w)$ is active or not at the beginning of a particular time-slot, where remaining nodes in $\mathbb{R}(x)$ sleep during that slot. The time-slot duration is taken as $f(\mathfrak{T})$, where $\mathfrak{T}$ is the amount of delay the application can tolerate. Along with the schedule broadcast, the primary node x also sends its level and parent information to all nodes in $\mathbb{R}(x)$, which is stored in the node w as the *desLevel* and *desParent* set, respectively. All these activities have to be completed before the second phase of tree management is initiated.

5.3 Second Phase of Tree Management

Sensor nodes are limited in resources as discussed earlier. Nodes may fail at any time either due to the energy exhaustion (called *ageing*) or due to some technical fault or natural disaster (called *node crash*). The failure of a node in any of the cases may lead to a disruption of the application services by affecting the connectivity and/or coverage in the network. After the successful execution of the first phase of tree management, in the second phase, the proposed TMM is responsible for the maintenance activities. In this phase, the proposed TMM and AMC interact with each other to ensure the efficient data forwarding even in presence of a node failure.

Activities of Redundant Nodes: Let $\mathcal{E}_{Res}(x)$ and $\mathcal{E}_{Th}$ denote the residual energy and the threshold of ageing for any primary node x, respectively. At the beginning of time-slot T_i, any node $w \in \mathbb{R}(x)$ wakes up, and probes node x.

- If $\mathcal{E}^{Res}(x) > \mathcal{E}_{Th}$, w goes to sleep.
- $\mathcal{E}^{Res}(x) < \mathcal{E}_{Th}$, on a successful probing, w receives a positive reply from x, and becomes primary (in active state) to replace node x.
- If x has failed due to a sudden crash, on probing, w does not receive any reply from x, and assuming x dead, w becomes primary (in active state) to replace node x.

TH-1251_11610108

The maintenance scheme manages the event of faulty nodes' leaving from the tree, and the new nodes' joining into the tree as discussed in following subsections, considering two types of node failure, *ageing* and *crash*.

5.3.1 Maintenance Scheme for Single Node Failure

Ageing: A node fails due to the ageing if its residual energy decreases to a certain threshold. In this case, the leaving node x informs all the neighbors by a *Leave* message broadcast. On receiving a *Leave* from a node x, all nodes $v \in Child(x)$ sets *block* to True, which in turn, triggers the AMC to stop the data forwarding as discussed in Subsection 5.3.3. If $x \in Child(w)$, on receiving a *Leave* from x, node w waits for a signal from AMC. The AMC sends a signal to the TMM when it is safe to remove x from the $Child(w)$ as stated in Subsection 5.3.3. On receiving the signal from the AMC, the node x is removed from the $Child(w)$ safely. If $u \in \mathbb{R}(x)$ is the node to replace x, on receiving a *Leave* from x, node u broadcasts a *Join* message with two arguments, the $desId(u)$, which is node x, and the $desParent(u)_{desId}$, which is $parent(x)$ as stated in lines 7-10 of Algorithm 5.10.

On receiving a *Join* message from the node u, node v sets the parent to u if $parent(v)$ had failed, and sets the *block* to False. Similarly, if $Child(v)$ had failed, on receiving a *Join*, v adds u in its $Child$ set. In both the cases, node v sends a *JoinAck* to u according to Algorithm 5.11.

On receiving a *JoinAck* message from v, node u sets the node v as either its parent or child based on the level checking as performed in Algorithm 5.12, and also, starts data forwarding by setting the *block* to False. This completes the repairing, where the failed node due to an ageing has been replaced by a redundant node, and the tree has been adjusted locally replacing the failed node by a redundant one.

Node Crash: Now, due to a technical or other fault it may happen that a node fails suddenly without being able to inform its neighbors. Thus, due to a sudden failure of node x, neither $parent(x)$ nor any node $Child(x)$ receives the *Leave* message as discussed before. However, they can detect the failure of x through the lower layer fault-detection technique. Let node $w \in \mathbb{R}(x)$ fail to probe the node x, and become active. Node w broadcasts a *Join* message with the arguments $desId$ and $parent(desId)$, where $desId = x$. On receiving the *Join*, all nodes in $Child(x)$ and $parent(x)$ perform the steps as stated in Algorithm 5.11, and the rest of the maintenance scheme is similar to the one discussed in case of *Ageing*.

Proof of Correctness :

TH-1251_11610108

Algorithm 5.10 On receiving *Leave* from v by u

1. **if** $v = parent(u)$ **then**
2. $block \leftarrow True$ /*stops data forwarding until new parent is set*/
3. **end if**
4. **if** $v \in Child(u)$ **then**
5. wait for the signal from AMC /*on receiving the signal, removes v from Child(u)*/
6. **end if**
7. **if** $v \in serveList$ **then**
8. $desId \leftarrow v$
9. broadcasts *Join(desId, desParent*$(u)_{desId}$*)*
10. **end if**

Algorithm 5.11 On receiving *Join(nid, pid)* from v by u

1. **if** $nid = parent(u)$ **then**
2. $parent(u) \leftarrow v$
3. send *JoinAck(level(u))* to v
4. $block \leftarrow False$
5. **else if** $nid \in Child(u)$ or $pid = desId(u)$ **then**
6. $Child(u) \leftarrow Child(u) \cup \{v\}$
7. send *JoinAck(level(u))* to v
8. **end if**

Algorithm 5.12 On receiving *JoinAck(l)* from v by u

1. **if** $l > level(u)$ **then**
2. $Child(u) \leftarrow Child(u) \cup \{v\}$
3. **else if** $l < level(u)$ **then**
4. $parent(u) \leftarrow v$
5. **end if**
6. $block \leftarrow False$ /*starts data forwarding*/

Theorem 5.6. *The tree maintenance cost for a single node failure is a constant in terms of both the control message communication and the delay.*

Proof. Let node x fail at time t_i, and a node $u \in \mathbb{R}(x)$ would come up to substitute the node x. In the worst case, u will come up at the time $t_i + f(\mathfrak{T})$, in case of failure due to a node crash introducing an extra $\mathfrak{T}$ amount of delay, where $\mathfrak{T}$ is the delay that the application can tolerate. Let $v \in Child(x)$ such that v receives the *Leave* message from x at the time $t_j > t_i$. As the *Leave* is a broadcast message, node u also receives the same *Leave* at the time $t_j \pm \epsilon$, where $\epsilon \to 0$. Finally, x receives a *Join* message at the time

TH-1251_11610108

$t_k > t_j$ from u. The interval $(t_j - t_i)$ or $(t_k - t_j)$, assumed to be Ω, is equal to the 1-hop propagation delay and is very small. Therefore, if due to the ageing or crash the parent node fails, the child can fix its parent in $\mathfrak{T} + 2\Omega$ in the worst case. The delay is same for the parent of the failed node also. The newly added node u receives a *JoinAck* from each of the nodes in $Child(x)$ and the $parent(x)$. Every *JoinAck* is sent as a response to the *Join* message, however, is sent independently. As the *Leave* message is broadcast, it can be assumed that an exchange of *Join* and *JoinAck* messages between every pair of nodes with respect to the node u is in parallel, and so, cumulatively they can introduce 2Ω amount of delay in the worst case. Overall, for a single node failure, the cost of tree maintenance in terms of delay can be estimated as $\mathfrak{T} + 2\Omega$ in the worst case, which is essentially $O(max\{\mathfrak{T}, \Omega\})$.

A child node as well as the parent of the failed node transmits 3 messages $(1_{Leave} + 1_{Join} + 1_{JoinAck})$, in total, to fix its parent due to the parent failure. The newly added node u receives a *JoinAck* from each of the nodes in $Child(x)$ as well as from $parent(x)$, and thus, transmits $\delta_{Join} + (\xi + 1)_{JoinAck} + 1_{Leave}$, where $\xi = |Child(x)|$. Therefore, for a single node failure, the total cost of control message communication can be estimated as $3 + \delta + \xi + 2 = \delta + \xi + 5$, which is essentially a constant. □

5.3.2 Maintenance Scheme for Multiple Node Failures

In case, multiple nodes fail simultaneously, and there exists more than 2-hop distance between any two failed nodes, then each of the failure cases can be handled as a single node failure as discussed in Subsection 5.3.1. This is because the effect of repairing for one node does not interfere with the repairing activities for another node. However, if more than one nodes fail in close vicinity, the maintenance scheme becomes slightly modified as discussed in this subsection. Let two nodes x and y fail at the same time either due to an ageing or due to a crash, such that $x = parent(y)$. Then, both the nodes $w_1 \in \mathbb{R}(x)$ and $w_2 \in \mathbb{R}(y)$ will forward a *Join* message. So, w_1 receives a *Join* with argument $desId = y$ from the node w_2, where node y is neither the parent nor a child of w_2. However, the *Join* message sent from w_2 also carries the $parent(y)$, that is of course node x. Therefore, w_1 adds node w_2 in its *Child* set satisfying the condition in line 5 of Algorithm 5.11, and sends w_2 a *JoinAck* message as a response. In this way, multiple simultaneous node failures in close vicinity can also be handled by the proposed scheme.

The worst Case Scenario: On simultaneous failure of two neighboring nodes, if the corresponding redundant nodes are communication unreachable to each other, the

TH-1251_11610108

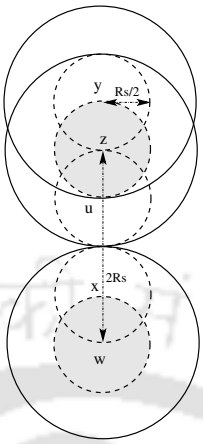

Figure 5.7: Node u and node x fails. Connectivity after recovery

worst case scenario occurs. Considering Figure 5.7, let u and x be two nodes such that $u = parent(x)$. Also, let w and z be two nodes such that $w \in \mathbb{R}(x)$ and $z \in \mathbb{R}(u)$. At the time t_i, both the nodes u and x fail, and nodes w and z come up as the substitutes. According to the figure, as both the nodes w and z are placed at an extreme distance $R_s/2$, from the nodes x and u, respectively, the distance between w and z becomes $2 \times R_s$. Now, as per the proposed maintenance scheme, $z = parent(w)$ in the repaired tree to maintain the connectivity as well as the sensing coverage. However, in a normal state, $R_c = R_s$, and thus, node w and node z can not communicate to each other. In this situation, node w does not receive *Join* message from the node z. After receiving a *Leave* message from x, node w waits for $f(\mathfrak{T})$ amount of time, where $\mathfrak{T}$ is the delay that can be tolerated by the application. If no *Join* message is received within $f(\mathfrak{T})$ time, w sets $R_c = 2R_s$, and broadcasts an *Urgent* message with $u = parent(desId(w))$ as an argument. Node z, on receiving an *Urgent* message, checks that $desId(z) = u$. Node z adds w in $Child(z)$, and replies back by an *UrgentAck* message after setting $R_c = 2R_s$. Node w sets $parent(w) = z$ on receiving the *UrgentAck* message. In the worst case, the distance between these two nodes in the repaired tree can be a maximum of $2R_s$. So, with a little adjustment in R_c and at the cost of one extra hand-shaking, the TMM can handle the simultaneous node failure in a close vicinity for this special case scenario as stated in Theorem 4.2. However, the worst case scenario is a rarely occurring event, and thus, the implementation cost can be afforded, subject to the application requirement.

TH-1251_11610108

Algorithm 5.13 Application Message Controller at node u

```
1.  if data D is received then
2.      put D in Buff[in]                                   /*stores data in buffer*/
3.      adjust in
4.      if D.lastFlag = True then
5.          send the signal to TMM with argument v          /*v is the node from which the D is
            received*/
6.      end if
7.  end if
8.  if block = False then
9.      send Buff[out] to parent(u)                         /*forward data from buffer*/
10.     adjust out
11. end if
```

5.3.3 Application Message Controller

The proposed Application Message Controller (AMC) at each node works cooperatively with the TMM to offer a reliable delivery of application messages. Most of the event-triggered applications require an independent data gathering where sensory data from each node is forwarded to the parent so that all data from the network are eventually delivered to the sink. Every node is assumed to generate sensory data packets, and store it into a local FIFO buffer, $Buff$. Let *in* and *out* are two pointers such that the new data is pushed in $Buff[in]$, and is dispatched from $Buff[out]$. Along with the self generated sensory data, data received from each child is pushed to $Buff[in]$. Once the tree construction is over, the variable *block* is set to False at every primary node such that they can start data sensing and forwarding. If a parent node fails, node u sets the *block* to True, and stops the forwarding as stated in lines 1-2 of Algorithm 5.10. On resetting the parent by the TMM, node u sets *block* to False again to start data forwarding as stated in line 6 of Algorithm 5.12. Thus, no data is lost due to parent changing. Further, the node x that is going to leave the network due to an energy exhaustion, appends a $lastFlag$ set to True with the last data it sends to its parent u. Node u, on receiving data D with $lastFlag = True$, processes the data, and sends a signal to the TMM with the argument x. On receiving the signal from the AMC, the TMM simply removes x from $Child(u)$ safely as stated in Algorithm 5.10. However, if a node crashes suddenly, the application data in transit are lost forever. The detail working of the AMC has been given in Algorithm 5.13.

TH-1251_11610108

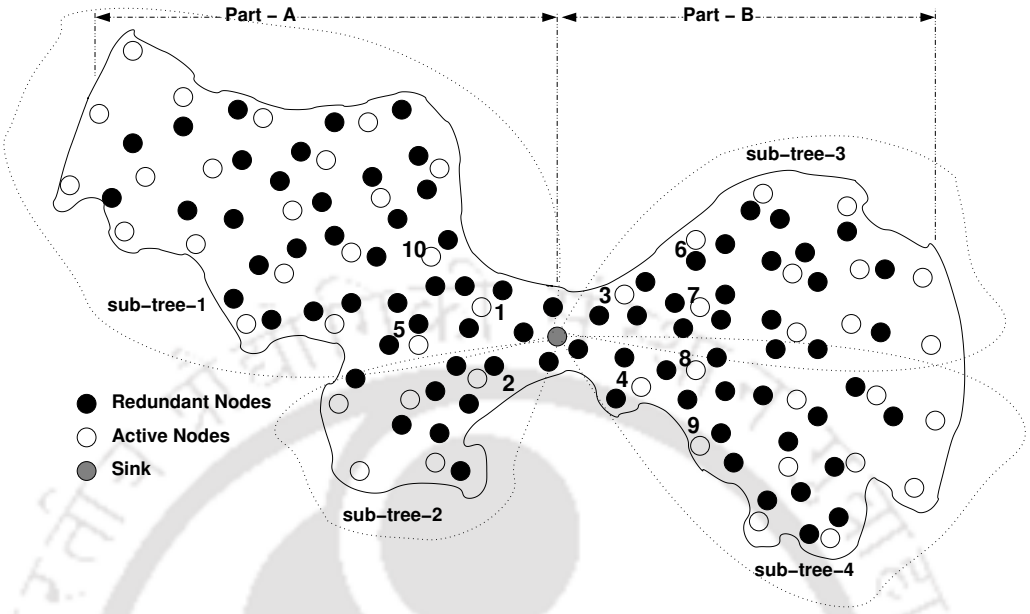

Figure 5.8: Simulation terrain with primary and redundant nodes

5.4 Simulation Results

The proposed scheme is implemented and simulated using NS-2.35 network simulator [4]. An uneven terrain, as shown in Figure 5.8, is used for the simulation purpose. The terrain can be divided into two parts, where Part A of the terrain represents an irregular area, and Part B of the terrain represents a regular area. This type of terrain is considered to simulate the effect of node distribution and the performance of data forwarding over regularity of the terrain. Sensor nodes are deployed in the terrain, and identified as primary or redundant nodes according to the proposed scheme. The sensing range of every sensor is taken as 200 m. The energy specifications of sensor nodes are taken according to Willow MicaZ MPR2400 sensor motes [3] where the power consumption for receive, transmit, and idle mode are 0.053 watt, 0.047 watt, and 0.0216 watt, respectively. The data forwarding tree for Part A of the terrain can have a maximum level of 8, whereas the data forwarding tree for Part B of the terrain can have a maximum level of 5.

IEEE 802.11 MAC protocol is considered as the MAC layer channel access for the sensor nodes. The channel data rate is assumed to be 1 Mbps. The Constant Bit Rate (CBR) data traffic is considered as an application layer traffic with packet size 64 bytes. The proposed scheme is compared with two other recently proposed schemes, the one proposed in Chapter 3 and the one proposed by Liao *et al.* [109]. Chapter

TH-1251_11610108

Table 5.2: Comparison of balanced and unbalanced tree based forwarding

Data Generation Rate	Energy Dissipation at Leaf watt-hour	Node ID	Energy Dissipation (unbalanced) watt-hour	Energy Dissipation (balanced) watt-hour
32 bps	75.22	1	84.11	81.27
		2	78.19	81.56
		3	85.81	81.67
		4	76.54	81.54
128 bps	78.86	1	85.01	84.92
		2	82.61	83.66
		3	88.01	83.52
		4	78.60	83.22
256 bps	80.02	1	92.24	92.11
		2	87.48	88.51
		3	93.22	89.13
		4	84.44	88.64
512 bps	82.08	1	106.67	106.51
		2	97.15	97.36
		3	104.66	100.85
		4	95.08	100.64

3 proposes an online tree repairing scheme based on the alternate parent calculation. This scheme is refereed as "Online Repairing" in all further references. Liao *et al.* [109] proposed a gradient based sensor deployment strategy. However, they have not considered the redundancy in the sensor node deployment. This scheme is refereed as "Gradient Deployment without Redundancy" in all further references. The proposed scheme in this Chapter is refereed as "with Redundancy" in all the figures.

5.4.1 Effect of Load Balancing on Energy Dissipation

Table 5.2 shows the energy dissipation of four primary nodes that are directly connected to the sink. These nodes are marked in Figure 5.8 as 1, 2, 3 and 4. Out of these four nodes, node 1 and node 2 belong to the Part A of the terrain, whereas node 3 and node 4

TH-1251_11610108

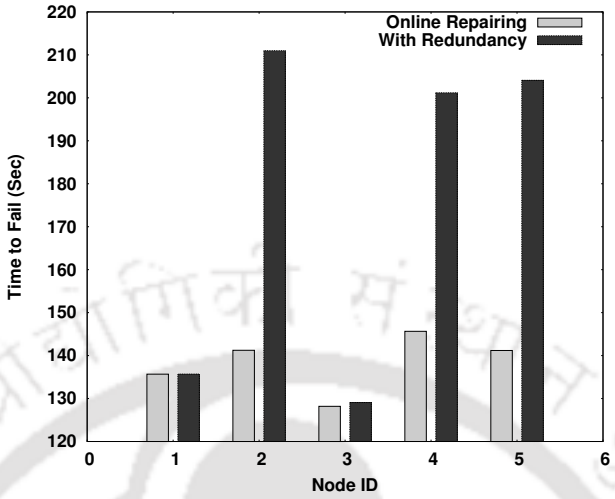

Figure 5.9: Comparison of fault tolerance using redundancy and online repairing

belong to the Part B of the terrain. The energy dissipation is shown in Table 5.2 for both the balanced tree construction and the unbalanced tree construction. The unbalanced tree is constructed using a distributed BFS tree construction algorithm similar to the one proposed in Chapter 3. The table shows the energy dissipation for different data generation rate at the sensor nodes. By analyzing the nature of the deployed nodes in the terrain, it can be observed that the absence of load-balancing can severely affect the lifetime of the nodes in the Part B of the terrain. Part B can have two sub-trees as pointed in Figure 5.8, sub-tree 3 and sub-tree 4. Node 7 and node 8 are in the communication range of both the nodes 3 and 4. So, for a distributed BFS tree construction, it is possible that node 3 can have three children, whereas node 4 can have a single child. In this case, the energy dissipation of node 3 would be more compared to the node 4. This can be observed from Table 5.2. For unbalanced tree, the energy dissipation of the node 3 is more compared to the energy dissipation of the node 4. Thus, node 3 would die out earlier than the node 4, though they belong to the same level of the data forwarding tree, in a regular part of the terrain. It can also be observed from the table that the balancing does not have much effect on the nodes 1 and 2 of the data forwarding tree, as the number of children in their rooted sub-trees are inherently different, according to the nature of the terrain.

5.4.2 Time to Fail: Comparison with "Online Repairing"

The time to fail can be defined as the lifetime of a node, where the failure is caused by the energy exhaustion. Five primary nodes are considered in the data gathering tree to show

TH-1251_11610108

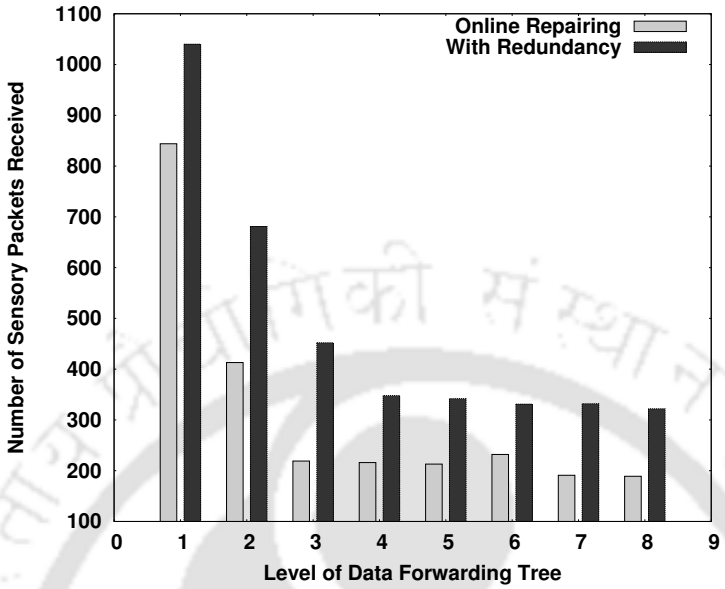

Figure 5.10: Amount of sensory packets received from Part A of the terrain

the results. To compute the time to fail, the initial energy at every node is taken as 5 Joule, and the data generation rate at every sensor is assumed to be 512 bps. The result is shown in Figure 5.9. The load of node 1 and node 3 is higher compared to all other nodes in Part A and Part B of the terrain, respectively. So, node 1 and node 3 fail first. The time to fail for node 1 and node 3 are almost the same for both the "Online Repairing" scheme and the proposed approach. Once node 1 fails, all the data traffic from the Part A of the terrain passes through the nodes 5 and 2. Similarly, all the data traffic from the Part B passes to the node 4. From Figure 5.9, it can be observed that node 2, node 4 and node 5 fail much earlier in the "Online Repairing" scheme compared to the proposed approach in this chapter. Once node 2 and node 4 fail, the network gets partitioned. The figure shows that the network lifetime is more for the proposed scheme compared to the "Online Repairing" scheme.

5.4.3 Amount of Sensory Packets Received

Figure 5.10 and Figure 5.11 show the amount of sensory packets received from Part A and Part B of the terrain, respectively. The data generation rate at every sensor is taken as 512 bps. For the "Online Repairing" scheme, the last data packet arrives just before the network gets partitioned. In the proposed scheme, sensory packets are received at the sink until the leaf nodes die out of energy. Both the figures show that the amount of sensory

TH-1251_11610108

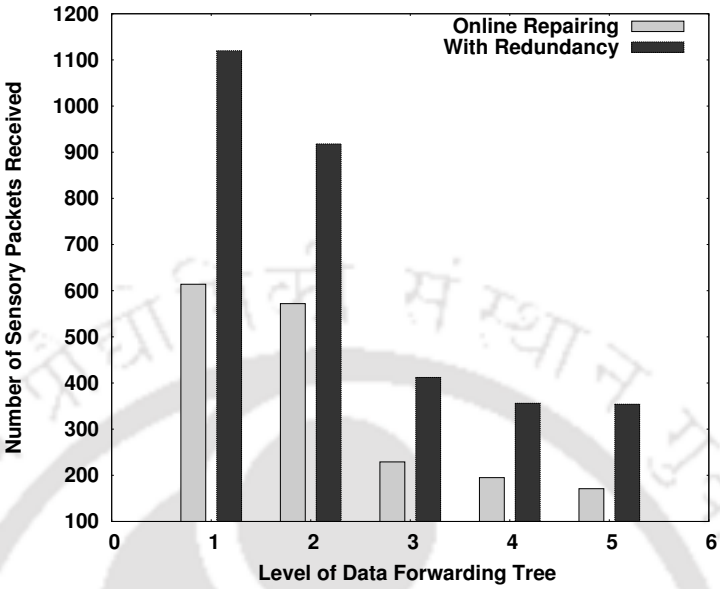

Figure 5.11: Amount of sensory packets received from Part B of the terrain

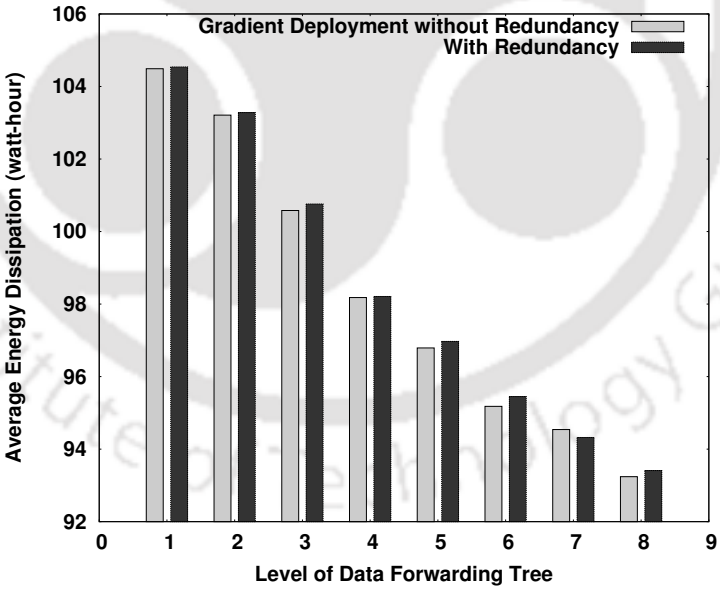

Figure 5.12: Energy dissipation: Gradient based deployment with and without redundancy

packets received in the proposed scheme is more compared to the "Online Repairing" scheme.

TH-1251_11610108

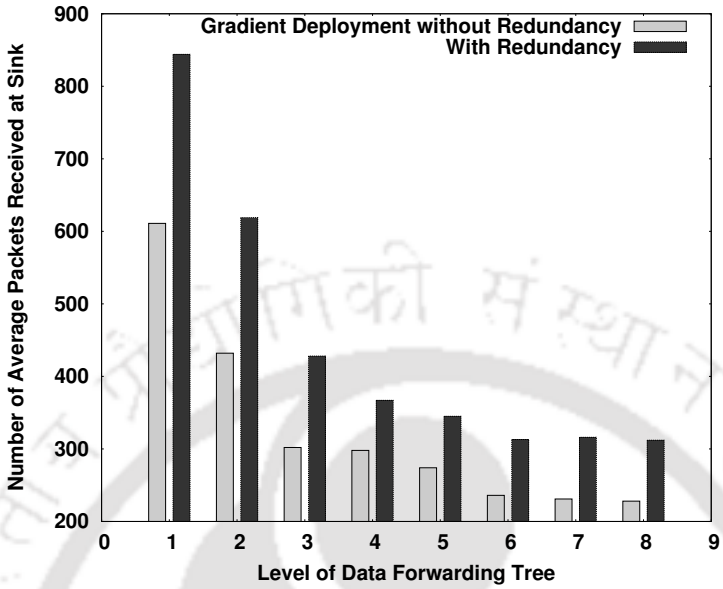

Figure 5.13: Packets received: Gradient based deployment with and without redundancy

5.4.4 Comparison with Gradient based Deployment without Redundancy

Figure 5.12 and Figure 5.13 show the comparison with the gradient based deployment framework as proposed in [109]. For this experiment, the data generation rate at every sensor is taken as 512 bps. As shown in Figure 5.12, the average energy dissipation is similar for both the frameworks proposed by Liao *et al.* [109] and in this chapter. However, from Figure 5.13, the amount of sensory data received is more in the gradient based deployment with redundancy compared to the one without redundancy. From analysis of the simulation traces, it has been observed that for the framework proposed in [109], the MAC layer contention near the sink is very high. As a result, lots of packets are dropped. The reason behind this high contention is that the overlapping is high near the sink because of the high density of the primary nodes. In the proposed scheme in this chapter, the density of the primary nodes is uniform throughout the terrain. The density of the redundant nodes increases from the boundary of the terrain towards the sink. Further the data gathering tree is load-balanced in this case. So, the packet drops due to the contention is significantly less in the proposed scheme of this chapter compared to the one in the literature.

TH-1251_11610108

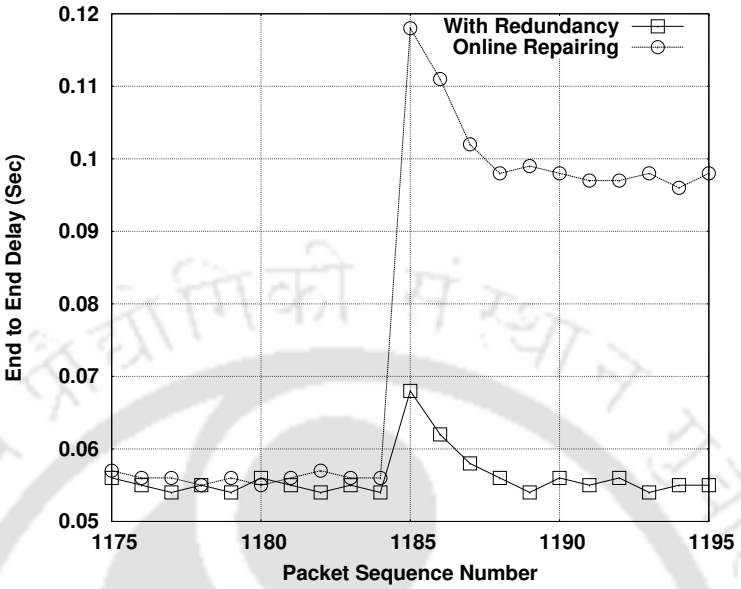

Figure 5.14: End-to-End Delay

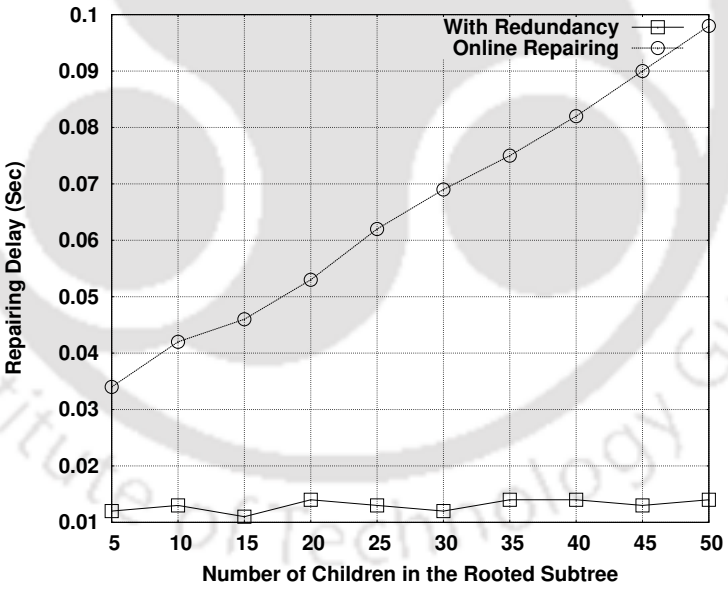

Figure 5.15: Repairing Delay

5.4.5 End-to-End and Repairing Delay

Figure 5.14 shows the end-to-end delay of some packets generated from the node 10 of the Part A of the terrain. The data generation rate at every sensor is taken as 512 bps. It can be seen from Figure 5.8 that node 10 is the child of node 1. Once node 1 fails, node

TH-1251_11610108

10 chooses node 5 as its alternate parent according to the "Online Repairing" scheme. Node 1 fails just before the packet with a sequence number 1185 is transmitted from node 10. It can be observed from Figure 5.14 that for the "Online Repairing" scheme, the end-to-end delay increases significantly for the packets transmitted after node 1 fails. This is because of the increase in the length of the forwarding path. For the proposed scheme in this chapter, four packets experience higher delay because of the parent failure. Once a redundant node in that region becomes active, all further transmitted packets experience similar delay to the earlier packets.

Figure 5.15 compares the repairing delay for the proposed scheme in this chapter with that of the "Online Repairing" scheme. In case of the "Online Repairing", the repairing delay depends on the number of nodes in the rooted sub-tree. The repairing delay increases as the number of nodes in the rooted sub-tree increases. For the proposed scheme in this chapter, the repairing delay is a constant, and significantly less compared to the one in the "Online Repairing" scheme. This shows that the "Online Repairing" scheme from node failure is not suitable for the delay-sensitive senor network applications.

5.5 Summary

In this chapter, a fault-tolerant tree based data gathering scheme has been proposed for critical delay sensitive applications of sensor network. Considering the constraints of the critical delay sensitivity and the resource limitation of sensor nodes, no existing work in the literature can provide a complete solution for the efficient fault-tolerant data gathering, maintaining the connectivity and the coverage in the network. Therefore, to fulfill these objectives, an efficient scheme was required to be designed starting from initial node deployment to data gathering tree construction and management. Here, considering the gradient based node deployment with redundancy as proposed in the previous chapter, a load-balanced BFS tree has been constructed in a distributed way for data gathering. Further, every node in the tree performs some local information processing during the tree initialization. Based on these information, an efficient tree repairing scheme has been proposed to handle a single as well as multiple node failures. Finally, the reliability of the application messages has also been assured through a local buffer management. The performance of the proposed scheme has been analyzed through the simulation.

To accommodate different application specific design objectives, the proposed theory of tree based data gathering has been modeled for two separate application scenarios as discussed in Chapter 1. Based on the application services, the sensing coverage criteria

TH-1251_11610108

in road sensor network (RSN) differs from the one in critical infrastructure information system (CIIS). The strictness of QoS requirements in terms of reliability and delay is also higher in CIIS than that in RSN. So, considering the universal design objectives of network lifetime improvement and node failure handling, the framework of tree based data gathering has been customized for these two different application scenarios according to their special needs. Chapter 6 proposes an adaptive data collection protocol, called *ADCROSS*, designed specially for traffic monitoring in RSN. In Chapter 7, multiple-tree based data gathering protocol, called *RelBAS*, has been proposed considering the inherent design challenges for the protection of any critical infrastructure.

TH-1251_11610108

TH-1251_11610108

Chapter 6

Adaptive Data Collection from Road Surveilling Sensors

The recent advancement in the field of Wireless Sensor Network (WSN) has sustained it as a better alternative for the design of Intelligent Transport Systems (ITS), because of its ability to provide cost-effective, reliable and scalable solutions [46]. Low-cost sensor devices are deployed over the road that sense different parameters of the moving vehicles such as vehicle speed, direction and pressure. Sensory data are then forwarded to the sinks, placed at certain intervals along the road-side, in multi-hop paths for further processing. Sinks form a wired infrastructure such that the collected sensory data can be processed in real-time for traffic and road-condition monitoring. Most of the past researches in the area of smart road design have suggested that the sensors should be placed on the moving vehicles [44] as already discussed in Chapter 2. However, feasibility of these schemes is limited as not all the vehicles are accommodated with the required infrastructure. Further, reliability in delivery of sensory data to the road-side sinks is low for these schemes, due to mobility of the vehicles. Magnetic sensors [2, 97], that can detect the moving vehicles from disturbances in the Earth's magnetic field, are widely used for in-road deployment, due to high sensitivity, small size and low power consumption. Deterministic deployment through manual intervention should be avoided for cost reduction purpose. Also, considering physical characteristics of the road-network, mounting sensors at the road-crossings or within the passing vehicles are not an efficient solution for providing real-time data analysis with a sufficient percentage of reliability. Randomly deployed sensors along the roadways form a data gathering tree rooted at the road-side sinks. During the deployment, sensors are placed with a sufficient number of redundancy such that any

TH-1251_11610108

arbitrary failure due to crash or sensor battery exhaustion does not affect the network connectivity, sensing coverage, etc. So, to extend the effective network lifetime, sensors follow a sleep-wakeup based schedule for critical energy saving as stated in Chapter 1. An efficient coordination among the sensors is required to maintain the global synchronization, as well as to satisfy the required design objectives. The principal objective of this design is to maintain a constant connectivity and the necessary sensing coverage in the network without affecting the network lifetime as well as the quality of service (QoS) for the running application. Communication through radio antennas are quite an amount costlier than the local computation for any sensor. Interaction among nodes through the exchange of control messages incurs a large overhead, which in effect reduces the overall network lifetime.

This chapter introduces *ADCROSS* (Adaptive Data Collection from Road Surveilling Sensors), an efficient sensory data collection protocol for road surveillance applications. The *ADCROSS* protocol proposes the concept of *k-strip length coverage*, that ensures any passing vehicle to be detected by at least k active sensors along the length of the road segment. A sleep-schedule for the sensors has been proposed, in which every node individually computes a back-off function to select the set of active nodes for any particular time-slot. This selection procedure satisfies the required *k-strip length coverage*, and maintains the required connectivity. The selection is globally consistent, and costs very low as only local computation is involved per node basis. For any particular time-slot, the active nodes form data gathering trees rooted at the nearest sinks, such that both in-flow and out-flow traffic analysis are possible with respect to any individual sensor. The reliability and other QoS metrics like delay have been ensured for the delivery of sensory data. The proposed solution is not only distributed, but also localized in nature (the computations are based on local information only), that significantly improves the network performance.

The rest of the chapter is organized as follows:

Section 6.1 gives the network architecture and the system model assumptions. The principal objectives and problem definitions are provided in Section 6.2. The centralized theory of the proposed scheduling scheme has been established in Section 6.3. Section 6.4 describes the localized distributed design of the proposed scheduling. The data collection tree construction and maintenance activities of the *ADCROSS* protocol to support node failure is provided in Section 6.5. In Section 6.6, the effectiveness of the *ADCROSS* protocol is presented using the simulation results. Finally, Section 6.7 concludes the chapter.

TH-1251_11610108

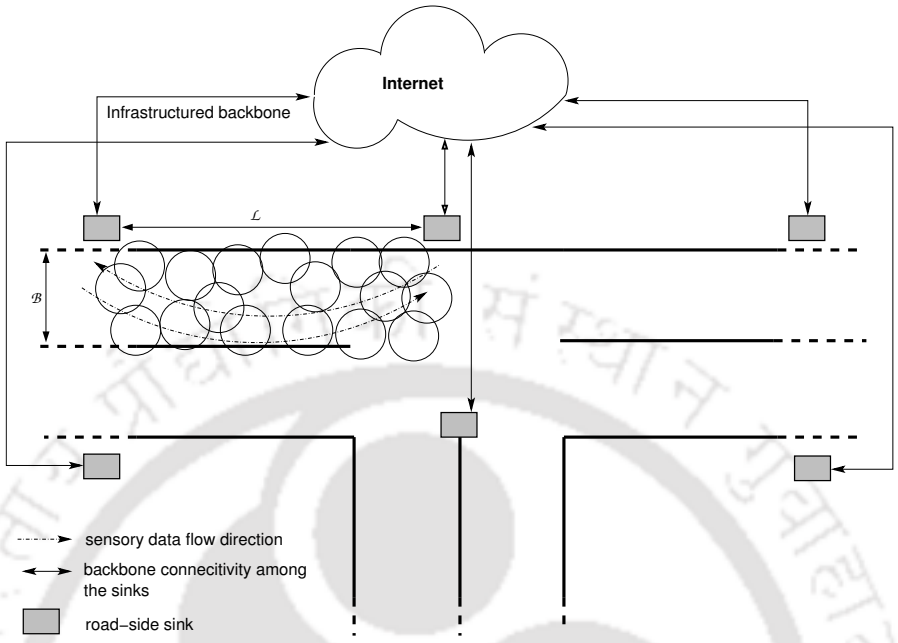

Figure 6.1: Road sensor network

6.1 Network Architecture

This chapter considers a sensor network built over the magnetic sensors [2] deployed along the road to offer a cost-effective solution for vehicle detection and traffic monitoring. Each sensor is equi-potential in terms of the resource and sensing capacity. A series of road-side base stations or sinks are placed to collect the sensory data. The sinks are placed at specific distances from each other, determined by the service providers. The sinks form an interconnected network for further processing of the sensory data. In this chapter, the terms 'sensing' and 'communication' denote the data sensing by the sensors, and forwarding of the sensory data to another sensor in the neighborhood, respectively. The sensing and the communication radius of each sensor is denoted by R_s and R_c, respectively, such that $R_c \geq 2R_s$ to maintain the connectivity as well as the sensing coverage. Further it can be noted that, throughout the chapter the term 'coverage' denotes the sensing coverage, unless mentioned otherwise.

Definition 6.1. *The disc Λ_u is called sensing covered by the node u, where $\Lambda_u = \pi R_s^2$, and u is at the center of Λ_u.*

Definition 6.2. *Let the Euclidean distance between any two nodes u and v be denoted by $d_{u,v}$. Then all nodes v, such that $d_{u,v} \leq R_c$, are called the communication neighbors of the*

TH-1251_11610108

node u, and are denoted by $cNeighbor(u)$.

Definition 6.3. *The set of all nodes v, are called the sensing neighbors of the node u, denoted by $sNeighbor(u)$, if $\Lambda_u \cap \Lambda_v \neq \phi$.*

A road may be segmented into multiple lanes such that the deployed sensors along each lane segment form data gathering tree rooted at the road-side sinks. Road-side sinks, deployed at a certain interval form a infrastructure backbone as shown in Figure 6.1. This chapter introduces the concept of *k-strip length coverage* along each lane segment between any two sinks. Let the lane segment between any two sinks s and s' be $\mathbb{L}_{s,s'}$, and the length of $\mathbb{L}_{s,s'}$ be $\mathcal{L}$ as shown in Figure 6.1. Then the *k-strip length coverage* can be defined as follows.

Definition 6.4. *Let n sensors be deployed along the lane segment $\mathbb{L}_{s,s'}$. Then to satisfy the k-strip length coverage, every passing vehicle along $\mathbb{L}_{s,s'}$ must be detected by at least k sensors, where $\lceil \mathcal{L}/R_c \rceil \leq k \leq n$.*

This chapter assumes that every sensor node calculates its relative position with respect to the sink using a pivot based approach as proposed in [104]. According to this scheme, the sinks work as the pivot points. It is mentionable that the proposed approach requires one time sensor localization, and the relative positioning is sufficient. Every sensor node also needs to be aware of the positions of its communication neighbors, which can be obtained through beaconing. In any wireless system, periodic beaconing is necessary for synchronization among the nodes, which can be utilized for the localization purpose.

The sensors follow a periodic sleep-wakeup based schedule to save the critical power resource. The objective of designing sleep-wakeup schedule for the sensors is to distribute the power consumption uniformly throughout the network while ensuring connected k-strip length coverage. The time domain is divided into a series of fixed sized intervals such that in every time-slot a set of nodes along $\mathbb{L}_{s,s'}$ remain in active state maintaining the *k-strip length coverage*, whereas the remaining nodes save energy in the sleep state during that time-slot. The set of active nodes in any particular time-slot form a data gathering tree rooted at the sink node, and involves in data sensing and forwarding activities. At the beginning of every time-slot, the active set of nodes for that slot are selected based on a set of performance metrics. The proposed adaptive sleep-wakeup scheduling can work cooperatively with any channel access protocol for data transmission.

TH-1251_11610108

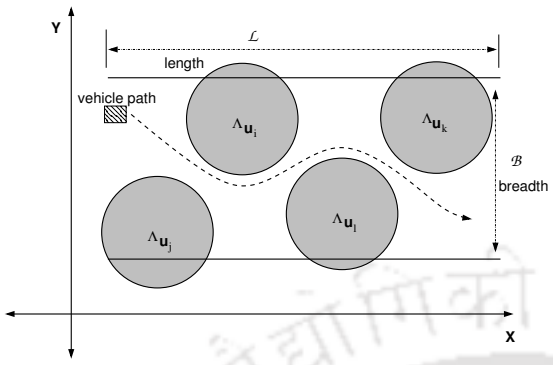

Figure 6.2: Barrier gap along the breadth

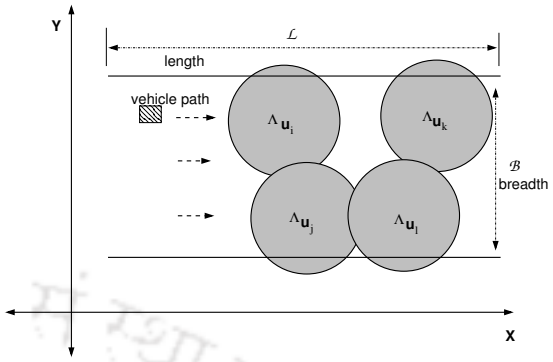

Figure 6.3: 1-barrier coverage in each barrier chain

6.2 Problem Definition and Objectives

As discussed earlier, every lane segment of a road should be *k-strip length* covered by the sensors to correctly detect the moving vehicles. To satisfy the *k-strip length coverage*, it is required that every passing vehicle is detected by at least k active sensors along $\mathbb{L}_{s,s'}$. It can be easily observed that if the sensors are deployed according to Figure 6.2, a passing vehicle may not be detected by any of the active sensors due to an existence of barrier gap. For correct detection of vehicle speed and direction, continuous sensing is required along the length of a road segment. To satisfy the *k-strip length coverage* along the length of the road, the design requirements are as follows:

- Barrier gaps created along the breadth of the lane must be avoided.
- Every passing vehicle must be detected by at least k active sensors along $\mathbb{L}_{s,s'}$.
- Along the length of the lane, the distance between any two active sensors must be bounded such that the connectivity of the network is always maintained.

Let $\mathcal{B}$ denote the breadth of a lane segment. Considering Figure 6.3, the length and the breadth of a lane segment are represented along the X-axis and the Y-axis, respectively, of a Cartesian co-ordinate system. Let the sensors be projected over the X-axis and the Y-axis, and $\mathcal{D}_X(u)$ and $\mathcal{D}_Y(u)$ be the projected sensing diameter of sensor u along the X-axis and the Y-axis, respectively. To satisfy the desired coverage such that there exists no barrier gap along the breadth of the road segment, it is required to form barrier chains with the active sensors as shown in Figure 6.3. Each barrier chain supports 1-barrier coverage breadth-wise, and existence of k such barrier chains along the length of the road

TH-1251_11610108

segment assures the required *k-strip length coverage* along the road segment. Moreover, all these barrier chains must be interconnected to offer the desired connectivity into the network. To find the optimal formation of the barrier chains with minimum number of sensors, the active sensors should be selected in such a way so that for any two adjacent active nodes within a barrier chain, the length-wise overlapping of the coverage area is maximum, and the breadth-wise overlapping of the coverage area is minimum. The active nodes are selected based on the following conditions.

C1 : For an active sensor u, another sensor $v \in sNeighbor(u)$ should get priority to become active, if $\mathcal{D}_X(u) \cap \mathcal{D}_X(v) > \mathcal{D}_X(u) \cap \mathcal{D}_X(w)$; $\forall w \in sNeighbor(u) \setminus \{v\}$.

C2 : For an active sensor u, another sensor $v \in sNeighbor(u)$ should get priority to become active, if $\mathcal{D}_Y(u) \cap \mathcal{D}_Y(v) < \mathcal{D}_Y(u) \cap \mathcal{D}_Y(w)$; $\forall w \in sNeighbor(u) \setminus \{v\}$.

C3 : For all active sensors v in a barrier chain $\left(\bigcup_{v \in \mathcal{S}_{BC}} \mathcal{D}_Y(v) \right) = \mathcal{B}$, where $\mathcal{S}_{BC}$ denotes the set of active nodes in a particular barrier chain.

In the proposed system, sensor nodes periodically follow a sleep-wakeup cycle such that the active sensors maintain the *k-strip length coverage* with the connectivity along a lane segment in every time-slot. A centralized scheduling scheme, discussed in the following section, has been designed to select the active sensors efficiently for any particular time-slot. In subsequent sections, a localized distributed variant of the centralized scheduling is proposed, where every sensor can determine its state from the local information processing, such that *k-strip length coverage* as well as the network connectivity is maintained globally.

6.3 Centralized Scheduling

Let there be n number of sensors, represented as $\mathbb{U} = \{u_1, u_2, ..., u_n\}$ such that $|\mathbb{U}| = n$, along the lane segment $\mathbb{L}_{s,s'}$. The scheduling mechanism should ensure the *k-strip length coverage* as well as the network connectivity among the active sensors. This problem is represented as a hierarchical graph, called the *connected strip length coverage (CSLC)* graph, according to Figure 6.4. As discussed in the previous section, breadth-wise 1-barrier coverage needs to be ensured satisfying conditions (C1)-(C3). Each barrier-chain is represented as the inner layer of the CSLC graph where every active sensor within the chain represents a vertex. An edge exists between two active sensors, if conditions (C1) and (C2) are satisfied. The inner layer of the CSLC graph is termed as the *B-chain*. Every

TH-1251_11610108

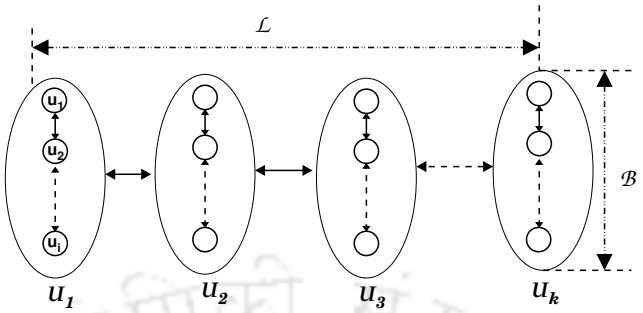

Figure 6.4: Hierarchical representation of the connected k-strip length coverage

such *B-chain* represents a vertex in the outer layer of the CSLC graph. There exists an edge between two vertices of the outer layer of the CSLC graph, if the corresponding two *B-chains* are connected. Two *B-chains* $\mathcal{U}_1$ and $\mathcal{U}_2$ are said to be connected, if there exists $u_i \in \mathcal{U}_1, u_j \in \mathcal{U}_2$, such that $u_i \in cNeighbor(u_j)$. The outer layer of the CSLC graph is termed as the *L-chain*. Then the scheduling problem for finding the optimal set of active sensors for any time-slot is reduced to the problem of finding the optimal CSLC graph, in terms of the connectivity, coverage and the number of active sensors, from the set of sensors.

Based on the CSLC graph, assuring the *connected k-strip length coverage* is equivalent to ensuring the connectivity among the k sequential *B-chains* along the *L-chain*, and *1-barrier coverage* along every *B-chain*. According to the centralized approach, for a given set of sensors, the CSLC graph is constructed in two steps. In the first step, all feasible solutions for the *L-chain* are constructed considering the connectivity with minimum *L-chain* length k. Then for every feasible solution of the *L-chain*, the optimal *B-chains* are constructed ensuring conditions (C1) and (C2). The optimal solution for the *L-chain* is determined from the set of all feasible solutions. This problem is different from the conventional connected $k-$barrier coverage, where the connectivity and the coverage are maintained in the same dimension. In this problem to satisfy connected $k-$strip length coverage, the connectivity and the coverage are required to be assured in different dimensions, where the formation of *L-chain* satisfies the connectivity with minimum path length k, and the *B-chains* are constructed based on the barrier coverage criteria. As both the *L-chain* and the *B-chain* are required to be optimized to find out the optimal CSLC graph, this problem belongs to the set of multi-objective optimization problems, which is known to be NP-hard [47]. A greedy approximation heuristic is designed for this purpose based on the interval graphs [63] and k-reachability problem [45].

TH-1251_11610108

6.3.1 Coverage Interval Stabbing

The concept of the *coverage interval stabbing (CovIS)* is introduced in this section to solve the optimal CSLC graph problem. As shown in Figure 6.5, there are 12 sensors $u_1, u_2, ..., u_{12}$ deployed along the target lane segment. Let $\langle \mathcal{C}_X(u_i), \mathcal{C}_Y(u_i) \rangle$ denote the Cartesian coordinate position of sensor u_i. Therefore, sensor u_i covers the lane segment $(\mathcal{C}_X(u_i) - R_s, \mathcal{C}_X(u_i) + R_s)$ along the length, and $(\mathcal{C}_Y(u_i) - R_s, \mathcal{C}_Y(u_i) + R_s)$ along the breadth. If all such covered lane segments are cut with the vertical lines through all the endpoints of the $\mathcal{D}_X$ and $\mathcal{D}_Y$, the *CovIS* is formed along the length and the breadth of the lane, as shown in the figure. The dotted vertical lines are called the *CovIS lines*. The *CovIS lines* provide following important information that are necessary to design the optimal CSLC graph.

(i) The *CovIS lines* divide the lane segment into subsegments (both length-wise and breadth-wise) based on the number of sensors that covers a particular subsegment. For example, the first length-wise subsegment (between the first and the second *CovIS lines*) can be covered by a maximum of two sensors, whereas the second length-wise subsegment (between the second and the third *CovIS lines*) can be covered by a maximum of three sensors, as shown in Figure 6.5(a). The number of sensors that cover a subsegment, is termed as the *CovIS number*.

(ii) The *CovIS lines* determine the amount of overlapping among the sensors.

(iii) If there are n number of sensors, then the *CovIS lines* can divide the lane segment into at most $(2n - 1)$ subsegments. Out of those subsegments, the ones with the minimum *CovIS* number are called the critical subsegments. To design a greedy heuristic for solving the optimal CSLC graph problem, the critical subsegments should be considered first to check for conditions (C1) and (C2), as they are covered by a minimum number of sensors.

The *CovIS sets* for every sensor are constructed from the *CovIS lines* along the length and the breadth of the lane. As a generic notation, let $\mathcal{D}(u_i)$ denote either $\mathcal{D}_X(u_i)$ or $\mathcal{D}_Y(u_i)$, whenever applicable. Then the *CovIS set* is defined as follows.

Definition 6.5. *Let $\mathbb{C}_l(u_i)$ denote the set of CovIS lines that intersect $\mathcal{D}(u_i)$. The CovIS set of sensor u_i, denoted as $\mathbb{S}_{cov}(u_i)$, contains all sensors u_j such that $\mathcal{D}(u_j)$ intersects the CovIS line $L_c | L_c \in \mathbb{C}_l(u_i)$.*

TH-1251_11610108

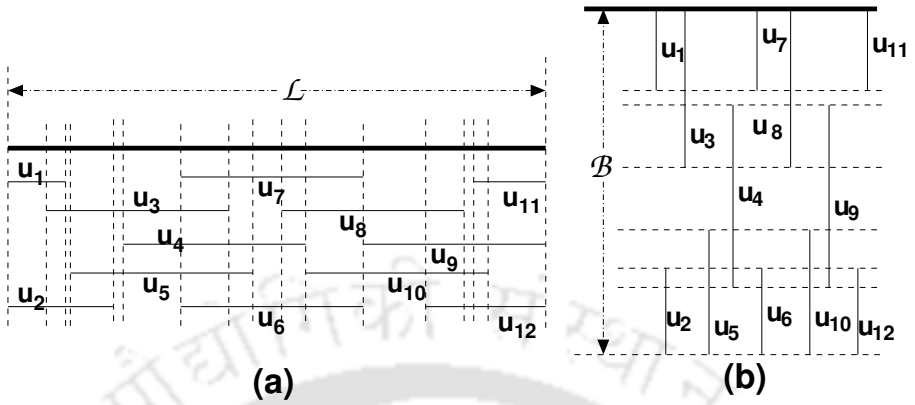

Figure 6.5: CovIS along the length and the breadth

The *CovIS set* can be either breadth-wise, denoted as $\mathbb{S}^{\mathcal{B}}_{cov}(u_i)$, when $\mathcal{D}(u_i) = \mathcal{D}_Y(u_i)$, or length-wise, denoted as $\mathbb{S}^{\mathcal{L}}_{cov}(u_i)$, when $\mathcal{D}(u_i) = \mathcal{D}_X(u_i)$. Following theorem provides the theoretical basis to compute $\mathbb{S}_{cov}(u_i)$ using a constant time algorithm.

Theorem 6.1. *Let $L^-_c(u_i)$ and $L^+_c(u_i)$ denote the leftmost and the rightmost CovIS lines that intersect $\mathcal{D}(u_i)$. Then sensor $u_j \in \mathbb{S}_{cov}(u_i)$ if and only if $\mathcal{D}(u_j)$ intersects either $L^-_c(u_i)$ or $L^+_c(u_i)$.*

Proof. Let $u_j \in \mathbb{S}_{cov}(u_i)$, and $\mathcal{D}(u_j)$ intersects $L'_c \in \mathbb{C}_l(u_i)$. Further Assume that $\mathcal{D}(u_j)$ intersects neither $L^+_c(u_i)$ nor $L^-_c(u_j)$. Let $\mathcal{P}(l_1, l_2)$ denote the perpendicular distance between the line segments l_1 and l_2. As $|\mathcal{D}(u_i)| = 2R_s$, and $\mathcal{D}(u_i)$ intersects both $L^+_c(u_i)$ and L'_c,

$$\mathcal{P}(L^+_c(u_i), L'_c) \leq 2R_s \tag{6.1}$$

Similarly,

$$\mathcal{P}(L^-_c(u_i), L'_c) \leq 2R_s \tag{6.2}$$

As, $|\mathcal{D}(u_i)| = 2R_s$,

$$\mathcal{P}(L^-_c(u_i), L^+_c(u_i)) = 2R_s \tag{6.3}$$

Now, $|\mathcal{D}(u_j)| = 2R_s$. If $\mathcal{D}(u_j)$ intersects L'_c but does not intersect $L^+_c(u_i)$ then,

$$\mathcal{P}(L'_c, L^+_c(u_i)) \geq 2R_s \tag{6.4}$$

TH-1251_11610108

From equations (6.1) and (6.4),

$$\mathcal{P}(L'_c, L_c^+(u_i)) = 2R_s \tag{6.5}$$

From equations (6.3) and (6.5), $L'_c = L_c^-(u_i)$, that contradicts the assumption.

The reverse proof is direct from Definition 6.5. □

Based on Theorem 6.1, for any target sensor u_i, $\mathbb{S}_{cov}(u_i)$ can be directly constructed by considering the *CovIS lines* that pass through the end points of $\mathcal{D}(u_i)$. This helps to reduce the complexity of the greedy approximation heuristic for solving the optimal CSLC graph.

6.3.2 Construction of the Set of Feasible Solutions for the L-chain

As the reference sensors for the *B-chains* are not known a priori, the feasible solutions for the *L-chain* are constructed based on a reference line along the length of the lane. This is because every line along the road must be covered by at least k sensors to ensure *k-strip length coverage*. For the sake of simplicity, one edge of the lane is assumed to be the reference line. According to Figure 6.5(b), the upper most *CovIS line* is considered as the reference line. The sensors that intersect this reference *CovIS line*, called the candidate sensors, are considered to determine all feasible solutions for the *L-chain*. In the figure, u_1, u_3, u_7, u_8 and u_{11} are the candidate sensors.

A connectivity graph $\mathbb{G}_{con}(V_{con}, E_{con})$ is constructed, where the set of candidate sensors form the vertex set, V_{con}. There exists a directed edge $(u_i \rightarrow u_j) \in E_{con}$ where $u_i, u_j \in V_{con}$, if $u_i \in cNeighbor(u_j)$, and $\mathcal{C}_X(u_i) < \mathcal{C}_X(u_j)$. Two more vertices, s and s', are included in V_{con}, where s and s' are the sinks at the two end points of the lane segment, as discussed earlier. There exists a directed edge $(s \rightarrow u_i) \in E_{con}$, if $s \in cNeighbor(u_i)$ and $\mathcal{C}_X(s) < \mathcal{C}_X(u_i)$. Similarly, there exists a directed edge $(u_j \rightarrow s') \in E_{con}$, if $u_j \in cNeighbor(s')$ and $\mathcal{C}_X(u_j) < \mathcal{C}_X(s')$. With this directed graph, the problem of finding all feasible solutions for the *L-chain* is equivalent to finding all possible paths of length κ in the directed graph, where $\kappa \geq k$. This problem is similar to the complement of the k-reachability problem, as described in [45], and is known to be NP-hard. However, using the 2-approximation method proposed in [45], the problem can be solved efficiently with $O(|V_{con}|^2)$ time complexity. This solves the first step of the optimal CSLC graph problem.

TH-1251_11610108

6.3.3 Optimal Solution of the B-chains

To solve the second step of the optimal CSLC graph problem, a candidate *L-chain* from the feasible solutions of the *L-chain*, denoted by χ, is considered. The second step of the optimal CSLC graph problem determines the optimal *B-chains* for all $u_i \in \chi$. For this purpose, consider the *CovIS* as described earlier. A directed acyclic graph, called the *CovIS graph*, is constructed from the CovIS sets of all sensors. Let $\mathbb{G}_{cov}(V_{cov}, E_{cov})$ denote the *CovIS graph*, where V_{cov} is the set of vertices and E_{cov} is the set of edges. The CovIS graph is constructed as follows.

First the vertex set V_{cov} is populated as follows:

$$V_{cov} = \{u_j | u_j \in \mathbb{S}^{\mathcal{B}}_{cov}(u_i) \cap \mathbb{S}^{\mathcal{L}}_{cov}(u_i)\} \cup \{u_i\}$$

For every sensor pair $u_j, u_k \in V_{cov}$, there exists a directed edge $u_j \rightarrow u_k$ if the following two conditions are satisfied,

(i) $u_k \in sNeighbor(u_j)$

(ii) $\mathcal{C}_Y(u_k) < \mathcal{C}_Y(u_j)$

A weight, $\mathcal{W}_{u_j \rightarrow u_k}$, is assigned to every directed edge $u_j \rightarrow u_k$. The weight is calculated as follows:

$$\mathcal{W}_{u_j \rightarrow u_k} = \frac{|\mathcal{C}_X(u_j) - \mathcal{C}_X(u_k)|}{|\mathcal{C}_Y(u_j) - \mathcal{C}_Y(u_k)|} \tag{6.6}$$

To minimize the breadth-wise overlapping of sensing area, the distance along Y direction between two active nodes in a barrier chain need to be maximized and vice-versa. So, minimization of the weight function as expressed in equation (6.6) reflects the conditions (C1) and (C2), described in Section 6.2. A dummy vertex u_d is also included in V_{cov}. There exists a zero weighted directed edge $u_j \rightarrow u_d$ if $\mathcal{D}(u_j)$ intersects the lower most *CovIS line*, as shown in Fig 6.5(b) (the opposite edge of the lane segment that is not considered as the reference line).

With the help of $\mathbb{G}_{cov}$, the problem of finding the optimal *B-chain* is reduced to the problem of finding the minimum weight path from vertex u_i to vertex u_d. The Goldberg-Radzik algorithm is applied for this purpose using the minimum path weight heuristics [72] (correspond to the critical subsegments, as discussed in Subsection 6.3.1), that can determine the optimal path with the time complexity $O(|V_{cov}||E_{cov}|)$.

TH-1251_11610108

Algorithm 6.1 Procedure findOPT_CSLC($\mathbb{U}$)

1. $OPT_{\chi} = NULL$
2. $OPT\_cost_{\chi} = \infty$
3. $\mathbb{F}_{\chi} \leftarrow$ findC_Lchain($\mathbb{U}$)
4. **for all** $\chi \in \mathbb{F}_{\chi}$ **do**
5. **for all** $u_i \in \chi$ **do**
6. $\mathcal{U}(u_i) \leftarrow$ findOPT_Bchain(u_i)
7. **if** $OPT\_cost_{\chi} > \aleph(\mathcal{U}(u_i)) \times \mathcal{Q}(\mathcal{U}(u_i))$ **then**
8. $OPT_{\chi} = \chi$
9. $OPT\_cost_{\chi} = \aleph(\mathcal{U}(u_i)) \times \mathcal{Q}(\mathcal{U}(u_i))$
10. **end if**
11. **end for**
12. **end for**
13. **Return** OPT_{χ}

6.3.4 Optimal Solution of the CSLC Graph

Assume $\mathcal{U}(u_i)$ denotes a optimal B-chain corresponding to the candidate sensor u_i. As discussed, $\mathcal{U}(u_i)$ can act as a outer layer vertex of the CSLC graph. Let $\mathcal{Q}(\mathcal{U}(u_i))$ denote the cost of the optimal *B-chain* corresponding to vertex $\mathcal{U}(u_i)$. The cost of a *B-chain* is the summation of the weights (calculated using equation (6.6)) of the edges in the chain. Let $\aleph(\mathcal{U}(u_i))$ denote the number of sensors in the optimal *B-chain* $\mathcal{U}(u_i)$. Further, let $\mathbb{F}_{\chi}$ denote the set of feasible solutions for the *L-chain*. As discussed, every feasible solution of the *L-chain* returns a set of candidate sensors. The optimal *L-chain*, denoted as χ^*, that corresponds to the optimal CSLC graph, is calculated based on the following unconstrained optimization.

$$\chi^* = \min_{\chi \in \mathbb{F}_{\chi}} \left\{ \left(\sum_{u_i \in \chi} (\mathcal{Q}(\mathcal{U}(u_i)) \times \aleph(\mathcal{U}(u_i))) \right) \right\} \tag{6.7}$$

χ^* returns the optimal *L-chain* corresponding to the optimal *B-chain* with minimum cost satisfying the overlapping conditions, as described in conditions (C1) and (C2) in Section 6.2. Therefore, it contains the optimal CSLC graph satisfying the *connected k-strip length coverage* along with the minimum number of sensors required to be active.

Algorithm 6.1 shows the procedure of finding the optimal CSLC graph for a set of sensors. The sub-procedure findC_Lchain($\mathbb{U}$) determines the set of candidate sensors for all feasible *L-chain* solutions. Every *L-chain* solution returns a set of candidate sensors ensuring connectivity such that minimum cardinality of the set is k. The sub-procedure

TH-1251_11610108

findOPT_Bchain(u_i) determines the optimal *B-chain* for a candidate sensor u_i. The following lemmas and theorems show the complexity of the proposed centralized heuristic.

Lemma 6.1. *$\mathbb{G}_{con}(V_{con}, E_{con})$ is a directed acyclic graph (DAG).*

Proof. Let there exists a cycle $v_1 \rightarrow v_2 \rightarrow ... \rightarrow v_k \rightarrow v_1 | \{v_1, v_2, .., v_k\} \in V_{con}$. From the construction of $\mathbb{G}_{con}$, there exists an edge $(v_i \rightarrow v_j) \in E_{con}$ if $\mathcal{C}_X(v_i) < \mathcal{C}_X(v_j)$. Therefore,

$$\mathcal{C}_X(v_1) < \mathcal{C}_X(v_2) < ... < \mathcal{C}_X(v_k) < \mathcal{C}_X(v_1)$$

which is a contradiction. Therefore, there exists no cycle in $\mathbb{G}_{con}$, and that completes the proof. □

Lemma 6.2. $|\mathbb{F}_\chi| = O(2^{|V_{con}|})$

Proof. Let $\mathbb{P}_{\mathbb{G}_{con}}$ denote the number of paths in the connectivity graph $\mathbb{G}_{con}(V_{con}, E_{con})$, from the source vertex s to the end vertex s'. The number of solutions returned by findC_Lchain($\mathbb{U}$) is equal to the number of κ-hop paths, $\forall \kappa \geq k$, which is upper bounded by $\mathbb{P}_{\mathbb{G}_{con}}$.

From Lemma 6.1, $\mathbb{G}_{con}$ is a DAG. The number of paths in $\mathbb{G}_{con}$ is computed as follows. For a vertex $v_i \in V_{con}$, let $OUT(v_i) = \{v_j | (v_i \rightarrow v_j) \in E_{con}\}$. Let there exists $\mathbb{P}_{\mathbb{G}_{con}(v_j)}$ number of paths from the vertex v_j to the end vertex s'. Then $\mathbb{P}_{\mathbb{G}_{con}}(v_i)$ is calculated recursively as follows:

$$\mathbb{P}_{\mathbb{G}_{con}}(v_i) = \sum_{v_j \in OUT(v_i)} \mathbb{P}_{\mathbb{G}_{con}}(v_j) \tag{6.8}$$

The upper bound of this recurrence is calculated using the induction over the maximum number of paths in a complete DAG without any duplicate edges. Considering a 2-node DAG, there can be a maximum of 1 path between two nodes. Now considering a 3-node DAG, the third node can be connected either of the two existing nodes, making a total of $1 + 1 = 2$ paths using the recurrence. Similarly, for a 4-node DAG, the fourth node can be connected to any one of the three existing nodes making a total of $1 + 2 + 1 = 4$ paths using the recurrence. Therefore, it can be seen that the total number of paths in a complete DAG follows a geometric series $1, 2, 4, 8, 16, 32, ...$ Therefore for $|V_{con}|$ number of nodes, the number of paths is upper bounded by $2^{|V_{con}|-2}$, which is $O(2^{|V_{con}|})$. □

Theorem 6.2. *The asymptotic running time of Algorithm 6.1 is bounded by* $O(\max\{n^2, |V_{cov}||E_{cov}||\chi|2^{|V_{con}|}\})$.

TH-1251_11610108

Proof. Using the k-reachability problem as described in [45], the procedure findC_Lchain($\mathbb{U}$) returns the candidate sensors for all the feasible solutions of the *L-chain* in $O(n^2)$ time complexity. Based on Theorem 6.1, $\mathbb{G}_{cov}$ can be constructed in a constant time. Therefore, the *Goldberg-Radzik* algorithm [72] returns the optimal solution of the *B-chains* in $O(|V_{cov}||E_{cov}|)$ time complexity.

According to Algorithm 6.1, the procedure findOPT_Bchain(u_i) needs to be enumerated for $|\chi| \times 2^{|V_{con}|}$ times, that makes the upper bound of Algorithm 6.1 to be $O(\max\{n^2, |V_{cov}||E_{cov}||\chi|2^{|V_{con}|}\})$ □

From Theorem 6.2, the running time of the centralized heuristic contains an exponential factor $2^{|V_{con}|}$. The value of this factor depends on the number of sensors on the reference line during the construction of the set of candidate sensors for all of the feasible solutions of the *L-chain* problem. This exponential factor can be reduced by using an approach of prioritizing the candidate sensors, such that only the candidate sensors with a high priority can participate in the optimal CSLC graph problem. As discussed earlier, the energy of a sensor node plays a major role in deciding the network lifetime. Therefore, the candidate sensors with the maximum remaining energy can be considered for the CSLC graph formation. Based on this heuristic, the localized distributed approach is designed, where the sensors are scheduled based on the following decision factors,

1. Remaining energy and k-connectivity for the optimal *L-chain* formation,
2. The weight function, as designed in equation (6.6), for the optimal *B-chain* formation.

It can be noted that based on the physical distribution of the sensors, there can be more than one optimal solutions of the CSLC graph problem. Therefore, the localized distributed solution may lead the network to an inconsistent state, if strict optimality criteria is followed. This is because, different sensors may try to reach in different non-convergent optimal solutions. Therefore, the strictness in the decision factors for the active sensors selection is relaxed, so that a sensor decides its own state if, either it has sufficient information, or it has waited sufficient time to obtain the information. This has been realized by designing a back-off function, as described in the following section.

6.4 Distributed Approach : Active Node Selection

As discussed earlier, every node follows a sleep-wakeup schedule in *ADCROSS* to efficiently utilize sensor battery power. In practical scenario, large number of sensors are distributed

TH-1251_11610108

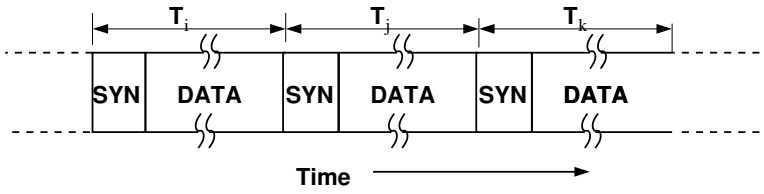

Figure 6.6: Slotted time intervals for the scheduling

along the target road, and therefore, they do not contain the global topology information. In a distributed environment, based on the computed relative positions of the neighbors, every sensor independently participates in active node selection procedure. The localized distributed scheduling scheme for active node selection has been developed on the basis of the centralized scheduling theory discussed in the previous section. The selection procedure aims to choose a set of sensors that can produce a near-optimal solution to the problem of finding the minimum number of active nodes to satisfy the desired *connected k-strip length coverage* along the lane segment $\mathbb{L}_{s,s'}$.

6.4.1 Basics

The time domain is divided into a series of time-slots of fixed length as shown in Figure 6.6. Each slot is further subdivided into two parts, *SYN* and *DATA*. During the initial *SYN* interval (on triggering of the *SYN_BEGIN* signal), every node participates in the active node selection procedure by deciding its state for the corresponding time-slot. Also, the activities at every node for data gathering tree construction is performed in the *SYN* interval. The active set of nodes participate in the sensing and the forwarding activities for the *DATA* interval of the slot, whereas the remaining nodes go to the sleep state to save energy during that time. To decide the state of a node, every sensor computes a back-off function, denoted by $\Xi_u^{T_i}$, that returns a timeout interval. Once the timeout occurs, every sensor decides its state for the next DATA interval. The back-off function is designed based on the decision factors - the remaining energy, connectivity with minimum path length k and the weight function. The sensors that are more competent for being active in the next DATA interval, compute a small back-off value, and therefore, are prioritized to take the decision for being active. As every sensor executes the same procedure to compute the back-off function, the proposed localized distributed scheduling scheme has been described in view of a particular sensor node u. Let for two nodes u and v, such that $u \in sNeighbor(v)$, if $\Xi_u^{T_i} < \Xi_v^{T_i}$, the waiting time for u is also less than that of v. Node u then broadcasts a *Wakeup* message in its 1-hop neighborhood declaring its active

TH-1251_11610108

state for the *DATA* interval of T_i, before node v could do the same. Hence, on receiving the *Wakeup* message from u, node v re-computes its back-off to check whether it is still required to become active or not. The basic theory of the proposed distributed scheduling for the active node selection, executed at every node, is provided by Algorithm 6.2.

Let the projected Euclidean distance between node u and node v along X-axis is $d^X_{u,v}$, and along Y-axis it is $d^Y_{u,v}$ (as a generic notation $d_{u,v}$). So, considering two nodes u and v, $d^X_{u,v} = |\mathcal{C}_X(u) - \mathcal{C}_X(v)|$ and $d^Y_{u,v} = |\mathcal{C}_Y(u) - \mathcal{C}_Y(v)|$. To maintain the connectivity as well as the *k-strip length coverage* along the road segment $\mathbb{L}_{s,s'}$, the distributed design of the problem puts a bound on $d^X_{u,v}$ such that $d^X_{u,v} \leq \mathcal{L}/k \leq R_c$ is always true. It follows that if $\mathbb{L}_{s,s'}$ is length-wise divided into k number of cells with a cell length of $\mathcal{L}/k$, *connected k-strip length coverage* will be assured. Considering the application to be road surveillance, it is necessary to enforce the *k-strip length coverage* into the network. As the complete area coverage is not required, it is sufficient to keep the lower bound of the Euclidean distance, $d_{u,v}$ between any two active sensors u and v as R_s. Let there exist two sets of nodes in $cNeighbor(u)$, denoted by $\mathcal{G}_O(u)$ and $\mathcal{G}_N(u)$, such that $\forall w \in \mathcal{G}_O(u), \mathcal{D}^X_w \cap \mathcal{D}^X_u \neq \phi$, and $\forall w \in \mathcal{G}_N(u), \mathcal{D}^X_w \cap \mathcal{D}^X_u = \phi$. Then the closest neighbors of u that are active, along the direction of both the X-axis and the Y-axis can be represented by four node variables, $u_{x+}, u_{x-}, u_{y+}, u_{y-} \in cNeighbor(u)$, such that the following is true. The initial values of these variables are set as $u_{x+} = u_{y+} = \infty, u_{x-} = u_{y-} = 0$.

- $u_{x+} \in \mathcal{G}_N(u) | \mathcal{C}_X(u_{x+}) = \min\{\forall w \in \mathcal{G}_N(u) | \mathcal{C}_X(w) > \mathcal{C}_X(u)\}$
- $u_{x-} \in \mathcal{G}_N(u) | \mathcal{C}_X(u_{x-}) = \min\{\forall w \in \mathcal{G}_N(u) | \mathcal{C}_X(w) < \mathcal{C}_X(u)\}$
- $u_{y+} \in \mathcal{G}_O(u) | \mathcal{C}_X(u_{y+}) = \min\{\forall w \in \mathcal{G}_O(u) | \mathcal{C}_Y(w) > \mathcal{C}_Y(u)\}$
- $u_{y-} \in \mathcal{G}_O(u) | \mathcal{C}_X(u_{y-}) = \min\{\forall w \in \mathcal{G}_O(u) | \mathcal{C}_Y(w) < \mathcal{C}_Y(u)\}$

6.4.2 Back-off Computation

The back-off function basically prioritizes the nodes that are more competent to become active. At the beginning of SYN interval of time-slot T_i, the signal SYN_BEGIN returns True at every node u according to Algorithm 6.2. Node u then computes its back-off, $\Xi^{T_i}_u$ to determine its competence priority. $\Xi^{T_i}_u$ is a function of four decision factors to find the optimal L-chains and B-chains as stated in Section 6.3. These are the energy factor $\mathscr{U}$, the connectivity factor $\mathscr{C}$, the breadth-cover factor $\mathscr{B}$, and the length-cover factor $\mathscr{L}$. To put more priority on a competent node for being active, the objective is to minimize $\Xi^{T_i}_u$.

TH-1251_11610108

Algorithm 6.2 Active node selection procedure at node u

```
1. if SYN_BEGIN then
2.     Compute Ξ_u^{T_i}
3. end if
4. if END_BACK_OFF then
5.     if d^Y_{u,u_{y+}} > 2R_s then
6.         active ← True
7.         Send Wakeup to w; ∀w ∈ cNeighbor(u)
8.     else
9.         if {d^X_{u_{x−},u} > L/k} ∨ {d^X_{u,u_{x+}} > L/k} then
10.            active ← True
11.            Send Wakeup to w; ∀w ∈ cNeighbor(u)
12.        else
13.            active ← False
14.        end if
15.    end if
16. end if
```

This is because a small back-off value at node u will enforce it to decide its active state before all its neighbors. To make the node selection consistent in 1-hop neighborhood, the back-off value is normalized before applying it to count the actual back-off timeout for a node. Thus, all the factors $\mathscr{U}, \mathscr{C}, \mathscr{B}$ and $\mathscr{L}$ are estimated in normalized form. The design of the back-off function as computed is shown in equation (6.9).

$$\Xi_u^{T_i} = 1 - (\mathscr{U} + \mathscr{C} + \mathscr{B} + \mathscr{L}) \tag{6.9}$$

The computation details of the function parameters $\mathscr{U}, \mathscr{C}, \mathscr{B}$ and $\mathscr{L}$ have been described as follows.

Energy Factor:

From Subsection 6.3.4, the optimal L-chain is formed by selecting the candidate sensors with the maximum residual energy. Let the residual energy at node u, at the time of $\Xi_u^{T_i}$ computation, be $\mathcal{E}_u^r$. The energy factor at node u can be defined as stated in equation (6.10). For an active node u, the nodes that are within the disc Λ_u are the potential competitors of u for being active in T_i. Hence, if $\mathcal{E}_u^r > \mathcal{E}_w^r;\ \forall w \in sNeighbor(u)$, the competence priority of node u will be high, as

TH-1251_11610108

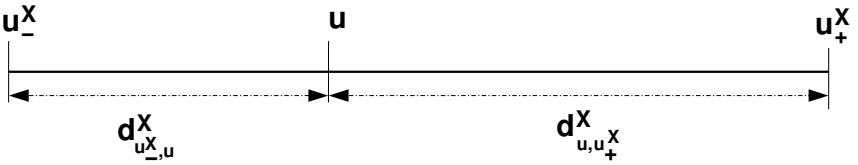

Figure 6.7: Computation of connectivity factor

well as the $\mathscr{U}$ value will be high. So, to minimize $\Xi_u^{T_i}$, it is required to maximize $\mathscr{U}$.

$$\mathscr{U} = \frac{\mathcal{E}_u^r}{\left(\sum_{z \in sNeighbor(u)} \mathcal{E}_z^r\right)} \tag{6.10}$$

Connectivity Factor:

From Subsection 6.3.4, the computation of optimal L-chain also depends on the connectivity criteria. As stated in Subsection 6.4.1, to maintain the network connectivity with *k-strip length coverage*, $d_{u,v}^X \leq (\mathcal{L}/k) \leq R_c$. As shown in Figure 6.7, node u is at $d_{u_{x-},u}^X$ and $d_{u,u_{x+}}^X$ distance apart from the two active nodes u_{x-} and u_{x+}, respectively. Also, no node is active in between u_{x-} and u, as well as between u and u_{x+}, along the X-axis according to the assumption. As the value of $d_{u_{x-},u}^X$ decreases, the competence priority of node u also decreases, and vice-versa. To minimize $\Xi_u^{T_i}$, it is required to maximize $\mathscr{C}$. As $d_{u_{x-},u}^X$ is upper bounded by $\mathcal{L}/k$, to select node u, the factor $\left(\mathcal{L}/k - d_{u_{x-},u}^X\right)^2$ should produce a low value. Considering both the active nodes u_{x-} and u_{x+} in either direction along the X-axis for node u, the objective is to minimize Z_u, where Z_u can be expressed in the form of a second order norm, as given in equation (6.11).

$$Z_u = \sqrt{\left(\mathcal{L}/k - d_{u_{x-},u}^X\right)^2 + \left(\mathcal{L}/k - d_{u,u_{x+}}^X\right)^2} \tag{6.11}$$

Now, all the nodes $w \in sNeighbor(u)$ are the potential competitors of node u for being active in T_i. So, u also calculates Z_w; $\forall w \in sNeighbor(u)$, and the ratio of these two values is taken to estimate the connectivity factor $\mathscr{C}$. To minimize $\Xi_u^{T_i}$, it is required to maximize $\mathscr{C}$ as expressed in equation (6.12).

$$\mathscr{C} = \frac{1/Z_u}{\sum_{w \in sNeighbor(u)} (1/Z_w)} \tag{6.12}$$

TH-1251_11610108

Breadth wise no barrier gap along the road segment can be ensured if the length-wise overlapping is large, and the breadth-wise overlapping is small between the sensing covered area of any two adjacent nodes in a barrier chain, as shown in Figure 6.8. Computation of the optimal B-chains depends on the validation of conditions (C1)-(C3) as stated in Section 6.2. Now, in distributed environment, any competent node ensures these conditions during the back-off calculation by computing the *breadth-cover factor* ($\mathscr{B}$) and the *length-cover factor* ($\mathscr{L}$).

Breadth-cover Factor:

From the condition (C2) of Section 6.2, the objective is to minimize the value of $\mathcal{D}_Y(u) \cap \mathcal{D}_Y(u_{y+})$. Under the coverage constraint, $d_Y(u, u_{y+}) \leq 2R_s$. So, if u_{y+} is already active, the selection of u as an active node would be the most appropriate, if the computed J_u is the minimum among all the potential competitors of node u, where J_u can be estimated as shown in equation (6.13).

$$J_u = (|2R_s - d_Y(u, u_{y+})|) \tag{6.13}$$

Let $\mathbb{Y}$ be the set of potential competitors of node u for being active in T_i, such that $\mathbb{Y} = \{w | \forall w \in sNeighbor(u) \text{ and } \Lambda_{u_{y+}} \cap \Lambda_w \neq \phi\}$. Therefore, along with J_u, u also calculates J_w, $\forall w \in \mathbb{Y}$, and the ratios of these two values are taken to compute the breadth-cover factor $\mathscr{B}$. As the objective is to minimize the back-off function, and therefore, to maximize the breadth-cover factor, $\mathscr{B}$ can be expressed as shown in equation (6.14).

$$\mathscr{B} = \frac{1/J_u}{\sum\limits_{w \in \mathbb{Y}} (1/J_w)} \tag{6.14}$$

Length-cover Factor:

From the condition (C1) of Section 6.2, the objective is to maximize the value of $\mathcal{D}_X(u) \cap \mathcal{D}_X(u_{y+})$ as well as the value of $\mathcal{D}_X(u) \cap \mathcal{D}_X(u_{y-})$, where both the nodes u_{y+} and u_{y-} are already active. If $\mathcal{D}_X(u) \cap \mathcal{D}_X(u_{y+}) = \phi$, $d^X_{u,x} = 2R_s$ is the maximum. So, $\mathcal{D}_X(u) \cap \mathcal{D}_X(u_{y+}) \leq 2R_s$, and also, $\mathcal{D}_X(u) \cap \mathcal{D}_X(u_{y-}) \leq 2R_s$. Considering both the cases where $\mathcal{C}_X(u) > \mathcal{C}_X(u_{y+})$ (or $\mathcal{C}_X(u) > \mathcal{C}_X(u_{y-})$) and $\mathcal{C}_X(u) < \mathcal{C}_X(u_{y+})$ (or $\mathcal{C}_X(u) < \mathcal{C}_X(u_{y-})$), the objective is to minimize the value of $|\mathcal{C}_X(u) - \mathcal{C}_X(u_{y+})| + |\mathcal{C}_X(u) - \mathcal{C}_X(u_{y-})|$ under the coverage constraint. Therefore, the selection of u will be the most appropriate in its neighborhood if condition (C1) is satisfied for both the nodes u_{y+} and u_{y-}. To minimize the back-off function,

TH-1251_11610108

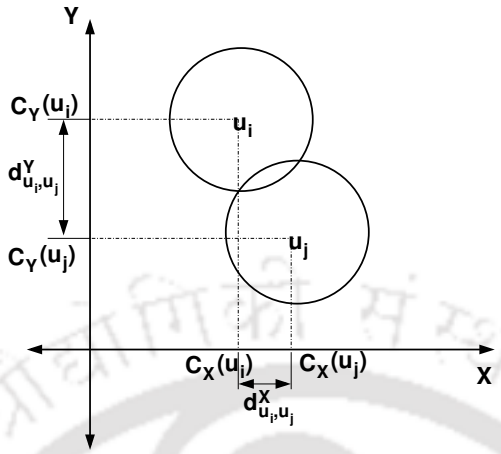

Figure 6.8: Computing breadth-cover and length-cover factors

the length-cover factor, $\mathscr{L}$ has to be maximized, and so, the factor H_u has to be maximized. H_u is estimated as given in equation (6.15).

$$H_u = \sum_{w \in \{u_{y+}, u_{y-}\}} 2R_s - |\mathcal{C}_X(u) - \mathcal{C}_X(w)| \tag{6.15}$$

To select node u, its computed back-off must be the minimum among all the potential competitors in its sensing neighborhood. Every node computes H_u for itself as well as for all nodes in $sNeighbor(u)$, and the ratio of these two values are considered to compute $\mathscr{L}$ as stated in equation (6.16).

$$\mathscr{L} = \frac{H_u}{\sum_{w \in sNeighbor(u)} H_w} \tag{6.16}$$

6.4.3 Scheduling Strategy

Once a node decides to become active, it broadcasts a *Wakeup* message in its 1-hop communication neighborhood as mentioned in Algorithm 6.2. Let a node $u \in cNeighbor(v)$ receives the *Wakeup* message from v at the SYN interval of the time-slot T_i. Let node u be not active at this point, and is following the back-off timer according to already computed $\Xi_u^{T_i}$. On receiving a *Wakeup* from v, u first stores the $\Xi_u^{T_i}$ value in the variable Ξ', and assigns the variables $u_{x+}, u_{x-}, u_{y+}, u_{y-}$ according to their corresponding values as stated in Algorithm 6.3. Then it re-computes the back-off function, as denoted by $\Xi_u^{T_i}$ in Algorithm 6.3. The back-off timer is set to a value equal to the difference of

TH-1251_11610108

Algorithm 6.3 On receiving *Wakeup* from node v by node u

1. $\Xi' \leftarrow \Xi_u^{T_i}$ /*Store the elapsed time of the back-off value*/
2. **if** $\{\mathcal{C}_X(v) < \mathcal{C}_X(u_{x+})\} \wedge \{\mathcal{C}_X(u) < \mathcal{C}_X(v)\}$ **then**
3. $\quad u_{x+} \leftarrow v$
4. **end if**
5. **if** $\{\mathcal{C}_X(v) > \mathcal{C}_X(u_{x-})\} \wedge \{\mathcal{C}_X(u) > \mathcal{C}_X(v)\}$ **then**
6. $\quad u_{x-} \leftarrow v$
7. **end if**
8. **if** $\{\mathcal{C}_Y(v) < \mathcal{C}_Y(u_{y+})\} \wedge \{\mathcal{C}_Y(u) < \mathcal{C}_Y(v)\}$ **then**
9. $\quad u_{y+} \leftarrow v$
10. **end if**
11. **if** $\{\mathcal{C}_Y(v) > \mathcal{C}_Y(u_{y-})\} \wedge \{\mathcal{C}_Y(u) > \mathcal{C}_Y(v)\}$ **then**
12. $\quad u_{y-} \leftarrow v$
13. **end if**
14. Compute $\Xi_u^{T_i}$
15. $\Xi_u^{T_i} \leftarrow \Xi_u^{T_i} - \Xi'$

these two computed back-offs. On timeout of the back-off timer, the END_BACK_OFF signal returns True. Node u then checks whether it is required to become active or not. First it checks whether the breadth-cover is satisfied according to lines 6-7 of Algorithm 6.2. If node u is required to become active to assure the breadth-coverage, *active* variable is set to True, and u broadcasts a *Wakeup* message in $cNeighbor(u)$. Otherwise, it checks whether the connectivity constraint is satisfied or not according to lines 9-11 of Algorithm 6.2. From all these conditions, if node u finds that it is not necessary for u to become active, it decides to sleep for the $DATA$ interval of the corresponding time-slot, and thus, sets *active* to False according to lines 12-13 of Algorithm 6.2.

6.4.4 Initialization

Initially, node u, for which there exists no active node in $cNeighbor(u)$, computes its $\Xi_u^{T_0}$ by considering $d^X_{u_{x-},u}$ and $d^X_{u,u_{x+}}$ as a uniformly generated random number in equation (6.11). While computing the breadth-cover factor, it assumes the node u_{y+} be positioned over the boundary of the lane from equation (6.14). Also, for the computation of length-cover factor, node u assumes the value of $|\mathcal{C}_X(u) - \mathcal{C}_X(w)|$ to be 0 in equation (6.15), and thus, the computed $\mathscr{L}$ factor contributes the maximum to the calculation of $\Xi_u^{T_0}$. If the computed $\Xi_u^{T_0}$ from equation (6.9) produces a negative value, it is considered as 0 (the highest priority), and the corresponding node becomes active with an immediate effect.

TH-1251_11610108

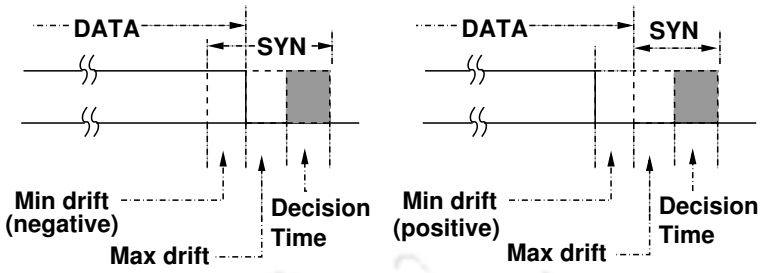

Figure 6.9: Effect of clock drift over SYN interval

6.4.5 Adjusting SYN-DATA Intervals

As discussed, every sensor needs to be active in the SYN interval for necessary synchronization and scheduling. Because of the clock drift and the absence of a global clock, it may be possible that the SYN intervals of different sensors do not overlap, which may lead to an inconsistency in the system. This problem can be resolved by a proper adjustment of SYN-DATA intervals, such that the clock drifts at the inactive nodes (the nodes that are in the sleep state) do not affect their SYN intervals. From the specification of the clock accuracy of all the available sensor motes, the maximum and the minimum clock drifts can be calculated a priori. Let ϑ_{max} and ϑ_{min} denote the maximum and the minimum clock drifts per second, respectively. Moreover, at least two nodes within the distance R_c need to be in the active state to maintain the connectivity. Therefore, once a sensor decides to be in the active state, the *B-chain* formation can be completed within a constant time. Further, 2-hop communication is sufficient to decide the candidates for the *L-chain* formation, because based on the active nodes at the left *B-chain* and at the right *B-chain*, a sensor node can decide whether it should remain in the active state to maintain k-connectivity. Let $\boldsymbol{T}_{DECS}$ denote the time for decision making, which equals to the 2-hop propagation delay plus a small constant time for computation and tolerance interval. Further assume that $\boldsymbol{T}_{DATA}$ and $\boldsymbol{T}_{SYN}$ denote the DATA and the SYN intervals respectively. $\boldsymbol{T}_{DATA}$ is determined based on the maximum clock drift such that the scheduling synchronization is maintained. Considering Figure 6.9, the minimum clock drift can be positive or negative. The SYN size should be adjusted in such a way, that in spite of the positive or negative clock drift, every sensor should remain in the wakeup state for the $\boldsymbol{T}_{DECS}$ time duration. Therefore,

$$\boldsymbol{T}_{SYN} \geq \boldsymbol{T}_{DATA}(\vartheta_{max} - \vartheta_{min}) + \boldsymbol{T}_{DECS} \tag{6.17}$$

The effect of the SYN size has been later analyzed using simulation results.

TH-1251_11610108

6.5 Tree based Data Collection

After selection of active nodes in SYN interval of a time-slot T_i, all active nodes form data gathering trees rooted at the sink node. Every active node u within the lane segment $\mathbb{L}_{s,s'}$ is included into two trees, one rooted at sink s and the other rooted at sink s', where s and s' are the two nearest sinks of the node u. This is because, to assure the required reliability in delivery of sensory data as well as for the statistical analysis, both in-flow and out-flow of data, with respect to any sensor is required to be collected. Let s and s' be positioned in the left side and the right side of a node u, respectively. The data gathering trees rooted at s and s' be denoted as $\mathbb{T}_s$ and $\mathbb{T}_{s'}$, respectively. The parent variables for the corresponding trees are represented as $Lparent(u)$ and $Rparent(u)$ for the node u, whereas the sinks for the corresponding trees are represented by $Ldest$ and $Rdest$. As the *Wakeup* message is broadcast in 1-hop communication neighborhood during the node selection phase, the active recipients of the messages set their parents according to the following conditions.

1. If $\exists v \in cNeighbor(u) | \mathcal{C}_X(v) < \mathcal{C}_X(u)$ and $d^X_{u,v} > d^X_{u,w}$; $\forall w \in cNeighbor(u)$, where $w \neq v$, set $Lparent(u) = v$
2. If $\exists z \in cNeighbor(u) | \mathcal{C}_X(z) > \mathcal{C}_X(u)$ and $d^X_{u,z} > d^X_{u,w}$; $\forall w \in cNeighbor(u)$, where $w \neq z$, set $Rparent(u) = z$

The tree construction is completed within the SYN interval before the sensory data collection starts in $DATA$ interval of each time-slot. The routing path establishment through data gathering tree construction does not require the transmission of any control message apart from the *Wakeup*. So, the cost of the tree based data collection, using the proposed localized distributed scheduling, depends only on the local computation at node level and the transmission cost of one control message, which is nominal. Let $\mathcal{M}$ be a sensory data packet. For reliable delivery of M to the sinks, node u acts as follows:

- For each $\mathcal{M}$, where $\mathcal{M}$ is self-generated, node u does the following:
  - Append $Ldest$ with one copy of $\mathcal{M}$, and forwards it to $Lparent(u)$.
  - Append $Rdest$ with another copy of $\mathcal{M}$, and forwards it to $Rdest(u)$.
- If $\mathcal{M}$ is received from another node $w \in cNeighbor(u)$ with destination sink identifier (denoted as sid), node u does the following:
  - If $sid = Ldest$, u forwards the corresponding packet $\mathcal{M}$ to $Lparent(u)$.

TH-1251_11610108

– If $sid = Rdest$, u forwards the corresponding packet $\mathcal{M}$ to $Rparent(u)$.

So, every sensory data packet is forwarded to two different sinks at the cost of double redundancy in received sensory data. As a result, the reliability in delivery of the sensory data is increased. In case of any arbitrary node failure, as the data forwarding in one path gets disrupted, the data packets can be delivered through another path reliably. According to the proposed scheme, all nodes perform a sleep-wakeup based schedule to select the active nodes for each time-slot. Therefore, any arbitrary node failure is handled by the *ADCROSS* protocol by simply discarding the faulty node during the active node selection period (during SYN interval). At the beginning of SYN interval, all nodes synchronize through 1-hop beaconing. In the worst case scenario, data forwarding activities may be stalled for a maximum of $DATA+SYN$ duration. The periodic scheduling for active node selection ensures that no communication hole or barrier gap is created into the network due to an arbitrary node failure. However, the protocol performs correctly until there exists enough redundant nodes to replace a failed node assuring both the connectivity and coverage into the network. It can be noted that the probability of multiple node failures in close proximity is very less as already discussed in Section 3.3 of Chapter 1. In such a case, if both the paths from a particular node towards two different sinks are affected due to multiple node failure, the paths are re-established during the active node selection in SYN interval of the following time-slot. So, in the worst case for an active node, if there exists no path to any of the sinks due to node failure, the maximum repairing delay could be equal to the duration of $DATA$ interval. In this way, the *ADCROSS* protocol can support any arbitrary node failure.

6.6 Simulation Results

The proposed *ADCROSS* protocol is simulated using NS-2.35 network simulation framework [4]. As *ADCROSS* focuses on the communication and the coverage of the network rather than the physical data sensing, a generic highway scenario is considered for the simulation purpose. The sensors are placed on the highway based on the Poisson distribution with a mean sensor distance that varies according to simulation scenarios. Different parameters, used for different simulation scenarios, are explained as and when required. However, generally every active sensor uses CSMA based channel access protocol for communication with the neighbors. The sensory data is considered to be small UDP packets with mean packet size of 64 bytes. The UDP packet generation rate depends on the sensing frequency that follows a log-normal distribution with mean sensing frequency

TH-1251_11610108

Table 6.1: Communication parameters of the sensor motes

Parameter	**value**
Data rates	250 Kbps
Transmit (TX) Power	0 dBm
Receive (RX) Power	-24 dBm
RX sensitivity	-94 dBm
Communication range	30 m
Sensing Range (magnetic sensors)	5 m
Current Draw (RX)	19.7 mA
Current Draw (TX)	17.4 mA
Current Draw (Idle)	15 mA
Current Draw (Sleep)	1 μA
External Voltage	2.7 Volt

as 10 events/second. The interface buffer size at every node is considered equal to the average bandwidth delay product. The sensors communicate with each other using 802.15.4 complaint MicaZ technology, that can support up to 250 Kbps data rate using 2.4 GHz channel bandwidth. The sink positioning is specific to the simulation scenario. The power consumption of every sensor during communication is considered according to the standard MicaZ motes [3], as shown in Table 6.1. With these settings, the average power consumption for the receive, transmit, idle and sleep modes are 53.19 mW, 46.98 mW, 40.5 mW and 0.0027 mW respectively. It can be noted that the Crossbow micaZ technology provides a flexible communication platform for the sensor nodes, above which the specific sensor boards (like magnetic sensors) can be mounted.

6.6.1 Analysis of the k-Strip Length Coverage

In this set of simulation scenarios, the effect of the *strip length coverage* for the road sensor network is analyzed. Let for *k-strip length coverage*, the *'coverage bound factor'* is defined as $k/\lceil \mathcal{L}/R_c \rceil$. As $k \geq \frac{\mathcal{L}}{R_c}$, this factor determines the frequency of the sensing events along the road. Increasing the value of this factor increases the coverage requirements, and therefore, the road is required to be covered by more number of sensors. Next, let the *'coverage factor'* be defined as $\lceil \mathcal{L}/R_c \rceil$. This factor also determines the number of sensors required to cover a region. A large value of this factor implies that more number of sensors are required to completely cover an area. This factor is used to compare the *k-strip length*

TH-1251_11610108

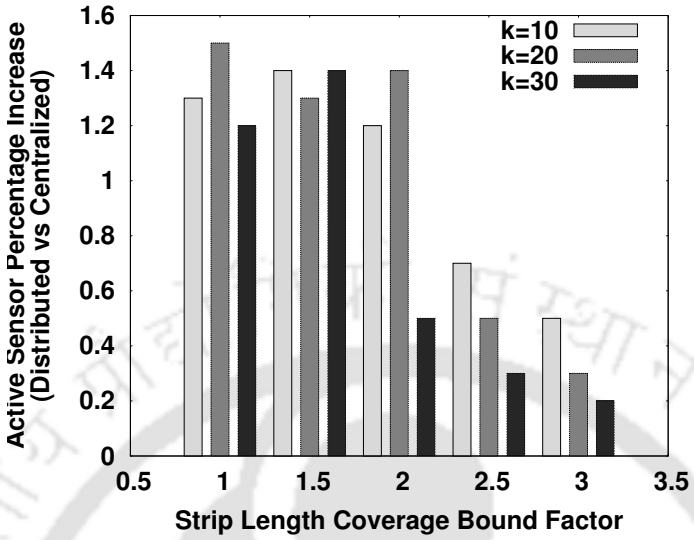

Figure 6.10: Effect of the distributed approach

coverage with the area coverage.

For this experiment, the value of k and R_c is kept fixed, and the value of $\mathcal{L}$ is varied. The value of R_c and R_s is taken as similar to the ones given in Table 6.1. The width of the road[1] is fixed to $15m$. The coverage factors are analyzed for two different cases - the *coverage factor* of the proposed localized distributed scenario with respect to the centralized optimal scenario, and the *coverage factor* of the proposed *k-strip length* scenario with respect to the full area coverage scenario.

k-Strip Length Coverage (Distributed vs Centralized):

Figure 6.10 shows the percentage increase in the number of active sensors for the localized distributed approach, compared to the centralized heuristic, with respect to the *coverage bound factor*. Let for a specific scenario, the centralized approach returns N_C numbers of active sensors, whereas the localized distributed approach returns N_D number of active sensors. Then the *'percentage increase'* is calculated as $(N_D - N_C)/N_C \times 100\%$. As $k \geq \lceil \mathcal{L}/R_c \rceil$, the *coverage bound factor* $= 1$ implies the scenario that requires minimum number of sensors to ensure the *k-strip length coverage*. The figure shows that the *percentage increase* is less than 1.5%. Therefore, in the worst case scenario, the localized distributed approach requires only 1.5% more number of sensors compared

[1]The terms *lane* and *road* are used interchangeably in the simulation descriptions. In general a road is considered to be of single lane, unless mentioned otherwise for specific scenarios

TH-1251_11610108

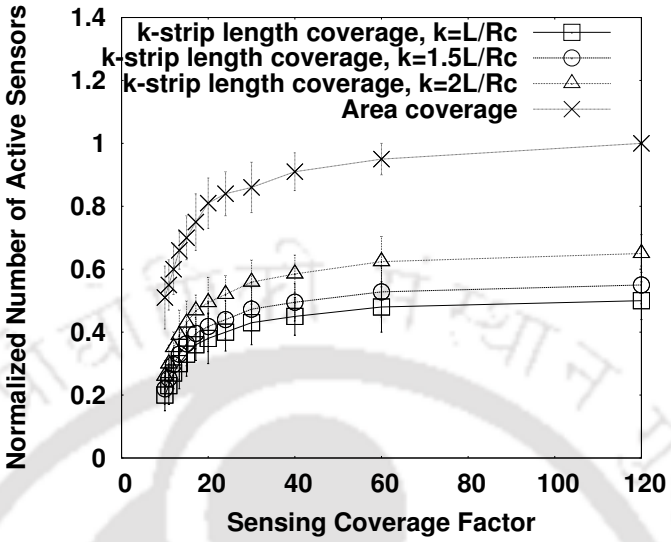

Figure 6.11: k-strip length coverage vs full area coverage

to the centralized heuristic that finds the optimal number of sensors to satisfy the *k-strip length coverage*. The figure reveals that the *percentage increase* is maximum when the number of sensor requirement is minimum. The *percentage increase* decreases as the number of sensor requirement increases.

k-Strip Length Coverage vs Full Area Coverage:

For this experiment, consider three different value of k, which are $\lceil \mathcal{L}/R_c \rceil$, $1.5\lceil \mathcal{L}/R_c \rceil$ and $2\lceil \mathcal{L}/R_c \rceil$. Figure 6.11 shows the comparison between *k-strip length coverage* and full area coverage with respect to the *coverage factor*. In this case, the localized distributed approach proposed in [69] is used for determining the number of sensors required for full area coverage in the same road segment. The simulation scenario is executed for 10 times with 10 different set of sensor positions according to the Poisson distribution with mean sensor distance as $R_s/2$. The average of these 10 scenarios are used to plot the graphs. The difference between the maximum outcome and the minimum outcome, called the *confidential interval*, is also shown in the graph using a vertical line. The Y-axis of the graph shows the normalized number of active sensors, calculated as the ratio between the outcome from individual experiment, to the maximum outcome. It can be seen from the figure that the maximum number of sensors are required for the full area coverage when the *coverage factor* is the maximum. Further, the figure reveals that the number of active sensors required for the full area coverage is significantly high compared to the number

TH-1251_11610108

of active sensors required for the *k-strip length coverage*. With an increase the value of k, the number of actives sensors required increases. This figure clearly motivates to use *k- strip length coverage* in the road sensor network for the purpose of vehicle detection and monitoring, from networking perspective to increase the network lifetime by using less number of active sensors for communication.

6.6.2 Performance Analysis of ADCROSS and Comparison with Other Protocols

This subsection analyzes the performance of the *ADCROSS* protocol for the scheduling and the data gathering in sensor network, and compares its performance with other state-of-the-art works. For comparison purpose, three other schemes have been considered as follows:

1. *SMAC* [193]: This is the most well adopted protocol for sleep-wakeup scheduling in sensor network. In SMAC protocol, every sensor waits for a random amount of time, and then broadcast a sleep-wakeup scheduling that the neighboring sensors follow. However, the SMAC protocol does not consider the coverage and connectivity criteria during scheduling.

2. *Coverage Aware Sleep Schedule* [161]: In this protocol, a sensor node goes to sleep mode if its coverage area is already covered by some other sensor nodes. The protocol works on the calculation of a probability factor with the concept, that the greater the overlap of a sensor's coverage with its neighbor, the higher the probability that it can go to the sleep state. The protocol follows a cluster based approach, where the cluster head determines the schedule for the sensors in a cluster.

3. *PEDAMACS* [59]: This protocol has been proposed for road sensor network [51], where the sensor nodes forward their topology information to the sinks, and the sinks determine the scheduling.

For this set of experiments, $\mathcal{L}$ and B for the road are taken as 150 m and 15 m, respectively, and R_s and R_c are considered similar to the values given in Table 6.1. As earlier, every simulation setup is executed for 10 different times with the sensor positions varying based on a Poisson distribution with mean sensor distance as $R_s/2$. It is assumed that $k = \lceil \mathcal{L}/R_c \rceil = 5$. For the *ADCROSS* protocol, the SYN and the $DATA$ intervals are considered as 2 ms and 98 ms, respectively. For all other three protocols, the target

TH-1251_11610108

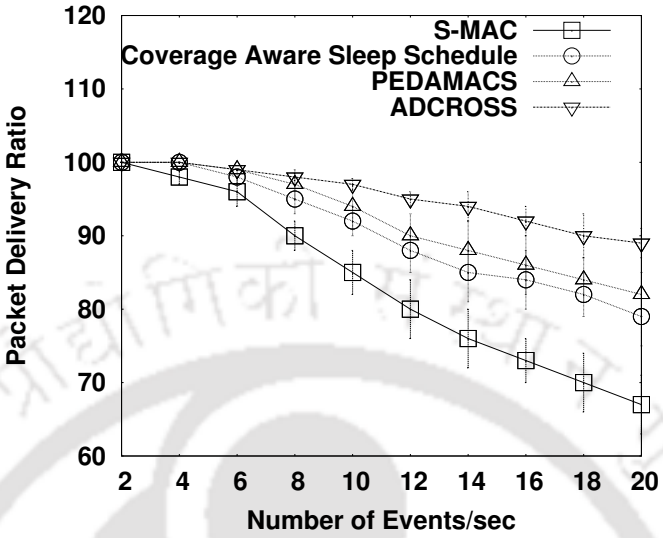

Figure 6.12: Average Packet Delivery Ratio

scheduling time is set to 100 ms, that is after 100 ms of data transmission, the nodes are rescheduled.

Effect on the Packet Delivery:

Figure 6.12 compares the four protocols in terms of average packet delivery ratio. The average packet delivery ratio is calculated as the percentage of packets received at the sinks, compared to the number of packets generated at the sensors. The figure reveals that the packet delivery ratio for SMAC is the minimum among the four protocols. Due to non-coordination among the scheduling and the forwarding mechanisms in SMAC, the interface buffer gets overflowed. Similar situation occurs for coverage aware sleep scheduling. However, in this case the buffer overflow is less because of the less number of rescheduling compared to the SMAC protocol. Though PEDAMACS provides a coordination mechanism between the scheduling and the packet forwarding, the decision time for scheduling is comparatively high, which makes the packets to be overflowed from the interface buffer. *ADCROSS* provides the maximum packet delivery ratio by maintaining a proper coordination among the scheduling and the data forwarding mechanisms.

Figure 6.13 compares the packet delivery delay for the four schemes. The delay for the SMAC protocol increases exponentially with an increase in the number of events/sec, because of the random sleep-wakeup scheduling. This randomness in the scheduling

TH-1251_11610108

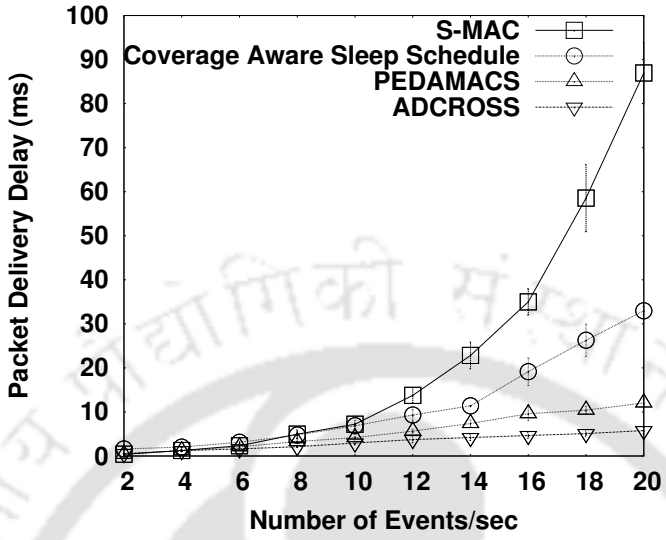

Figure 6.13: Average Packet Delivery Delay

increases the waiting time of the packets at the individual sensor's interface queue. The delay for the coverage aware sleep scheduling and the PEDAMACS is comparatively high from *ADCROSS*, because of the scheduling delay, that increases the average per-packet waiting time. Further, the forwarding paths of these three protocols, except *ADCROSS*, are determined based on a dynamic routing mechanism, that further imposes an extra overhead. The SYN interval for *ADCROSS* is a small fraction of the DATA interval. Moreover, the forwarding path in *ADCROSS* is predetermined during the scheduling information broadcast through the *Wakeup* message. Therefore, it results in a minimum packet delivery delay.

Analysis of the Scheduling Overhead:

For these sets of experiments, the number of events triggered at every sensor is considered to be 10 events/sec. Figure 6.14 compares the average scheduling delay among the four schemes. PEDAMACS requires the maximum scheduling delay, as every sensor needs to forward the topology information to the sink, and the sink determines the schedule, followed by a schedule broadcast to the individual sensor. Though the schedule update does not require much computational overhead at a sink, still the sensors need to forward the topology information. This incurs an extra overhead if the number of sensors in the network is large enough. The scheduling delay for SMAC is the minimum among the three schemes, except *ADCROSS*, because every sensor requires only 1-hop broadcast to

TH-1251_11610108

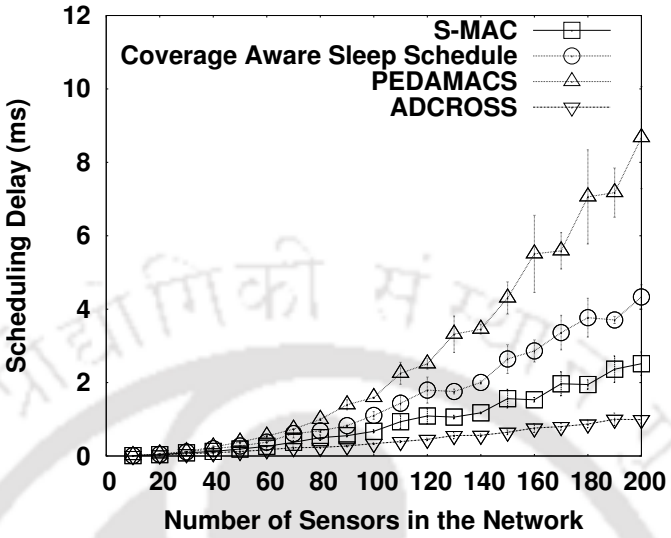

Figure 6.14: Average Scheduling Delay

inform the scheduling information to the neighbors. The figure shows that the scheduling delay for *ADCROSS* is the minimum among all the schemes. It can be noted that the scheduling delay for *ADCROSS* is calculated as the time duration from the start of the SYN interval, to the last *Wakeup* message broadcast. This shows the actual time in the SYN duration that is utilized for scheduling message broadcast. However, the sensors, that decide to be in the sleep state, generate larger back-off value compared to the sensors that decide to be in the active state. Because of this synchronization requirement among the sensors, the complete SYN duration is utilized to ensure that every sensor decides its state correctly.

Figure 6.15 compares the scheduling overhead among the four schemes in terms of the control packets broadcast. Though SMAC requires less duration for the scheduling generation, the number of control packets broadcast is high, because the random scheduling generated at the individual sensor needs to be synchronized throughout the network. This incurs extra control packets broadcast. The scheduling delay for PEDAMACS is high, but the number of control packets is less, as every sensor only needs to forward the topology information to the sink. The number of control packets broadcast is considerably less for the *ADCROSS* protocol, because only the sensors that decide to be in active state, broadcast a *Wakeup* message. Both these graphs reveal that the control overhead is significantly low for the *ADCROSS* protocol, compared to the other schemes.

TH-1251_11610108

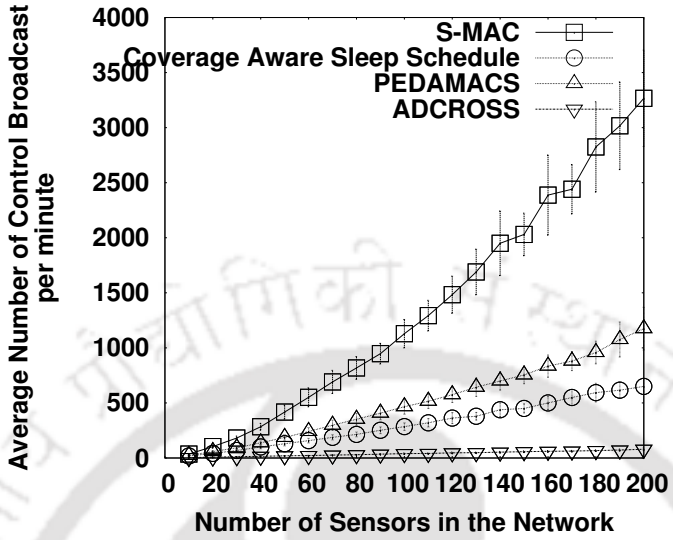

Figure 6.15: Average Scheduling Overhead

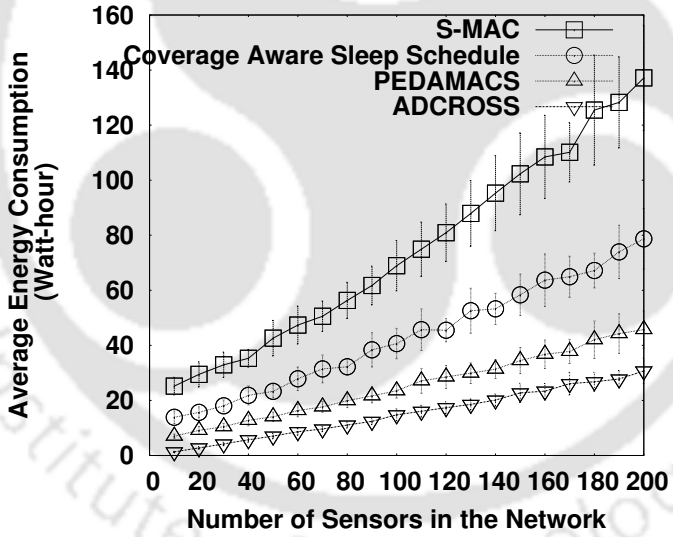

Figure 6.16: Average Energy Consumption

Analysis of Energy Consumption:

Considering the average number of events triggered per second to be 10, Figure 6.16 compares the average energy consumption of the *ADCROSS* protocol with the other three schemes. SMAC consumes the maximum energy because of the scheduling and synchronization message overhead. The figure shows that the average energy consumption for *ADCROSS* is the minimum among the four schemes. *ADCROSS* saves energy by

TH-1251_11610108

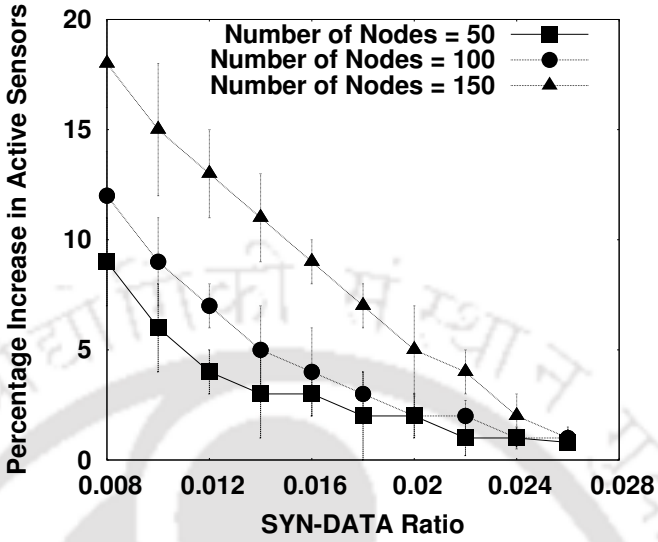

Figure 6.17: SYN-DATA Ratio

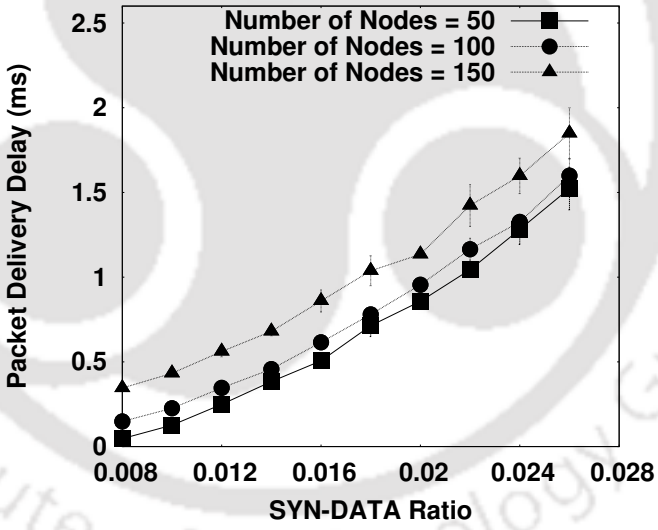

Figure 6.18: SYN-DATA Ratio: End-to-end Delay

considering *k-strip length coverage* that requires less number of sensors than that in full area coverage, and by reducing the control message broadcast.

6.6.3 Effect of the SYN-DATA Ratio

The *'SYN-DATA Ratio'* is defined as the ratio of the SYN size to the $SYN + DATA$ size. For these set of experiments, the $SYN + DATA$ interval is considered to be 100 ms, and

TH-1251_11610108

the SYN size is varied to check the effect on the network parameters. The value of k is takes as $\lceil \mathcal{L}/R_c \rceil$. Two parameters are considered for evaluation, the effect of SYN size over the localized distributed computation of the active nodes, and the effect over the average packet delivery delay. Figure 6.17 shows the effect of the SYN size over the localized distributed computation of the active nodes in the *ADCROSS* protocol. A 0 value of the *percentage increase* denotes the similar return of active nodes by both the centralized heuristic and the distributed approach. As the *SYN-DATA* ratio is increased (implies an increase in the SYN size), the *percentage increase* in the number of active sensors returned by the localized distributed computation is decreased. As the *percentage increase* tends to zero with an increase in the *SYN-DATA* ratio, the distributed computation gives similar outcome with that of the centralized heuristic. This indicates that a large SYN size gives sufficient time to the sensors to decide their states optimally close to the centralized heuristic. However, Figure 6.18 reveals that the large SYN size also increases the average packet delivery delay by increasing the control overhead time. Therefore, the *SYN-DATA* ratio can be selected based on the application specific delay sensitivity with minimum number of sensors involved in data gathering using the distributed scheduling scheme.

6.7 Summary

An adaptive sleep-wakeup scheduling of sensors has been proposed in this chapter for data collection in road sensor network, ensuring both the sensing coverage and the connectivity, along with an improved network lifetime. The concept of *k-strip length coverage* is introduced that the proposed *ADCROSS* protocol aims to support. A centralized heuristic is designed to determine the optimal set of active sensors for any time-slot, such that any passing vehicle along the road is detected by at least k sensors. A localized distributed scheduling scheme is designed by exploring the properties of the centralized heuristic, where every sensor node decides its state after waiting for a back-off period. The effectiveness of the proposed scheme is analyzed through the simulation results, and compared with other state-of-the-art works. Many other applications for critical infrastructure monitoring requires to satisfy the k-barrier coverage for detecting any potential intrusion across the boundary of a target territory. The design objectives for those cases are different from the one discussed in this chapter for the road sensor network. Therefore, to meet the special requirements of connectivity, sensing coverage, reliability and delay altogether, a multiple sink-based data gathering scheme, for the protection of critical infrastructure, has been introduced in the following chapter.

TH-1251_11610108

Chapter 7

Sensory Information Collection for Critical Infrastructure Monitoring

Critical Infrastructure (CI) like power generation and distribution centers, water supply plants, hospital, bank, bridge, international border area and transportation system, etc. are the indispensable assets for the socio-economical foundation of any country. The increasing threats of catastrophe or invasion on the security of the CI demand a robust and stable monitoring system to be developed. As already discussed in Chapter 1, wireless sensor network has emerged to contribute to the imperative protection of the large-scale CI by providing flexible, cost-effective and automated solutions to achieve smooth services [150]. A large number of low-cost micro-electronic sensors are deployed along the boundary of any target CI to sense different physical phenomena like temperature, pressure, speed and light, etc. according to various application requirements. The deployed sensors assure the k-barrier coverage such that the movement of an object across the CI boundary is detected by at least k sensors, and the corresponding alert can be raised for taking necessary actions. The type of sensor depends on the application requirement that defines the nature of the object to be detected. For an example, in international border area any intrusion in terms of person, vehicle or robots can be detected by sensing pressure, speed and direction of the object, etc. The sensors form an interconnected information system such that the sensory data is forwarded to the base station or gateway through a set of intermediate sinks for statistical analysis.

From Chapter 2, it is clear that tree based data gathering [77, 204] is widely used in sensor network because of its efficiency in the data collection with minimum contention and forwarding delay. Sensor nodes form a tree rooted at the sink, where the data is

TH-1251_11610108

aggregated at the intermediate nodes based on the application requirement, before being forwarded to the parent. The conventional store and forward policy for ensuring the reliability [10,128,134] is inefficient for applications that cannot tolerate the repairing and retransmission delay. Data forwarding through multiple node-disjoint paths [62,147,188] is an alternative option for the delay-bounded applications to assure the reliability in spite of node or link failure. Finding multiple node-disjoint paths towards a single sink is difficult due to the inherent randomness in sensor deployment. However, multiple node-disjoint paths towards different sinks are easier to compute, because the paths are sparsely located in different directions. Further, multiple paths towards different sinks provide the support for reliability during potential sink failure. With this approach, reliable data delivery is guaranteed in spite of a node or sink failure, without any repairing or retransmission delay (which is much higher than the forwarding time).

Considering potential vulnerability towards the protection of the CI and the failure prone nature of sensor nodes, the inherent design challenge is to ensure the reliable delivery of the sensory data to the base station even in presence of arbitrary node failure. Sensor nodes may fail due to power outage, technical fault or tampering. Although sinks are considered to be more resource advantageous compared to sensors, they may also fail as a result of tampering. Therefore, the overall network lifetime, in terms of the connectivity and the sensing coverage, is reduced in absence of the data gathering tree maintenance from a series of node failures. On failure of a set of nodes, reconstruction of data gathering trees starting from the scratch is not a good option, as it incurs large overhead. Also, the failure of one or multiple sensors or sinks may create sensing and/or communication holes into the network, introducing threats for potential attacks and disrupting application services. Even if the data is forwarded through multiple paths towards multiple sinks, the re-establishment of the affected path due to a node failure is necessary. Sensor deployment with high redundancy can extend the network lifetime by supporting node failures, as the failed node is replaced by a suitable redundant node, such that the required network connectivity and the sensing coverage are maintained throughout. The redundancy based tree maintenance scheme is efficient, as it offers an improved network lifetime with uninterrupted application services at a nominal cost of local path repairing.

This chapter proposes a framework and the working principles of the reliable data gathering protocol, *RelBAS* (Reliable Data Gathering from Border Area Sensors) for critical infrastructure monitoring. The primary design objective of *RelBAS* demands the reliable delivery of sensory data to the base station in spite of node or sink failures, such that every potential intrusion can be detected without any delay in reporting. The major

TH-1251_11610108

contributions are summarized as follows.

- *RelBAS* uses a multi-sink data gathering approach that is different from the conventional multicast. Multiple data gathering trees rooted at each of the multiple sinks are constructed, such that from every sensor node, there exist multiple node-disjoint paths to multiple sinks.

- The active set of sensors that participate in data sensing and forwarding activities, are selected such that the $k-$barrier coverage is maintained along the boundary of the target CI.

- During the tree construction at every sensor node, for a particular sink, the best forwarding path is selected among all possible options based on the hop-count distance to the sink and the node's residual energy. This offers minimum delay in data forwarding while ensuring the energy efficiency.

- *RelBAS* deals with single node failure by designing a local tree-repairing scheme through a suitable redundant node selection for replacing the failed node, while maintaining the connectivity and the sensing coverage. It also supports the identification of an affected zone due to multiple node failures.

The rest of the chapter is organized as follows:

The system model and network architecture along with an overview of the proposed *RelBAS* protocol are provided in Section 7.1. The description of the *RelBAS* protocol has been presented in three phases. The initialization phase includes the scheme of active node selection maintaining the k-barrier coverage, as provided in Section 7.2. Section 7.3 describes the second phase of *RelBAS*, that focuses on the data gathering tree construction rooted at multiple sinks. The correctness of the proposed scheme has also been established through the proofs of theorems and lemmas. Next section discusses the last phase of *RelBAS* for handling a single node failure and identifying an affected zone due to the failure of multiple nodes in close vicinity. A theoretical analysis to show the effect of *sink-connectivity* over the reliability has been provided in Section 7.5. The performance of *RelBAS* has been compared with existing schemes in the literature, those support reliability in data forwarding, by analysis of the simulation results, as presented in Section 7.6. Finally, Section 7.7 concludes the chapter.

TH-1251_11610108

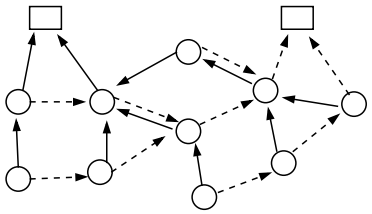

Figure 7.1: *RelBAS* protocol with *sink-connectivity* = 2

7.1 System Model and Protocol Overview

This section presents system model assumptions and network architecture details along with a brief overview of the proposed *RelBAS* protocol.

7.1.1 System Model and Network Architecture

Let sufficiently large number of sensors, say $\mathcal{N}$, be placed uniformly in the target area. Considering the risk factor associated with the smooth functioning of CI based applications, it is required to assure k-barrier coverage for any intrusion detection. With the k-barrier coverage, any movement across the *belt region*[1] [101] will be detected by at least k number of active sensors, for a desired value of k ($k > 2$). Each sensor node is assumed to be identified by a unique location based identifier, and is homogeneous in terms of the sensing, processing and the transmitting capacity. Every sensor is assumed to be able to calculate its relative position with respect to the sink using a pivot based location computation scheme as proposed in [104], where the sinks act as pivot points. To serve the purpose, it is sufficient to compute the relative positions of the sensors based on the neighbor information collected through beaconing. In any wireless system, periodic beaconing is necessary for synchronization among the nodes, which can be utilized for the localization purpose. Let $\langle \mathcal{C}_X(u), \mathcal{C}_Y(u) \rangle$ denote the position of the sensor node u in the Cartesian coordinate system. Let R_c and R_s be the radius of the communication disc and the sensing disc in the two dimensional plane, respectively, such that $R_c \geq 2R_s$ to assure both the connectivity and the sensing coverage [183]. Sensory data from the set of sensors is forwarded to the sink for further processing. For efficient forwarding, sensor nodes form a data gathering tree rooted at the sink node, such that each intermediate node forwards the self-generated data as well as the data forwarded from its children to the parent node in the tree. Multiple interconnected sinks are placed uniformly in the

[1] A *belt region* is bounded by two parallel curves uniformly separated by a distance $\mathcal{W}$, called the belt width.

TH-1251_11610108

target area to collect all the sensory data from the network, and then, forward them to a gateway or base station through a dedicated link for the user access. The base station is responsible for taking all administrative decisions based on the analysis of the collected sensory data. The sinks act as intermediate data collectors, and perform all maintenance related activities like the data gathering tree construction and the management. To assure the reliable delivery of sensory data to the sinks, there exist multiple data gathering trees with each one rooted at every sink node. Thus, data from every node is delivered to multiple sinks through multiple node-disjoint paths. The *sink-connectivity* can be defined as the number of sinks to which every sensor node maintains a path. The *sink-connectivity* is decided for any critical infrastructure and sensitive site, depending on the severity of potential threats. Figure 7.1 shows the network architecture for the proposed *RelBAS* protocol considering dual *sink-connectivity*. A data gathering tree rooted at the sink s_i is denoted by $\mathbb{T}_{s_i}$, whereas the set of nodes and tree edges in $\mathbb{T}_{s_i}$ are represented by $V_{\mathbb{T}_{s_i}}$ and $E_{\mathbb{T}_{s_i}}$, respectively. Let the Euclidean distance between two nodes u and v, projected along the X-axis and the Y-axis, be $d^X_{u,v}$ and $d^Y_{u,v}$, respectively (as a generic notation $d_{u,v}$ is used wherever required). The communication neighbors, denoted by $cNeighbor(u)$, can be defined as the set of all nodes such that $\forall v \in cNeighbor(u), d_{u,v} \leq R_c$. Let Λ_u denote the area that is sensing covered by a node u such that $\Lambda_u = \pi R_s^2$. Therefore, the sensing neighbors, denoted by $sNeighbor(u)$, can be defined as the set of all nodes such that $\forall v \in sNeighbor(u), \Lambda_u \cap \Lambda_v \neq \phi$. Sensor nodes are assumed to be able to communicate in a secure channel reliably, and thus, communication vulnerability issues have not been addressed here.

7.1.2 Protocol Overview

The *RelBAS* protocol has been designed to ensure the reliable delivery of the sensory data, from the sensors deployed in the target area to the base station, in spite of arbitrary sensor or sink failure. Considering the failure prone nature of sensors as well as the potential threats in critical infrastructure sites, sensors are deployed with high redundancy. The proposed *RelBAS* protocol works in three phases. First, a set of active sensors is selected for sensing and forwarding the data while satisfying the required coverage criteria as discussed in Section 7.2. Once the set of sensors is set to be active, they remain in that state until they die out of energy or crash due to tampering. Remaining nodes follow a sleep based schedule to wake up and check periodically for the possible failure of any node in its neighborhood. If a sensor node, say x, fails, one redundant node w becomes active to act as a replacement for the node x. Second, a set of data gathering trees is constructed

TH-1251_11610108

with the active set of nodes, such that every tree is rooted at different sinks, as described in Section 7.3. After the deployment is completed and the active set of sensor nodes are selected, the tree construction is initiated by every sink node through a hop-by-hop *Token* message broadcast. As the *Token* wave propagates outwards from the sink, every sensor node decides a suitable parent for the corresponding tree. The parent selection, ensuring the delivery of the data in the best path to a particular sink, is based on the calculation of a selection metric, which is a function of the hop-count distance to the sink and the node's residual energy. For m *sink-connectivity*, there exists m node-disjoint paths to m different sinks from every active node to ensure the reliable delivery of the sensory data even in the presence of arbitrary node failure. Third, on arbitrary failure of a node, another node in its neighborhood from the set of redundant nodes is set to active to repair the affected path in the tree. The proposed tree maintenance activities are performed locally such that the required sensing coverage and the connectivity can be ensured with minimum cost, after the tree repairing from node failure is finished successfully. The detection of a zone due to the failure of multiple nodes within a close proximity can also be reported to the base station according to the proposed protocol, as shown in Section 7.4.

7.2 RelBAS : Initialization

During the initialization phase, the active nodes are selected that participate in the tree based data forwarding. For movement detection in the boundary region, it is required to maintain the k-barrier coverage along the belt region. Further, for the reliable delivery of sensory data at the base station, it is required to maintain the connectivity among the active sensors. However, it is not possible to design a localized scheduling scheme deterministically for saving sensor energy as well as maintaining the k-barrier coverage for an arbitrary deployment [101]. For critical infrastructure based delay sensitive applications, using sleep-wakeup schedule is not efficient as it incurs extra delay overhead. Also, in distributed sensor network environment, the centralized control is not feasible in the application point of view. In this chapter, a simple distributed and completely localized heuristic has been proposed to select the initial set of active nodes satisfying both the k-barrier coverage and the network connectivity. Once active, the nodes remain in the active state until they crash, or die out of energy. The rest of the nodes in the network follow a periodic sleep-wakeup based schedule, and act as a replacement in case of any active node failure. The active node selection technique has been described in this section, whereas the activities of the redundant nodes are discussed in Section 7.4.

TH-1251_11610108

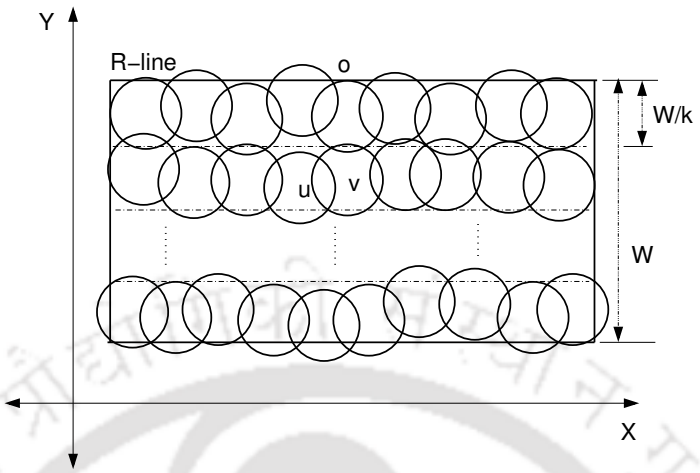

Figure 7.2: Sensing overage and network connectivity onto the belt region

7.2.1 Active Node Selection

Let the active nodes form k number of barrier chains, with each barrier chain supporting 1-barrier coverage for the given belt region of width $\mathcal{W}$. To support the k-barrier coverage, assume the belt region is virtually divided into k number of thin belts (called as *t-belt*), each of width $(\mathcal{W}/k)$, as shown in Figure 7.2. The length and the width of the belt are measured along the X-axis and the Y-axis, respectively. Let 'R-line' denote the upper curve of the belt region on which the sinks are mounted, and any point $o \in$ R-line can be considered as the pivot for computing the relative position of each sensor. Two nodes u and v belong to the same *t-belt* if Axiom 7.1 is true.

Axiom 7.1. $\left\lfloor \frac{d^Y_{o,u}}{(\mathcal{W}/k)} \right\rfloor = \left\lfloor \frac{d^Y_{o,v}}{(\mathcal{W}/k)} \right\rfloor$

Proof. As the width of each *t-belt* is $\mathcal{W}/k$, the axiom directly follows. □

Every sensor independently computes a back-off function, denoted by $\Xi(F_{tb}, F_{cc})$, that depends on two factors, the t-belt maintenance factor F_{tb} and the connectivity-coverage maintenance factor F_{cc}. Initially, all nodes are in the active state (status 'ON'). The back-off function determines the waiting time of any node for deciding whether to remain 'ON', or act as a redundant (status 'OFF'). Initially, the back-off timer $\tau_\Xi = f(\Xi)$ is set to a random amount of time for each sensor. Therefore, on timeout of τ_Ξ, few nodes randomly decide to become 'ON', and broadcast an *Active* message in 1-hop neighborhood. Node u, that receives an *Active* message from another node $v \in cNeighbor(u)$, computes its $\Xi(F_{tb}, F_{cc})$ as stated in Algorithm 7.1. If the competence of being 'ON' for the node u is

TH-1251_11610108

greater than all other nodes $w \in sNeighbor(u)$, then the computed τ_Ξ value for the node u is less than that of all other nodes $w \in sNeighbor(u)$. Thus, node u decides to be 'ON' before all other nodes in $sNeighbor(u)$ could decide the same, through an *Active* message broadcast in neighborhood. On receiving an *Active* message from node v, if $d_{u,v} < R_s$, node u can safely decide to be 'OFF', and acts as a redundant.

Algorithm 7.1 On receiving *Active* from node v by node u

1: **if** $d_{u,v} < R_s$ **then**
2: $status \leftarrow OFF$
3: **else**
4: **if** {Axiom 7.1} $\wedge$ $\{|\mathcal{C}_Y(u) - \mathcal{C}_Y(v)| > (\mathcal{W}/2k)\}$ **then**
5: Recompute τ_Ξ /*defers the decision time*/
6: **else**
7: **if** $|\mathcal{C}_X(u) - \mathcal{C}_X(v)| \geq 2R_s$ **then**
8: Recompute τ_Ξ /*defers the decision time*/
9: **else**
10: $status \leftarrow ON$
11: **end if**
12: **end if**
13: **end if**

Computation of the Back-off Function:

As discussed earlier, the back-off timer $\tau_\Xi = f(\Xi)$ prioritizes a node that is more competent in its neighborhood for being 'ON', by producing a lower back-off value. Ξ can be computed as shown in equation (7.1). The competence of a node for being 'ON' depends on two factors, F_{tb} and F_{cc}, to satisfy the desired k-barrier coverage along the belt as well as the network connectivity. Let the node v is already 'ON' in a *t-belt*. Also, let a node $u \in cNeighbor(v)$ be the most competent in $sNeighbor(u)$ for being 'ON'.

$$\Xi(F_{tb}, F_{cc}) = 1 - \{F_{tb} + F_{cc}\} \tag{7.1}$$

1. *t-belt maintenance factor :* To maintain the k-barrier coverage into the belt region, it is required to assure 1-barrier coverage into each of the k number of *t-belts*. The width of each *t-belt* is $(\mathcal{W}/k)$. Therefore, to maintain the corresponding *t-belt*, the objective is to minimize $d^Y_{u,v}$; and thus, to maximize $|(\mathcal{W}/2k) - d^Y_{u,v}|$ with respect to all potential competitors of the node v. Again, from equation (7.1), the node v will

TH-1251_11610108

be most competent in its neighborhood, if Ξ produces a lower value; and thus, F_{tb} produces a larger value. So, the *t-belt maintenance factor*, F_{tb} can be computed as given in equation (7.2).

$$F_{tb} = \frac{|(\mathcal{W}/2k) - d^Y_{u,v}|}{\sum\limits_{\forall w \in sNeighbor(u)} |(\mathcal{W}/2k) - d^Y_{u,w}|} \tag{7.2}$$

2. *connected-coverage maintenance factor :* To maintain the connectivity along the length of each *t-belt*, $d^X_{u,v} \leq R_c$ must be true. From the system model assumption, $R_c \geq 2R_s$. For any two active nodes u and v, $\Lambda_u \cap \Lambda_v \neq \phi$ must be true to assure 1-barrier coverage into each *t-belt*. However, to limit the number of active nodes, the objective is to satisfy $\Lambda_u \cap \Lambda_v \rightarrow \phi$. Hence, node u will be the most competent for being 'ON', maintaining 1-barrier coverage as well as the connectivity along the length of each *t-belt*, if $R_s < d^X_{u,v} < 2R_s$ and $|d^X_{u,v} - R_s| < |d^X_{w,v} - R_s|, \forall w \in sNeighbor(u)$. Again, from equation (7.1), node v will be the most competent in its neighborhood if Ξ produces a lower value; and thus, F_{cc} produces a larger value. So, the *connected-coverage maintenance factor*, F_{cc} can be computed as given in equation (7.3).

$$F_{cc} = \frac{1/\left(|d^X_{u,v} - R_s|\right)}{\sum\limits_{\forall w \in sNeighbor(u)} 1/\left(|d^X_{w,v} - R_s|\right)} \tag{7.3}$$

Once the active nodes are selected, the redundant nodes are also identified. The activities of the redundant nodes are described in Section 7.4 as a background of the second phase of the proposed *RelBAS* protocol. The selected active nodes participate in the data gathering tree construction for carrying out the sensory data collection and the forwarding activities. To construct multiple data gathering trees as discussed earlier, each one rooted at every sink, a distributed localized heuristic has been designed in the following section to describe the second phase of the proposed *RelBAS* protocol.

7.3 RelBAS : Data Gathering Tree Construction

For n number of sinks in the network and m *sink-connectivity* ($n > m$), n data gathering trees are constructed, such that each tree is rooted at each sink. The core of every tree is called the *skeleton* and denoted by $\mathfrak{S}$. All nodes within a particular skeleton are dedicated to forward the sensory data that are meant to be forwarded towards the corresponding sink. To ensure that there exist m node-disjoint paths to m different sinks from every

TH-1251_11610108

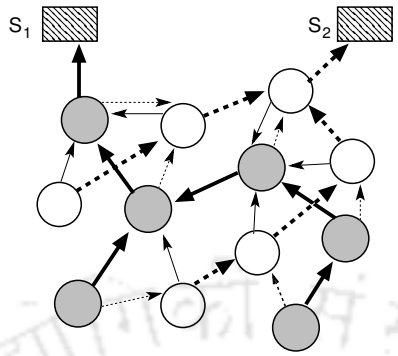

Figure 7.3: The topological architecture of multi-sink data gathering trees (2 *sink-connectivity*)

node, the construction and the distribution of the skeletons are based on certain criteria as discussed in Subsection 7.3.2. Node $u \in \mathfrak{S}_{s_i}$ is called a *relay* for the sink s_i as it forwards all the data received from its children to the sink s_i through its parent $v \in \mathfrak{S}_{s_i}$. All the nodes $w \in cNeighbor(u)$, such that $w \notin \mathfrak{S}_{s_i}$, are called the *terminal* nodes for $\mathfrak{S}_{s_i}$. Therefore, all terminal nodes for a particular skeleton are connected to a relay of that skeleton through a direct edge. Whether a node is a relay or a terminal with respect to a particular skeleton, is decided per node basis such that the global objective of m *sink-connectivity* is satisfied. In general, every active node maintains a set of parent variables, one for each of the m sinks, through which it forwards the self-generated sensory data. For any node $u \in \mathfrak{S}_{s_i}$, among these m nodes in the neighborhood, only one parent acts as the relay of $\mathfrak{S}_{s_i}$. For remaining $(m-1)$ nodes, u acts as a terminal node. Figure 7.3 presents the topological architecture of multi-sink data gathering trees with 2 *sink-connectivity*. Sinks are represented by rectangles, whereas sensor nodes are represented by circles. The relay nodes are darkly shaded for the skeleton corresponding to $\mathbb{T}_{s_1}$, and lightly shaded for the skeleton corresponding to $\mathbb{T}_{s_2}$. A thick arrow represents a tree edge included to a skeleton, whereas a thin arrow connects a terminal node to a relay node. The tree edges in $\mathbb{T}_{s_1}$ are marked with the continuous lines, whereas the tree edges in $\mathbb{T}_{s_2}$ are marked with the dotted lines.

On completion of deployment of sensors, data gathering tree construction is initiated by every sink node through a *Token* message broadcast in its neighborhood. Every active node that receives a *Token* corresponding to the sink s_i, participates in the tree construction by making certain decisions through local computation of neighborhood information. First, it selects m suitable nodes in its neighborhood that can act as parents to forward the sensory data towards m different sinks. Second, it determines the skeleton

TH-1251_11610108

Table 7.1: State variables at each node u

Variable Name	Type	Meaning
$sFlag$	Boolean	True if u has become a relay for any skeleton
sid	Node	Sink ID for which node u is relay
$parent_s(u)$	Node	The parent of the relay node in the corresponding skeleton
$pList$	Set	The set of (sink ID, parent ID) pair for m different trees
$sList$	Set	The set of unique (sink ID, $\mathcal{Z}$) pair for m different trees
$tList$	Set	The list of (node ID, sink ID, hop-count) tuples from which a *Token* is received
$tCounter$	Set	The list of (sink ID, count) where denotes the number of times a *Token* is received for the sink s_i
$eList$	Set	The list of (node ID, residual energy) pair for all nodes in $cNeighbor(u)$

corresponding to a particular sink, from m different sinks, for which it should act as a relay. The activities for the parent selection and the skeleton formation are executed in parallel at every node on receiving a *Token*. However, for the sake of clarity in understanding, they are discussed in the following two subsections. Every node maintains a set of state variables as shown in Table 7.1. Each *Token* message contains a set of parameters, listed as follows:

1. the sink node identifier, s_i,
2. the distance between the sender v and the sink s_i in hop-count, $\mathfrak{h}_v$, and
3. the residual energy of the sender v, $\mathfrak{e}_v$.

7.3.1 Parent Selection

On receiving a *Token* corresponding to the sink s_i from a node v, if $s_i \notin sList$, node u sets the node v as its parent for $\mathbb{T}_{s_i}$ by including v in $pList$. If u chooses to play the relay for the skeleton corresponding to s_i, it also sets the variable $parent_s(u)$ to v as stated in Algorithm 7.3 (skeleton construction method is described in detail in the following subsection). It also updates $sList$ and $tList$ accordingly, as stated in Algorithm 7.2. If $s_i \in sList$, node u checks its $pList$. If the node v offers a better path to the sink s_i than the existing parent $w \in \mathbb{T}_{s_i}$, node u updates its $pList$ (and $parent_s(u)$, if $u \in \mathfrak{S}_{s_i}$)

TH-1251_11610108

by replacing w with v. This parent update event is executed based on the computation of a selection metric called $\mathcal{Z}_{s_i}$, and can be defined as $\mathcal{Z}_{s_i} = f(\mathrm{H}, \mathrm{R})$. H and R are the normalized parameters, the hop-count and the energy, respectively.

Computation of the Selection Metric:

If there exist multiple paths from a node to a particular sink, the selection metric ensures that the selected parent offers the best path among them. $\mathcal{Z}_{s_i}$ at every node is initialized to zero, and can be calculated as given in equation (7.4).

$$\mathcal{Z}_{s_i} = \frac{\alpha}{\mathrm{H}} + (1-\alpha)\mathrm{R},\ 0 \leq \alpha \leq 1 \tag{7.4}$$

The higher value of $\mathcal{Z}_{s_i}$, computed for the node v, makes it more competent to be selected as a parent for $\mathbb{T}_{s_i}$ than all its potential competitors. Therefore, if the computed $\mathcal{Z}_{s_i}$ value is the highest for the node $v \in cNeighbor(u)$ among all nodes in $tList_{s_i}$, v offers the best path based on its residual energy and the hop-count distance to the sink s_i. The positive constant α can be chosen accordingly to meet the application requirements as discussed with the simulation results in Section 7.6. *sList* at every node is updated each time *pList* is updated. To select the best path to the sink s_i, following objectives have been considered in choosing the corresponding parent,

(a) the node that offers the minimum hop-count to the sink, H, and

(b) the node with the highest residual energy, R.

Parameter H

Let $\mathcal{H}_{u,s_i}$ denote the hop-count distance of a node u to the sink s_i. If node $v \in cNeighbor(u)$ for $\mathbb{T}_{s_i}$ is the parent of u, $\mathcal{H}_{v,s_i}$ is the minimum for all nodes $w \in \{cNeighbor(u) \cap V_{\mathbb{T}_{s_i}}\}$. Therefore, if there exist many paths from the node u to the sink s_i, the hop-count parameter ensures the selection of the shortest path to the sink. H can be estimated as shown in equation (7.5).

$$\mathrm{H} = \frac{\mathcal{H}_{v,s_i}}{\sum\limits_{\forall w \in \{cNeighbor(u) \cap V_{\mathbb{T}_{s_i}}\}} \mathcal{H}_{w,s_i}} \tag{7.5}$$

TH-1251_11610108

Parameter R

Let $\mathcal{E}_{Res}(u)$ be the residual energy of a node u at any instance of time. If node $v \in cNeighbor(u)$ for $\mathbb{T}_{s_i}$ is the parent of u, $\mathcal{E}_{Res}(v)$ is the maximum for all nodes $w \in \{cNeighbor(u) \cap V_{\mathbb{T}_{s_i}}\}$. Therefore, among multiple nodes offering the same hop-count to the sink, the selected parent provides the maximum node lifetime, as ensured by the energy parameter. R can be estimated as shown in equation (7.6).

$$\mathrm{R} = \frac{\mathcal{E}_{Res}(v)}{\sum\limits_{\forall w \in \{cNeighbor(u) \cap V_{\mathbb{T}_{s_i}}\}} \mathcal{E}_{Res}(w)} \tag{7.6}$$

Axiom 7.2. *Let there exist multiple paths $\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_k$ to a particular sink s_i via nodes $v_1, v_2, ..., v_z \in cNeighbor(u)$, respectively. If v_i is selected as the parent for the sink s_i, the corresponding path $\mathbb{P}_i$ is the best in $\{\mathbb{P}_j\}$; $\forall j = 1..z$.*

Proof. This directly follows from equation (7.4). □

Algorithm 7.2 On receiving *Token*$(s_i, \mathfrak{h}_v, \mathfrak{e}_v)$ from node v by node u

1: Calculate $\mathcal{Z}'_{s_i}$ using $\mathfrak{h}_v, \mathfrak{e}_v$
2: $tCounter_{s_i} \leftarrow tCounter_{s_i} + 1$
3: $tList \leftarrow tList \cup \{v, s_i, \mathfrak{h}_v\}$
4: **if** $\mathcal{Z}'_{s_i} > \mathcal{Z}_{s_i}$ **then**
5: $pList_u(s_i) \leftarrow v$
6: $sList_u(s_i) \leftarrow \mathcal{Z}'_{s_i}$
7: $\mathcal{H}_{u,s_i} \leftarrow \mathfrak{h}_v + 1$
8: **end if**
9: **if** $\{|sList_u| \leq m\} \wedge \{sFlag = False\}$ **then**
10: Call $waitTimer(f(\Omega))$ /*Ω is per hop link delay on average*/
11: **end if**

7.3.2 Skeleton Formation

To ensure the desired reliability in delivery of sensory data to the base station by maintaining m node-disjoint paths from every node to m different sinks, the formation of the skeletons must satisfy following two objectives.

1. Every node in the network must act as a relay for exactly one skeleton.
2. All skeletons must be node-disjoint.

TH-1251_11610108

Algorithm 7.3 On timeout of *waitTimer*$(f(\Omega))$

1: **if** $\{|sList_u| < m\} \wedge \{tCounter_{s_i} \leq \lfloor(\delta_u + 1)/m\rfloor\}$ **then**
2: $eFlag \leftarrow True$
3: $tmp \leftarrow s_i$
4: **end if**
5: **if** $|sFlag| = m$ **then**
6: $eFlag \leftarrow True$
7: $tmp \leftarrow s_j | s_j \in sList$ is uniformly chosen
8: **end if**
9: **if** $eFlag = True$ **then**
10: Broadcast *Token*$(tmp, \mathcal{H}_{u,tmp}, \mathcal{E}_{Res}(u))$
11: $sFlag \leftarrow True$
12: $sid \leftarrow tmp$
13: $parent_s(u) \leftarrow v$
14: $eFlag \leftarrow False$
15: **end if**

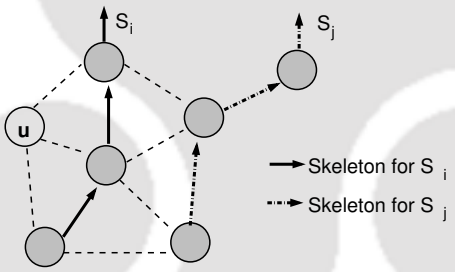

Figure 7.4: Unfair distribution of skeletons

The main responsibility of relay nodes is to expand the wave of *Token* message propagation into the whole network such that the skeletons are formed. On receiving a *Token* from a node $v \in \mathbb{T}_{s_i}$, a node decides whether to act as a relay for the corresponding skeleton or not, based on the computation of a localized heuristic. A node may get *Token* messages from many nodes corresponding to different skeletons, but it acts as a relay for only one skeleton. It can be noted that every node, even each of the boundary nodes, acts as a relay for some skeleton. This is because the boundary nodes forward the *Token* like all other intermediate nodes. However, there exist no node in $cNeighbor(u)$, where u is a boundary node that can further expand the corresponding skeleton. The proposed localized heuristic is developed to meet global objectives of fair skeleton distribution while selecting a particular skeleton for which the node can act as a relay.

TH-1251_11610108

Fairness in Skeleton Distribution

For the communication graph as shown in Figure 7.4, it can be observed that due to an uneven distribution of the skeletons, there does not exist any relay for the skeleton $\mathfrak{S}_{s_j}$ in $cNeighbor(u)$. Global fair distribution of skeletons assures that every node acts as a relay for one skeleton, and as a terminal for remaining $(m-1)$ skeletons within its 1-hop neighborhood. Following postulates are required to be true to meet global design objectives.

Postulate 1. *To satisfy the desired m sink-connectivity, there must exist at least one relay node from each of the m skeletons within* 1*-hop neighborhood of every node.*

Postulate 2. *To achieve global fair distribution of skeletons, m number of relays for the corresponding skeletons must be equally distributed in the neighborhood of every node.*

Let the degree of a node u in the network communication graph be denoted by $\delta_u = |cNeighbor(u)|$, whereas the maximum and the minimum degree of a node over the whole network be δ_{MAX} and δ_{min}, respectively. Hence, according to the Postulates 1 and 2, every skeleton can have a share of $(\delta_u + 1)/m$ number of relay nodes in $cNeighbor(u)$, including the contribution of the node u in one skeleton. To evenly distribute the neighbors among m skeletons in neighborhood, node u first reserves $\lfloor(\delta_u + 1)/m\rfloor$ number of neighbors for each of the m skeletons. On receiving a *Token*, node u selects the corresponding skeleton $\mathfrak{S}_{s_i}$ for which it acts as a relay, and forwards the *Token* in the 1-hop neighborhood, if following conditions are satisfied according to Algorithm 7.2 and Algorithm 7.3.

Condition 1. Node u is not a relay node yet $(sFlag = False)$

Condition 2. The number of relays for $\mathfrak{S}_{s_i}$ in $cNeighbor(u)$ (denoted by $tCounter_{s_i}$ in Algorithm 7.3) is less than $\lfloor(\delta_u + 1)/m\rfloor$, the share of $\mathfrak{S}_{s_i}$

Condition 3. Node u has not yet received *Token* messages from m different sinks $(|sList| < m)$

However, the decision making of being a relay for a particular sink at every node is deferred through a *waitTimer* call, the timeout duration of which is a function of Ω, the average per hop link delay. This restricts the unwanted *Token* forwarding from two neighbors for the same skeleton at the same time; and thus, one node's decision is reflected correctly to the other node's decision. While ensuring a fair distribution of the skeletons

TH-1251_11610108

over the whole network, the initial deployment of the sensors play an important role as suggested by Proposition 7.1 and Proposition 7.2.

Proposition 7.1. *To assure the m sink-connectivity throughout the network, $m \leq \delta_{min} \leq \delta_{MAX}$ must be true.*

Proposition 7.2. *To assure a fair distribution of the skeletons globally, the condition $\lfloor(\delta_{MAX}+1)/m\rfloor = \lfloor(\delta_{min}+1)/m\rfloor$ always holds true considering the uniform distribution of the deployed sensors.*

If the deployment density of the network satisfies the condition of global fairness as stated in Proposition 7.2, the local fairness in skeleton distribution at every node can be assured based on Postulate 1, as proved in Theorem 7.3. Further, some other important properties of the proposed *m sink-connectivity* based data gathering scheme are established with the help of Theorems 7.4-7.6.

Lemma 7.1. *Every node receives at least m Token messages from m different sinks, if Proposition 7.2 holds true.*

Proof. Let node u receive $(m-1)$ number of *Token*, for m *sink-connectivity*. From Proposition 7.1, $m \leq \delta_u$. According to Condition 2, there exists a reservation of $\lfloor(\delta_u+1)/m\rfloor$ nodes in $cNeighbor(u)$ for the relay corresponding to each of the m skeletons. Assume all the neighbors of the node u are of degree equal to δ_{MAX} and $\delta_u = \delta_{min}$ in the worst case. Let $q = (m-1)\lfloor(\delta_{MAX}+1)/m\rfloor$. So, u may receive q number of *Token* messages corresponding to $(m-1)$ different sinks in total, at the worst case. Assuming Proposition 7.2 is true,

$$\lfloor(\delta_{MAX}+1)/m\rfloor = \lfloor(\delta_{min}+1)/m\rfloor = C, \text{say} \tag{7.7}$$

Then, for any three positive integers a, b and m (the *sink-connectivity*), from equation (7.7), δ_{MAX} and δ_{min} can be expressed as,

$$\delta_{MAX} = mC + a \text{ and } \delta_{min} = mC + b \text{ such that } a > b \text{ and } a, b < m \tag{7.8}$$

To prove that every node receives m number of *Token* from m different sinks, it is sufficient to prove that $q - \delta_{min} < 0$.

$$\begin{aligned} q - \delta_{min} =& (m-1)\lfloor(\delta_{MAX}+1)/m\rfloor - \delta_{min} \\ &= (m-1)\lfloor(mC+a+1)/m\rfloor - (mC+b) \text{ , from (7.8)} \\ &= (m-1)C - mC + b \text{ as } a < m \\ &= -(C+b) < 0 \text{ as } C, b > 0 \end{aligned}$$

TH-1251_11610108

Hence, there will always be at least one node in $cNeighbor(u)$ through which it can receive a *Token* corresponding to a sink, other than the ones for which it has already received $(m-1)$ number of *Token*. Hence, proved. □

Lemma 7.2. *Every node $u \in V_{act}$ forwards a Token corresponding to any sink exactly once, where V_{act} denotes all active nodes in the network.*

Proof. Let node $u \in V_{act}$ receive a *Token* from node $v \in \mathfrak{S}_{s_i}$. So, one of the following cases may occur as described in Algorithm 7.2.
Case 1 : $sFlag = True$ at node u. This means u has already forwarded a *Token* according to line 11 of Algorithm 7.3.
Case 2 : $sFlag = False$ at node u. In this case, u has not yet forwarded any *Token*; and thus, following sub-cases are possible.

1. If $|sList| < m$, u waits for the $waitTimer$ timeout. On timeout, if the number of received *Token* for s_i, $tCounter_{s_i}$ is less than its assigned share, $\lfloor(\delta_u+1)/m\rfloor$, u forwards the *Token* according to Algorithm 7.3.
2. If $|sList| < m$ and $tCounter_{s_i} > \lfloor(\delta_u+1)/m\rfloor$, it waits for another *Token* corresponding to some s_j, such that the condition at line 1 of Algorithm 7.3 is satisfied. From Lemma 7.1, node u will receive all the *Token* from m different sinks. Once the condition is satisfied on receiving a *Token* for some s_j, node u forwards the *Token* according to Algorithm 7.3.
3. If $|sList| = m$ at node u, but $sFlag$ is still False, after waiting for a $waitTimner$ timeout duration, node u selects one s_i uniformly from all m sinks in $sList_u$. Then u forwards a *Token* corresponding to that selected skeleton in its neighborhood according to lines 5-15 of Algorithm 7.3.

Therefore, for all possible cases, node u forwards a *Token* corresponding to a sink. Once the *Token* is sent, node u changes the status of the Boolean variable $sFlag$ to True, for which it cannot forward another *Token* for the sink s_i. Hence, proved. □

The following corollary can be derived directly from Lemma 7.2.

Corollary 7.1. *If $u \in \mathfrak{S}_{s_i}$, then $u \notin \mathfrak{S}_{s_j}$ for any $j = 1..m$ and $i \neq j$.*

Theorem 7.3. *For every node u, there exists at least one relay node corresponding to each of the m skeletons in $cNeighbor(u)$.*

TH-1251_11610108

Proof. As every node forwards a *Token* exactly once, every node in the network acts as a relay for exactly one skeleton, from Lemma 7.2. So, for any node u, every node $v \in cNeighbor(u)$ is a relay for some skeleton. From Proposition 7.1, $\delta_u \geq m$. Therefore, u receives at least m number of *Token* corresponding to m different skeletons according to Lemma 7.1. Therefore, from Lemma 7.1 and Lemma 7.2, it follows that there exists at least one relay node corresponding to each of the m skeletons in $cNeighbor(u)$. □

Theorem 7.4. *All the skeletons in the network are node-disjoint.*

Proof. Two skeletons $\mathfrak{S}_i$ and $\mathfrak{S}_j$ are called node-disjoint if $\mathfrak{S}_i \cap \mathfrak{S}_j = \phi$. Let $\exists x \in \mathfrak{S}_i \cap \mathfrak{S}_j$, which means, both $x \in \mathfrak{S}_{s_i}$ and $x \in \mathfrak{S}_{s_j}$ are true. However, from Corollary 7.1, if $x \in \mathfrak{S}_{s_i}$, then $x \notin \mathfrak{S}_{s_j}$ such that $i \neq j$, and vice-versa, which contradicts the assumption. The above is true for any pair of skeletons. Hence, proved. □

The following corollary can be derived directly from Theorem 7.4.

Corollary 7.2. *All the skeletons in the network are edge-disjoint.*

Theorem 7.5. *All the paths from node u, denoted by $\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_m$ to the sinks $s_1, s_2, ..., s_m$ respectively, are node-disjoint.*

Proof. Two paths $\mathbb{P}_i$ and $\mathbb{P}_j$ are called node-disjoint if $\mathbb{P}_i \cap \mathbb{P}_j = \phi$. Let $pList_u(s_i)$ denote the parent of node u for $\mathbb{T}_{s_i}$. Let $\mathbb{P}_i$ and $\mathbb{P}_j$ be two paths from node u such that $\mathbb{P}_i = u, pList_u(s_i), ..., s_i$ and $\mathbb{P}_j = u, pList_u(s_j), ..., s_j$. Assume that $\exists w \in \mathbb{P}_i \cap \mathbb{P}_j$. Node u can be a relay of one and only one skeleton according to Lemma 7.2. From Theorem 7.4, three cases are possible as follows:
Case 1 ($u \in \mathfrak{S}_{s_i}$): $\mathbb{P}_i \subset \mathfrak{S}_{s_i}$. Therefore, from Theorem 7.3, $pList_u(s_j) \in \mathfrak{S}_{s_j}$, which follows $\mathbb{P}_j \subset \mathfrak{S}_{s_j}$. From Corollary 7.1, $\mathfrak{S}_{s_i} \cap \mathfrak{S}_{s_j} = \phi$, which contradicts the assumption.
Case 2 ($u \in \mathfrak{S}_{s_j}$): Similar to the previous case.
Case 3 ($u \in \mathfrak{S}_{s_k}$ where $k \neq i, j$): From Theorem 7.4, $\mathbb{P}_i \subset \mathfrak{S}_{s_i}$ and $\mathbb{P}_j \subset \mathfrak{S}_{s_j}$ for $s_i \neq s_j$, as both the nodes $pList_u(s_i)$ and $pList_u(s_j)$ are relay for $\mathfrak{S}_{s_i}$ and $\mathfrak{S}_{s_j}$, respectively.
Hence proved. □

Lemma 7.3. $\bigcup_{i=1}^{n} V_{\mathfrak{S}_{s_i}} = \bigcup_{i=1}^{n} V_{\mathbb{T}_{s_i}} = V_{act}$, *where $V_{\mathbb{T}_{s_i}}$ denotes the active nodes within $\mathbb{T}_{s_i}$*

Proof. This can be directly proved from Lemma 7.2 and Theorem 7.4. □

Lemma 7.4. *No cycle is created during the construction of any tree $\mathbb{T}_{s_i}$.*

TH-1251_11610108

Proof. From Theorem 7.3, if node $u \in \mathfrak{S}_i$ is a relay for the sink s_i, it acts as a terminal for all other $(m-1)$ sinks such that there exists one relay from each of the corresponding skeleton in its neighborhood. Therefore, all the terminal nodes for $\mathfrak{S}_{s_i}$ are connected to all the relay nodes of $\mathfrak{S}_{s_i}$ through a direct communication edge, and do not contribute to the construction of any cycle.

Now during the skeleton construction, a cycle is formed within the tree $\mathbb{T}_{s_i}$, rooted at the sink s_i, if there exists a path $\mathbb{P} \in \mathfrak{S}_{s_i}$ such that $\mathbb{P} = u_1, u_2, ..., u_p$ and $u_1 = parent_s(u_2), u_2 = parent_s(u_3), ..., u_p = parent_s(u_1)$. To select node u_p as its parent for the skeleton $\mathfrak{S}_{s_i}$, the node u_1 has to receive a *Token* from the node u_p. If $u_1 \in cNeighbor(u_p)$, it receives the *Token*. However, as u_p belongs to the descendant chain of the node u_1, $\mathcal{H}_{u_p,s_i} > \mathcal{H}_{parent_s(u_1),s_i}$. Therefore, u_1 does not update its parent to u_p ever, according to Algorithm 7.2. Hence, proved. □

Theorem 7.6. *Multiple trees, each one rooted at every sink, are constructed correctly.*

Proof. The proof is based on following inferences.

1. From Lemma 7.3, there exists no node in the network that is not a part of any tree.
2. From Theorem 7.4, all the skeletons are node-disjoint.
3. From Theorem 7.3 and Theorem 7.5, there exists m node-disjoint paths to m different sinks.
4. From Lemma 7.4, no cycle is created during the construction of any tree $\mathbb{T}_{s_i}$.

Hence, proved. □

Theorem 7.7. *The complexity of the tree construction scheme in terms of control message communication is $O(|V|)$, where V denotes the set of active nodes in the terrain.*

Proof. From Lemma 7.2, every node broadcasts a *Token* message exactly once. This directly follows that the message complexity of the tree construction scheme is $O(|V|)$, where V is the set of all active nodes in the terrain. □

If more than m number of *Token* messages are received at a node u such that $|sList| > m$, they are simply ignored. In most of the cases, the sink, from which the $(m+1)^{th}$ *Token* is received, does provide a better path than all other existing paths corresponding to m sinks. So, when m number of paths have already been established with m different sinks, ignoring the *Token* from a sink does not affect the performance of

TH-1251_11610108

the protocol in most of the cases. Therefore, every node maintains exactly m number of node-disjoint paths to m different sinks for m *sink-connectivity*. Let $\mathcal{M}_v$ be a sensory data packet for the running application, received at the node u from the node $v \in cNeighbor(u)$. $\forall v \in cNeighbor(u)$, $\mathcal{M}_v$ is sent to $parent_s(u)$. However, the self-generated sensory data at the node u, $\mathcal{M}_u$ is sent to all nodes $w \in cNeighbor(u)$ such that each w is a relay for one skeleton ($w \in pList_u$). It can be noted that $parent_s(u) \in pList_u$. Therefore, the self-generated sensory data at every node is delivered to m sinks through m different skeletons assuring the reliability. However, due to sudden failure of any node, any path in the data gathering tree may get affected. A local tree maintenance scheme has been proposed for this purpose through redundant node activities, as discussed in the following section.

7.4 RelBAS : Tree Repairing and Failure Management

In wireless communication environment, every node, whether an active or a redundant, broadcast a beacon frame periodically for synchronization and information collection in its 1-hop neighborhood. Therefore, the failure of a node x can be detected by all nodes in $cNeighbor(x)$ through usual beaconing[2]. As every node is connected to the sinks through multiple node-disjoint paths, sudden disruption of connectivity in one path due to a node failure does not affect the performance of the running application. Sensory data are delivered to other sinks through different paths reliably. However, the affected path must be re-established without any delay to support further node failures. Construction of a new tree starting from the scratch is not a good option as it incurs large overhead in terms of delay and communication complexity. A local tree repairing technique has been introduced in this section through the replacement of the failed node by a suitable redundant node, such that the affected path is re-established with minimum cost. The repairing delay has also no effect in the performance of the running application. The activities of the redundant nodes, as discussed in the following subsection, play a major role in performing the local tree maintenance at a nominal cost.

7.4.1 Activities of the Redundant Nodes

Once the redundant nodes (status 'OFF') are identified, they follow a sleep-wakeup based schedule to act as the replacement on an active node failure. The time domain is divided

[2]It can be noted that this 1-hop beaconing is mandatory in any wireless communication environment, and thus, does not incur any extra overhead

TH-1251_11610108

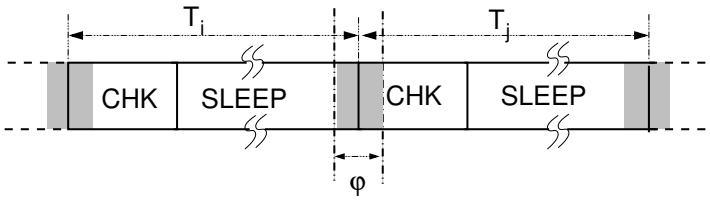

Figure 7.5: Periodic Renewal Interval with CHK and SLEEP duration

into periodic renewal slots, where each slot is further subdivided into 'CHK' and 'SLEEP' intervals as shown in Figure 7.5. Let the 'CHK' and the 'SLEEP' intervals for any time-slot be denoted by T_{CHK} and T_{SL}, respectively. During the 'CHK' interval of every time-slot, each node with status 'OFF' wakes up, and communicates with its communication neighbors through 1-hop beaconing for synchronization and neighborhood information update. In this way, all redundant nodes remain updated with the information of $sList$, $pList$ and $parent$ variable of an active node for which they serve. Let during some T_{CHK}, a node u, on detection of the failure of an active node $x \in sNeighbor(u)$, is selected as the replacement for the node x. Node u changes its status from 'OFF' to 'ON' to repair the sensing hole and the communication hole created due to failure of the node x, while the rest of the redundant nodes go to sleep for the T_{SL} duration. During the 'CHK' interval, if no node in neighborhood is found to be failed, the redundant nodes go to sleep again for the 'SLEEP' interval of the corresponding time-slot. While sleeping during the 'SLEEP' interval, the redundant nodes may lose the time synchronization. At the beginning of 'CHK' interval, as all redundant nodes wake up, they are then synchronized through usual 1-hop beaconing. From the specification of the clock accuracy of all available sensor motes, the maximum and the minimum clock drifts can be calculated a priori. Let ϑ be the drift factor, that is the maximum deviation of clock time per second among all sensors. Therefore, there must be a tolerance of duration φ at the end of each 'SLEEP' interval, such that ϑ is bounded above by φ. Thus, all sleeping nodes are available for synchronization at the beginning of 'CHK' interval of the next time-slot. Let $\mathfrak{T}$ denote the maximum delay in data delivery that is tolerable to the running application. So, to handle a node failure through the replacement by a redundant node, without affecting the performance of the application, the duration of the 'SLEEP' interval can be estimated as shown in equation (7.10).

$$\varphi \geq T_{SL} \times \vartheta \tag{7.9}$$

TH-1251_11610108

$$T_{SL} = min\left(\frac{\varphi}{\vartheta}, \mathfrak{T}\right); \qquad \text{from (7.9)} \tag{7.10}$$

7.4.2 Handling Single Node Failure

Let $x \in \mathfrak{S}_{s_i}$ fail. The selection of a node u as the replacement of the node x would be correct if following two objectives are fulfilled.

1. The sensing hole created in the corresponding *t-belt* must be sensing covered by the node u.
2. Node u must re-establish the affected path that included the node x in all the involved trees.

All nodes in $sNeighbor(x)$, that detect the failure of the node x at the beginning of the 'CHK' interval of any time-slot, are redundant nodes from Subsection 7.2.1. So, all nodes in $sNeighbor(u)\backslash\{u\}$, denoted by $\wp_x$, are the potential competitors of the node u for being selected as the replacement for the node x. Now, to meet objective 1, it is required that $d_{u,x} = \min\{d_{w,x},\ \forall w \in \wp_x\}$. On detection of failure of the node x, every node in $sNeighbor(x)$ computes Ξ_R, the back-off function for the redundant node selection. Ξ_R, shown in equation (7.11) can be designed similar to the one given in equation (7.1) for active node selection. F^R_{tb} and F^R_{cc} are the corresponding variation of the *t-belt maintenance factor* (F_{tb} in equation (7.2)), and the *connectivity-coverage factor* (F_{cc} in equation (7.3)).

$$\Xi_R(F^R_{tb}, F^R_{cc}) = 1 - (F^R_{tb} + F^R_{cc}) \tag{7.11}$$

$$F^R_{tb} = \frac{|d^Y_{u,x} - (\mathcal{W}/2k)|}{\sum\limits_{\forall w \in \wp_x} |d^Y_{w,x} - (\mathcal{W}/2k)|} \tag{7.12}$$

$$F^R_{cc} = \frac{\left(|d^X_{u,x} - R_s|\right)}{\sum\limits_{\forall w \in \wp_x} \left(|d^X_{w,x} - R_s|\right)} \tag{7.13}$$

Node u will be selected as the most suitable redundant node that can replace the failed node x satisfying the required coverage criteria if the computed Ξ_R for the node u is less than that of all other nodes in $\wp_x$. On detection of failure of the node x, all redundant nodes $w \in sNeighbor(x)$ wait for a $\tau(\Xi_R)$ amount of time before an *Active* message broadcast in $cNeighbor(w)$. As $u \in sNeighbor(x)$ produces the minimum value of Ξ_R, node u broadcast an *Active* message, on timeout of $\tau(\Xi_R)$, announcing its 'ON' status before any other node can do the same. On receiving the *Active* message from the node u, all nodes in $\wp_x$ go to sleep.

TH-1251_11610108

Now, from Section 7.2, $R_s < d^X_{u,v} < 2R_s$ for any two active node u and v within a $t - belt$; and also, according to the node selection metric, as expressed in equation (7.1), $d^X_{u,v} \to R_s$. Therefore, the selection of a single redundant node u on failure of the node x, such that $d^X_{u,x} \to 0$ according to the equation (7.11), is sufficient to maintain the required sensing coverage in most of the times. The set of nodes $\mathfrak{V}_x = \{v \in cNeighbor(x)\}$ is called the victims, such that for each $v \in \mathfrak{V}_x$, $x \in pList_v$. Let $\Gamma(u)$ denote the communication disk of the node u such that $\Gamma(u) = \pi R_c^2$. For the failed node x, and its corresponding replacement node u, if $d_{u,x} \to 0$, $\Gamma(x) \setminus \Gamma(u) \to Null$. So, $\forall w \in \mathfrak{V}_x$, $w \in \Gamma(u)$ will be true in most of the cases if $d_{u,x} \to 0$. Therefore, to meet objective 2, following two conditions have to be satisfied.

(C1) For all nodes $w \in \mathfrak{V}_x$, each entry in $pList_w$ corresponding to x must be updated with the replacement, node $u \in cNeighbor(w)$

(C2) The replacement of the node x (which is assumed to be node u) must be connected to all nodes $z \in pList_x$,

All nodes $w \in \mathfrak{V}_x$ update their corresponding state variables $sList$, $pList$ and $parent_s(w)$ by replacing x with u on receiving the *Active* message from the node u, which satisfies condition (C1). Node u can also set its $sList$, $pList$ and $parent_s(w)$, based on the updated information collected from all nodes in $cNeighbor(u)$ during the 1-hop beaconing, which satisfies condition (C2). Thus, the selected redundant node that acts as a replacement for the failed node x, repairs the tree locally, while ensuring the desired sensing coverage and the connectivity in the network. The cost of this repairing is only the cost of *Active* message broadcast in 1-hop neighborhood. Let Ω denote the per hop link delay on average. The repairing delay, denoted by Δ, to handle any single node failure according to the proposed *RelBAS* protocol is upper bounded by a small value, as shown in equation (7.14).

$$\Delta \leq T_{SL} + \varphi + \Omega \tag{7.14}$$

7.4.3 Handling Multiple Node Failures

The proposed *RelBAS* protocol can handle any single node failure by local tree repairing where the faulty node is replaced by a designated redundant node. The repairing is possible until there exist enough number of redundant nodes within Λ_x of the failed node x to act as the replacement. The failure of multiple nodes that are sparsely located in the network can be handled by the proposed *RelBAS* protocol, where each of the failure cases is treated separately as a single node failure case, and handled accordingly. However, the

TH-1251_11610108

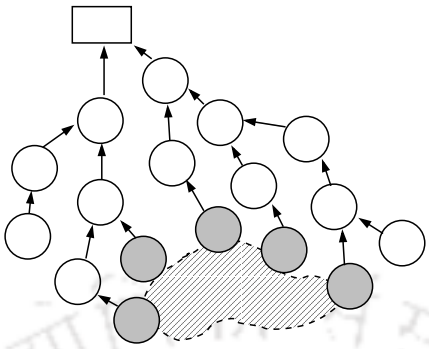

Figure 7.6: Affected zone due to invasion

situation is more critical if multiple nodes fail at the same time; and are in close proximity, such that both the sensing coverage and the connectivity of a zone gets affected. Multiple node failures in the same zone indicates the occurrence of a potential attack with high probability. Hence, this event needs to be reported to the base station separately. The challenge lies in detecting the affected zone; and then, reporting to the base station without any delay for manual intervention.

If a zone gets affected due to the invasion, then the active nodes as well as the redundant nodes in that zone get destroyed. Let the active nodes located near the periphery of the affected zone (darkly shaded nodes shown in Figure 7.6) are called the *guard* nodes. In case of a zone failure, these guard nodes perform following actions to detect the event as well as to report to the sinks. Let the affected zone be denoted by $\mathfrak{A}$.

- Every guard node g can detect the failure of its child, say c, individually through a beacon as discussed in the previous subsection.
- A guard node g waits for receiving an *Active* message, broadcast from a redundant node w that replaces the node $c \in cNeighbor(g)$, on detection of the failure of c. If all redundant nodes that could wakeup to serve the node c are located within $\mathfrak{A}$, g does not receive any *Active* message. On timeout of the $reportTimer$, the case is reported to the sinks as a potential threat of a zone failure by forwarding a *Report* message with the node ID to all of its parents.

In this way, every guard node detects the non-recoverable failure of the nodes in its neighborhood, and individually reports to the sinks via respective parents. On receiving multiple such *Report* messages forwarded from the sinks, the base station can identify the location of the affected zone exploiting the information from the source ID, received with

TH-1251_11610108

each *Report* message. Once the affected zone is detected, the necessary actions can be performed to reform the network manually.

7.5 Theoretical Analysis : The Effect of Sink-connectivity over the Reliability

Let m be the *sink-connectivity*. Let ϱ_i, $i = \{1, 2, ..., m\}$ denote the probability that the data is successfully delivered through the path $\mathbb{P}_i$. The proposed *RelBAS* protocol aims at finding the node-disjoint path routing to multiple sinks. As the paths towards different sinks are node-disjoint, all the probabilities ϱ_i for $i = \{1, 2, ..., m\}$ can be considered independent and identically distributed. Let b_i be an indicator, such that $b_i = 1$, if data is delivered successfully through the path $\mathbb{P}_i$, or $b_i = 0$ otherwise. Let $B_m : \{0, 1\}^m \rightarrow \mathbb{N}$ is the random variable that denote the number of successful data delivery. Then $B_m = \sum_{\forall i} b_i$. The expected value of B_m, $E[B_m]$ can be expressed as follows:

$$E[B_m] = \sum_{i=1}^{m} \varrho_i b_i \tag{7.15}$$

Let $\mathscr{P}\{fail(u)\}$ denote the failure probability of a node. The normal failure of a node depends on its energy dissipation and other external factors such as technical fault or tampering. The failure probability of a node can be expressed using the Poisson distribution as follows:

$$\mathscr{P}\{fail(u)\} = K_P.e^{\lambda} \tag{7.16}$$

Where λ is the average energy dissipation rate of a sensor, and K_P is a constant that depends on the external factors. As the sensors are uniformly distributed, let there be a $\mathfrak{N}_i$ number of sensors in the path $\mathbb{P}_i$. The data is successfully delivered through the path $\mathbb{P}_i$ if and only if no sensor in the path fails. Then ϱ_i can be expressed as,

$$\varrho_i = (1 - \mathscr{P}\{fail(u)\})^{\mathfrak{N}_i} \tag{7.17}$$

$\mathfrak{N}_i$ depends on the distance (number of hops) between the node and the sink. As the sensor nodes are uniformly distributed, $\mathfrak{N}_i$ would be uniform based on the position of the source node.

Figure 7.7 plots the expected number of successful data receptions with respect to the failure probability, as per the theoretical analysis. It can be seen from the figure that the increasing *sink-connectivity* improves the performance in terms of the expected number

TH-1251_11610108

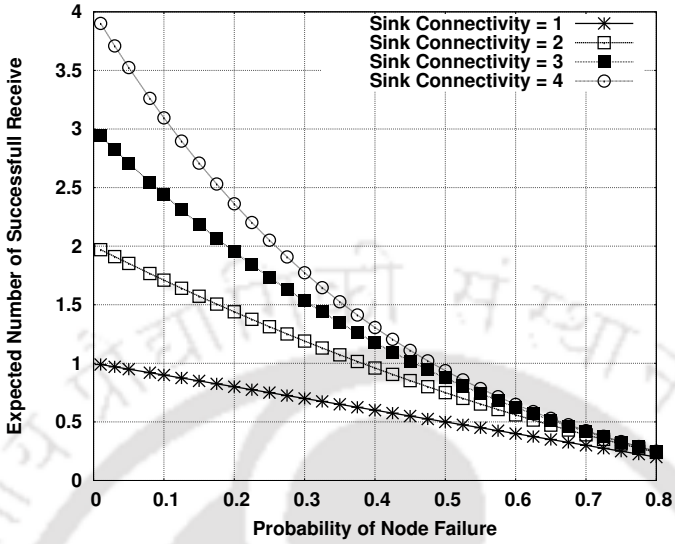

Figure 7.7: Exp. no. of successful data reception

of successful data reception. With a failure probability around 0.4, at least one copy of data is received at the sink for *sink-connectivity* = 2. An increasing *sink-connectivity* further improves the reliability in data transmission. In next section, the reliability of the proposed *RelBAS* protocol is analyzed using the simulation results.

7.6 Simulation Results

The proposed *RelBAS* protocol has been implemented in NS-2.35 [7] discrete event network simulation framework, and the performance is analyzed and compared with other protocols using the simulation results. *RelBAS* is implemented as a loadable module in NS-2 protocol stack, between the data link layer and the network layer. For comparison purpose, three other representative protocols have been selected from each of the following classes.

(i) *Multi-sink forwarding:* The MUSTER protocol [133] has been used as a representative protocol for multi-sink based data forwarding in a sensor network. MUSTER builds multiple paths towards a set of sinks in a distributed manner. However, it aims to merge different paths from a single source to reduce the number of sensors involved in the data forwarding.

(ii) *Multi-path single sink forwarding:* The scheme proposed in [62] has been used as a representative protocol for multi-path single sink data forwarding. The scheme builds a set of paths towards a single sink based on computation of a path metric

TH-1251_11610108

Table 7.2: Communication parameters of the sensors

Parameter	**value**
Data rates	250 Kbps
Transmit (TX) Power	0 dBm
Receive (RX) Power	-24 dBm
RX sensitivity	-94 dBm
Communication range	25 m
Sensing Range (magnetic sensors)	10 m
Current Draw (RX)	19.7 mA
Current Draw (TX)	17.4 mA
Current Draw (Idle)	15 mA
Current Draw (Sleep)	1 μA
External Voltage	2.7 Volt

that minimizes the end-to-end data delivery time. The protocol is localized in nature, and provides good reliability assurance while ensuring the timeliness for the data delivery.

(iii) *Traditional single-path forwarding:* A robust gradient based data delivery protocol, as proposed in [192] has been considered that represents the traditional single-sink data forwarding mechanisms. The protocol builds and maintains a cost field for providing each sensor the direction to which the sensing data needs to be forwarded. The authors have shown that the protocol can deliver data with high degree of reliability in spite of multiple node failures.

The representative protocols for each classes are chosen in a way, that they have shown to provide better performance than other existing protocols from the corresponding class in the literature. These protocols are also implemented in NS-2 as a modification over the existing routing protocols. It can be noted that these protocols do not assure the connectivity and the coverage during node failures; and therefore, the performance of the proposed *RelBAS* protocol is compared with these protocols in terms of reliability, redundancy, transmission delay and energy dissipation.

7.6.1 Scenario set-up

In the simulation scenario, the sensing and the communication radius of a sensor are assumed to be 10 m, and 25 m respectively. The arena is considered to be 400 $\times$ 50

TH-1251_11610108

m^2. The sensor nodes are distributed in the arena using a uniform distribution with mean sensor distance as 1 m. Therefore, in every sensing disk in arena, there are on average 10 numbers of sensors deployed. The maximum *sink-connectivity* used for the simulation is 6, which is less than the minimum degree of connectivity for any sensor in the arena (which is on average 8). Uniform distribution of sensors are used to satisfy the optimal performance of the proposed *RelBAS* protocol, as discussed in Subsection 7.3.2. However, the effect of deviation from the uniformity due to sensor node failure is also analyzed through the simulation results. The communication parameters for the sensor nodes are assumed based on MicaZ technology, as shown in Table 7.2. With these settings, the average power consumption in receive, transmit, idle and sleep states are $53.19mW$, $46.98mW$, $40.5mW$ and $0.0027mW$, respectively. It can be noted that the Crossbow micaZ technology provides a flexible communication platform for sensor nodes, above which the specific sensor boards can be mounted.

Every active sensor uses CSMA based channel access protocol for communication with the neighbors. The sensory data is considered to be small UDP packets with mean packet size of 64 bytes. The UDP packets are generated on the triggering of an event, called the *'epoch'*. At every *epoch*, one UDP packet is generated, and transmitted towards the sinks based on the forwarding protocol. The sensors communicate with each other using 802.15.4 complaint MicaZ technology, that can support up to $250Kbps$ data rate using $2.4GHz$ channel bandwidth. The sinks are placed at uniform distances from each-other. Number of sinks placed in the arena is considered to be two more than the required *sink-connectivity*. For ensuring the $k-$barrier coverage, the value of k is assumed to be 5.

During simulation, it has been considered that sensor nodes fail based on a Poisson distribution. A sensor node can fail because of two reasons - either due to energy exhaustion, which is the normal failure, or due to a sudden crash. Therefore, the failure of a sensor node can be modeled as a self-similar renewal and ergodic process[3] (normal failure) with implicit randomness in it (the sudden failure due to crash). Therefore, the lifetime of a sensor can be considered as a Poisson variable, as modeled in [149], with mean as the lifetime for the normal failure.

Every simulation scenario is executed for 10 different times with different seed values used for positioning the sensors through uniform distribution. The epochs are also generated using binomial distribution, based on the seed value. Binomial distribution

[3]The process is renewal, because after a sensor node fails, it is replaced by another sensor with similar lifetime characteristics. Similarly, the process is also ergodic, as the self-similar nature of the sensory failures can be modeled by considering one representative sensor only.

TH-1251_11610108

is considered for the epoch generation, as they are event triggered[4]. The average result from the 10 simulations are used to plot the graphs.

7.6.2 Metrics used for the performance analysis and the comparison

Following metrics are used for analyzing the performance of the proposed *RelBAS* protocol and comparing the result with other schemes.

- *Reliability:* The reliability is defined as the ratio of the total number of epochs generated at the sensors, to the total numbers of epochs detected. An epoch is considered to be detected, if at least one corresponding UDP packet is correctly received at the sinks. The value of the reliability metric lies in between 0 and 1, where 0 denotes complete unreliable, and 1 denotes complete reliable.

- *Redundancy-on-Failure:* Let $\Im$ denote all epochs generated in the network. Further assume that ∇_i denotes the total number of UDP packets received at the sinks corresponding to epoch i. Then the redundancy-on-failure is represented as follows,

$$\text{Redundancy-on-Failure} = \frac{\sum_{i \in \Im} \frac{\max\{0, \nabla_i - 1\}}{\nabla_i}}{|\Im|}$$

  $\nabla_i \in [0, m]$, where m is the *sink-connectivity*. The value of $\nabla_i \to 0$, as probability of node failure tends to 1. Therefore, the redundancy-on-failure represents the fraction of packets that are received redundant. The maximum value of this parameter can be $(m-1)/m$, when all the UDP packets for all the epochs are received at the sinks.

- *Reliability to Redundancy-on-Failure Ratio (RRF Ratio):* The RRF ratio is the ratio of the reliability to the redundancy-on-failure. This ratio represents the performance gain in terms of reliability where the cost is the redundancy. The RRF ratio is used to analyze the trade-off between reliability and redundancy in the proposed scheme.

7.6.3 Reliability vs Redundancy

For these sets of simulations, it has been considered that every sensor generates epochs based on a log-normal distribution, with 12 number of epochs per minute. The parameter α is taken as 0.5, that gives equal weight to the hop-count and the energy metrics, used for the data gathering tree construction. Three metrics, as discussed already, are considered to evaluate the trade-off in reliability and redundancy in the proposed *RelBAS* protocol.

TH-1251_11610108

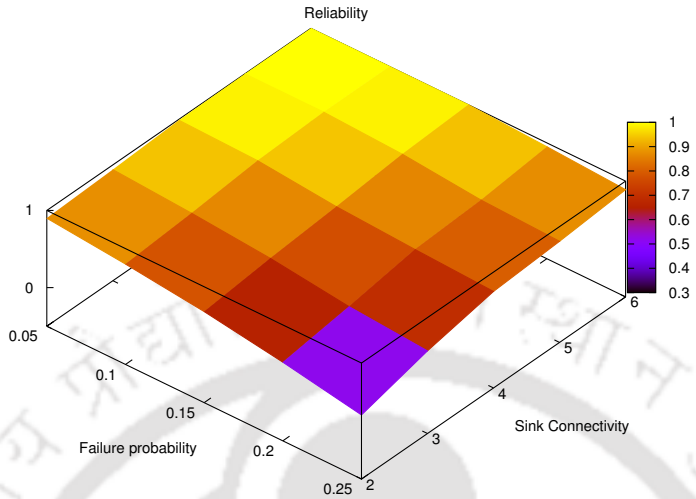

Figure 7.8: Reliability: On failure probability and *sink-connectivity*

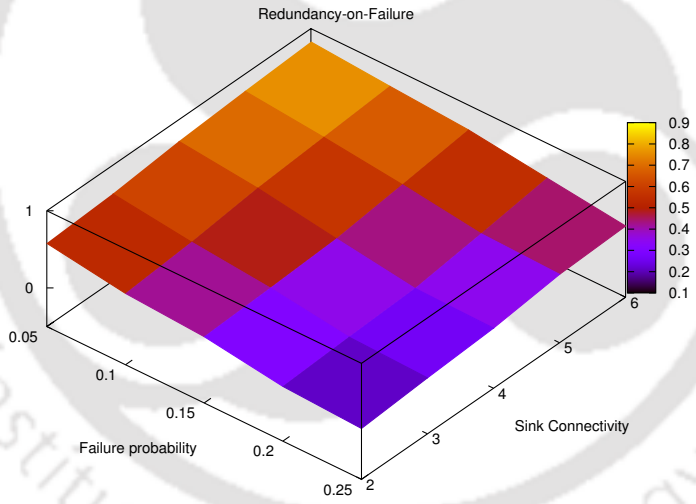

Figure 7.9: Redundancy-on-Failure: On failure probability and *sink-connectivity*

Figure 7.8 shows the reliability with respect to the failure probability and the *sink-connectivity*. The figure reveals that the reliability decreases as the failure probability increases. On the contrary, the reliability increases as the *sink-connectivity* increases. The reliability is minimum for *sink-connectivity* 2 and failure probability 0.25. Figure 7.9 depicts the redundancy-on-failure versus the failure probability and the *sink-connectivity*.

[4]Either the event generates an epoch, or remains silent. Therefore there are two possibilities with equal probabilities

TH-1251_11610108

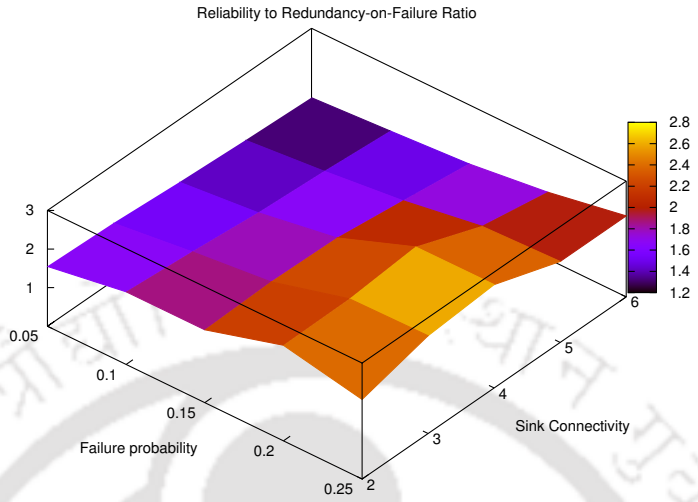

Figure 7.10: RRF Ratio: On failure probability and *sink-connectivity*

It can be seen from the figure that the redundancy gets reduced when the failure probability is high and the *sink-connectivity* is low. The interesting observation is inferred from Figure 7.10 that shows the RRF ratio, with respect to the failure probability and the *sink-connectivity*. For a fixed failure probability, the figure shows the minimum *sink-connectivity* that is sufficient to provide maximum reliability with minimum redundancy. For example, when the failure probability is less than 0.15, dual *sink-connectivity* is sufficient to provide maximum reliability with minimum redundancy. On the other hand, for failure probability greater than 0.15, triple *sink-connectivity* is required for this purpose. The analysis reveals that for the optimum performance of the *RelBAS* protocol, the *sink-connectivity* needs to be decided based on the failure probability. The failure probability can be estimated from the specification of the sensor lifetime and the nature of the deployment region.

7.6.4 Performance comparison: RelBAS vs other protocols

These group of simulations are executed in three sets. For the first set, the performance for the *RelBAS*, along with other three representative protocols as mentioned earlier, are evaluated with respect to the failure probability. For this set, the *sink-connectivity* and the mean number of epochs generated per minute at every sensor are considered to be 2 and 12, respectively. In the second set of simulations, the performance of four protocols are compared with respect to the *sink-connectivity*, by keeping failure probability fixed at 0.15. Similar to the earlier scenario, the mean number of epochs generated per minute at every

TH-1251_11610108

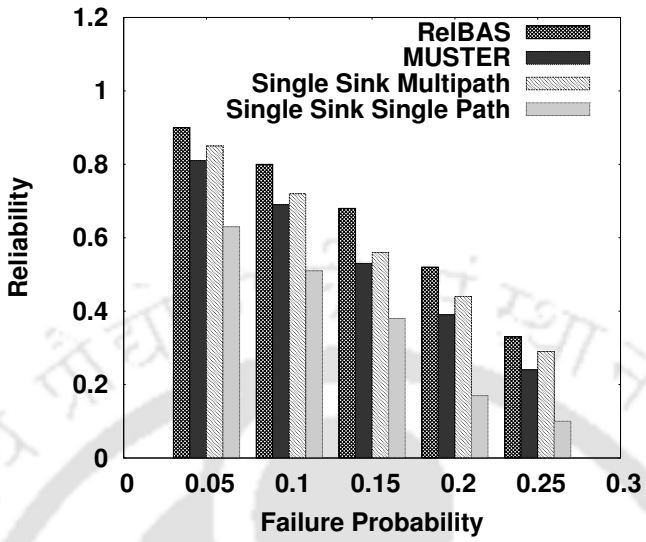

Figure 7.11: *RelBAS* vs others: Reliability over failure probability

sensor is fixed at 12. The reliability and the redundancy is compared in these two sets of simulations. The third set of simulations compare four protocols in terms of the average transmit delay, average energy dissipation and the variance in energy dissipation. For this set of simulations the failure probability and the *sink-connectivity* is considered as 0.15 and 2, respectively. For all these scenarios, α is taken as 0.5, and the epochs are generated based on a log-normal distribution. It can be noted that for single-sink multi-path protocol the term *'sink-connectivity'* is not appropriate; and therefore, the term *'path-connectivity'* is used instead. Path-connectivity, denoted by γ, represents the maximum number of paths from a source to the sink. According to the Poisson distribution failure probability equals to 0.25 gives a very high chance of network disconnection, as 25% of the active nodes can fail simultaneously.

Reliability and redundancy

Figure 7.11 and Figure 7.12 compare the protocols in terms of reliability. It is obvious that the *sink-connectivity* or the path-connectivity does not make any sense for the traditional single path forwarding protocol; and therefore, this is included in the comparison results only for the scenario where the performance is compared with respect to the failure probability. From Figure 7.11, the single path forwarding provides minimum reliability among four protocols. The multi-path forwarding protocol performs better than MUSTER with respect to the failure probability as well as the *sink-connectivity* or the *path-*

TH-1251_11610108

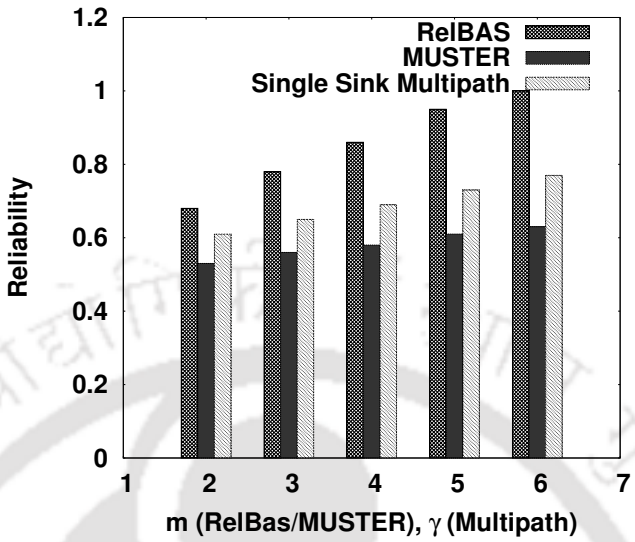

Figure 7.12: *RelBAS* vs others: Reliability over *sink-connectivity*

connectivity. The major problem with MUSTER is that it tends to merge the paths to the same sink to reduce the number of nodes involved in forwarding. However, this approach affects the reliability in packet delivery, because a single node failure in the path may result in the loss of all packets. All paths in a multi-path single sink forwarding protocol converges towards the same sink. The sensor nodes near a common sink have to forward an equal amount of data, and the energy dissipation rate of a sensor is directly proportional to the amount of data it forwards. Therefore, all such sensors near a single sink have equal probability of failure, which may result in data loss from all paths, even with multi-path forwarding. As a result, *RelBAS* provides better reliability compared to MUSTER, as shown in Figure 7.11 and Figure 7.12, by forwarding data into different sinks through edge disjoint paths. This increases the possibility that at least one copy of data would be delivered to one of the sinks.

Figure 7.13 and Figure 7.14 compare the redundancy-on-failure for three protocols, *RelBAS*, MUSTER and multi-path with respect to the failure probability and the *sink-connectivity*. The redundancy-on-failure value for the single path protocol is always zero, as no redundant packet is transmitted ever. The figures reveal that the redundancy for MUSTER is less than that for other two protocols. However, if the parameter is compared with respect to the RRF ratio, as shown in Figure 7.15 and Figure 7.16, the proposed *RelBAS* protocol provides maximum reliability with minimum redundancy. Therefore, it can be said that the gain (reliability) per cost (redundancy) is more in the proposed

TH-1251_11610108

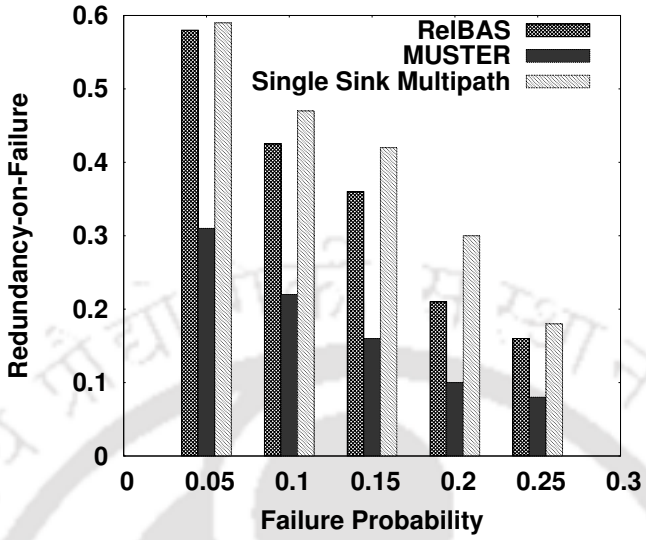

Figure 7.13: *RelBAS* vs others: Redundancy-on-failure over failure probability

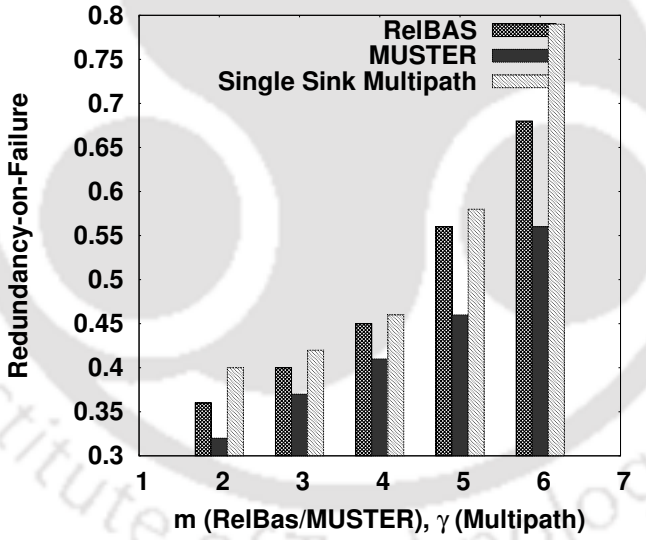

Figure 7.14: *RelBAS* vs others: Redundancy-on-failure over *sink-connectivity*

protocol, and it effectively exploits the redundancy in data delivery to improve the network reliability. The RRF ratio for the proposed *RelBAS* protocol becomes significantly higher than that for other two schemes, which explores the redundancy to improve the reliability, as shown in Figure 7.15. Further, Figure 7.16 reveals that the RRF ratio for the proposed *RelBAS* protocol is maximum with *sink-connectivity* 3 (failure probability 0.15), which also reflects the result obtained in Figure 7.10. One interesting observation can be made

TH-1251_11610108

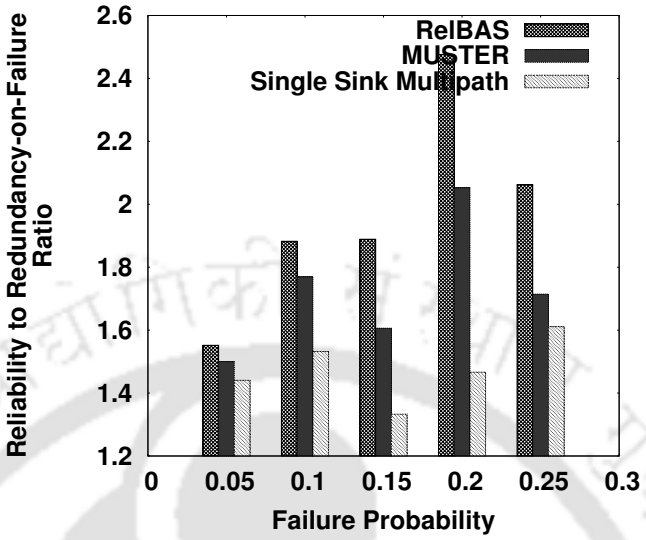

Figure 7.15: *RelBAS* vs others: RRF ratio over failure probability

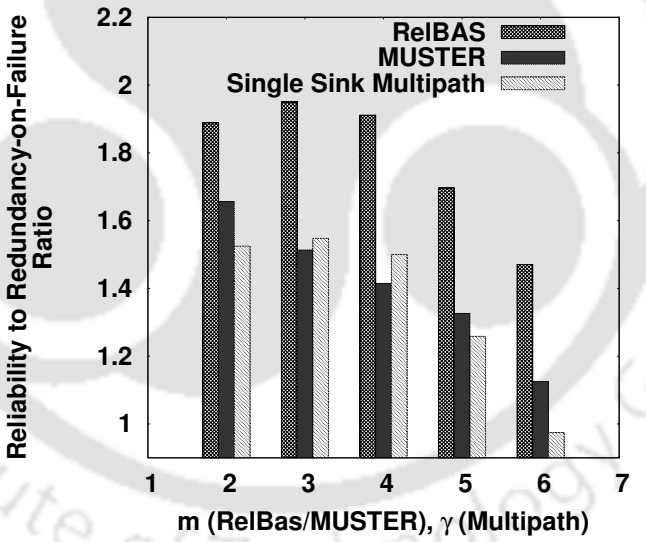

Figure 7.16: *RelBAS* vs others: RRF ratio over *sink-connectivity*

by comparing Figure 7.11, Figure 7.13 and Figure 7.15, that *RelBAS* provides more reliability with less redundancy. To explain this, let us consider Figure 7.17 that shows the distribution of packet reception for the multi-path routing protocol with respect to the failure probability for *sink-connectivity* 2. The figure shows that either both copies of the packet is lost, or they are received through both the paths. Only a fraction of packets are received through either of the paths. The reason behind this is the localization of the

TH-1251_11610108

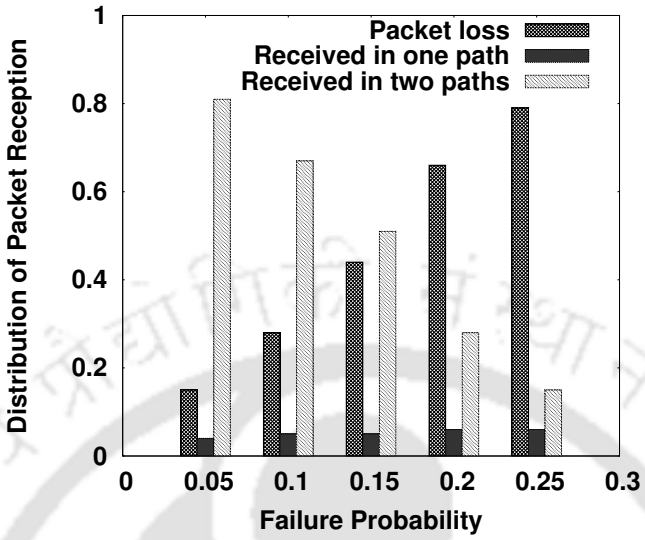

Figure 7.17: Multi-path forwarding: Distribution of packet reception

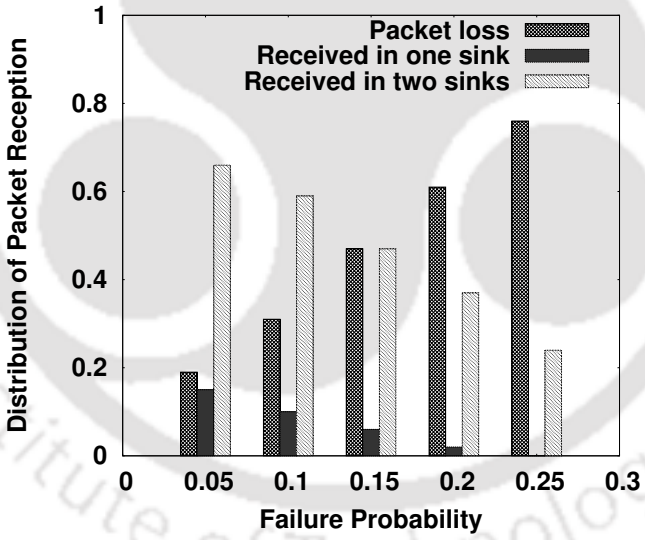

Figure 7.18: MUSTER: Distribution of packet reception

sensor failures - the probability of failure for all sensors near a sink are similar, as discussed earlier. Therefore, because of frequent sensor failure at the same locality, both the paths fail to forward data in most of the cases. Similar trend is observed for the MUSTER protocol, as shown in Figure 7.18. As a consequence of path merging, a single node failure affects all paths towards it. On the contrary, Figure 7.19 reveals that the proposed *RelBAS* protocol takes maximum advantages of the redundancy. The paths towards multiple sinks

TH-1251_11610108

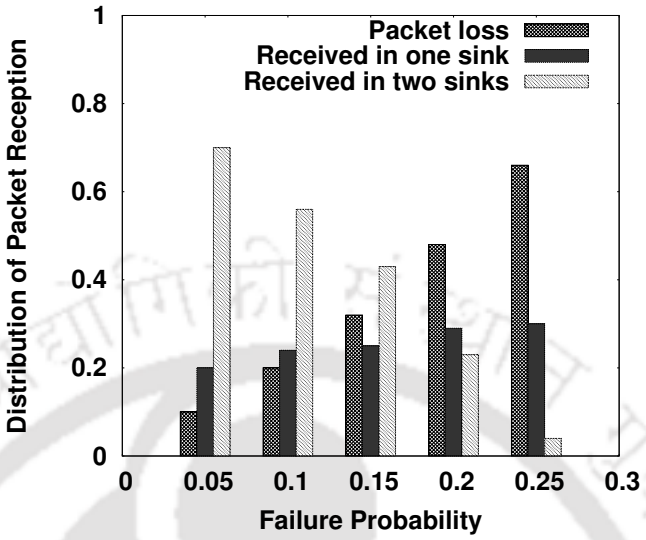

Figure 7.19: *RelBAS*: Distribution of packet reception

are edge-disjoint; and therefore, in most of the cases, sensor failure affects only one of the paths resulting in correct delivery of data through another path. The figure shows that when the failure probability is high, most of the data is forwarded through either of the paths, which is just the opposite behavior observed in the MUSTER and the multi-path single-sink forwarding protocols.

Delay and Energy Dissipation

The performance of three protocols, *RelBAS*, MUSTER and the multi-path forwarding, in terms of the average transmission delay, average energy dissipation and the variance in energy dissipation has been compared. These parameters are compared with respect to the average number of epochs generated per minute at a sensor. Figure 7.20 to Figure 7.22 show the comparison among three protocols with respect to transmission delay and power consumption (average and variance). As discussed earlier, every simulation is conducted for 10 times with different seed values, and the average is taken to plot the graphs. The vertical lines in these graphs demonstrate the difference between the maximum and the minimum outcomes from every simulation, called the confidence interval.

Figure 7.20 compares the average transmit delay for these three protocols. The delay is calculated as the time between the packet for an epoch is generated and the epoch is detected (the first packet for that epoch is received at one of the sinks). Similar to the earlier scenario, the average transmit delay for the multi-path forwarding protocol is

TH-1251_11610108

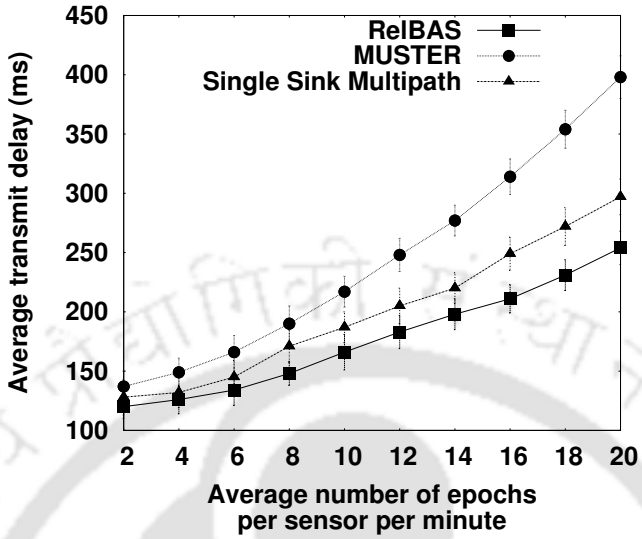

Figure 7.20: *RelBAS* vs others: Average transmission delay

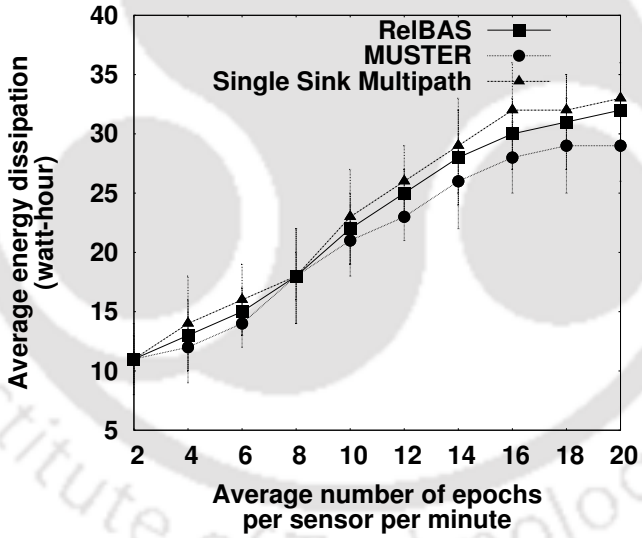

Figure 7.21: *RelBAS* vs others: Average energy dissipation

more because of the convergence of all the paths from a source towards a common sink. Convergence of multiple paths towards a common sink increases the network congestion near the sink, and the transmission of the packets get delayed. The path merging technique proposed by the MUSTER protocol also results in the similar results by increasing the network congestion to a single path. *RelBAS* shows minimum delay among the three protocols by distributing the traffic load among all active sensors; and henceforth, reducing

TH-1251_11610108

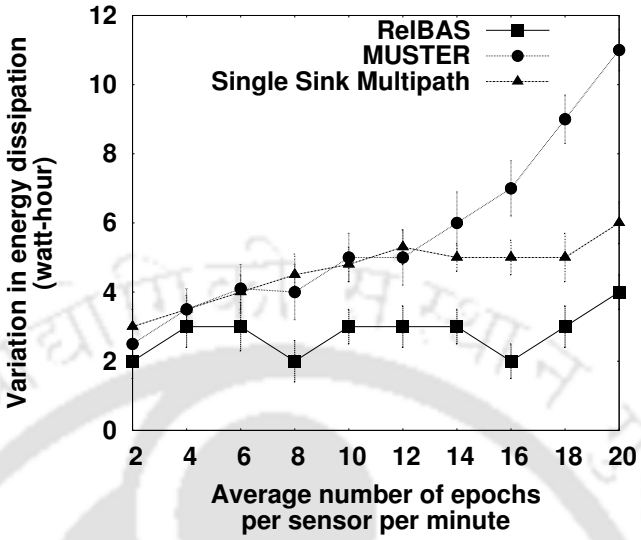

Figure 7.22: *RelBAS* vs others: Variance in energy dissipation

the network congestion.

The energy dissipation of a sensor node depends on the duration in which the senor involves in data communication. Figure 7.21 compares three protocols in terms of the average energy dissipation among all sensors. The figure shows that average energy dissipation rate of all three protocols are almost similar. However, Figure 7.22 reveals that the variation in energy dissipation deviates significantly among three protocols. A high variation in energy dissipation indicates that some nodes of the network dissipates energy more than some other nodes, making a possibility of network disconnection. The figure shows that the variance in energy dissipation tends to increase exponentially for MUSTER, as the epoch generation rate increases. MUSTER aims at path merging to reduce the number of nodes involved in forwarding. As a result, the nodes that act as a forwarder, dissipate more energy than others, leaving the possibility of early failure. Similar situation happens for the nodes near the sink in case of the multi-path forwarding protocol. In the proposed *RelBAS* protocol, every sensor becomes the part of exactly one skeleton corresponding to one sink; and therefore, the forwarding load is distributed uniformly among the sensors. Accordingly, the variance in energy dissipation is reduced to a significant amount. From these results, it is observed that the average lifetime of the network (the time duration starting from the network initialization till the first unrecoverable hole is created, in terms of either communication or sensing coverage) is more in the proposed *RelBAS* protocol, compared to others.

TH-1251_11610108

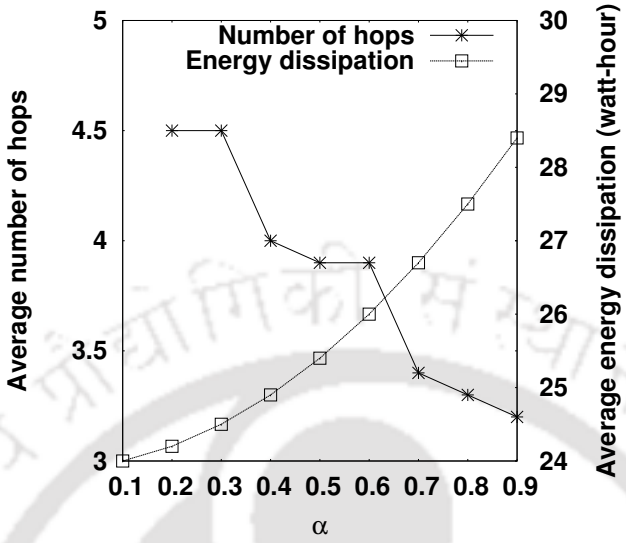

Figure 7.23: *RelBAS*: Effect of α value

7.6.5 Effect of α value on the design of RelBAS protocol

In the design of the proposed *RelBAS* protocol, the parameter α is used to trade-off among the shortest-hop path and the energy distribution during the parent selection. If α value is more, priority is given to the node that provides the shortest-hop path. On the contrary, small α value prioritizes the nodes with higher remaining energy to be selected as the parent of the data gathering tree. This simulation analyzes the effect of the parameter α over the performance of the proposed *RelBAS* protocol. For this simulation, the failure probability and the *sink-connectivity* are considered to be 0.15 and 2, respectively. Every sensor node generates epochs using log-normal distribution with mean 12 epochs per minute. Figure 7.23 shows the effect of α value over the hop-counts and the average energy dissipation. As expected, increasing α value decreases average hop-counts used for communication, and increases the average energy dissipation. In practical scenario, α value needs to be selected based on the delay-sensitivity of the application. If strict bound on the timeliness of data delivery is required, large α value needs to be selected. On the other hand, if delay-bounded delivery is sufficient and minimum delay delivery is not necessary, then the α value can be selected that provides bounded delay within the requirement, and minimizes the average energy dissipation.

TH-1251_11610108

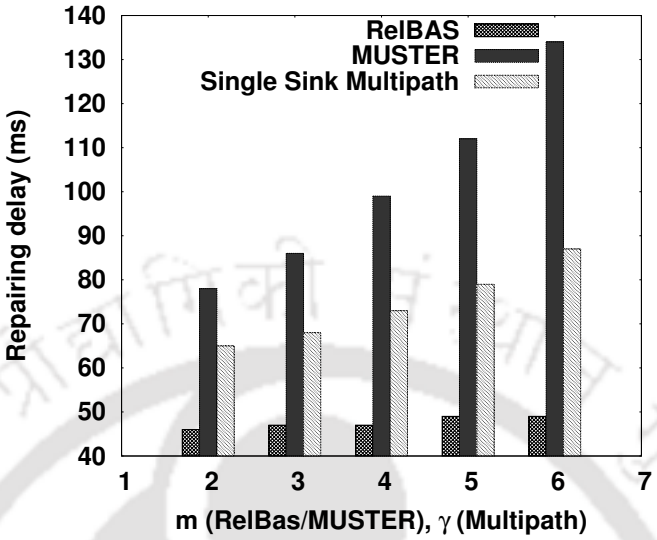

Figure 7.24: *RelBAS* vs others: Repairing delay

7.6.6 Analyzing the effect of node failure

Figure 7.24 shows the repairing delay with respect to the *sink-connectivity* or the *path-connectivity*. Single node failure is considered in this simulation, where every sensor generates epochs using a log-normal distribution with mean 12 epochs per minute. The value of α is considered to be 0.5. The figure reveals that as the *sink-connectivity* increases, the repairing delay for MUSTER increases exponentially. In case of the MUSTER protocol, a single failure affects all paths, as a result of path merging. Therefore, all paths need to be repaired, which increases the repairing delay significantly. The repairing delay for single sink multi-path forwarding is also very high, because a single failure requires the complete path to be re-established using the path searching mechanism. The proposed *RelBAS* protocol rely on the local repairing only, where a failed node is replaced by another redundant node, and based on the local computation the paths are re-established. As a consequence, the path repairing delay for the proposed protocol is significantly low, as shown in Figure 7.24.

Figure 7.25 compares the repairing delay for three protocols in terms of the number of simultaneous failure in the same locality. *Sink-connectivity* or *path-connectivity* is considered to be 2 in this scenario, and the value of α is 0.5. The epoch generation rate at every sensor is considered similar to the earlier scenario. With the increase in the number of simultaneous node failure in the same locality, the repairing delay for MUSTER increases exponentially, as earlier, because of the effect of path merging. The

TH-1251_11610108

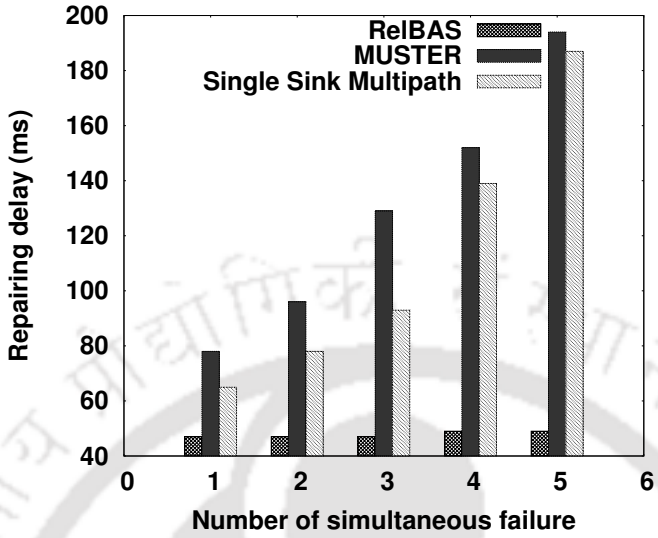

Figure 7.25: *RelBAS* vs others: Repairing delay on simultaneous failure

repairing delay for the single source multi-path forwarding also increases exponentially as the number of simultaneous failure in the same locality increases. The reason is the convergence of all paths towards a single sink, where multiple simultaneous failure in the same locality may affect all paths. The proposed *RelBAS* protocol incurs minimum delay through local repairing, as long as simultaneous failures do not affect the connectivity. As the repairing is based on the local coordination only, all paths are repaired within a small delay. However, if the paths cannot be recovered and a zone failure is detected, the information is notified to the sinks, as discussed in the previous section.

Figure 7.26 compares the remaining connectivity to the sinks (after disconnection from the sinks due to repeated node failures) for three protocols with respect to the total number of epochs generated in the network. For multi-sink (or multi-path) connectivity, the remaining connectivity is defined as the number of sinks with which a node can maintain a path. This simulation considers only the normal failure due to energy exhaustion, and does not consider the sudden failure. A single node failure affects multiple paths for MUSTER. At the same time, because of the path merging property, failure is localized in the MUSTER protocol. The nodes that act as forwarder, tend to fail fast due to energy exhaustion. As a result, for MUSTER, the loss in *sink-connectivity* starts later compared to *RelBAS* (because of less dissipation in average energy), but occurs faster (because of high variance in average energy) as reflected in the figure. The disconnection to the third and the fourth sinks occur simultaneously in MUSTER, as a result of which

TH-1251_11610108

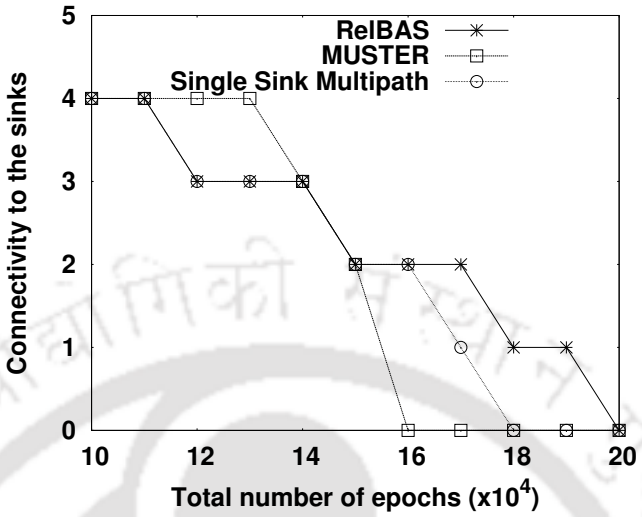

Figure 7.26: *RelBAS* vs others: Connectivity loss

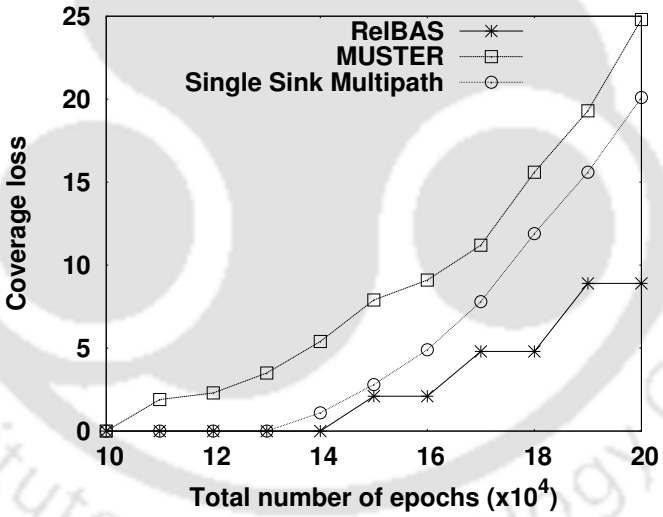

Figure 7.27: *RelBAS* vs others: Coverage loss

the network connectivity gets affected earlier than the *RelBAS* protocol. This graph shows that *RelBAS* improves the network connectivity compared to other protocols.

Figure 7.27 shows the coverage loss for three protocols with respect to the number of epochs generated in the network. The coverage loss is defined as the percentage of area that remains uncovered due to a series of node failures (or simultaneous node failures in the same locality). This simulation considers only the normal failure due to energy exhaustion,

TH-1251_11610108

and does not consider the sudden failure. The *sink-connectivity* is considered to be 3 in this scenario. Because of high variation in energy dissipation, the node failures are more localized in the MUSTER protocol, which results in significant coverage losses. For multi-path single sink forwarding, coverage loss occurs faster near the sinks. The proposed *RelBAS* protocol considers both the connectivity and the coverage during the scheduling, data gathering tree construction and the failure handling. As a result, the proposed protocol provides improved support in terms of coverage, as depicted in Figure 7.27.

7.7 Summary

In this chapter, a reliable data gathering protocol has been proposed for critical infrastructure monitoring with the sensor network technology. The proposed *RelBAS* protocol assures the reliability in delivery of sensory data to the base station by constructing multiple trees rooted at different sinks, such that from every node there exists multiple node-disjoint paths to a set of sinks. Thus, any single node failure can be handled through the local tree repairing maintaining both the connectivity and the sensing coverage. The proposed protocol offers high degree of reliability in data delivery with the aim of satisfying all the metrics like the energy efficiency, timeliness in delivery of data and the network connectivity and sensing coverage during the node failures. The detection technique for any zone failure due to multiple simultaneous node failures has also been incorporated in the proposed *RelBAS* protocol. The simulation results have shown that the proposed protocol provides better reliability with less redundancy, and performs better than MUSTER and single sink multi-path data delivery protocols in terms of the reliability, transmit delay and the network lifetime. Further, the proposed protocol provides better support for the network connectivity and the sensing coverage with minimum time for the path repairing during node failures, compared to other two protocols.

TH-1251_11610108

Chapter 8

Conclusion and Future Scope

8.1 Conclusion

Sensor nodes deployed in the terrain of interest sense physical phenomena and the generated sensory data are accumulated at a common sink for further analysis. An efficient collection of sensory data satisfying the desired QoS, such as bounded delay, no loss, etc., determines the performance of the running application. To provide an uninterrupted smooth service, the topology management protocols for sensory data collection are designed to work transparently with both the application interface and the hardware platform. The set of protocols proposed in this thesis is designed as the loadable module that work transparently in the network layer and below the application layer of the TCP/IP protocol stack. The Topology Management Module (TMM) and the Application Message Controller (AMC) have been designed as the submodules that work cooperatively to offer reliable application services. The TMM is responsible for data gathering tree construction and maintenance activities to handle node failures, where as the AMC ensures reliable delivery of sensory data to the sink by controlling the application message flow. Considering the pattern of node failure due to energy depletion, the effect of node failure on the network connectivity and the sensing coverage, and different application specific design requirements various research challenges are explored in the thesis. The major contributions of the thesis can be summarized as follows:

First, a set of distributed message-passing algorithms has been proposed for data gathering tree construction that eventually becomes a BFS tree through incremental level improvement. Arbitrary node failure is supported by the proposed TMM that precomputes an alternate parent at every node during the tree construction. So, on failure of the parent node, the affected path is repaired locally as the victim node establishes the new tree edge

TH-1251_11610108

with its alternate parent. The proactive activities of TMM offers low overhead, both in terms of repairing delay and control message. Also, the TMM has been extended that works reactively to support multiple simultaneous node failures. Then, a theory of initial deployment with redundancy has been developed considering the potential node failure due to energy exhaustion. The effectiveness of the proposed scheme has been verified through simulation results for different deployment pattern of sensor nodes.

In second contribution, a gradient based node deployment strategy has been proposed based on the uneven load distribution in the target terrain. Nodes near the sink are highly loaded compared to the ones in the terrain periphery. So, they die out of energy faster creating network holes. To avoid the interruption in application services due to arbitrary node failure, the density of the deployed nodes has been estimated based on the gradient of energy dissipation that decreases from the sink towards the terrain periphery. The gradient of the node density at a particular node in the data gathering tree depends on the number of nodes in its rooted subtree. The proposed theory has been developed considering both the steady sensing and the periodic sensing model of sensor nodes. The effectiveness of the gradient estimation has been established through the worst case analysis of sensor network calculus. Simulation results proves the correctness of the estimated gradient, which is bounded above the actual traffic-load dependent gradient value.

In third contribution, considering the gradient based node deployment with redundancy, the activities of the TMM and the AMC have been described. In this case, the TMM constructs a load balanced BFS tree where the uneven load between two adjacent children is shared at every intermediate node. The node failure is supported by selection of a suitable redundant node as the replacement maintaining the network connectivity and the sensing coverage. The AMC handles the sensory data flow at every node to avoid data loss or unwanted delay in the delivery. The cooperative interaction between the TMM and the AMC offers efficient data collection with node failure support, the validation of which has been provided by the simulation results.

Finally, to investigate the performance of topology management protocol under various application specific QoS requirements like delay and reliability, two separate application scenarios are considered. Two different topology management protocols, namely '*ADCROSS*' and '*RelBAS*' have been designed for sensory data collection in road sensor network and critical infrastructure information system, respectively. The design of these two protocols are based on different sensing coverage requirement and other metrics specific to the desired QoS of the applications. The highlights of these last two contributions are summarized as follows:

TH-1251_11610108

1. *ADCROSS*: Considering the requirement of road traffic monitoring the concept of $k-$strip length coverage has been introduced, which demands every passing vehicle will be detected by at least k active sensors along the length of the road segment. Sensors are deployed along the roadways in high redundancy such that the nodes support sleep based energy saving. In every time slot a set of active sensors are selected maintaining the connectivity as well as $k-$strip length coverage along the length of the road segment. A centralized schedule has been proposed for optimum selection of active nodes that has been modeled as a graph optimization problem. A distributed variation of the active node selection strategy has also been proposed for practical implementation. The application message flow is handled by allowing every node to be included into two separate data gathering trees, rooted at the nearest sinks. Thus, disruption in one path due to node failure does not affect other paths. Simulation results show that ADCROSS performs better than the similar existing protocols in the literature for road monitoring.

2. *RelBAS*: A robust topology management protocol has been proposed for reliable data gathering in critical infrastructure information system. Multiple data gathering trees are constructed by the TMM, each one rooted at different sink. From every active node there exists multiple node disjoint paths to multiple sinks such that the disruption in one path due to node failure does not affect the other paths. Strong requirement of reliable data delivery as well as barrier coverage in the border area of a critical infrastructure are satisfied by the activities of the TMM in the proposed RelBAS protocol. The analysis of simulation results reveals that the performance gain in terms of reliability justifies the cost in terms of redundancy. RelBAS performs better than other existing protocols in the literature for critical infrastructure monitoring for ensuring reliable delivery of sensory data.

8.2 Future Scope

From the summary it can be observed that the thesis contributes to the field of topology management protocol designing for efficient data gathering in sensor network considering fault-tolerance, energy efficiency, QoS requirement and overhead issues. Several research challenges have been addressed while designing the TMM and the AMC to ensure reliable data gathering. Further, some new design objectives and interesting research challenges have been identified during the progress of this research. Some of the potential directions to which the contribution of the thesis can be extended are discussed as follows:

TH-1251_11610108

In this thesis the issues of initial deployment have been explored and a gradient based sensor node deployment strategy has been proposed. The proposed gradient based deployment scheme assumes the failure of nodes as a result of energy exhaustion. Any arbitrary single node failure due to tampering or technical fault does not affect the performance of the proposed scheme. However, if multiple nodes fail simultaneously within a close proximity, the event can be considered as a potential threat of a zone failure, as discussed for *RelBAS* protocol in Chapter 7. *RelBAS* is able to detect such event of zone failure and then report the same to the base station for manual intervention. The critical security requirement in the border areas demand the recovery of the affected zone as soon as the detection is performed. Sensor nodes are required to be re-deployed in the affected zone maintaining the sensing coverage in the affected zone as well as the connectivity with the existing communication network. The challenges and issues inherent with the re-deployment of nodes in a particular affected zone can be explored as an extension of the problems addressed in the thesis.

All the works contributed to this thesis are based on homogeneous sensor network environment such that every sensor is equi-potential in terms of sensing and processing capacity and battery power. Another direction of future work can be considered in the heterogeneous sensor network environment where few sensor nodes are equipped with actuator and are able to move in 2-dimensional space. These special nodes are available in specific areas of user defined interest. Whenever a network hole is created due to the failure of any arbitrary single node or a set of nodes, these special nodes adjust their positions such that the network connectivity and the sensing coverage is maintained. The self-organization of sensor nodes to reconfigure the underlying topology with a support to potential node failure is an interesting problem to address. The main challenges in this problem would include the identification of the network hole, selection of the nodes to perform the task, movement of the nodes towards the correct direction, optimization of the solution space and so on.

Security of the sensory data packets is an important requirement to satisfy overall reliability of the application services. The proposed *RelBAS* protocol assures the reliable data delivery to the base station by forwarding data packets in multiple node-disjoint paths. The disruption in one path due to node failure does not affect other paths, which assures the delivery of sensory data without any loss due to node failure. The *RelBAS* protocol is developed on the assumption that sensor nodes are trusted and they communicate in a secure channel. However, considering the increasing threat of potential invasion on the security of critical infrastructure, the issues related to the existence of

TH-1251_11610108

malicious nodes in the network could be analyzed as future scope of the work. Few sensor nodes equipped with intrusion detection system can be deployed for detecting any vulnerabilities in the network caused due to the violation of security criteria. Path selection based on the trust value, which evolves through information collection from different IDSs, would be an interesting topic of investigation for adding another dimension of reliability to the application services considering various challenges in implementation point of view.

TH-1251_11610108

TH-1251_11610108

References

[1] 50 sensor applications for a smarter world. [Online]. Available: http://www.libelium.com/top_50_iot_sensor_applications_ranking

[2] Magnetic sensors, honeywell. [Online]. Available: http://www.magneticsensors.com/vehicle-detection-solutions.php

[3] MicaZ sensor motes MPR2400. [Online]. Available: http://www.willow.co.uk/html/mpr2400-_micaz_zigbee.html

[4] Ns-2 network simulator, version 2.35. [Online]. Available: http://www.isi.edu/nsnam/ns/

[5] "21 ideas for the 21st century," *Business Week*, pp. 78–167, August 30 1999.

[6] "10 emerging technologies that will change the world," *MIT Technology Review*, vol. 106, no. 1, pp. 33–49, February 2003.

[7] (2013) NS-2 Network Simulator, version 2.35. [Online]. Available: http://www.isi.edu/nsnam/ns/

[8] R. Abedi, N. Aslam, and S. Ghani, "Fault Tolerance Analysis of Heterogeneous Wireless Sensor Network," in *proceedings of 24th Canadian Conference on Electrical and Computer Engineering*, 2011, pp. 175–179.

[9] F. Ahdi, V. Srinivasan, and K.-C. Chua, "Topology control for delay sensitive applications in wireless sensor networks," *Mobile Networks and Applications*, vol. 12, no. 5, pp. 406–421, 2007.

[10] Ö. Akan and I. Akyildiz, "Event-to-sink reliable transport in wireless sensor networks," *IEEE Transactions on Networking*, vol. 13, no. 5, pp. 1003–1016, 2005.

TH-1251_11610108

[11] A. Aksu, P. Krishnamurthy, D. Tipper, and O. Ercetin, "On security and reliability using cooperative transmissions in sensor networks," *Mobile Networks and Applications*, vol. 17, no. 4, pp. 526–542, 2012.

[12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Journal of Computer Network*, vol. 38, pp. 393–422, 2002.

[13] M. Albano, S. Chessa, and R. Di Pietro, "Information assurance in critical infrastructures via wireless sensor networks," in *proceedings of the 4th International Conference on Information Assurance and Security*, 2008, pp. 305–310.

[14] C. Alippi and G. Vanini, "Wireless sensor networks and radio localization: a metrological analysis of the MICA2 received signal strength indicator," in *proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, 2004, pp. 579–580.

[15] H. Alwan and A. Agarwal, "A survey on fault tolerant routing techniques in wireless sensor networks," in *proceedings of the 3rd International Conference on Sensor Technologies and Applications*, 2009, pp. 366–371.

[16] H. M. Ammari and S. K. Das, "Fault tolerance measures for large-scale wireless sensor networks," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 1, pp. 2:1–2:28, 2009.

[17] ——, "Fast track article: Scheduling protocols for homogeneous and heterogeneous k-covered wireless sensor networks," *Pervasive Mobile Computing*, vol. 7, no. 1, pp. 79–97, 2011.

[18] H. Attiya and J. Welch, *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Wiley-IEEE, 2004.

[19] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks: A survey," *ACM Transactions on Sensor Networks*, vol. 8, no. 4, pp. 34:1–34:33, 2012.

[20] A. Bari, A. Jaekel, and S. Bandyopadhyay, "Energy aware fault tolerant routing in two-tiered sensor networks," in *proceedings of the 12th International Conference on Distributed Computing and Networking*, 2011, pp. 293–302.

TH-1251_11610108

[21] A. Bari, A. Jaekel, J. Jiang, and Y. Xu, "Design of fault tolerant wireless sensor networks satisfying survivability and lifetime requirements," *Computer Communications*, vol. 35, no. 3, pp. 320–333, 2012.

[22] I. Bekmezci, *Wireless Sensor Networks: A Military Monitoring Application.* VDM Verlag, 2009.

[23] W. Bin, L. Wenxin, and L. Liu, "A survey of energy conservation, routing and coverage in wireless sensor networks," in *proceedings of the 7th International Conference on Active Media Technology*, 2011, pp. 59–70.

[24] Z. Bojkovic and B. Bakmaz, "A survey on wireless sensor networks deployment," *WSEAS Transactions on Communications*, vol. 7, no. 12, pp. 1172–1181, 2008.

[25] C. Boulinier, A. K. Datta, L. L. Larmore, and F. Petit, "Space efficient and time optimal distributed bfs tree construction," *Information Processing Letters*, vol. 108, pp. 273–278, 2008.

[26] E. Bulut and I. Korpeoglu, "Sleep scheduling with expected common coverage in wireless sensor networks," *Wireless Networks*, vol. 17, no. 1, pp. 19–40, 2011.

[27] C. Buratti, A. Conti, D. Dardari, and R. Verdone, "An overview on wireless sensor networks technology and evolution," *Sensors*, vol. 9, no. 9, pp. 6869–6896, 2009.

[28] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: ultra-low power data gathering in sensor networks," in *proceedings of the 6th International Conference on Information Processing in Sensor Networks*, 2007, pp. 450–459.

[29] G. Campobello, A. Leonardi, and S. Palazzo, "Improving energy saving and reliability in wireless sensor networks using a simple CRT-based packet-forwarding solution," *IEEE Transactions on Networking*, vol. 20, no. 1, pp. 191–205, 2012.

[30] I. Cardei and M. Cardei, "Energy-efficient connected-coverage in wireless sensor networks," *International Journal of Sensor Networks*, vol. 3, no. 3, pp. 201–210, 2008.

[31] M. Cardei and J. Wu, "Energy-efficient coverage problems in wireless ad-hoc sensor networks," *IEEE Computer communications*, vol. 29, no. 4, pp. 413–420, 2006.

[32] J. Carle and D. Simplot-Ryl, "Energy-efficient area monitoring for sensor networks," *IEEE Computer*, vol. 37, no. 2, pp. 40–46, 2004.

TH-1251_11610108

[33] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 298–313, 2003.

[34] C. Ceken, "An energy efficient and delay sensitive centralized MAC protocol for wireless sensor networks," *Computer Standards and Interfaces*, vol. 30, no. 1-2, pp. 20–31, 2008.

[35] M. Ceriotti, M. Corra, L. D'Orazio, R. Doriguzzi, D. Facchin, S. Guna, G. Jesi, R. Lo Cigno, L. Mottola, A. Murphy, M. Pescalli, G. Picco, D. Pregnolato, and C. Torghele, "Is there light at the ends of the tunnel? wireless sensor networks for adaptive lighting in road tunnels," in *proceedings of the 10th International Conference on Information Processing in Sensor Networks*, 2011, pp. 187–198.

[36] Y. Challal, A. Ouadjaout, N. Lasla, M. Bagaa, and A. Hadjidj, "Secure and efficient disjoint multipath construction for fault tolerant routing in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1380–1397, 2011.

[37] K. Y. Chan and T. Dillon, "On-road sensor configuration design for traffic flow prediction using fuzzy neural networks and taguchi method," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 1, pp. 50–59, 2013.

[38] A. Chen, T. H. Lai, and D. Xuan, "Measuring and guaranteeing quality of barrier-coverage for general belts with wireless sensors," *ACM Transactions on Sensor Networks*, vol. 6, no. 1, pp. 2:1–2:31, 2010.

[39] L.-W. Chen, Y.-H. Peng, and Y.-C. Tseng, "An infrastructure-less framework for preventing rear-end collisions by vehicular sensor networks," *IEEE Communications Letters*, vol. 15, no. 3, pp. 358–360, 2011.

[40] S. Chen, M. Huang, S. Tang, and Y. Wang, "Capacity of data collection in arbitrary wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 1, pp. 52–60, 2012.

[41] S. Chen, S. Tang, M. Huang, and Y. Wang, "Capacity of data collection in arbitrary wireless sensor networks," in *proceedings of the 29th IEEE Computer and Communications*, 2010, pp. 1–5.

TH-1251_11610108

[42] T.-S. Chen, H.-W. Tsai, and C.-P. Chu, "Gathering-load-balanced tree protocol for wireless sensor networks," in *proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 2, 2006, pp. 8–13.

[43] ——, "Adjustable convergecast tree protocol for wireless sensor networks," *Computer Communications*, vol. 33, pp. 559–570, 2010.

[44] W. Chen, L. Chen, Z. long Chen, and S. liang Tu, "WITS: A wireless sensor network for intelligent transportation system," in *proceedings of the 1st International Multi-Symposium of Computer and Computational Sciences*, 2006, pp. 635–641.

[45] J. Cheng, Z. Shang, H. Cheng, H. Wang, and J. X. Yu, "K-Reach: Who is in your small world," in *proceedings of the 38th International Conference on Very Large Data Bases*, vol. 5, no. 11, 2012, pp. 1292–1303.

[46] S.-Y. Cheung and P. Varaiya, "Traffic surveillance by wireless sensor networks: Final report," University of California, Berkeley, Tech. Rep. UCB-ITS-PRR-2007-4, 2007.

[47] A. Chinchuluun and P. M. Pardalos, "A survey of recent developments in multi-objective optimization," *Annals of Operations Research*, vol. 154, no. 1, pp. 29–50, 2007.

[48] W. Choi and S. K. Das, "Coverage-adaptive random sensor scheduling for application-aware data gathering in wireless sensor networks," *Computer Communications*, vol. 29, no. 17, pp. 3467–3482, 2006.

[49] W. Choi, G. Ghidini, and S. K. Das, "A novel framework for energy-efficient data gathering with random coverage in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 8, no. 4, pp. 36:1–36:30, 2012.

[50] C.-Y. Chong and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges," *proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[51] S. Coleri, S. Y. Cheung, and P. Varaiya, "Sensor networks for monitoring traffic," in *proceedings of the Allerton Conference on Communication, Control and Computing*, 2004.

[52] A. Coman, M. A. Nascimento, and J. Sander, "Exploiting redundancy in sensor networks for energy efficient processing of spatiotemporal region queries," in *Proceedings of the 14th ACM international conference on Information and knowledge management*, 2005, pp. 187–194.

TH-1251_11610108

[53] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2001.

[54] B. Cărbunar, A. Grama, J. Vitek, and O. Cărbunar, "Redundancy and coverage detection in sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 1, pp. 94–128, 2006.

[55] B. Deb, S. Bhatnagar, and B. Nath, "ReInForM: Reliable information forwarding using multiple paths in sensor networks," in *proceedings of the 28th Annual IEEE International Conference on Local Computer Networks*, 2003, pp. 406–415.

[56] S. S. Dhillon and K. Chakrabarty, *Sensor placement for effective coverage and surveillance in distributed sensor networks*, 2003, vol. 3.

[57] O. Durmaz Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 1, pp. 86–99, 2012.

[58] D. England, B. Veeravalli, and J. B. Weissman, "A robust spanning tree topology for data collection and dissemination in distributed environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, pp. 608–620, 2007.

[59] S. C. Ergen and P. Varaiya, "Pedamacs: Power efficient and delay aware medium access protocol for sensor networks," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 920–930, 2006.

[60] T. Erlebach, T. Grant, and F. Kammer, "Maximising lifetime for fault-tolerant target coverage in sensor networks," in *proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures*, 2011, pp. 187–196.

[61] Z. Fei and M. Yang, "A proactive tree recovery mechanism for resilient overlay multicast," *IEEE Transactions on Networking*, vol. 15, pp. 173–186, 2007.

[62] E. Felemban, C.-G. Lee, and E. Ekici, "MMSPEED: Multipath multi-SPEED protocol for QoS guarantee of reliability and timeliness in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 738–754, 2006.

[63] M. R. Fellows, D. Hermelin, F. Rosamond, and S. Vialette, "On the parameterized complexity of multiple-interval graph problems," *Theoretical Computer Science*, vol. 410, no. 1, pp. 53–61, 2009.

TH-1251_11610108

[64] P. Gajbhiye and A. Mahajan, "A survey of architecture and node deployment in wireless sensor network," in *proceedings of the 1st International Conference on the Applications of Digital Information and Web Technologies*, 2008, pp. 426–430.

[65] N. Gallego, A. Mocholi, M. Menendez, and R. Barrales, "Traffic monitoring: Improving road safety using a laser scanner sensor," in *proceedings of the Electronics, Robotics and Automotive Mechanics Conference*, 2009, pp. 281–286.

[66] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 4, pp. 11–25, 2001.

[67] M. Garland and P. S. Heckbert, "Fast polygonal approximation of terrains and height fields," CS Dept., Carnegie Mellon University, Tech. Rep. CMU-CS-95-181, 1995.

[68] A. Ghosh and S. K. Das, "Review: Coverage and connectivity issues in wireless sensor networks: A survey," *Pervasive Mobile Computing*, vol. 4, no. 3, pp. 303–334, 2008.

[69] S. Ghosh and S. Rao, "Sensor network design for smart highways," in *proceedings of the 5th annual IEEE International Conference on Automation Science and Engineering*, 2009, pp. 353–360.

[70] J. Girao, D. Westhoff, and M. Schneider, "CDA: Concealed data aggregation for reverse multicast traffic in wireless sensor networks," in *proceedings of the IEEE International Conference on Communications*, vol. 5, 2005, pp. 3044–3049.

[71] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, 2009, pp. 1–14.

[72] A. V. Goldberg and T. Radzik, "A heuristic improvement of the bellman-ford algorithm," *Applied Mathematics Letters*, vol. 6, no. 3, pp. 3–6, 1993.

[73] B. gu Lee and J. han Kim, "Algorithm for finding the moving direction of a vehicle using magnetic sensor," in *proceedings of the IEEE Symposium on Computational Intelligence in Control and Automation*, 2011, pp. 74–79.

TH-1251_11610108

[74] H. Gupta, V. Navda, S. Das, and V. Chowdhary, "Efficient gathering of correlated data in sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 1, pp. 4:1–4:31, 2008.

[75] N. Haddadou, A. Rachedi, and Y. Ghamri-Doudane, "Advanced diffusion of classified data in vehicular sensor networks," in *proceedings of the 7th International Wireless Communications and Mobile Computing Conference*, 2011, pp. 777–782.

[76] S. Halder, A. Ghosal, and S. D. Bit, "A pre-determined node deployment strategy to prolong network lifetime in wireless sensor network," *Computer Communications*, vol. 34, no. 11, pp. 1294–1306, 2011.

[77] S. Hariharan and N. B. Shroff, "On optimal energy efficient convergecasting in unreliable sensor networks with applications to target tracking," in *proceedings of the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2011, pp. 24:1–24:10.

[78] T. Herault, P. Lemarinier, O. Peres, L. Pilard, and J. Beauquier, "Self-stabilizing spanning tree algorithm for large scale systems," in *proceedings of the 8th International Conference on Stabilization, Safety, and Security of Distributed Systems*, 2006, pp. 574–575.

[79] C.-C. Huang, J.-L. Huang, J.-A. Yan, and L.-Y. Yeh, "An in-network approximate data gathering algorithm exploiting spatial correlation in wireless sensor networks," in *proceedings of the 27th Annual ACM Symposium on Applied Computing*, 2012, pp. 550–555.

[80] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 519–528, 2005.

[81] A. Iranli, M. Maleki, and M. Pedram, "Energy efficient strategies for deployment of a two-level wireless sensor network," in *proceedings of the International Symposium on Low Power Electronics and Design*, 2005, pp. 233–238.

[82] R. Iyengar, K. Kar, and S. Banerjee, "Low-coordination topologies for redundancy in sensor networks," in *proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2005, pp. 332–342.

TH-1251_11610108

[83] ——, "Low-coordination topologies for redundancy in sensor networks," in *proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2005, pp. 332–342.

[84] N. Jaggi and A. A. Abouzeid, "Energy-efficient connected coverage in wireless sensor networks," in *proceedings of the 4th Asian International Mobile Computing Conference*, 2006, pp. 77–86.

[85] I. Jawhar, N. Mohamed, and K. Shuaib, "A framework for pipeline infrastructure monitoring using wireless sensor networks," in *proceedings of the Wireless Telecommunications Symposium*. IEEE, 2007, pp. 1–7.

[86] J. Jeong, S. Guo, T. He, and D. Du, "APL: Autonomous passive localization for wireless sensors deployed in road networks," in *proceedings of The 27th IEEE Conference on Computer Communications*, 2008, pp. 583–591.

[87] ——, "Autonomous passive localization algorithm for road sensor networks," *IEEE Transactions on Computers*, vol. 60, no. 11, pp. 1622–1637, 2011.

[88] C. Johnen, "Memory efficient, self-stabilizing algorithm to construct BFS spanning trees," in *proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, 1997.

[89] P. Jurcik, A. Koubâa, R. Severino, M. Alves, and E. Tovar, "Dimensioning and worst-case analysis of cluster-tree sensor networks," *ACM Transactions on Sensor Networks*, vol. 7, no. 2, pp. 14:1–14:47, 2010.

[90] K. Kalpakis, "Everywhere sparse approximately optimal minimum energy data gathering and aggregation in sensor networks," *ACM Transactions on Sensor Networks*, vol. 7, no. 1, pp. 9:1–9:26, 2010.

[91] A. R. M. Kamal, C. Bleakley, and S. Dobson, "Packet-level attestation (PLA): A framework for in-network sensor data reliability," *ACM Transactions on Sensor Networks*, vol. 9, no. 2, pp. 19:1–19:28, 2013.

[92] K. Kar, S. Banerjee *et al.*, "Node placement for connected coverage in sensor networks," in *proceedings of the Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

TH-1251_11610108

[93] A. Kesselman and D. Kowalski, "Fast distributed algorithm for convergecast in ad hoc geometric radio networks," in *proceedings of IEEE Global Telecommunications Conference*, 2003, pp. 3523–3530.

[94] R. A. M. Khan and H. Karl, "Multihop performance of cooperative preamble sampling MAC(CPS-MAC) in wireless sensor networks," in *proceedings of the 10th International Conference on Ad-Hoc, Mobile, and Wireless Networks*, 2011, pp. 145–149.

[95] H. Kim, Y. Seok, N. Choi, Y. Choi, and T. Kwon, "Optimal multi-sink positioning and energy-efficient routing in wireless sensor networks," in *Information Networking. Convergence in Broadband and Mobile Networking*. Springer, 2005, pp. 264–274.

[96] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *proceedings of the 6th International Symposium on Information Processing in Sensor Networks*. IEEE, 2007, pp. 254–263.

[97] A. N. Knaian, "A wireless sensor network for smart roadbeds and intelligent transportation systems," Massachusetts Institute of Technology, USA, Tech. Rep., 2000.

[98] F. Kong and J. Tan, "A collaboration-based hybrid vehicular sensor network architecture," in *proceedings of the International Conference on Information and Automation*, 2008, pp. 584–589.

[99] R. H. Kori, A. S. Angadi, M. K. Hiremath, and S. M. Iddalagi, "Efficient power utilization of wireless sensor networks: A survey," in *proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing*, 2009, pp. 571–575.

[100] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *proceedings of the 22nd International Conference on Distributed Computing Systems Workshops*, 2002, pp. 575–578.

[101] S. Kumar, T. H. Lai, and A. Arora, "Barrier coverage with wireless sensors," in *proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, 2005, pp. 284–298.

TH-1251_11610108

[102] S. Kumar and S. Chauhan, "A survey on scheduling algorithms for wireless sensor networks," *International Journal of Computer Applications*, vol. 20, no. 5, pp. 7–13, 2011.

[103] S. Kwon, J.-G. Kim, and C. Kim, "An efficient tree structure for delay sensitive data gathering in wireless sensor networks," in *proceedings of the 22nd International Conference on Advanced Information Networking and Applications-Workshops*, 2008, pp. 738–743.

[104] G. Latsoudas and N. D. Sidiropoulos, "A fast and effective multidimensional scaling approach for node localization in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 55, no. 10, pp. 5121–5127, 2007.

[105] S. H. Lee, S. Lee, H. Song, and H. S. Lee, "Wireless sensor network design for tactical military applications: Remote large-scale environments," in *proceedings of the 28th IEEE Conference on Military Communications*, 2009, pp. 911–917.

[106] W. Li, E. Chan, M. Hamdi, S. Lu, and D. Chen, "Communication cost minimization in wireless sensor and actor networks for road surveillance," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 2, pp. 618–631, 2011.

[107] X.-Y. Li, Y. Wang, and Y. Wang, "Complexity of data collection, aggregation, and selection for wireless sensor networks," *IEEE Transactions on Computers*, vol. 60, pp. 386–399, 2011.

[108] W. Liang and Y. Liu, "Online data gathering for maximizing network lifetime in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 1, pp. 2–11, 2007.

[109] W.-H. Liao and M.-S. Lin, "An energy-efficient sensor deployment scheme for wireless sensor networks," in *proceedings of the IEEE International Conference on Vehicular Electronics and Safety*, 2011, pp. 76–81.

[110] P. G. Liaskovitis and C. Schurgers, "Leveraging redundancy in sampling-interpolation applications for sensor networks: A spectral approach," *ACM Transactions on Sensor Networks*, vol. 7, no. 2, pp. 12:1–12:28, 2010.

[111] K.-W. Lim, W. S. Jung, and Y.-B. Ko, "Multi-hop data dissemination with replicas in vehicular sensor networks," in *proceedings of the IEEE Vehicular Technology Conference*, 2008, pp. 3062–3066.

TH-1251_11610108

[112] S. Lindsey and C. Raghavendra, "PEGASIS: Power-efficient gathering in sensor information systems," in *proceedings of the IEEE Aerospace Conference*, vol. 3, 2002, pp. 1125–1130.

[113] S. Lindsey, C. Raghavendra, and K. M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 924–935, 2002.

[114] H. Liu, Z.-L. Zhang, J. Srivastava, and V. Firoiu, "PWave: A multi-source multi-sink anycast routing framework for wireless sensor networks," in *Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*. Springer, 2007, pp. 179–190.

[115] X. Liu, Q. Wang, W. He, M. Caccamo, and L. Sha, "Optimal real-time sampling rate assignment for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 2, pp. 263–295, 2006.

[116] K. Lorincz, D. J. Malan, T. R. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton, "Sensor networks for emergency response: challenges and opportunities," *Pervasive Computing, IEEE*, vol. 3, no. 4, pp. 16–23, 2004.

[117] W. Lou and Y. Kwon, "H-SPREAD: a hybrid multipath scheme for secure and reliable data collection in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 4, pp. 1320–1330, 2006.

[118] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for tree-based data gathering in sensor networks: Research articles," *Wireless Communications and Mobile Computing*, vol. 7, no. 7, pp. 863–875, 2007.

[119] K. Lu, L. Huang, Y. Wan, and H. Xu, "Energy-efficient data gathering in large wireless sensor networks," in *proceedings of the 2nd International Conference on Embedded Software and Systems*, 2005, pp. 327–331.

[120] X. Lu, S. Chen, W. Chen, and D. Li, "Sweep coverage with mobile sensors on two-way road," in *Advances in Wireless Sensor Networks*, 2013, pp. 335–345.

TH-1251_11610108

[121] C. Luo, F. Wu, J. Sun, and C. W. Chen, "Compressive data gathering for large-scale wireless sensor networks," in *proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 2009, pp. 145–156.

[122] H. Luo, Z. Zhang, and Y. Liu, "ReCoDa: reliable forwarding of correlated data in sensor networks with low latency," in *proceedings of the International Conference on Wireless Communications and Mobile Computing*, 2007, pp. 278–283.

[123] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.

[124] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002, pp. 88–97.

[125] Y. Manolopoulos, D. Katsaros, and A. Papadimitriou, "Topology control algorithms for wireless sensor networks: a critical survey," in *proceedings of the 11th International Conference on Computer Systems and Technologies*, 2010, pp. 1–10.

[126] X. Mao, X. Xu, S. Tang, and X.-Y. Li, "Providing and finding k-road-coverage efficiently in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 12, no. 12, pp. 1053–1065, 2012.

[127] S. Megerian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Worst and best-case coverage in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 1, pp. 84–92, 2005.

[128] D. Messina, M. Ortolani, and G. L. Re, "Reliable data gathering in tree-based IEEE 802.15.4 wireless sensor networks," in *proceedings of the 4th Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, 2007, pp. 1–7.

[129] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: the backpressure collection protocol," in *proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010, pp. 279–290.

[130] M. Moh, M. Dumont, and T.-S. Moh, "A survey on fault tolerant routing techniques in wireless sensor networks," in *proceedings of the 5th IEEE International Symposium on Signal Processing and Information Technology*, 2005, pp. 170–175.

TH-1251_11610108

[131] A. Momen, P. Azmi, F. Bazazan, and A. Hassani, "Optimised random structure vehicular sensor network," *IET Intelligent Transport Systems*, vol. 5, no. 1, pp. 90–99, 2011.

[132] H. Moser and B. Thallner, "Construction of a fault-tolerant wireless communication topology using distributed agreement," in *proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, 2006, pp. 35–44.

[133] L. Mottola and G. Picco, "MUSTER: Adaptive energy-aware multisink routing in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 12, pp. 1694–1709, 2011.

[134] A. Naderi and S. M. Mazinani, "A differentiated tree-based protocol along with re-routing policy in wireless sensor networks," in *proceedings of the International Symposium on Computer, Consumer and Control*, 2012, pp. 495–498.

[135] H. R. Oh and H. Song, "Energy efficient MAC protocol for delay-sensitive data transmission over wireless sensor network," *Wireless Communications and Mobile Computing*, vol. 12, no. 9, pp. 755–766, 2012.

[136] F. Oldewurtel and P. Mä, "Efficiency analysis and derivation of enhanced deployment models for sensor networks," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 9, no. 1, pp. 25–41, 2012.

[137] K. Onodera and T. Miyazaki, "An autonomous algorithm for construction of energy-conscious communication tree in wireless sensor networks," in *proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops*, 2008, pp. 898–903.

[138] P. Ostovari, M. Dehghan, and J. Wu, "Connected point coverage in wireless sensor networks using robust spanning trees," in *proceedings of the 31st International Conference on Distributed Computing Systems Workshops*, 2011, pp. 287–293.

[139] G. Palubinskas, F. Kurz, and P. Reinartz, "Detection of traffic congestion in optical remote sensing imagery," in *proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, vol. 2, 2008, pp. 426–429.

[140] S. Pantavungkour and R. Shibasaki, "Three line scanner, modern airborne sensor and algorithm of vehicle detection along mega-city street," in *proceedings of the*

TH-1251_11610108

2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas, 2003, pp. 263–267.

[141] A. Pascale, M. Nicoli, F. Deflorio, B. Dalla Chiara, and U. Spagnolini, "Wireless sensor networks for traffic management and road safety," *IET Intelligent Transport Systems*, vol. 6, no. 1, pp. 67–77, 2012.

[142] M. Patan and D. UcińSki, "Configuring a sensor network for fault detection in distributed parameter systems," *International Journal of Applied Mathematics and Computer Science*, vol. 18, no. 4, pp. 513–524, 2008.

[143] C. Pelczar, K. Sung, J. Kim, and B. Jang, "Vehicle speed measurement using wireless sensor nodes," in *proceedings of the IEEE International Conference on Vehicular Electronics and Safety*, 2008, pp. 195–198.

[144] W. Y. Poe and J. B. Schmitt, "Sink placement without location information in large-scale wireless sensor networks," in *proceedings of the Asian Internet Engineering Conference*, 2009, pp. 69–76.

[145] D. Puccinelli and M. Haenggi, "Reliable data delivery in large-scale low-power sensor networks," *ACM Transactions on Sensor Networks*, vol. 6, no. 4, pp. 28:1–28:41, 2010.

[146] "Qualnet 5.0.1 network simulator." [Online]. Available: http://www.scalable-networks.com/products/qualnet/

[147] M. Radi, B. Dezfouli, K. Bakar, and M. Lee, "Multipath routing in wireless sensor networks: Survey and research challenges," *Sensors*, vol. 12, no. 1, pp. 650–685, 2012.

[148] M. Radi, B. Dezfouli, K. A. Bakar, and M. Lee, "Multipath routing in wireless sensor networks: Survey and research challenges," *Sensors*, vol. 12, no. 1, pp. 650–685, 2012.

[149] V. Rai and R. N. Mahapatra, "Lifetime modeling of a sensor network," in *proceedings of the conference on Design, Automation and Test in Europe - Volume 1*, 2005, pp. 202–203.

[150] R. Roman, C. Alcaraz, and J. Lopez, "The role of wireless sensor networks in the area of critical information infrastructure protection," *Information Security Tech. Report*, vol. 12, no. 1, pp. 24–31, 2007.

TH-1251_11610108

[151] S. Rothery, W. Hu, and P. Corke, "An empirical study of data collection protocols for wireless sensor networks," in *proceedings of the Workshop on Real-World Wireless Sensor Networks*, 2008, pp. 16–20.

[152] A. Sahoo and S. Chilukuri, "DGRAM: A delay guaranteed routing and MAC protocol for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 10, pp. 1407–1423, 2010.

[153] J. A. Sanchez, R. Marin-Perez, and P. M. Ruiz, "Beacon-less geographic multicast routing in a real-world wireless sensor network testbed," *Wireless Networks*, vol. 18, no. 5, pp. 565–578, 2012.

[154] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Computing Surveys*, vol. 37, no. 2, pp. 164–194, 2005.

[155] S. S. Satapathy and N. Sarma, "TREEPSI: tree based energy efficient protocol for sensor information," in *proceedings of the IFIP International Conference on Wireless and Optical Communications Networks*, 2006.

[156] J. B. Schmitt and U. Roedig, "Sensor network calculus - a framework for worst case analysis," in *proceedings of the First IEEE international conference on Distributed Computing in Sensor Systems*, 2005, pp. 141–154.

[157] A. Seetharam, A. Bhattacharyya, M. K. Naskar, and A. Mukherjee, "Estimation of node density for an energy efficient deployment scheme in wireless sensor network," in *proceedings of the 3rd International Conference on Communication System software and Middleware*, 2008, pp. 95–98.

[158] C. Sergiou and V. Vassiliou, "Source-based routing trees for efficient congestion control in wireless sensor networks," in *proceedings of the IEEE 8th International Conference on Distributed Computing in Sensor Systems*, 2012, pp. 378–383.

[159] F. K. Shaikh, A. Khelil, and N. Suri, "On modeling the reliability of data transport in wireless sensor networks," in *proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, 2007, pp. 395–402.

[160] E. Shakshuki, X. Xing, and H. Zhang, "Agent-based fault detection mechanism in wireless sensor networks," in *proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2007, pp. 31–34.

TH-1251_11610108

[161] F. Shen, M.-T. Sun, C. Liu, and A. Salazar, "Coverage-aware sleep scheduling for cluster-based sensor networks," in *proceedings of the 2009 IEEE Conference on Wireless Communications and Networking Conference*, 2009, pp. 2480–2485.

[162] W. Shen and Q. Wu, "Exploring redundancy in sensor deployment to maximize network lifetime and coverage," in *proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2011, pp. 557–565.

[163] Z. Shen, Y. Chang, X. Zhang, and C. Cui, "An efficient topology maintenance algorithm based on shortest path tree for wireless sensor networks," in *proceedings of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2005, pp. 288–292.

[164] L. Shu, Y. Zhang, Z. Zhou, M. Hauswirth, Z. Yu, and G. Hynes, "Transmitting and gathering streaming data in wireless multimedia sensor networks within expected network lifetime," *Mobile Networks and Applications*, vol. 13, no. 3-4, pp. 306–322, 2008.

[165] L. Shu, Z. zhou, A. Aguilar, and M. Hauswirth, "Stream data gathering in wireless sensor networks within expected lifetime," in *proceedings of the 3rd International Conference on Mobile Multimedia Communications*, 2007, pp. 62:1–62:6.

[166] P. Sthapit, Y. T. Park, and J.-Y. Pyun, "Medium reservation based MAC for delay-sensitive wireless sensor network," in *proceedings of the 2009 9th IEEE International Conference on Computer and Information Technology*, vol. 02, 2009, pp. 122–127.

[167] C. Suh, D. M. Shrestha, and Y.-B. Ko, "An energy-efficient MAC protocol for delay-sensitive wireless sensor networks," in *proceedings of the International Conference on Emerging Directions in Embedded and Ubiquitous Computing*, 2006, pp. 445–454.

[168] K. Sung, J. J. Yoo, and D. Kim, "Collision warning system on a curved road using wireless sensor networks," in *proceedings of the IEEE 66th Vehicular Technology Conference*, 2007, pp. 1942–1946.

[169] H. Ö. Tan and I. Körpeolu, "Power efficient data gathering and aggregation in wireless sensor networks," *ACM SIGMOD Record*, vol. 32, no. 4, pp. 66–71, 2003.

TH-1251_11610108

[170] K. K. Tan, S. N. Huang, Y. Zhang, and T. H. Lee, "Distributed fault detection in industrial system based on sensor wireless network," *Computer Standards and Interfaces*, vol. 31, no. 3, pp. 573–578, 2009.

[171] A. S. Tanenbaum, C. Gamage, and B. Crispo, "Taking sensor networks from the lab to the jungle," *IEEE Computer*, vol. 39, no. 8, pp. 98–100, 2006.

[172] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 2, pp. 28–36, 2002.

[173] J. Troya and A. Vallecillo, "A domain specific visual language for modeling power-aware reliability in wireless sensor networks," in *proceedings of the 4th International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages*, 2012, pp. 3:1–3:6.

[174] M. Tubaishat, Q. Qi, Y. Shang, and H. Shi, "Wireless sensor-based traffic light control," in *proceedings of the 5th IEEE Consumer Communications and Networking Conference*, 2008, pp. 702–706.

[175] V. Turau and C. Weyer, "Fault tolerance in wireless sensor networks through self-stabilisation," *Intenational Journal of Communication Networks and Distributed Systems*, vol. 2, no. 1, pp. 78–98, 2009.

[176] T. Uchiyama, K. Mohri, H. Itho, K. Nakashima, J. Ohuchi, and Y. Sudo, "Car traffic monitoring system using MI sensor built-in disk set on the road," *IEEE Transactions on Magnetics*, vol. 36, no. 5, pp. 3670–3672, 2000.

[177] S. Upadhayayula, V. Annamalai, and S. Gupta, "A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks," in *proceedings of IEEE Global Telecommunications Conference*, vol. 6, 2003, pp. 3525–3530.

[178] V.Annamalai, S. Gupta, and L.Schwiebert, "On tree-based convergecasting in wireless sensor networks," in *proceedings of IEEE Wireless Communication and Networking Conference*, 2003, pp. 1942–1947.

[179] J. Wang and N. Zhong, "Efficient point coverage in wireless sensor networks," *Journal of Combinatorial Optimization*, vol. 11, no. 3, pp. 291–304, 2006.

TH-1251_11610108

[180] J. Wang, Y. Liu, and S. K. Das, "Energy-efficient data gathering in wireless sensor networks with asynchronous sampling," *ACM Transactions on Sensor Networks*, vol. 6, no. 3, pp. 22:1–22:37, 2010.

[181] L. Wang and Y. Xiao, "A survey of energy-efficient scheduling mechanisms in sensor networks," *Mobile Networks and Applications*, vol. 11, no. 5, pp. 723–740, 2006.

[182] Q. Wang, M. Hempstead, and W. Yang, "A realistic power consumption model for wireless sensor network devices," in *proceedings of Sensor and Ad Hoc Communications and Networks*, vol. 1, 2006, pp. 286–295.

[183] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, 2003, pp. 28–39.

[184] E. U. Warriach, K. Tei, T. A. Nguyen, and M. Aiello, "Fault detection in wireless sensor networks: a hybrid approach," in *proceedings of the 11th International Conference on Information Processing in Sensor Networks*, 2012, pp. 87–88.

[185] P. Wightman and M. Labrador, "A3Cov: A new topology construction protocol for connected area coverage in WSN," in *proceedings of IEEE Wireless Communications and Networking Conference*, 2011, pp. 522–527.

[186] Y. Wu, Z. Mao, S. Fahmy, and N. B. Shroff, "Constructing maximum-lifetime data gathering forests in sensor networks," *IEEE Transactions on Networking*, vol. 18, no. 5, pp. 1571–1584, 2010.

[187] W. Xie, X. Zhang, and H. Chen, "Wireless sensor network topology used for road traffic," in *proceedings of the IET Conference on Wireless, Mobile and Sensor Networks*, 2007, pp. 285–288.

[188] C. Xu, L. Cao, Z. Bao, S. Zhu, G. Zhang, and H. Zhou, "Up-down links dualpath greedy routing protocol for wireless sensor networks," *Wireless Personal Communications*, vol. 64, no. 2, pp. 323–345, 2012.

[189] X. Xu and S. Sahni, "Approximation algorithms for sensor deployment," *IEEE Transactions on Computers*, vol. 56, no. 12, pp. 1681–1695, 2007.

[190] T. Yan, Y. Gu, T. He, and J. A. Stankovic, "Design and optimization of distributed sensing coverage in wireless sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 7, no. 3, pp. 33:1–33:40, 2008.

TH-1251_11610108

[191] S. Yang, F. Dai, M. Cardei, J. Wu, and F. Patterson, "On connected multiple point coverage in wireless sensor networks," *International Journal of Wireless Information Networks*, vol. 13, no. 4, pp. 289–301, 2006.

[192] F. Ye, G. Zhong, S. Lu, and L. Zhang, "Gradient broadcast: a robust data delivery protocol for large scale sensor networks," *Wireless Networks*, vol. 11, no. 3, pp. 285–298, 2005.

[193] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[194] X.-S. Yi, P.-J. Jiang, X.-W. Wang, and S.-C. Zhang, "Survey of energy-saving protocols in wireless sensor networks," in *proceedings of the 1st International Conference on Robot, Vision and Signal Processing*, 2011, pp. 208–211.

[195] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[196] S. Yoon and C. Shahabi, "The clustered aggregation (CAG) technique leveraging spatial and temporal correlations in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 3, no. 1, 2007.

[197] R. Yu, Q. Chen, X. Wang, and S. K. Das, "Efficient data gathering in partially connected and delay-tolerant wireless sensor networks," in *proceedings of the 5th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, 2008, pp. 6:1–6:7.

[198] X. Yu, Y. Liu, Y. Zhu, W. Feng, L. Zhang, H. Rashvand, and V. O. K. Li, "Efficient sampling and compressive sensing for urban monitoring vehicular sensor networks," *IET Wireless Sensor Systems*, vol. 2, no. 3, pp. 214–221, 2012.

[199] Y. Yu., B. Krishnamachari, and V. Prasanna, "Energy-latency tradeoffs for data gathering in wireless sensor networks," in *proceedings of the 23rd IEEE International Conference on Computer Communications*, vol. 1, 2004.

[200] Z. Yun, X. Bai, D. Xuan, T. H. Lai, and W. Jia, "Optimal deployment patterns for full coverage and $k-$connectivity ($k <= 6$) wireless sensor networks," *IEEE Transactions on Networking*, vol. 18, no. 3, pp. 934–947, 2010.

TH-1251_11610108

[201] Y. Zeng, K. Xiang, and D. Li, "Applying behavior recognition in road detection using vehicle sensor networks," in *proceedings of the International Conference on Computing, Networking and Communications*, 2012, pp. 751–755.

[202] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 1, no. 1-2, pp. 89–124, 2005.

[203] L. Zhang, R. Wang, and L. Cui, "Real-time traffic monitoring with magnetic sensor networks," *Journal of Information Science and Engineering*, vol. 27, pp. 1473–1486, 2011.

[204] Q. Zhang, Z. Xie, W. Sun, and B. Shi, "Tree structure based data gathering for maximum lifetime in wireless sensor networks," in *proceedings of the 7th Asia-Pacific Web Conference on Web Technologies Research and Development*, 2005, pp. 513–522.

[205] T. Zheng, S. Radhakrishnan, and V. Sarangan, "Modeling and performance analysis of DMAC for wireless sensor networks," in *proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2011, pp. 119–128.

[206] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Models and solutions for radio irregularity in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 2, pp. 221–262, 2006.

[207] C. Zhu, C. Zheng, L. Shu, and G. Han, "A survey on coverage and connectivity issues in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 35, pp. 619–632, 2012.

[208] S. Zou, I. Nikolaidis, and J. J. Harms, "ENCAST: Energy-critical node aware spanning tree for sensor networks," in *proceedings of the 3rd Annual Communication Networks and Services Research Conference*, 2005, pp. 249–254.

[209] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Transactions in Embedded Computing Systems*, vol. 3, no. 1, pp. 61–91, 2004.

TH-1251_11610108

TH-1251_11610108

Publications Related to Thesis

Book Chapter

- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi, Sushanta Karmakar, "Sensory Data Gathering for Road-Traffic Monitoring: Energy Efficiency, Reliability and Fault-tolerance", in Modeling and Optimization in Science and Technology, Editor: Prof. Srikanta Patnaik, Springer (in Press).

Journal

- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "Convergecast Tree Management from Arbitrary Node Failure in Sensor Network", in Ad Hoc Networks, Elsevier, Volume 11, Issue 6, August 2013, Pages 1796-1819.

- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "ADCROSS: Adaptive Data Collection from Road Surveilling Sensors", in IEEE Transactions on Intelligent Transportation Systems (In Press).

- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "Topology Management Ensuring Reliability in Delay Sensitive Sensor Networks with Arbitrary Node Failures", in International Journal of Wireless Information Networks, Springer (Communicated).

- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "Impact of Redundant Sensor Deployments over Data Gathering Performance: A Model based Approach", in ACM Transactions on Sensor Networks (Communicated).

TH-1251_11610108

- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "Reliable Data Collection for Mission Critical Sensor Networks: A Node-Disjoint Multi-Path Multi-Sink Forwarding Approach with Inbuilt Fault Management", in Pervasive and Mobile Computing, Elsevier (Communicated).

Conference Proceedings

- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "RelBAS: Reliable Data Gathering from Border Area Sensors", in proceedings of the 18th IEEE Symposium on Computers and Communications (IEEE ISCC '13), July 7-10, Split, Croatia.
- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "Exploring Gradient in Sensor Deployment Pattern for Data Gathering with Sleep based Energy Saving", in proceedings of the 9th IEEE International Wireless Communications and Mobile Computing Conference (IEEE IWCMC 13), July 1-5, 2013, Cagliari, Sardinia, Italy.
- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "Energy-efficient Data Gathering for Road-side Sensor Networks ensuring Reliability and Fault-tolerance", in proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications (IEEE AINA '13), March 25-28, 2013, Barcelona, Spain.
- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "A Novel Crash-Tolerant Data Gathering in Wireless Sensor Networks", in proceedings of the 13th IEEE/IFIP Network Operations and Management Symposium (IEEE/IFIP NOMS '12), pp. 940 - 946, April 16-20, 2012, Maui, Hawaii, USA.
- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "A Tree-Based Local Repairing Approach for Increasing Lifetime of Query Driven WSN", in proceedings of the 2011 International Symposium on Pervasive Systems, Algorithms, and Networks (IEEE I-SPAN '11), pp. 475 - 482, August 24-26, 2011, Dalian, China.

TH-1251_11610108

Brief Biography of the Author

Suchetana Chakraborty was born in Uttarpara, Hooghly, West Bengal, India on 13th March, 1987. After completing her schooling in Uttarpara, Hooghly, she joined at the *Department of Computer Science and Engineering, Academy of Technology*, Hooghly, India (under *West Bengal University of Technology*, West Bengal, India), from where she has received the B.Tech degree in the year 2009. She completed her M.Tech. degree from the *Department of Computer Science and Engineering, Indian Institute of Technology Guwahati*, Assam, India in 2011, where she continued working towards the PhD degree under the supervision of Dr. Sushanta Karmakar. She has been awarded *TCS Research Fellowship* from TATA Consultancy Services, India and National Internet Exchange of India (NIXI) Fellowship. Her research interests include wireless sensor networks, fault-tolerant computing, distributed algorithms and bio-inspired computing. She has also worked in different fields like wireless mesh networks, network security, mobile middleware etc., in cooperation with other research groups at IIT Guwahati.

TH-1251_11610108

TH-1251_11610108

Other Publications of the Author

Journal

- **Suchetana Chakraborty**, Sandip Chakraborty, Sushanta Karmakar and Sukumar Nandi, "Dynamic Tree Switching for Distributed Message-Passing Applications", in Journal of Network and Systems Management, Springer (in Press).

- **Suchetana Chakraborty**, Ferdous Barbhuiya, Ankush Rai, Arijit Sur, Santosh Biswas and Sukumar Nandi, "Topology Adaptive Computation of Distributed IDS Set for Detecting Attacks on STP", in Journal of Information Assurance and Security, ISSN 1554-1010, Vol. 7, 2012.

Conference Proceedings

- Sandip Chakraborty, **Suchetana Chakraborty**, Sushanta Karmakar and Hridoy Sankar Dutta, "Hierarchical Topology Adaptation for Distributed Convergecast Applications", in proc. of the 29th ACM Symposium on Applied Computing, Dependable and Adaptive Distributed Systems (ACM SAC 2014), March 24 -28, 2014, Gyeongju, Korea.

- Sandip Chakraborty, Subhrendu Chattopadhyay, **Suchetana Chakraborty** and Sukumar Nandi, "Defending Concealedness in IEEE 802.11n", in proc. of the 6th International Conference on Communication Systems and Networks (IEEE/ACM COMSNETS 2014), January 07-09, 2014, Bangalore, India.

- Sandip Chakraborty, **Suchetana Chakraborty** and Sukumar Nandi, Beyond Conventional Routing Protocols: Opportunistic Path Selection for IEEE 802.11s Mesh Networks in proceedings of the 24th Annual IEEE International Symposium on

TH-1251_11610108

Personal, Indoor and Mobile Radio Communications (IEEE PIMRC '13), September 8-11, 2013, London, UK.

- **Suchetana Chakraborty**, Sandip Chakraborty and Sukumar Nandi, "Side Channel Attacks : Current Researches and Future Scopes", in proceedings of the National Workshop on Network Security, March 15-16, 2013, Tezpur, Assam, India.

- Suddhasil De, **Suchetana Chakraborty**, Sukumar Nandi and Diganta Goswami, "Supporting Tuple Space based Mobile Middleware over Unreliable Mobile Infrastructures : Design and Formal Specifications", in proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (IEEE ANTS '12), December 16-19, 2012, Bangalore, India.

- Suddhasil De, **Suchetana Chakraborty**, Diganta Goswami and Sukumar Nandi, "Formalization of Discovery and Communication Mechanisms of Tuple Space Based Mobile Middleware for Underlying Unreliable Infrastructure", in proceedings of the 2nd IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC '12), December 6-8, 2012, Solan, India.

- Suddhasil De, Diganta Goswami, Sukumar Nandi and **Suchetana Chakraborty**, "Formalization of a Fully-Decoupled Reactive Tuple Space model for Mobile Middleware", in proceedings of the 5th International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications (MOBILWARE '12), November 13-14, 2012, Berlin, Germany.

- Ankush Rai, Ferdous Ahmed Barbhuiya, Arijit Sur, Santosh Biswas, **Suchetana Chakraborty** and Sukumar Nandi, "Exploit Detection Techniques for STP using Distributed IDS", in proceedings of the World Congress on information and Communication Technologies (WICT '11), pp. 939 - 944, December 12-14, 2011, Mumbai, India.

- **Suchetana Chakraborty** and Sushanta Karmakar, "A Novel Approach for Adaptive Data Gathering in Sensor Networks by Dynamic Spanning Tree Switching" , in proceedings of the International Conference on Advances in Computing and Communications (ACC '11), Springer (CCIS), vol-191, pp. 585–594, July 22-24, 2011, Kochi, India.

- **Suchetana Chakraborty**, Sandip Chakraborty, Sukumar Nandi and Sushanta Karmakar, "A Reliable and Total Order Tree Based Broadcast in Wireless Sensor

TH-1251_11610108

Network", in proceedings of the 2nd International Conference on Computer and Communication Technology (ICCCT '11), pp. 618 -623, September 15-17, 2011, Allahabad, India.

- **Suchetana Chakraborty** and Sushanta Karmakar, "Adaptive Convergecast by Distributed Topology Switching" , in proceedings of the 29th Brazilian Symposium on Computer Networks and Distributed Systems (SBRC '11), pp. 545 - 557, May 30-June 3, 2011, Campo Grande, Brazil.

TH-1251_11610108

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Guwahati 781039, India

TH-1251_11610108