Delaunay Triangulation based Spanners for MANET

A thesis submitted
in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

D. Satyanarayana

Under the guidance of

Dr. S. V. Rao

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati - 781 038, INDIA
February 2009

Dedicated

to

My father

Late Shri D. Lakshmaiah

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

CERTIFICATE

This is to certify that the thesis entitled **"Delaunay Triangulation based Spanners for MANET"**, submitted by **D. Satyanarayana**, a research scholar in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, for the award of the Degree of **Doctor of Philosophy**, is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the Institute. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr. S. V. Rao,
Associate Professor,
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati-781 039, India.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Acknowledgments

The successful completion of any task would be incomplete without honoring the most important people without whom I would not have been what I am today.

At the outset, I would like to respectfully express my heartfelt thanks to my supervisor Dr. S. V. Rao for his guidance, support and encouragement. He has been an inspiration throughout my Ph.D program. His moral support gave me confidence of achieving what I aimed at. I once again thank him for everything.

I am tremendously grateful to Prof. Gautam Barua, Director, for his support during my research. I thank him for all that he has given me. I would like to express my sincere thanks to Dr. G. Sajith and Dr. Purandar Bhaduri for their directions during the PhD programme. I thank them for directing me through the proper channel.

I would like to thank my doctoral committee members Dr. Diganta Goswami, Prof. Sukumar Nandi, and Dr. Abijith Mitra for their valuable suggestions. I am exceptionally grateful to all the faculty of Computer Science and Engineering Department, IIT Guwahati for all their support.

I am extremely thankful to my wife Sathyashree. I have been gifted with her constant love, valuable support, understanding, and encouragement that have been of great help to complete my PhD. I am grateful to my family members without whom I would not have been here. Their love and affection has given me the strength during my research.

Last but not the least, I would like to express my deepest thanks to my friends and foes whose warmth and care have really mattered a lot.

I am also indebted to all the masters involved in the enlightenment of my life.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Abstract

Many position based routing protocols use Unit Disk Graph (UDG) as an underlying network topology for routing. Due to large number of edges in UDG, these protocols suffer from channel contention overhead, frequent packet collisions, and heavy resource consumption. To overcome these problems, many researchers proposed various local topology control algorithms to retain only linear number of links in the underlying network graph based on geometric neighborhood. These graphs are called geometric spanners. In this thesis, we study these spanners under various network requirements like less number of transmissions, frequent node failure, mobility, and fault tolerance.

Geometric spanners, like planarized local Delaunay triangulation (PLDel), relative neighborhood graph (RNG), and gabriel graph (GG) which are based on neighborhood properties contain shorter edges. Because of these shorter edges the number of transmissions between source and destination increases which in turn increases the end-to-end packet delay and jitter. We present three planar constrained based geometric graphs called constrained local Delaunay triangulation (CDT), constrained relative neighborhood graph (CRNG), and constrained Gabriel graph (CGG), to reduce the number of hops by introducing longer constraint edges.

In adhoc networks, nodes can go down due to various reasons, such as insufficient battery power, environmental effects like eruption of volcano, cyclones, and floods, and accidents like landslides and debris. Moreover, to conserve the energy, nodes can switch off their transmitter or go to the sleep mode. There will be heavy packet loss if these nodes exist in any routing path. Similarly, a new node can join the network or an existing node wakes up from sleep mode. We have proposed three dynamic spanners called dynamic local Delaunay triangulation (DLDel), dynamic relative neighborhood graph (DRNG), and dynamic Gabriel graph (DGG), which change their network topology dynamically to pre-

serve the spanner properties and reduce heavy packet loss.

Various resource limitations and environmental constraints make frequent link and node failures in adhoc networks, which make the network unreliable. For example, the edge disconnections occur due to buildings, walls, mountains, and obstacles between the wireless nodes. Similarly, the node failures occur due to the exhausted battery power, accidents, landslides, debris, eruption of volcano, and cyclones. So, network topology should be fault tolerant to take care of these failures. In this thesis, we have proposed the algorithms for fault tolerant versions of PLDel, RNG and GG called fault tolerant local Delaunay triangulation (FTLDel), fault tolerant relative neighborhood graph (FTRNG), and fault tolerant Gabriel graph (FTGG), respectively, by choosing most stable nodes.

The existing spanners assume that the nodes in the network are static. The frequent topology change due to node mobility disturbs various geometric properties of the spanner such as neighborhood relations, spanning ratio, and planarity. Moreover, some of the edges may become invalid links and may lead to disconnected network. In this thesis, we propose the algorithms for mobile local Delaunay triangulation (MLDel), mobile relative neighborhood graph (MRNG), and mobile Gabriel graph (MGG), to maintain their counter part spanners PLDel, RNG, and GG, respectively, under mobility.

These proposed spanners are simulated using the network simulator ($ns2.28$) and their performance are better than their counter parts, which consolidate our claims.

Contents

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

List of Figures

Chapter 1

Introduction

A wireless network allows two or more computers to communicate with each other without a network cable. Recent technological advancements in wireless communication and portable devices lead to proliferation of wireless networks. These networks are broadly classified into two kinds: infrastructured and adhoc networks. In infrastructured wireless networks, the mobile terminals communicate through an access point or base station. These access points are communicated to the internet by a high speed wired network, as shown in the Figure 1.1. Most popular infrastructured wireless networks include wireless local area network (WLAN) and cellular networks.

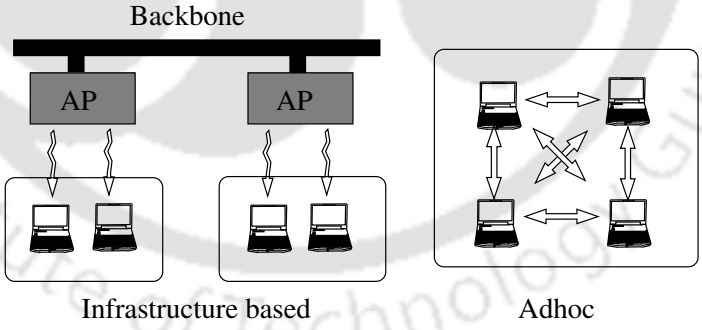


Figure 1.1: Infrastructured and infrastructureless wireless networks.

The other kind of wireless networks represent a complex distributed system that comprises wireless mobile nodes which can freely and dynamically self-organize into arbitrary and temporary network called adhoc network. An adhoc network is a collection of wireless nodes communicating via radio without any pre-existing infrastructure. A significant attention has been given to adhoc networks, as they provide ubiquitous connectivity with-

out any infrastructure. The idle characteristics made adhoc networks suitable for many application areas such as battlefield surveillance, disaster recovery, emergency services, education, environmental, agriculture, and biological applications. However, constraints like limited battery, shared medium, mobility, and infrastructureless nature pose various research challenges at different levels of the network architecture, see the Figure 1.2.

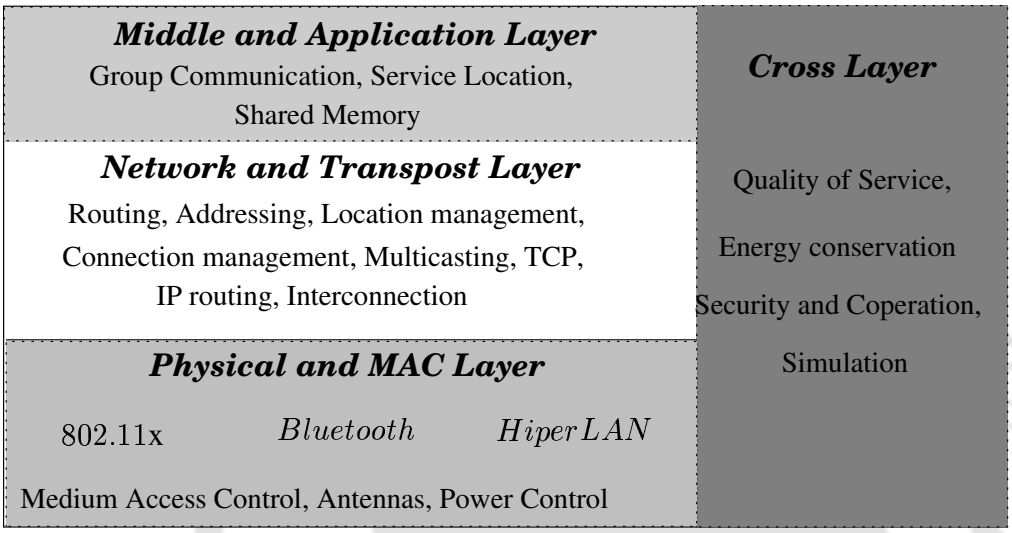


Figure 1.2: Research challenges at different layers.

The physical layer describes several challenges relating to radio wave modulation and signaling characteristics for data transmission, where as the medium access control (MAC) describes different flexible and fair access methods to the communication medium. The aim of networking protocols is to use the one-hop transmission services provided by the enabling technologies to construct end-to-end (reliable) delivery services, from a sender to one (or more) receiver(s). Apart from military, the mobile adhoc networks (MANETs) have attracted considerable attention and interests from commercial business industry [HHCW05, XC06, QHM+06], as well as the standard community. These applications introduce several research areas in MANET, such as service discovery, group communication, and shared memory. The cross layer issues, such as QoS, energy conservation, and security, span through all the layers of the network.

Unlike wired networks, the frequent disconnections due to node mobility and transient characteristics of wireless channel pose new research challenges to adhoc routing. In addition, due to the infrastructureless nature of adhoc network, each node has to act as both

a host and a router. This makes the routing in adhoc networks different from conventional infrastructure based wireless networks.

1.1 Routing

The protocols for routing in adhoc networks have been an active area of research since the early 1990s. Early research in this area focused on adapting the internet routing protocols for adhoc networks. But, it was not successful, as the characteristics of MANET are different from the internet.

In adhoc network, two nodes communicate directly, if they are in the communication range of each other. If they are far away then the nodes communicate with intermediate relay nodes and it is called multi-hop routing. Royds et al [RE04] describes the characteristics of single hop and multi-hop adhoc networks. The multi-hop routing in wireless adhoc network is one of the challenging tasks due to its distributed nature. Numerous routing protocols have been proposed for multi-hop routing in adhoc networks [PR99, JM96, PC97, PB94, MGLA96, KK00, BMSU01, DRWT97, HPS02, GLAS99, Toh97, CE95, KV98, KN97, KVCP97, SSB99, AWD04, BMJ+98]. A comprehensive survey on the routing protocols for mobile adhoc networks can be found in [RT99, TLW02, AWD04, BMJ+98, CM99]. These protocols can be classified into various categories.

1.1.1 Classification

Protocols can be classified into many different ways based on their properties. Many classifications of routing protocols exist in the literature. Some of these classifications include pro-active and reactive, flat and hierarchical based, and ID-based and location based routing protocols.

Proactive or table-driven routing protocols maintain the information of network topology in the form of tables. In other words, each node maintains routing information of every other node in the network. The examples include destination sequenced distance vector (DSDV) [PB94], wireless routing protocol (WRP) [MGLA96], and source tree adaptive routing (STAR) [GLAS99]. The reactive or on-demand routing protocols obtain the rout-

ing information only on demand. That is, whenever a source wants to send data, it first finds a route to the destination and sends the data on that path. For example, adhoc on-demand distance vector (AODV) [PR99], dynamic source routing (DSR) [JM96], temporally ordered routing algorithm (TORA) [PC97], and associativity based routing (ABR) protocol [Toh97] are some of the popular protocols.

The flat routing protocols regard the whole network as uniform in which functions and responsibilities of each node is same. That is, each node has the uniform responsibility for constructing routes. On the other hand, hierarchical routing protocols organize the network as a tree of clusters, where the roles and functions of nodes are different at various levels of the hierarchy. Routes are constructed according to the node's position in the virtual hierarchy. Some of the popular routing protocols in this category include zone routing protocol (ZRP) [HPS02], hierarchical state routing (HSR) [PGHC99], and core extraction based distributed adhoc routing (CEDAR) [SSB99].

Another way of classifying these routing protocols is shown in the Figure 1.3, in which all the routing protocols are classified into three categories unicast, multicast, and broadcast.

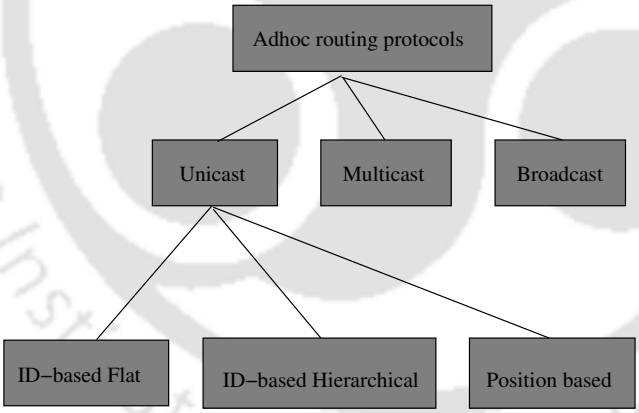


Figure 1.3: Adhoc routing protocol classification.

In unicast routing, a source node sends a packet to a certain destination node. The multicast routing protocols send the packets to a specific set of destination nodes from a given source. The multicast communications have emerged as one of the most researched areas as it is important in many applications, such as video conference. Some of the multicast routing protocols include adhoc multicast routing protocol (AMRoute) [XTML02], on-demand multicast routing protocol (ODMRP) [LSG02], a multicast protocol for ad-

hoc wireless networks (AMRIS) [WT99], and hierarchical QoS multicast routing protocol (HQMRP) [LC05]. In the broadcast routing, a source node sends a packet to all the nodes in the network. For many wireless applications such as group conference and digital audio/video broadcast, it is necessary to send the data to all devices that form the network. Some of the broadcast protocols for MANET include scalable broadcast algorithm (SBA) [PL00], adhoc broadcast protocol (AHBP) [PL02], and lightweight and efficient network-wide broadcast (LENWB) [SM00].

The unicast routing protocols are further divided into three categories: ID-based flat, ID-based hierarchical, and position based. The ID-based flat routing protocols route the packets according to node's ID or address. This requires nodes to disseminate their ID or address through their neighbors. Some of the ID-based flat routing protocols include AODV [PR99], DSDV [PB94], DSR [JM96], and boundary state routing (BSR) [GYS05].

Unlike flat routing protocols that regard the entire network as uniform, ID-based hierarchical routing protocols divide the network nodes into different groups, or clusters. The routing decisions are made based on a subset of nodes called cluster heads. These cluster heads form a backbone of the network, through which the other nodes communicate. Some examples include cluster-based routing protocol (CBRP) [AN06], dynamic address routing (DART) [EFK07], zone routing protocol (ZRP) [HPS02], core extraction distributed adhoc routing (CEDAR) [SSB99], and global state routing (GSR) [CG98].

Position based routing protocols take the advantage of position information of nodes to direct the packet flow. There are two issues that need to be addressed for enabling the use of location information in routing. The first one is the availability of a location service, which involves keeping track of up-to-date locations of nodes and responding to the location queries. The second one is making use of the location information of destination in routing protocols. Some of the position based routing protocols include greedy perimeter stateless routing (GPSR) [KK00], FACE [BMSU01], greedy other adaptive FACE routing (GOAFR) [KWZZ03], and location-aided routing (LAR) [KV98].

1.1.2 Position based routing

Adhoc networks can be classified into two categories, static and mobile. In static adhoc networks, position of nodes does not change once they become a part of the network. The

examples for static networks are rooftop networks [BVGLA99]. On the other hand, in mobile adhoc networks, nodes can move arbitrarily. Examples for this category are a network of hand-held devices and vehicular network. The routing in such networks is a challenging task because of their frequent topology changes without prior notice. There are two approaches to handle the routing in such networks: ID based routing and position based routing. ID based routing protocols use neighbor information to perform packet forwarding. The route discovery process in reactive methods and explicit route maintenance in proactive methods can be very expensive in terms of communication costs, which in turn reduces the response time of the networks. The other problems include scalability, limited resources, and computational overhead.

One way of overcoming these problems is by using the position information of the nodes. That is, each node in the network has to know its physical position. Each node is equipped with a global positioning system (GPS) [KH06, CHH01] to know its position or estimates the same by using signal strength and time delay with respect to its neighbors [Poo94, Caf99, QKS06].

Position based routing methods forward the data packets based on the position of destination and its neighbors. These are also called localized routing methods, since each node uses only its neighbors information for data forwarding. In localized routing, the communication and computation overhead is reduced compared to ID based routing protocols because of the absence of routing table maintenance. Moreover, localized routing methods are scalable. Some of the popular position based routing protocols include location aided routing (LAR) [KV98], greedy perimeter stateless routing (GPSR) [KK00], FACE routing [BMSU01], adaptive FACE routing (AFR) [KWZ02], geographic distance routing (GEDIR) [SL01], geographic adaptive FACE routing (GOAFR$^+$) [KWZZ03], geographical routing using partial information (GRPI) [JPS01], two-level link state routing (TLSR) [JNL99], and route discovery optimization (RDO) [BSC02]. In these methods, sender need to know the position of destination, which can be done by location service. Some of the main location services are distance routing effect algorithm for mobility (DREAM) [BCSW98], Quorum-based location service [SLJ08], grid location service (GLS) [LJD$^+$00], predictive location service (PLS) [LCN05], and hierarchical location service (HLS) [KFWM04]. A complete survey on position based routing is given

in [Sto02, MWH01, ID03].

1.2 Routing based on Geometric spanners

In position based routing, the most generic model of adhoc network is unit disk graph (UDG) [KZ03]. The UDG assumes that all the nodes in the network contain the same transmission range of one unit. It is a network graph which contains an edge $\overline{uv}$ if and only if $|\overline{uv}| \leq 1$ unit, where $|\overline{uv}|$ represents the length of the edge $\overline{uv}$, see the Figure 1.4. The number of edges in UDG can be as large as $O(n^2)$, where n is the number of nodes. Many position based routing protocols use UDG as an underlying network topology for routing. Due to large number of edges in UDG, these protocols suffer from channel contention overhead, frequent packet collisions, and heavy resource consumption. To overcome these problems, many researchers proposed various local topology control algorithms to retain only linear number of edges in the underlying network graph based on geometric neighborhood. These graphs are called geometric spanners.

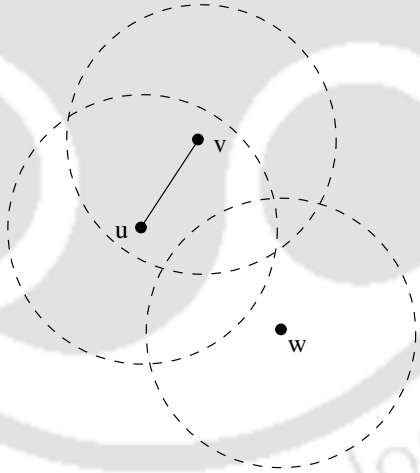


Figure 1.4: Unit Disk Graph model.

In addition to the optimization of power consumption, the spanners avoid packet loss due to local maxima and local loops, which is common in greedy forwarding strategies. For more details on local maxima and local loops, please refer the Section 2.5.

A class of position based routing protocols [KK00, KWZ02, BMSU01, KWZZ03, BM99] use various geometric spanners (subgraphs of UDG) as an underlying network graph for efficient routing. A geometric spanner is a spanning subgraph H of a given

graph G such that the path length between a pair of nodes in H is not more the t times the path length between the same pair of nodes in graph G, where the parameter t is called stretch factor. The number of edges in the spanner is in $O(n)$. Some of the spanners used in wireless networks are minimum spanning tree (MST), nearest neighborhood graph (NNG), relative neighborhood graph (RNG), Gabriel graph (GG), Delaunay triangulation (Del), and Yao graph (Yao). A detailed discussion on various geometric spanners and their routing protocols are given in **Chapter 2**.

1.3 Motivation

The existing geometric spanners are defined based on various neighborhood properties. But, there is no consideration of various network requirements in spanner construction. For example, due to the neighborhood relation property of the spanner, the smaller edges in the network graph leads to larger hop count which in turn leads to other problems such as larger delay, low throughput, and high energy consumption. Another important requirement of adhoc network is scarce energy resource. In wireless sensor networks a failure occurs when the entire battery power is exhausted. But, these nodes failure cause the violation of geometric properties, especially the neighborhood relation property. In addition, the node failure cause heavy packet loss if the nodes are in the routing path. Another problem with the existing spanners is that the nodes in the network are assumed to be static, where the MANET can not be limited by the constraint. The topological changes due to node mobility cause the violation of geometric properties. In other words, if the changes are not reflected in the network graph due to the node mobility, there will be heavy packet loss.

The above problems have been the motivation to study the geometric spanners under various network requirements. Since the spanners PLDel, RNG, and GG are more frequently used in wireless networks, we are interested to study the various properties of PLDel, RNG, and GG under certain network conditions.

1.4 Objective

The main objective of this thesis is the study of geometric spanners under various network requirements. Precisely, the need of study is on neighborhood relation property of geomet-

ric spanners and the problems imposed by this property on network performance. In addition, there is a need to find the solutions for these problems and evaluate the performance of these solutions. The next objective of the thesis is the study of dynamic maintenance of the spanners under frequent node failure conditions which is very common in adhoc and sensor networks. There is a need to find efficient localized algorithms for the dynamic maintenance of the spanners and evaluate the performance of these algorithms. The other objective is to study the need of stability based spanners at various network conditions. As fault tolerance is an important requirement in resource constrained adhoc networks, especially in wireless sensor networks, there is a need to append fault tolerance to the spanners. The last objective of the thesis is the study of node mobility on spanners and find the strategies to mobility management. As the node mobility disturbs the geometric properties of the spanner, there is a need to find the solutions which should consider both the geometric properties of the spanner and improve the network performance.

In the literature, the spanners have been studied more on their structural properties, rather than evaluating the performance of the spanners by looking at various network parameters such as delivery ratio, hop count, delay, jitter, throughput, and power consumption. In other words, it is important to conduct the simulation experiments on the spanners and evaluate their performance.

1.5 Contributions

This thesis focuses on the geometric spanners PLDel, RNG, and GG. We study these spanners under various network requirements like less number of transmissions, frequent node failure, mobility, and fault tolerance.

Constrained based geometric graphs: Geometric spanners, like PLDel, RNG, and GG which are based on neighborhood properties contain shorter edges. Because of these shorter edges the number of transmissions between source and destination increases which in turn increases the end-to-end packet delay and jitter. One can use constrained Delaunay triangulation (CDT) [Che87, She00] based spanners with suitable constraint edges. However, the method proposed in [Che87, Lee97, CR90, CW99, She00, Pet] are centralized and not suitable for adhoc networks. We present three planar constrained based geo-

metric graphs called constrained local Delaunay triangulation (CDT), constrained relative neighborhood graph (CRNG), and constrained Gabriel graph (CGG), by introducing long edges as constraint edges. Non-intersecting constraint edges are added to maintain the planarity. We have given two algorithms to place the constraint edges. More the number of constraint edges lesser the average hop count. Simulation results on these constrained geometric graphs show improvement in hop count, delay, and jitter over their respective non-constrained spanners.

Dynamic spanners: In adhoc networks, nodes can go down due to the various reasons, such as insufficient battery power, environmental effects like eruption of volcano, cyclones, and floods, and accidents like landslides and debris. Moreover, to conserve the energy, nodes can switch off their transmitter or go to the sleep mode. There will be heavy packet loss if these nodes exist in any routing path. Similarly, a new node can join the network or an existing node wakes up from sleep mode. In these cases, it is necessary to change network topology dynamically as and when required for efficient routing. In other words, the spanners for adhoc networks need to change dynamically to retain the geometric properties of the spanner for efficient routing. One can use Ming li's dynamic Delaunay triangulation [MXP05] approach, which periodically reconstructs the entire spanner. But this approach takes large computational and communication cost. We have proposed three dynamic spanners called dynamic local Delaunay triangulation (DLDel), dynamic relative neighborhood graph (DRNG), and dynamic Gabriel graph (DGG), which change their network topology dynamically to preserve the spanner properties and heavy packet loss. The spanners DLDel, DRNG, and DGG reconfigure the graph locally, instead of global reconstruction, to recover quickly from any packet loss occur in PLDel, RNG, and GG, respectively.

Fault tolerant spanners: Various resource limitations and environmental constraints make frequent link and node failures in adhoc networks, which make the network unreliable. For example, the edge disconnections occur due to buildings, walls, mountains, and obstacles between the wireless nodes. Similarly, the node failures occur due to the exhausted battery power, accidents, landslides, debris, eruption of volcano, and cyclones. So, network topology should be fault tolerant to take care of these failures. In the literature, two types of fault tolerant spanners are considered [LNS02, CZ03, Luk99], namely, k-

node fault tolerant spanners (k-NFTS) and k-link fault tolerant spanners (k-LFTS). These are centralized algorithms which are not suitable for adhoc networks. In this thesis, we have proposed the algorithms for fault tolerant versions of PLDel, RNG and GG called fault tolerant local Delaunay triangulation (FTLDel), fault tolerant relative neighborhood graph (FTRNG), and fault tolerant Gabriel graph (FTGG), respectively, by choosing most stable nodes.

The spanners under mobility: The existing spanners assume that the nodes in the network are static. The frequent topology change due to node mobility disturbs various geometric properties of the spanner such as neighborhood relations, spanning ratio, and planarity. Moreover, some of the edges may become invalid links and may lead to disconnected network. When these spanners are used as underlying network graph by routing protocols, there will be degradation in the network parameters like delivery ratio, delay, and jitter. One way of solving this problem is to reconstruct the entire spanner periodically. But constructing the entire spanner from the scratch involves large communication and computational costs. In this thesis, we propose the algorithms for mobile local Delaunay triangulation (MLDel), mobile relative neighborhood graph (MRNG), and mobile Gabriel graph (MGG), to maintain their counter part spanners PLDel, RNG, and GG, respectively, under mobility. Our approach consists of more frequent less cost local updates and less frequent more cost global updates.

These proposed spanners are simulated using the network simulator ($ns2.28$) and their performance are better than their counter parts, which consolidate our claims.

1.6 Organization of the thesis

In the next chapter, we given a survey on geometric spanners and related work. In the **Chapter 3**, we discuss the constrained geometric graphs CDT, CRNG, and CGG. The fourth Chapter proposes three algorithms DLDel, DRNG, and DGG to maintain the spanners PLDel, RNG, and GG, respectively, at various dynamic environments. The **Chapter 5** describes the fault tolerant versions of the spanners PLDel, RNG, and GG. The **Chapter 6** proposes three spanners MLDel, MRNG, and MGG to maintain the graphs under mobility conditions. Finally, the **Chapter 7** concludes the thesis with some pointer to future

research directions.

Chapter 2

Geometric Spanners

One of the perceptible requirement of network design is to construct a subgraph such that the shortest path connecting any two nodes in the subgraph is not much longer than the shortest path connecting them in original unit disk graph. Such subgraphs are called spanners. In other words, a spanner G' is a spanning subgraph of the given graph G such that the length of the shortest path between any two nodes in G' is bounded in the graph G. Formally, a graph G' is a t-spanner of UDG if and only if $|\Pi_{G'(u,v)}| \leq t \cdot |\Pi_{UDG(u,v)}|$, where $\Pi_{G'(u,v)}$ represents the shortest path between the two nodes u and v in graph G' and $|\Pi_{G'(u,v)}|$ represents length of the path. The value t is called the stretch factor of the spanner. More precisely, the value t is called length stretch factor, hop stretch factor, and power stretch factor if length of the path is measured, respectively, in terms of Euclidean distance, number of hops, and power required to transmit.

Both computational geometry scientists and network engineers have studied various geometric spanners to utilize the useful geometric properties in several applications of wireless networks. We review some of these geometric spanners.

2.1 Delaunay triangulation based geometric spanners

The graphs based on Delaunay trianglation are well studied in the field of computation geometry. However, it is widely used in the area of wireless networks because the benefits provided are as good as complete graphs [DFS90]. Some of the Delaunay based graphs are described below.

Minimum Spanning Tree (MST): The smallest spanner that connects all the nodes in the network is a minimum spanning tree. The MST is well studied in the literature [CSRL01, BR03]. The MST is not used in adhoc networks, because its stretch factor is too high. One can verify from the Figure 2.1 that the length and hop stretch factor can be as large as $n-1$.

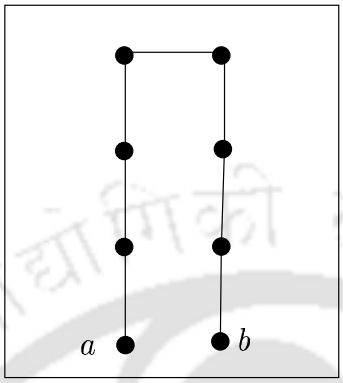


Figure 2.1: Stretch factor is $n-1$.

Nearest Neighbor Graph (NNG): The edges in the nearest neighborhood graph [EPY97] are determined by minimum distance. For any node p, there exists an edge $\overline{pq}$ if and only if node q is the nearest neighbor of p. A generalization of NNG is called k-nearest neighbor graph (k-NNG), for $k \geq 1$. Each node p is connected to a node q, if p is i^{th} nearest neighbor of q for some $i \leq k$. Although NNG incorporates a useful notation, it has the disadvantage that the resulting graph may be disconnected as shown in the Figure 2.2.

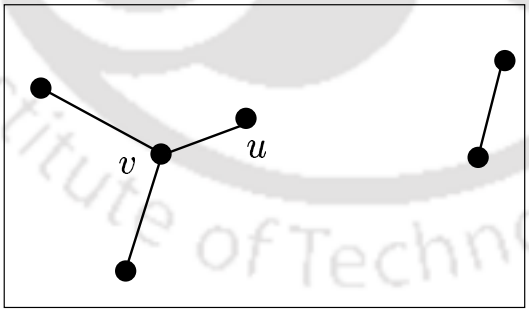


Figure 2.2: Disconnected NNG.

Relative Neighborhood Graph (RNG): Two nodes u and v of a set v of nodes, are connected in RNG if and only if their lune $lune(u,v)$ does not contain any other node $w \in V$. The lune $lune(u,v)$ is defined as the intersection of the two circles of radius $d(u,v)$ centered at u and v respectively, as shown in the Figure 2.3. The algorithm to

construct RNG for wireless networks is given in [KK00]. The spanning ratio of RNG is in $O(n)$ [BDEK02]. RNG is a connected graph since MST $\subseteq$ RNG [Sup83].

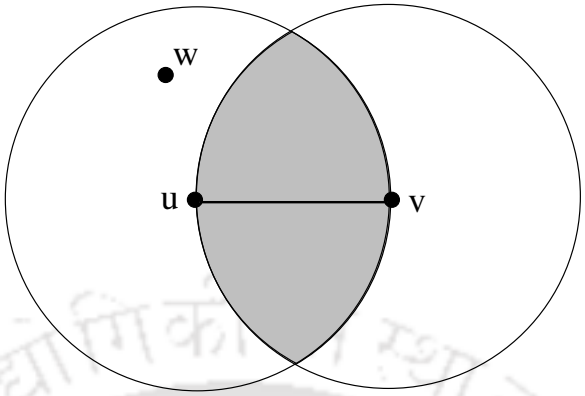


Figure 2.3: RNG Rule.

Gabriel Graph (GG): The edge $\overline{uv}$ exists between two nodes u and v in GG if there is no node w inside the circle with $\overline{uv}$ as the diameter as shown in the Figure 2.4. In other words, the edge $\overline{uv} \in GG$ if and only if the circle centered at $(u+v)/2$ with the radius $d(u,v)/2$ does not contain any other node inside, where $d(u,v)$ denotes distance between u and v. Bose and Martin have given a localized algorithm [BMSU01] to construct GG for adhoc networks. The spanning ratio of GG is bounded by $O(\sqrt{n})$ [BDEK02, Epp96].

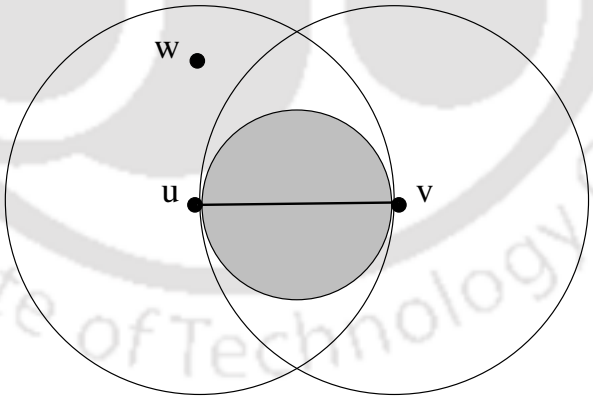


Figure 2.4: GG Rule.

Delaunay triangulation (Del): The Delaunay triangulation (Del) [PS90] of V is the triangulation of all the nodes such that the circumcircle of each triangle should not contain any other node $x \in V$, see the Figure 2.6. Delaunay triangulation is a planar graph with spanning ratio of 2.42 [KG89, KG92]. Hence, it is a t-spanner of UDG. RNG and GG

are subgraphs of Del, still it contains linear number of edges in the graph. However, it is not suitable to compute Delaunay triangulation in localized manner [LCWW03, LCW02]. Moreover, Del is also not suitable for adhoc networks, because it contains longer edges than the transmission range. For example, consider the Figure 2.5, the triangle $\triangle abc$ is a Delaunay triangle, as it does not contain any other node inside the circumcircle of three nodes a, b, and c. But this triangle is not suitable in adhoc networks, because the distance $|\overline{ab}|$ is more than the transmission range.

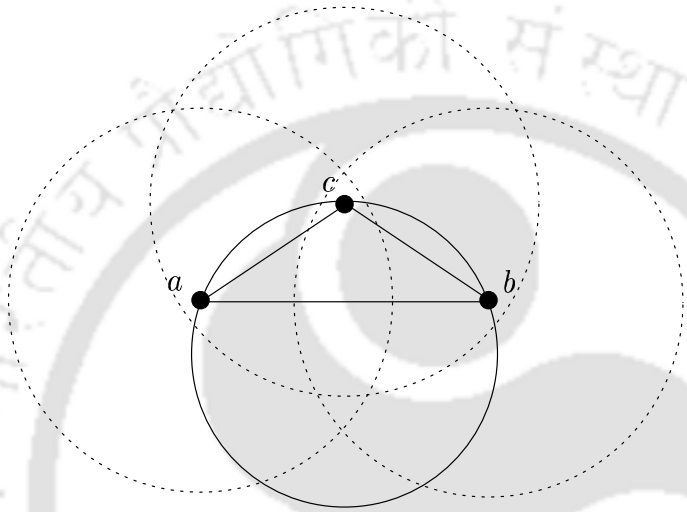


Figure 2.5: Edge $\overline{ab} \in$ Del and its length is longer than transmission range of a and b.

Unit Delaunay triangulation (UDel): The unit Delaunay triangulation UDel [LCW02] can be obtained from Del by removing all the edges longer than one unit. UDel is a t-spanner of UDG [LCW02, GGH$^+$01]. Though the UDel is suitable for adhoc networks, it is not known how to construct UDel locally [LCW02].

k-localized Delaunay triangulation ($LDel^k$): $LDel^k$, for $k \geq 1$, contains all the Gabriel edges and k-localized Delaunay triangles. A triangle $\triangle abc$ is called k-localized Delaunay triangle if the circumcircle of the nodes a, b, and c does not contain any other node which is a k-neighbor of the nodes a, b, or c. In the example of LDel^1 as shown in the Figure 2.6, the circumcircle of the triangle $\triangle uvw$ do not contain any other node x such that $x \in n_1(u) \cup n_1(v) \cup n_1(w)$, where $n_1(u)$ denotes the set of 1-hop neighbors of node u. Some of the properties of these spanners are below:

Lemma 2.1.1 *[LCWW03, LCW02]* $UDel \subseteq LDel^k$, *for all* $k \geq 1$.

Lemma 2.1.2 *[LCWW03, LCW02]* $LDel^{k+1} \subseteq LDel^{k}$.

Lemma 2.1.3 *[LCWW03, LCW02]* $LDel^{k}$ *are planar graphs for all* $k \geq 2$.

Lemma 2.1.4 *[LCWW03, LCW02]* $LDel^{1}$ *is not a planar graph.*

Planarized local Delaunay triangulation (PLDel): Xiang li et al. [LCWW03] gave a localized algorithm to construct Delaunay triangulation for adhoc networks called planarized localized Delaunay triangulation. In this algorithm, first it constructs LDel^1 with one hop neighbors. LDel^1 is a t-spanner of UDG [LCW02], but is not a planar graph. A planarization algorithm is applied on LDel^1 to make it a planar graph and it is called planarized local Delaunay triangulation (PLDel). PLDel is a planar 2.42-spanner of UDG [LCWW03, LCW02].

The following inclusion property summarizes the relations among various geometric spanners.

Lemma 2.1.5 $MST \subseteq RNG \subseteq GG \subseteq LDel^{2} \subseteq PLDel \subseteq LDel^{1}$.

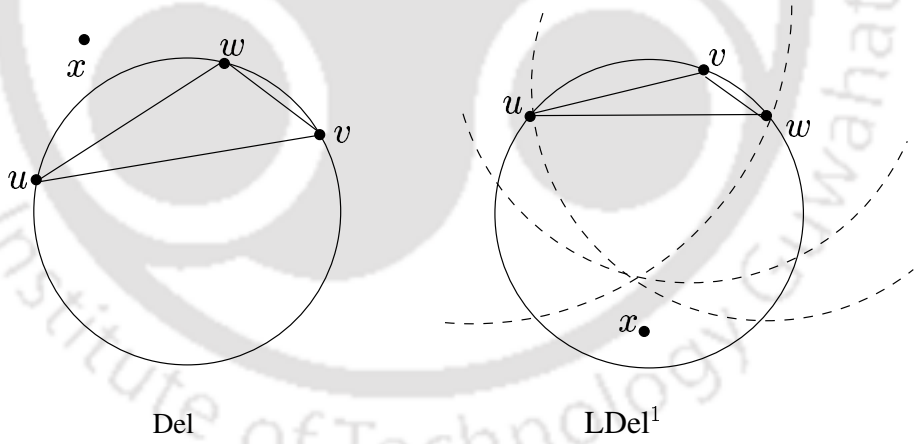


Figure 2.6: Rules of Del and LDel^1.

Partial Delaunay Triangulation (PDT): The partial Delaunay triangulation (PDT) [LSW04] is especially designed for bluetooth wireless technologies. The spanner PDT can be constructed in a localized manner. PDT contains all edges of Gabriel graph.

Let r be the node inside the $disk(p, q)$ with largest angle $\angle prq$, where $q \in n_1(p)$. The edge $\overline{pq}$ is added to PDT if the following conditions hold:

1. There is no node from $n_1(p)$ that lies on the different side of $\overline{pq}$ with respect to r and inside the circumcircle passing through p, q, and r, see the Figure 2.7.

2. $\sin(\theta) > \frac{d}{R}$, where R is the transmission range, d is the diameter of the circumcircle of $\triangle pqr$, and $\theta = \angle prq$.

The PDT is a subgraph of unit Delaunay triangulation (UDel).

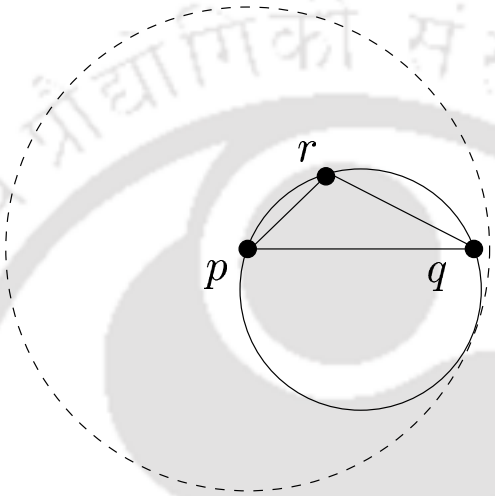


Figure 2.7: PDT rules.

Restricted Delaunay Graph (RDG): Gao et al [GGH$^+$01, GGH$^+$05] proposed a planar spanner called restricted Delaunay graph (RDG), which combines the node clustering algorithm with Delaunay triangulation graph. The spanner has constant length and hop stretch factors.

The construction of RDG is carried out in two phases. In the first phase, each node u collects its 1-hop neighbors $n_1(u)$, by receiving the broadcasted *hello_packets*. Each node u computes the Delaunay triangulation $Del(n_1(u))$ and broadcast this information. In the second phase, node u checks for consistency as follows: for each edge $\overline{uv} \in Del(n_1(u))$ and each node $w \in n_1(u)$, if $u, v \in n_1(w)$ and $\overline{uv} \notin Del(n_1(w))$, then node u deletes the edge $\overline{uv}$. The resulting graph is called RDG. The above algorithm is computed on cluster-heads and gateways, rather than on full node set.

Clustered Delaunay Graph (CDG): An improved version of RDG called CDG [ALW$^+$03, WL02] is proposed by Alzoubi et al. It is bounded degree planar spanner which integrates

the connected dominating set with the local Delaunay triangulation to form the backbone of the wireless network. The spanner is constructed in two phases: First, compute the connected dominating set (CDS) using the Alzoubi's method [Alz02, AWF02]. Apply the planarized local Delaunay triangulation on top of induced connected dominating set (ICDS) to obtain the spanner.

2.2 Other spanners

Yao Graph (Yao): In Yao graph [WLBW01, LWW01], the transmission range circle of a node u is divided into k equal sectors, where $k \geq 6$, as shown in Figure 2.8. In each sector, closest node is connected with u by an edge, if exists. Ties are broken arbitrarily. The spanning ratio of Yao graph is bounded by a constant $\frac{1}{1-2sin\frac{\pi}{k}}$ [LWW01, WLBW01]. Unlike GG and RNG, Yao graph is not a planar graph.

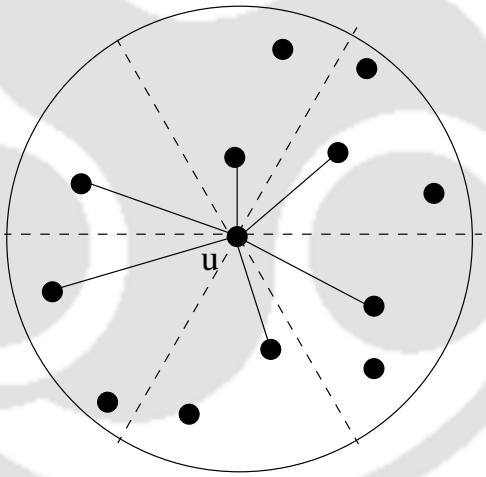
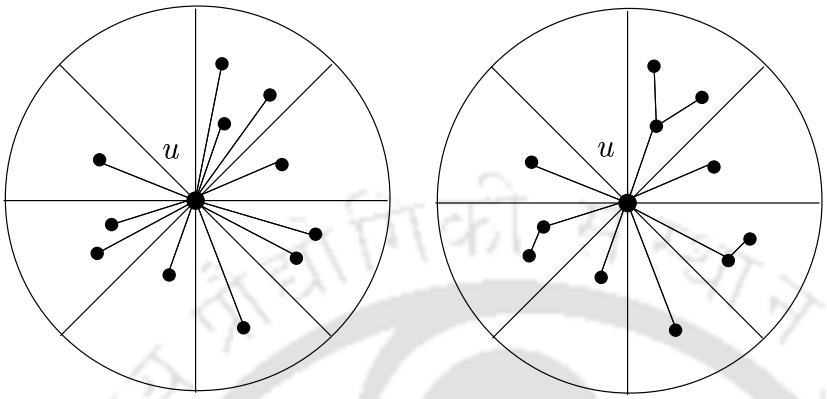


Figure 2.8: Yao Graph Rule.

Bounded degree spanners Yao-Yao graph [LWWF02] and symmetric Yao graph [LWWF02], variation of Yao graph, are proposed to reduce various network resources, such as power and memory.

Sink structure: Even though Yao graph provides bounded out degree, some nodes may have a very large in-degree, which causes large overhead. Li et al. [LWW01] and Arya et al. [ADM+95] have given a technique called sink structure for constructing the bounded degree and bounded power stretch factor from Yao graph. In this technique, the directed

star consisting of all the edges toward a node u is replaced by the directed tree, as shown in the Figure 2.9. The power stretch factor of the sink structure is at most $(\frac{1}{1-(2sin\frac{\pi}{k})^{\beta}})^2$, the maximum degree of the graph is $(k+1)^2-1$, and the maximum out degree is k, where β is the path loss component and k is the number of sectors [LWW01].



(i) (ii)

Figure 2.9: (i) Star formed by links toward u. (ii) Directed tree sinked at u.

2.3 Spanner metrics

Some of the main metrics on which spanners are evaluated are given below.

1. **Stretch factor:** The length of the shortest path between two nodes in the spanner is bounded with respect to shortest path in UDG. It is desirable that the stretch factor of the spanner is as less as possible.

2. **Planarity:** The planarity of spanner is required for many geographic routing protocols like GPSR, FACE, AFR, GFG, and GOAFR, to guarantee the delivery of packets to destination.

3. **Bounded degree:** The bounded node degree will make the network to consume fewer resources, like memory, power, and bandwidth. This is crucial for adhoc networks since nodes have limited resources.

4. **Sparseness:** The sparseness of the topology reduces the channel contention at the MAC level because the network graph contains fewer number of links. If the network

graph is too sparse like MST, the spanning ratio increases, which is not preferable. Thus, there is a need for optimum sparseness. Sparseness enables most routing protocols to run efficiently.

5. **Localized construction:** Distributed local algorithms are desirable for construction and maintenance of the spanners in adhoc networks because, it is easier to maintain the information of nodes within a constant number of hops.

2.4 Routing protocols

This section describes the protocols which use various geometric spanners as an underlying network graph.

Greedy perimeter stateless routing (GPSR) [KK00]: In GPSR, the data packet is forwarded greedily toward the destination. In other words, the current node forward the packet to the node in its 1-hop neighbor which has the least distance to the destination node. If a node does not find any relay node in its 1-hop neighbors it uses perimeter routing until the current node finds closest neighbor to the destination. This procedure is followed until the packet reaches the destination. The spanners RNG and GG are used as the underlying network graph when GPSR uses the perimeter routing.

FACE routing [BMSU01]: In the FACE routing the entire network graph is divided into faces which are surrounded by the edges of the polygon. This is achieved by constructing the Gabriel graph. In the forwarding process, the packet is traversed through all the edges of each face which crosses the virtual line from source to destination. The packet traversal on the face is based on right hand rule.

Greedy FACE greedy (GFG) [BMSU01]: The GFG is an improved version of FACE routing. In GFG, the packet traversal is taken place through only some of the edges of the face, instead of traversing the entire face, as in FACE routing.

Adaptive FACE routing (AFR) [KWZ02]: The AFR is an enhancement of FACE routing. The path searching in AFR is restricted by ellipse. In other words, the packet forwarding through the edges of GG faces should be within the ellipse. If the best route has the cost c in the graph then AFR finds a route which always has the cost less than or equal to c^2.

Geographic adaptive FACE routing (GOAFR$^+$) [KWZZ03]: The GOAFR$^+$ is the com-

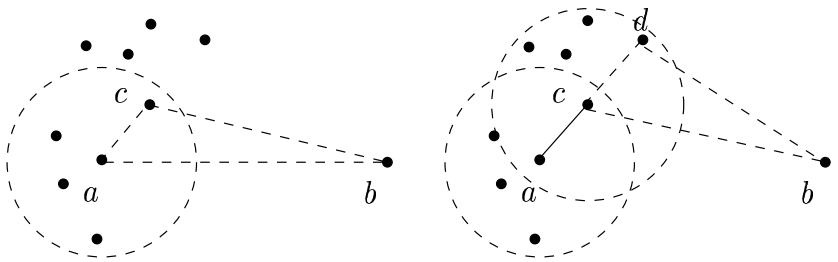


Figure 2.10: Greedy routing (i) Node a selecting relay node c as $|\overline{cb}|$ is smallest. (ii) Node c selecting next relay node d as $|\overline{db}|$ is smallest.

bination of greedy forwarding and FACE routing. In GOAFR$^+$, the packet is forwarded greedily toward the destination. That is, the packet is forwarded to the neighboring node which is closest to the destination. Once a node finds local maxima, it uses FACE routing to forward the packet. That is, using right hand rule, it forward the packet on the faces of the network graph. During the FACE routing, GOAFR$^+$ restricts the search area as in AFR.

In addition, the following are some of the important greedy routing protocols in the literature.

1. **Greedy routing (Grdy) [BMSU01]:** As shown in the Figure 2.10(i), let a be the current node and the dotted circle is the transmission range of node a. The node b is destination node. To forward a packet from node a to the destination node b, the node a selects a node c as the next relay node in its transmission range such that the euclidean distance $|\overline{cb}|$ is the smallest among a's 1-hop neighbors. Similarly, in the next step, as shown in the Figure 2.10(ii), the current node c selects the next relay node d such that the distance $|\overline{db}|$ is the smallest among c's 1-hop neighbors. This process is repeated until the packet reaches the destination node b.

2. **Compass routing (Cmp) [BCSW98, KSU99]:** As shown in the Figure 2.11(i), let the nodes a and b be the current and destination nodes respectively. If the node a is to transmit a packet to the destination node, it sends a packet to the next relay node c which makes the smallest angle. That is, $\angle cab$ is the smallest among all a's neighbors. Similarly, in the next step the current node c forward the packet to the next relay node d such that $\angle dcb$ is the smallest among c's 1-hop neighbors, as shown in

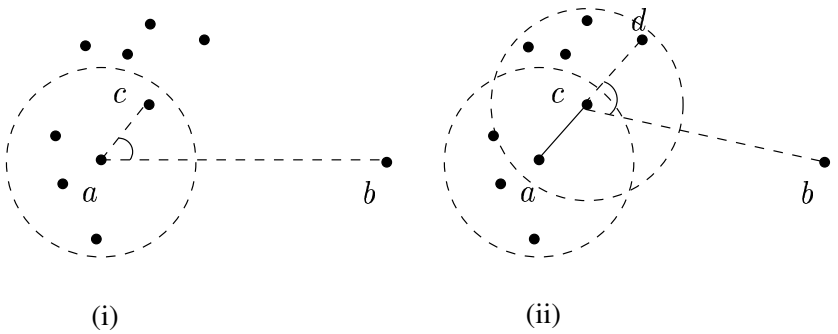


Figure 2.11: Compass routing (i) Node a selecting relay node c as $\angle cab$ is smallest. (ii)Node d selected as the next relay node since $\angle dcb$ is smallest.

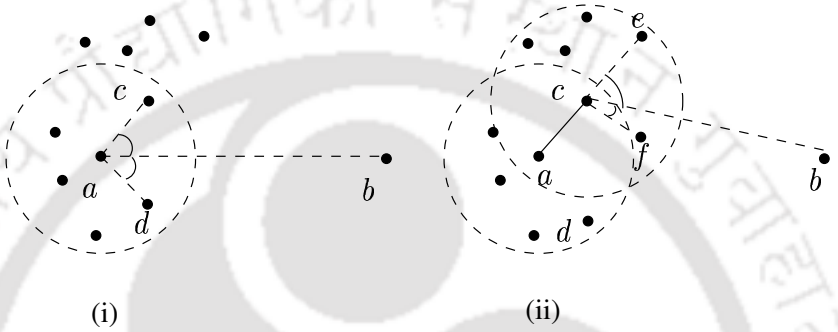


Figure 2.12: Random compass routing (i) Node a selects either node c or node d randomly as relay node. (ii) Node c selects either node e or node f randomly as the next relay node.

the Figure 2.11(ii). This process is repeated until the packet reaches the destination node.

3. **Random compass routing (RandCmp) [KSU99]:** Assume that the node a and b be the current and destination nodes respectively. For a packet traversal from node a to the destination node b, the current node a computes two nodes c and d, above and below the line $\overline{ab}$ respectively, as shown in the Figure 2.12(i). Here, the $\angle cab$ and $\angle dab$ are the minimum angles above and below the line $\overline{ab}$ respectively, among a's neighbors. The current node a sends the packet to the destination node by randomly selecting either node c or node d as the next relay node. In the next step, the current node c computes two nodes e and f as the next relay nodes and selects either of them as the relay node for transmitting the packet, as shown in the Figure 2.12(ii). This process is repeated until the packet reaches the destination node.

4. **Most forward routing (MFR) [TK84]:** As shown in the Figure 5.9(i), the current node a selects the next relay node c to transmit the packet to the destination node b

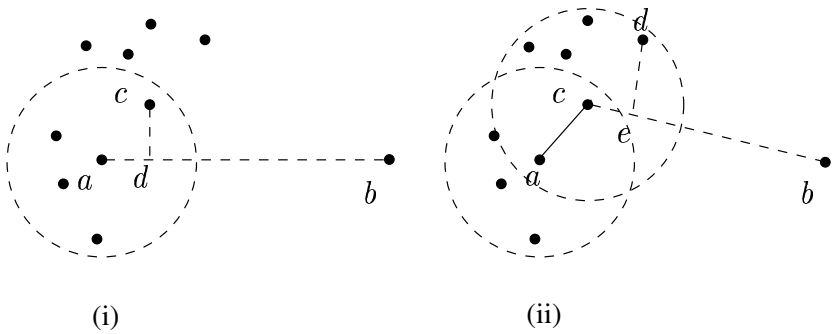


Figure 2.13: Most forward routing (i) Node a selects node c as $|\overline{db}|$ is smallest. (ii) Node c selects node d as $|\overline{eb}|$ is smallest.

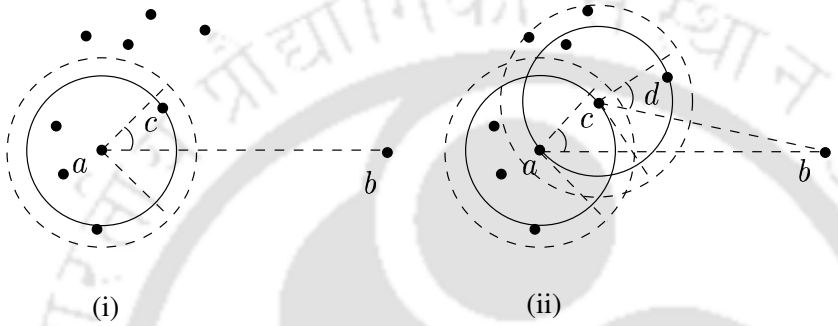
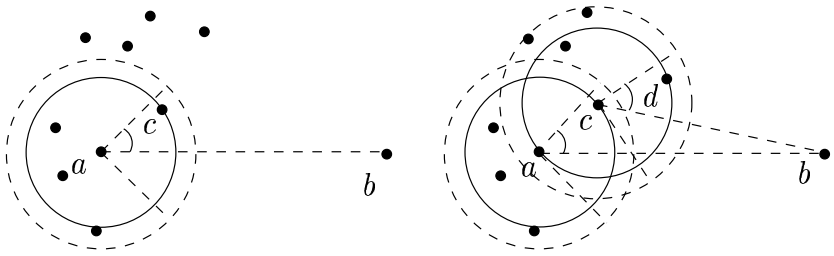


Figure 2.14: Nearest neighbor routing (i) Node a selects node c as the relay node. (ii) Node c selects node d as the next relay node.

such that the distance $|\overline{db}|$ is the smallest among a's neighbors where d is perpendicular projection point of node c to the line $\overline{ab}$. In the next step, as shown in the Figure 5.9(ii), the current node c forward the packet to the relay node d such that the distance $|\overline{eb}|$ is the least among c's 1-hop neighbors, where e is the d's perpendicular projection point to the line $\overline{cb}$. This process is repeated until the packet reaches the destination node.

5. **Nearest neighbor routing (NNR) [HL86]:** Let a and b be the current and destination nodes respectively, as shown in the Figure 5.10(i). For a particular angle α, the node a selects the next relay node c, such that $\angle cab \leq \alpha$ and c is the nearest neighbor of a. Similarly, in the next step, to forward the packet the current node c selects the next relay node d such that $\angle dcb \leq \alpha$ and d is the nearest neighbor of node c, see the Figure 5.10(ii). This process is repeated until the packet reaches the destination node.

6. **Farthest neighbor routing (FNR) [LCW02]:** In this routing protocol, for the given



(i) (ii)

Figure 2.15: Farthest neighbor routing (i) Node a selects node c as the relay node. (ii) Node c selects node d as the next relay node.

angle α, as shown in the Figure 5.11(i), the current node a selects the next relay node c such that $\angle cab \leq \alpha$ and c is the farthest neighbor of node a. Similarly, in the next step, to forward the packet, the current node c selects the next relay node d such that $\angle dcb \leq \alpha$ and d is the farthest neighbor of node c, as shown in the Figure 5.11(ii). This process is repeated until the packet reaches the node b.

The greedy routing, compass routing, and random compass routing provide guarantee delivery of packets if the underlying network topology is a Delaunay triangulation [KK00, BMSU01].

2.5 Geometric routing with Guaranteed delivery

The message delivery between hosts is an important and difficult problem in a MANET. In most of the greedy forwarding strategies, a source node first collects the location of the destination node using some location service and then forward the packet to a neighboring node which is closer to the destination. This process is repeated until the packet is delivered to the destination. This type of greedy heuristic does not guarantee the delivery of packets to the destination because the greedy forwarding strategies suffer from the problem of local maxima. In local maxima, a packet get stuck at a node which does not have a closer neighbor to the destination. Consider the Figure 2.16, the dotted circles represent the transmission ranges of nodes D and x. Even though the nodes u and v are neighbors of x, the node x does not find any better node to forward the packet to the destination. Hence, the node x is called local maxima.

Another problem of greedy heuristic is local loop. For example, consider the Figure 2.17. Let a and b be the source and destination nodes respectively. In compass routing, the packet is forwarded to the destination node b, through the nodes c, d, e, f, g, and h. In other words, the forwarded packet from the source node a is come back to the node a itself and never reaches the node b. Hence, this problem is called local loop. Some of the protocols based on greedy forwarding strategies, which suffer from these problems include geographic distance routing (GEDIR) [SL01], directional routing (DIR) [BCSW98], location aided routing (LAR) [KV99], Grdy [BMSU01], Cmp [BCSW98, KSU99], Rand-Cmp [KSU99], MFR [TK84], NNR [HL86], and FNR [LCW02].

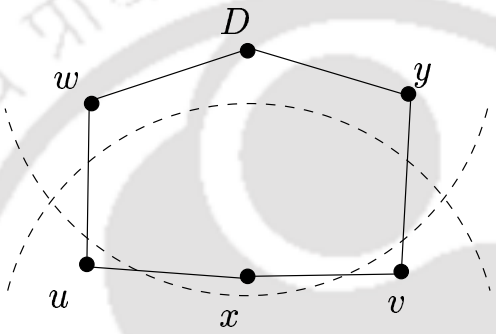


Figure 2.16: Local maxima.

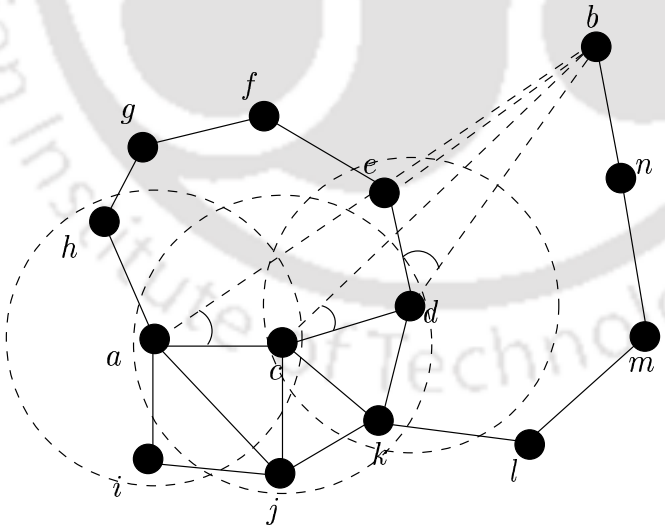


Figure 2.17: Local loop.

One way of solving these problems is by flooding. The flooding creates the problem of redundancy and heavy resource consumption. In contrast to this, some researchers have

used geometric spanner to overcome the local maxima and the local loop problems efficiently. For example, the GPSR [KK00] uses two spanners relative neighborhood graph (RNG) [BDEK02] and Gabriel graph (GG) [BDEK02, Epp96] as the underlying network topology. The GPSR forward the packet greedily toward the destination. If the packet forwarding faces local maxima problem then it uses right hand rule to forward the packet through the edges of the spanner. Once the local maxima problem is solved then it continues to send the packet greedily. Another famous geometric routing which guarantees the delivery of packet is FACE routing [BMSU01]. In FACE, the spanner GG is computed before the data transmission. The packet is forwarded through the faces of planar graph GG and moves toward the destination.

The routing protocols FACE, GPSR, AFR, GFG, and GOAFR$^+$ uses GG as the underlying network topology to generate delivery of packets, whereas GPSR can solve the local maxima problem using either RNG or GG.

2.6 Conclusions

In this chapter, we have studied various geometric spanners used in wireless networks. In addition, we have discussed several localized routing algorithms which can use these spanners as an underlying network graph. There are few routing algorithms which provide the guaranteed delivery of packets by utilizing the geometric spanners. Especially, the Delaunay triangulation based spanners are frequently used by many position based routing algorithms in wireless adhoc network. However, due to the neighborhood property of the Delaunay graphs, the smaller edges present in the network graph lead to larger hop count which in turn lead to high latency and low throughput. In the next chapter, we study the constrained based geometric graph which reduce the hop count between nodes in the network.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 3

Constrained Geometric Graphs

3.1 Introduction

Geometric spanners as an underlying network graph make various routing protocols efficient. In geometric spanners based on neighborhood relations, even though the nodes exist in the communication range of each other, they do not have a direct edge between them. Consider the PLDel shown in the Figure 3.1(i), the dotted circle indicates the transmission range of node a. Nodes b, c, d, and e are neighbor nodes of node a. Even though the node d is in the transmission range of node a, data packet traversal from node a to node d takes more than one hop. The edge $\overline{ad}$ cannot be placed in the PLDel due to the violation of Delaunay property. The similar case occurs in GG and RNG as shown in the Figure 3.1(ii) and the Figure 3.1(iii), respectively. They cannot include the edge $\overline{ac}$ in their respective graphs due to their geometric property. In other words, geometric spanners based on neighborhood relations restrict them to have longer edges, which increase the number of hops between source and destination, and in turn increases delay, jitter, and decreases throughput. But the multimedia, real time, and web applications cannot tolerate long delays caused by long path.

In this chapter, we consider the above problem and propose three new spanners called localized constrained Delaunay triangulation (LCDT), localized constrained relative neighborhood graph (LCRNG), and localized constrained Gabriel graph (LCGG) by introducing longer edges as constraint edges to reduce the number of hops and the delay between the nodes. Hereafter, we refer these graphs as contrained Delaunay triangulation

(CDT), constrained relative neighborhood graph (CRNG), and constrained Gabriel graph (CGG). In order to support our claim we have done the simulation using network simulator ($ns2.28$) [NS205]. The simulation results show that CDT, CRNG, and CGG give better performance than their counters PLDel, RNG, and GG, respectively. In addition to simulation work, we have proved some interesting properties of these spanners.

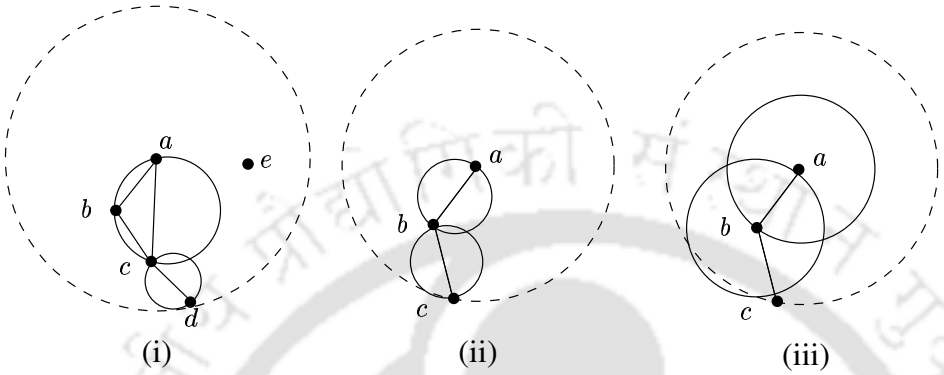


Figure 3.1: More hops in PLDel, GG, and RNG.

The remaining part of this chapter is organized as follows: The Section 3.2 describes the constrained based geometric graphs CDT, CRNG, and CGG, and their properties. The simulation work is presented in the Section 3.3. Finally, the Section 3.4 concludes the chapter.

3.2 Constrained Geometric spanners

The main idea is to add longer constraint edges to their basic spanner to reduce the number of hops between the nodes. For instance, if we have an edge between the nodes a and d in the Figure 3.1(i), a and c in the Figure 3.1(ii), or a and c in the Figure 3.1(iii), it reduces the number of hops between the nodes and in turn leads to many other benefits such as reduction in the delay, jitter, and energy consumption. But, these edges may violate the geometric properties of their respective spanners. Other edges in the spanners satisfy the geometric properties. This gives improvement in the hop count and at the same time satisfying various geometric properties by non-constraint edges.

We have considered the system model for constrained geometric graphs as an asynchronous wireless ad hoc network. In this network, each node has unique identification

and all nodes are aware of their location information through some positioning technique. All the nodes in the network have omni-directional antenna with same transmission range and the network contains symmetric communication links. The network has reliable communication, that is, if a node transmits a packet then all the nodes in the vicinity receive the same packet.

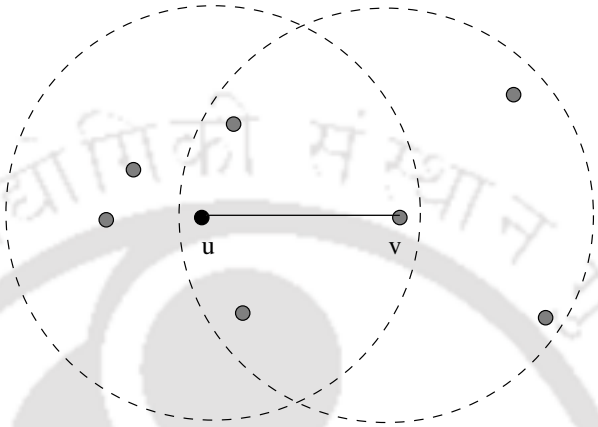


Figure 3.2: Non intersecting constraint edge computed by first method.

3.2.1 Constraint Edges

Placing non intersecting constraint edges in a network graph is an interesting problem. It can be done in many different ways. We have proposed two different methods. In the first algorithm, we set the $color$ of all the nodes to $WHITE$ and change the $color$ to $BLACK$ if the node contains least id among its *1-hop* neighbors. Each $BLACK$ node places a constraint edge with the longest distance node among its *1-hop* neighbors. To ensure non-intersection with other constraint edges, mark all $WHITE$ nodes to $BROWN$, which are 1-hop neighbor of u and v when $\overline{uv}$ is the constraint edge, as shown in Figure 3.2.

The formal algorithm is given in Algorithm *Constraint edge1*. This algorithm uses the following messages: The message *Hello_Packet* that carries the id and position information, is used for collecting the 1-hop neighborhood information. The messages *proposal* and *accept* contain two node ids. The *makegray* packet does not carry any information.

Algorithm: Constraint edge1

1. Each node u, sets its $color$ to $WHITE$, broadcast its id and position information, and listens to the messages from other nodes to collect 1-hop neighbors $n_1(u)$.

2. Each node u marks its color to $BLACK$ if it is a least id node among its 1-hop neighbors and finds the longest edge with the node $v \in n_1(u)$ as the constraint edge and broadcast this information using the message ***proposal(u,v)***.

3. $WHITE$ node v receiving the message ***proposal(u,v)***, sets its $color$ to $BROWN$ and broadcast the message ***accept(u,v)***.

4. $BLACK$ node u confirms the constraint edge after receiving the message ***accept(u,v)*** and broadcast the ***makegray*** packet.

5. Each $WHITE$ receiving the packets ***makegray*** or ***accept(u,v)***, marks its $color$ to $BROWN$.

6. If the $BLACK$ node u does not receive ***accept(u,v)*** message, it chooses the next longest edge from $n_1(u)$.

This localized algorithm places the non intersecting constraint edges using *1-hop* information. But this algorithm introduces a small number of constraint edges. If the graph contains large number of constraint edges then the average hop count becomes smaller. Hence, we have introduced the algorithm, which uses *2-hop* information.

In this algorithm, we initially set the $color$ of all nodes to $WHITE$ and change the $color$ to $BLACK$ if the node has least id among its *2-hop* neighbors. These $BLACK$ nodes select the longer edges as constraint edges with the nodes in *1-hop* neighbor list, by making sure that the angle between them must be greater than or equal to $60°$. The minimum angle between constraint edges ensures longer edges around each black node as shown in the Figure 3.3. The formal algorithm is given in Algorithm *Constraint edge2*. Unlike in Algorithm *Constraint edge1*, this algorithm does not require any mechanism to avoid the intersection of constraint edges, as the $BLACK$ nodes are at least three hops

away. The minimum angle between the constraint edges, belonging to the same $BLACK$ node, can be user defined.

The algorithm Constraint edge2 uses the following control messages: Initially, the message *Hello_Packet* is broadcased for collecting 1-hop neighborhood information. The message *Neighbor_Packet* which carries a list of node ids, is used for collecting 2-hop information. The message *makebrown_Packet* that carries a list of node ids, is used for marking the list to brown color.

Algorithm: Constraint edge2

---

1. Each node u collects the 1-hop neighborhood information in $n_1(u)$ and marks its $color$ to $WHITE$.

2. Each node u broadcasts $n_1(u)$ and collects the 2-hop neighbor information.

3. Nodes having lowest id among its 2-hop neighbors, set their $color$ to $BLACK$.

4. Each BLACK node chooses a set N of nodes from its 1-hop neighbors using the following method, see Figure 3.11.

   n_1 = farthest neighbor

   N = $\{n_1\}$

   for i = 2, 3, . . .

   n_i = choose i^{th} farthest neighbor

   if n_i makes the angle 60° or more with all nodes in N

   then $N = N \cup n_i$.

5. Each $BLACK$ node adds the constraint edges to the nodes in N and mark the $color$ of a node to $BROWN$ if the node is the other side of constraint edge.

In ideal conditions, the long constraint edges in the network do not affect the link quality, as per the UDG model. That is, every node in the vicinity of another node receives the transmitted packet [KZ03, GGH+05, LCWW03, GGH+01, ALW+03, LCW02, WL02, LSW04]. But in reality, the quality of signal strength decreases with the increase in distance to the receiving node [ZK07, CWPE05, Rap01]. The constraint edges may have poor link

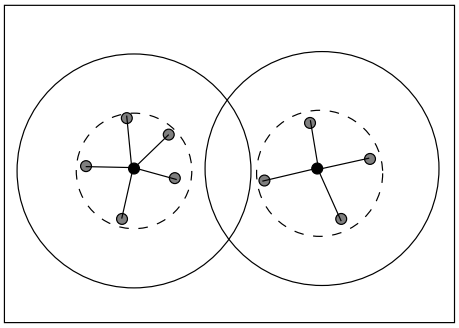


Figure 3.3: Constraint edges computed using 2-hop information in the third method.

quality. In addition to the distance, link quality also depends on the surroundings such as walls, buildings, mountains, and weather conditions. So, one can consider the link quality also as a parameter along with the distance to choose constraint edges.

Properties of Constraint Edges

The set of constraint edges computed by this method has interesting properties on their distribution and density. These are discussed in the next subsection.

Lemma 3.2.1 *The maximum number of constraint edges incident at a $BLACK$ node is six.*

Proof: At any $BLACK$ node, the minimum angle between two adjacent constraint edges is $60°$. The maximum number of constraint edges are six, since the total angle at any point is $360°$. ∎

Lemma 3.2.2 *The upper bound on the minimum number of constraint edges at $BLACK$ node is four.*

Proof: The minimum angle between two adjacent constraint edges is $60°$. The minimum number of constraint edges can be obtained by keeping the maximum angle less than $120°$ between the constraint edges as shown in the Figure 3.4. That is four constraint edges. ∎

The minimum number of hops between any two BLACK nodes can be three as shown in the Figure 3.5. But, it cannot be less as shown in the lemma below.

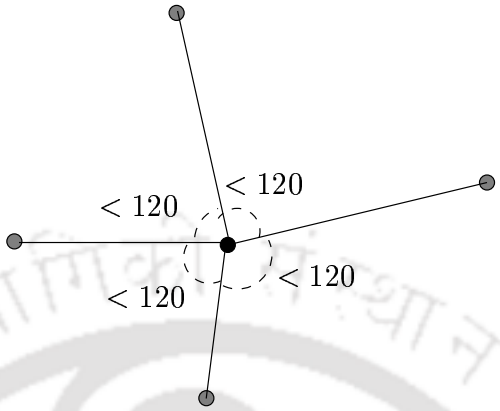


Figure 3.4: Minimum Number of constraint edges.

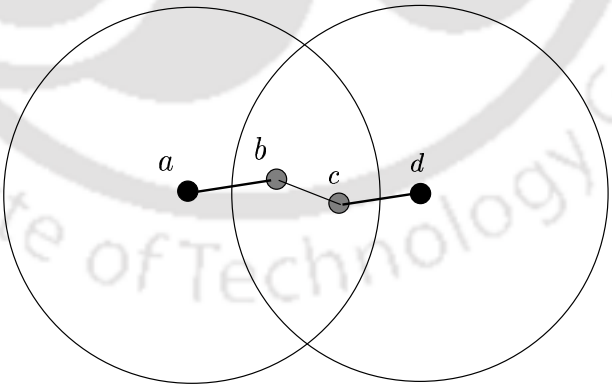


Figure 3.5: Constraint edges $\overline{ab}$ and $\overline{cd}$ do not intersect.

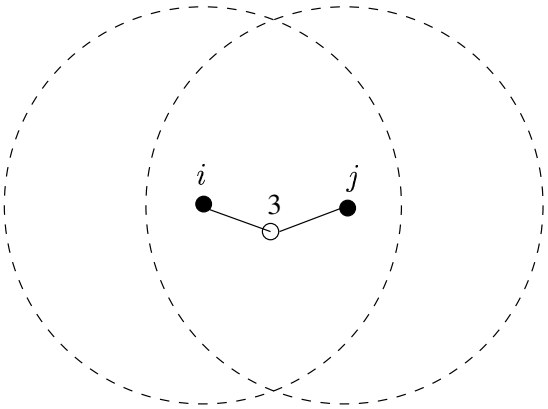


Figure 3.6: Minimum number of hops between $BLACK$ nodes.

Lemma 3.2.3 *The minimum number of hops between any two $BLACK$ nodes is not less than three.*

Proof: We prove the statement by *contradiction*. Let us assume that the minimum number of hops between two $BLACK$ nodes is less than three hops. That means it can be either one or two hops.

Let us consider the network graph shown in the Figure 3.6, where nodes i and j are $BLACK$ nodes. From the definition of $BLACK$ node, each $BLACK$ node has least *id* among its *2-hop* neighbors. That is, node i can not be a smallest *id* node in its *2-hop* neighbor, because node j is in its *2-hop* neighborhood. Similarly, node j cannot be a smallest *id* node in its *2-hop* neighbor, because node i is in its *2-hop* neighborhood. Similarly, we can argue for *1-hop* also. Hence proved. ∎

Lemma 3.2.4 *The maximum number of constraint edges incident to a $BROWN$ node is one.*

Proof: According to the algorithm, only $BLACK$ nodes initiate the constraint edges. So, one end of the constraint edge is $BLACK$ node and the other end is $BROWN$ node, as shown in the Figure 3.7.

The $BROWN$ node's degree becomes greater than 1 if it has another constraint edge of another $BLACK$ node. This means, two $BLACK$ nodes are two hop apart, which contradicts the $Lemma$ 3.2.3. ∎

From the Algorithm *Constraint edge2*, one can make the following observation.

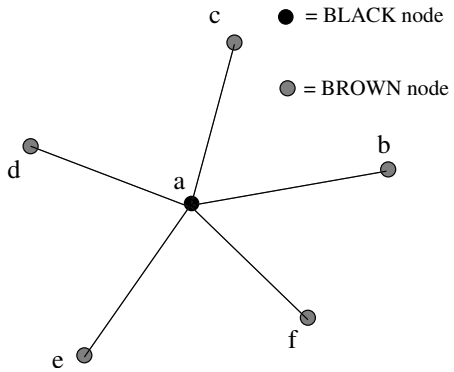


Figure 3.7: Constraint edges at $BROWN$ nodes.

Observation 3.2.5 *No $WHITE$ node contains constraint edges.*

Lemma 3.2.6 *The minimum and maximum numbers of constraint edges at any node in constraint geometric spanners are zero and six, respectively.*

Proof: From $Lemma$ 3.2.1, $Lemma$ 3.2.4, and $Observation$ 3.2.5, we can conclude that the maximum and minimum number of constraint edges at any node in the graph are six and zero, respectively. ∎

Lemma 3.2.7 *The minimum number of hops between two constraint edges incident on different $BLACK$ nodes is one.*

Proof: Let us assume the contradiction that the minimum number of hops between two constraint edges of two different $BLACK$ nodes is zero. That is, these constraint edges are joining at a single node, which must be $BROWN$. So, the degree of $BROWN$ node becomes more than one, which contradicts the $Lemma$ 3.2.4. ∎

Lemma 3.2.8 *No two constraint edges intersect.*

Proof: Any edge incident on a $BLACK$ node is within the 1-unit circle centered at that $BLACK$ node. The 1-unit circles centered at two different $BLACK$ nodes are disjoint, since the minimum number of hops between any two $BLACK$ nodes is at least three by the $Lemma$ 3.2.3, see the Figure 3.8. ∎

Lemma 3.2.9 *The number of $BLACK$ nodes in a circle of k-units radius is bounded by a constant for any constant k.*

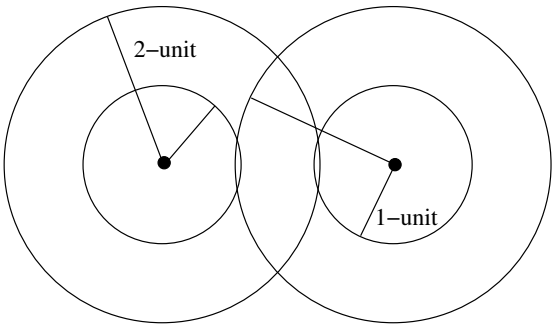


Figure 3.8: Constraint edges do not intersect, because minimum hops between any two BLACK nodes is three.

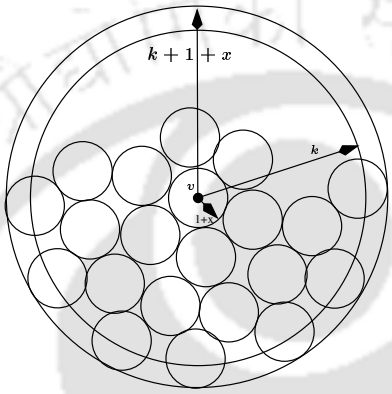


Figure 3.9: The $(1+x)$-units circles packed in $(k+1+x)$-units circle.

Proof: Two $BLACK$ nodes with $(1+x)$ units radius circles are disjoint, where $0 < x \leq 0.5$. The maximum number of $BLACK$ nodes that can be placed in a k-units radius circle can be obtained by placing $(1+x)$-units disjoint circles in a $(k+1+x)$-units radius circle, as shown in the Figure 3.9. Maximum number of disjoint $(1+x)$ units circles that can be packed in $(k+1+x)$ units circle are:

$\leq \frac{\pi(k+1+x)^2}{\pi(1+x)^2}$

$= O(k^2) = O(1)$, for any constant k. ∎

Lemma 3.2.10 *The number of $BROWN$ nodes in a k-unit radius circle is bounded by a constant for any constant k.*

Proof: The maximum number of $BLACK$ nodes that can be packed in a k-units radius is constant, from $Lemma$ 3.2.9. Each $BLACK$ node contains maximum of six $BROWN$ nodes, as per the $Lemma$ 3.2.1. Thus, the total number of $BROWN$ nodes that can be packed in the k-units circle becomes constant. Hence proved. ∎

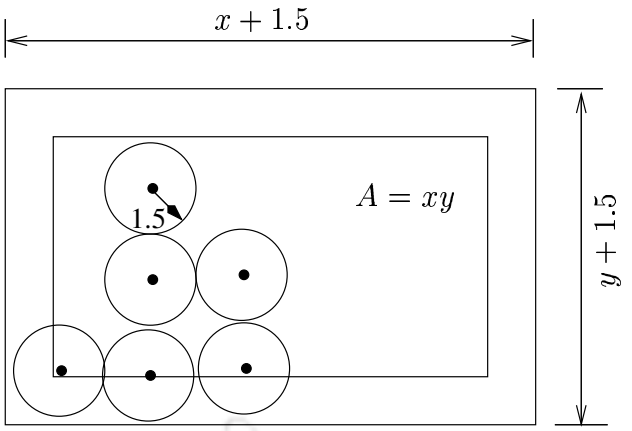


Figure 3.10: $BLACK$ nodes packed in an area A.

Lemma 3.2.11 *The total number of $BLACK$ nodes is in $O(n)$ for uniformly distributed nodes in an area A.*

Proof: Let c be the expected number of nodes in an unit area, where the nodes are deployed uniformly.

Thus, the expected number of nodes in the area A is $A * c = n$. That is, $A = \frac{n}{c}$.

The expected number of $BLACK$ nodes in the area A is obtained by packing the 1.5-unit radius circles in $A + D$ area, as shown in the Figure 3.10. Here D is the area, which has been added to A by increasing the length and breadth by 1.5 units. Thus, the expected number of $BLACK$ nodes in the area A becomes $\frac{A+D}{\pi * 1.5^2}$.

$$= \frac{\frac{n}{c}+D}{\pi * 1.5^2}$$
$$\in O(n)$$

Hence the lemma is proved. ∎

Theorem 1 *The total number of constraint edges in the constrained geometric graph is in $O(n)$.*

Proof: The total number of $BLACK$ nodes in the graph is in $O(n)$, from $Lemma$ 3.2.11. For each $BLACK$ node, there can be maximum of six constraint edges, by $Lemma$3.2.1. Thus, the total number of constraint edges in the graph is in $O(n)$. ∎

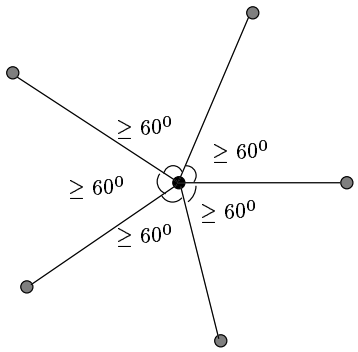


Figure 3.11: Constraint edges selection.

3.2.2 Constrained Delaunay triangulation (CDT)

Planarized localized Delaunay triangulation (PLDel) [LCWW03, LCW02] is a planar geometric spanner with the spanning ratio of 2.5 [LCWW03, LCW02], which is independent of the number of nodes in the graph. However, its hop stretch factor can be in $O(n)$. This is because, there may not be a direct edge between a pair of nodes even though they are in the communication range of each other. In this section, we propose a constrained Delaunay triangulation (CDT), which reduces the hop stretch factor by introducing long constraint edges.

The CDT of a set of nodes containing prespecified constraint edges should be as close as possible to the Delaunay triangulation [Che87, Lee97, CR90, CW99, She00]. The CDT is a well studied problem in the area of computational geometry. Paul Chew proposed a divide-and-conquer algorithm for CDT [Che87]. This algorithm is not suitable for wireless adhoc networks, because it is a centralized algorithm. In this section, we propose a localized algorithm to construct CDT for wireless networks. CDT has the following properties. (1) All the edges in CDT have the length not more than one unit. (2) CDT is a planar graph. (3) CDT is a spanner of UDG.

The spanner CDT construction is broadly divided into three parts. First, we construct the planarized local Delaunay triangulation (PLDel) [LCWW03, LCW02] with one hop neighborhood information. In the second part of the algorithm, we place non-intersecting constraint edges on top of PLDel using the Algorithm *Constraint edge2* discussed in the previous section. The third part planarizes the graph by deleting the edges crossing the constraint edges and adding the triangles around the constraint edges.

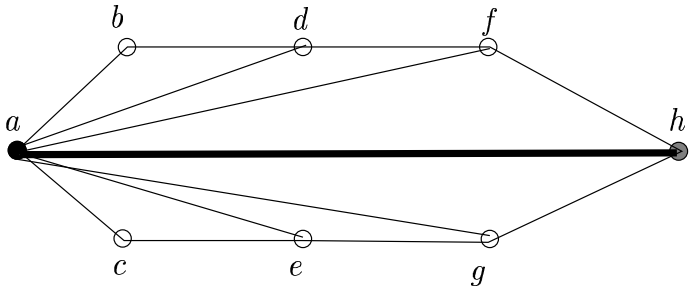


Figure 3.12: $\triangle adb$, $\triangle afd$, $\triangle ahf$, $\triangle aec$, $\triangle age$, and $\triangle ahg$ are created in CDT.

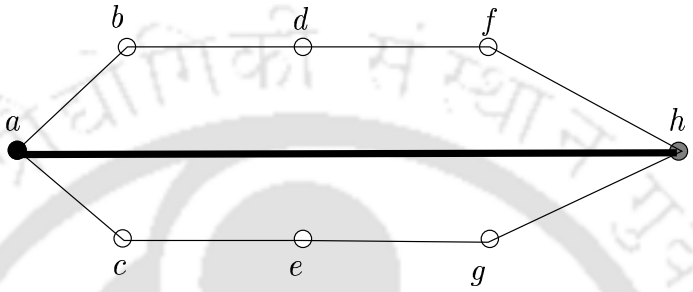


Figure 3.13: Convex polygon after removing edge crossing.

For the sake of completeness, we briefly describe the PLDel algorithm. Each node u computes Delaunay triangles $Del(u)$ with its 1-hop neighbors. The Gabriel edges are computed, which will never be deleted from PLDel. An edge $\overline{ab}$ is called Gabriel edge if it is a common edge for two Delaunay triangles $\triangle abc$, $\triangle abd$, and the angles $\angle acb$ and $\angle adb$ must be less than 90° (see the Figure 3.14). A triangle $\triangle abc$ is called consistent if the nodes a, b, and c have computed the triangle $\triangle abc$ as Delaunay triangle. Remove all inconsistent triangles. The resulting graph is called LDel^1, which may not be a planar graph. A separate planarization algorithm is applied on LDel^1 to induce a planarized local Delaunay triangulation (PLDel).

Compute the non-intersecting constraint edges using the Algorithm *Constraint edge2*. The graph becomes non-planar as shown in the Figure 3.15. Nodes at either ends of constraint edges are colored either $BLACK$ or $BROWN$ by the Algorithm *Constraint edge2*. These nodes broadcast the constraint edges incident on them using a ***Constraint_Packet***. After receiving the ***Constraint_Packet***, all non-$BLACK$ nodes identifies edges which are crossing the constraint edges and remove them. The polygons formed on either side of a constraint edge can be either convex or simple polygons as shown the Figure 3.13 or the Figure 3.16, respectively. If these polygons are convex, they can be triangulated by adding

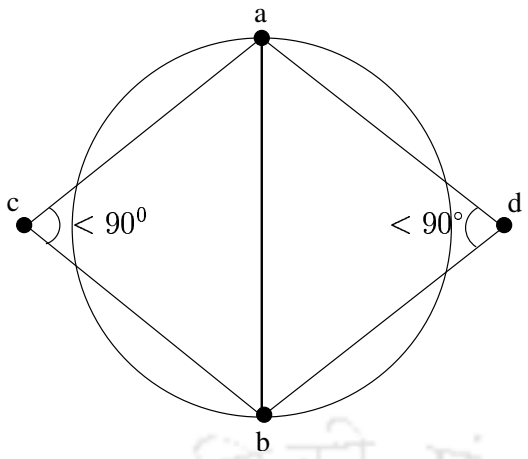


Figure 3.14: Gabriel edges.

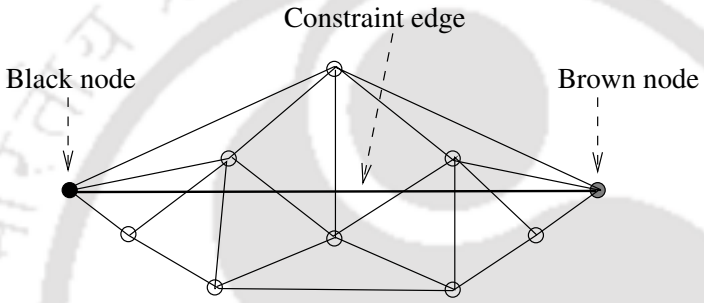


Figure 3.15: Constraint edge is added to PLDel.

edges from each vertex on the polygon to any one end of the constraint edge, as shown in the Figure 3.12. These edges do not satisfy the Delaunay property. If the polygon formed is not convex, the triangulation is done using the *Triangulation* algorithm. In this algorithm, the triangles are computed using Delaunay triangulation of two different set of nodes. The complete description is given in the next page. The resulting graph is called constrained Delaunay triangulation (CDT). The formal description of the algorithm is given below.

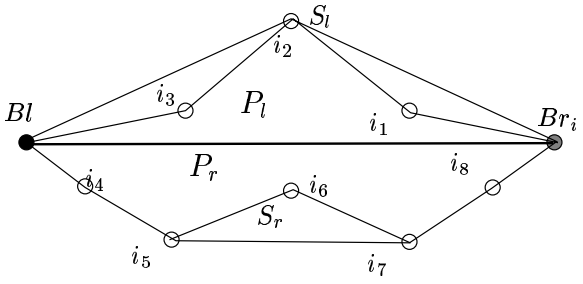


Figure 3.16: Constraint polygon, after removing edge crossing.

Algorithm: Constrained Delaunay Triangulation

1. Construct PLDel [LCWW03, LCW02] from the given set of nodes.
2. Place the non-intersecting constraint edges using Algorithm *Contraint edge2*.
3. BLACK and BROWN nodes broadcast the constraint edge information using the message ***Constraint_Packet***.
4. Each WHITE and BROWN nodes remove edges which cross constraint edges, see the Figure 3.16. This information is broadcasted using ***Edgecross_Packet***, which contain edges deleted from it and the list of undeleted edges.
5. Each $BLACK$ node broadcast the packet ***Polygon_Packet***, which contain the information received in ***Edgecross_Packet***.
6. After receiving the ***Polygon_Packet*** from $BLACK$ node Bl, each $BROWN$ node Br_i triangulate using the algorithm *Triangulation*.

The *Triangulation* algorithm is responsible for triangulating the polygons, which were created during the edge removal. Each BROWN node computes the polygon through the information obtained by the packets ***Polygon_Packet*** and ***Edgecross_Packet***. The polygon is computed as follows: Starting with the BROWN nodes, Br_i in the Figure 3.17, select an edge, $\overline{Br_i i_1}$, which makes least clockwise angle with the constraint edge $Br_i Bl$. Now the node i_1, which is the other side of the selected edge, selects an edge, $\overline{i_1 i_2}$ which makes least clockwise angle with the previously selected edge $\overline{Br_i i_1}$. Similarly, the nodes i_2, i_3, $\cdots$ i_8 select the edges which make the least clockwise angle with the edges $\overline{i_1 i_2}$, $\overline{i_2 i_3}$, $\cdots$ $\overline{i_7 i_8}$, respectively. This process is continued till it reaches the BROWN node. The set of nodes S and edges traversed in this path forms the polygon.

The set S of nodes is divided into two subsets S_l and S_r which forms two polygons called left polygon P_l and right polygon P_r, respectively, on the left and right sides of the constraint edge $\overline{Bl Br_i}$. All the vertices left (right) of $\overline{Bl Br_i}$ belongs to left (right) polygon P_l (P_r). Whether a node u is left (right) of an edge $Bl Br_i$ can be found by the determinant:

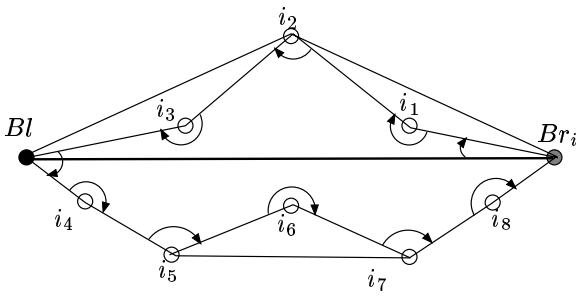


Figure 3.17: The polygon formation.

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

where (x_1, y_1), (x_2, y_2), and (x_3, y_3) are coordinates of the nodes Bl, Br_i, and u, respectively. Node u is on left (right) if the determinant is negative (positive). If the determinant value is zero, then add the node u to both S_l and S_r. The Delaunay triangulation is computed, individually, with the sets S_l and S_r, and remove the edges which are outside the polygons and the edges containing euclidean distance more than one unit.

Algorithm: Triangulation

1. Merge the information received, from $BLACK$ node Bl, in the ***Polygon_Packet*** with the received ***Edgecross_Packet***.

2. Identifies the set S of nodes, which deleted the edges crossed by the constraint edge $\overline{BlBr_i}$. S=$i_1, i_2, i_3, \cdots i_8$ for the network given in the Figure 3.16.

3. Divide the S into S_l and S_r, which are on left and right side of the constraint edge $\overline{BlBr_i}$ respectively.

4. Find the Delaunay triangulation DT of S_l and S_r using any centralized algorithm.

5. Identify the polygons P_l and P_r formed by the vertices S_l and S_r respectively. For the Figure 3.16

   $P_l = Br_i, i_1, i_2, i_3, Bl, Br_i$

   $P_r = Br_i, Bl, i_4, i_5, i_6, i_7, i_8, Br_i$

6. Remove edges from DT which are outside the P_l and P_r, and also the edges not part of UDG.

7. Each $BROWN$ node broadcast the edges of DT using the packet ***Delaunay_Packet***.

8. Each $BLACK$ node, after receiving the packet ***Delaunay_Packet***, rebroadcasts the same again.

9. Each node, after receiving the packet ***Delaunay_Packet***, updates their edge list.

Properties of CDT

This section proves some interesting results on CDT.

Lemma 3.2.12 *CDT is a planar graph*

Proof: The CDT is the combination of PLDel and the triangles formed within the constraint polygons. Since the PLDel is a planar graph [LCWW03], the non planarity must be involved within the constraint polygon. In the *Triangulation* algorithm, each $BROWN$ node computes the Delaunay triangulation of the constraint polygon. Since the Delaunay triangulation is planar graph, CDT is a planar graph. ∎

We analyze the message complexity for CDT. In addition to the messages of planarized local Delaunay triangulation [LCWW03, LCW02], CDT uses the following control messages:

1. ***Neighbor_Packet***: This packet is used to broadcast the 1-hop neighborhood information.

2. ***Constraint_Packet***: This packet is used to broadcast the constraint edge information.

3. ***Edgecross_Packet***: This packet is used to broadcast the edge cross information.

4. ***Polygon_Packet***: This packet is used to broadcast the edge cross information by $BLACK$ nodes.

5. ***Delaunay_Packet***: This packet is used to broadcast the Delaunay triangles.

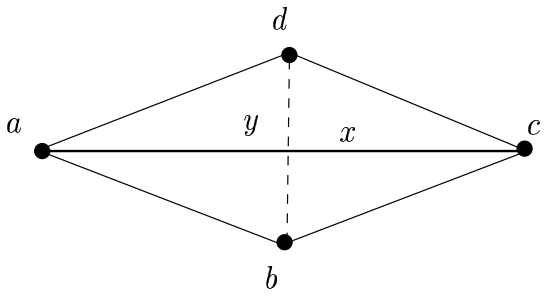


Figure 3.18: Length stretch factor.

The message ***Neighbor_Packet*** is broadcasted at most once by each node in the network graph. So, the message complexity for ***Neighbor_Packet*** is in $O(n)$, where n is the total number of nodes. Only $BLACK$ and $BROWN$ nodes use the message ***Constraint_Packet***. Each $BLACK$ and $BROWN$ node broadcast the message ***Constraint_Packet*** at most once. The message complexity for the broadcast of ***Constraint_Packet*** is in $O(n)$. The message ***Edgecross_Packet*** is used by only $WHITE$ and $BROWN$ nodes. Each $WHITE$ and $BROWN$ node uses ***Edgecross_Packet*** at most once in CDT. So the message complexity for this packet is in $O(n)$. Only $BLACK$ nodes use the ***Polygon_Packet*** and the message complexity is in $O(n)$. Each $BROWN$ node use the packet ***Delaunay_Packet*** once and each $BLACK$ node uses at most six times. So, the message complexity for this packet is in $O(n)$. The following theorem follows, since the message complexity of PLDel is in $O(n)$.

Theorem 2 *The message complexity to construct CDT is in* $O(n)$.

The length stretch factor of CDT can be very high. Consider an example network given in the Figure 3.18, where the edge $\overline{ac}$ is a constraint edge and edge $\overline{bd}$ is deleted due to the edge cross with the constraint edge. In CDT, the distance between the nodes b and d is *min(*$|\overline{ba}| + |\overline{ad}|$, $|\overline{bc}| + |\overline{cd}|$*)* and can be very high compared to $\overline{bd}$. Similarly, the number hops between b and d can be very high in CDT, as shown in the Figure 3.19. Hence, the spanning ratio is also very large. However, these are extreme cases, as it does not happen frequently in real environments.

3.2.3 Constrained relative neighborhood graph (CRNG)

The construction of CRNG is slightly different from CDT, which can be divided into three phases. In the first phase, the relative neighborhood graph (RNG) is constructed from the

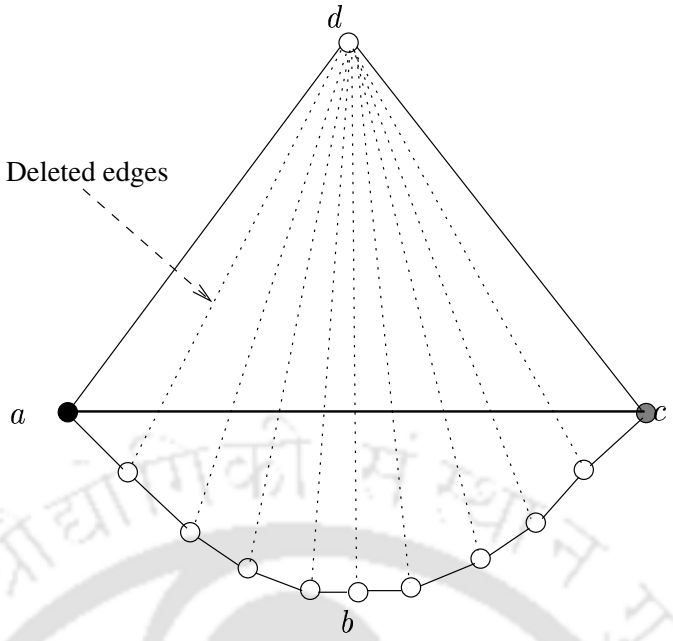


Figure 3.19: Hop stretch factor.

given set of nodes. The construction of RNG is as follows: A node u in the graph computes the lune with a node $v \in n_1(u)$ and tests whether any node $w \in n_1(u)$ exists inside the lune or not. The node u places the edge $\overline{uv}$ if the lune does not contain any other node. This process is continued for all the nodes in the graph and the resulting graph is RNG.

The second phase of the algorithm places the constraint edges using the Algorithm *Constraint edge2*. In the third phase, we remove the edges which cross the constraint edges. The resulting graph is called constrained relative neighborhood graph (CRNG). The formal description of the algorithm *CRNG* is given below.

Algorithm: CRNG

---

1. Broadcast ***hello_packet*** with ID and location information and collect 1-hop information $n_1(u)$.

2. Each node u follow the steps to find RNG edges

   for each $a_i \in n_1(u)$ do

   if $lune(u, a_i)$ does not contain any other node $a_j \in n_1(u)$

   add edge $\overline{ua_i}$ to CRNG.

3. Add the constraint edges as per the Algorithm *Constraint edge2*.

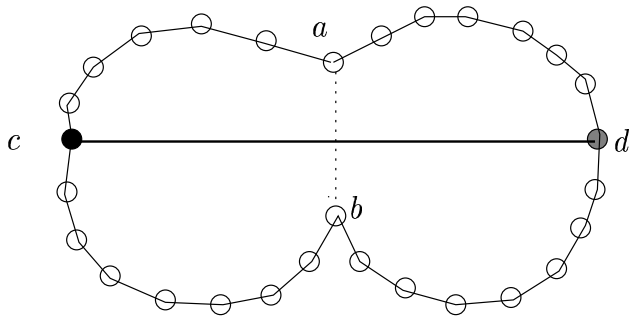


Figure 3.20: Spanning ratio of CRNG.

4. BLACK and BROWN nodes broadcast the constraint edge information using the message ***Constraint_Packet***.

5. Each $WHITE$ and $BROWN$ node delete all the edges, which cross constraint edges.

Properties of CRNG

Some properties of CRNG are discussed in this section.

Theorem 3 *The CRNG is a planar graph.*

Proof: First phase of the algorithm constructs the RNG, which is a planar graph [LCW02]. The resulting graph may become non-planar after adding the constraint edges in the *step 3*. But, the *step 5* assures planarity by removing the RNG edges, which cross the constraint edges, refer the Figure 3.20. ∎

Next, we analyze the message complexity of CRNG. The following messages are used for constructing CRNG.

1. ***Hello_Packet***: This packet is used to collect the neighbor list information.

2. ***Neighbor_Packet***: This packet is used to broadcast the 1-hop neighborhood information.

3. ***Constraint_Packet***: This packet is used to broadcast the constraint edge information.

4. ***Edgecross_Packet***: This packet is used to broadcast the edge cross information.

The message complexity for ***Hello_Packet*** is in $O(n)$. The packet ***Neighbor_Packet*** is broadcasted at most once by each node and the complexity becomes $O(n)$. The message ***Constraint_Packet*** is broadcasted by only BLACK and BROWN nodes at most once. Thus the message complexity of ***Constraint_Packet*** is in $O(n)$. Only the WHITE and BROWN nodes broadcast ***Edgecross_Packet*** at most once. Thus, the message complexity of ***Edgecross_Packet*** is in $O(n)$. The Theorem 4 follows, since the message complexity of RNG construction is in $O(n)$.

Theorem 4 *The message complexity for constructing CRNG is in* $O(n)$.

3.2.4 Constrained Gabriel graph (CGG)

The second and third phases in the construction of CGG are same as that of CRNG. But in the first phase of CGG, the Gabriel graph is constructed. The graph GG is constructed as follows. Each node u in the graph tests whether any node w exists in the circle with the diameter $|\overline{uv}|$, where $v, w \in n_1(u)$. The node u places the edge $\overline{uv}$ if the circle does not contain any other node. This process is continued for all the nodes and their neighbor lists, and the resulting graph is GG.

Algorithm: CGG

1. Broadcast ***hello_packet*** with ID and location information and collect 1-hop information in $n_1(u)$.

2. Each node u follows the steps to find GG edges

   for each $a_i \in n_1(u)$ do

   if $disk(u, a_i)$ does not contain any other node $a_j \in n_1(u)$

   add edge $\overline{ua_i}$ to CGG

3. Add the constraint edges as per the Algorithm *Constraint edge2*.

4. BLACK and BROWN nodes broadcast the constraint edge information using the message ***Constraint_Packet***.

5. Each $WHITE$ and $BROWN$ node delete all the edges, which cross constraint edge.

Properties of CGG

Theorem 5 *The CGG is a planar graph.*

Proof: We know that GG is planar graph [LCW02]. The graph may become non planar after adding the constraint edges. But, the WHITE and BROWN nodes remove their edges, which cross the constraint edge. Thus the graph becomes planar. ∎

The CGG uses four control messages ***Hello_Packet***, ***Neighbor_Packet***, ***Constraint_Packet***, and ***Edgecross_Packet*** in its graph construction. The purposes of these control messages are same as that of the control messages in CRNG. The number of messages broadcasted in CGG by these control packets is same as that of CRNG. Hence follows the theorem.

Theorem 6 *The message complexity for constructing CGG is in* $O(n)$.

Lemma 3.2.13 *CRNG* $\subseteq$ *CGG* $\subseteq$ *CDT.*

Proof: Let an edge $\overline{uv} \in RNG$. If it does not cross any constraint edge $\overline{pq}$, then the edge $\overline{uv} \in CRNG$. Since $RNG \subseteq GG$, the edge $\overline{uv}$ belongs to GG. The edge $\overline{uv}$ belongs to CGG, as it does not cross constraint edge $\overline{pq}$. If the edge $\overline{uv} \in RNG$ and cross the constraint edge $\overline{pq}$ then the edge $\overline{uv}$ will be removed in both the graphs CRNG and CGG. Thus, CRNG $\subseteq$ CGG. Similarly, one can show CGG $\subseteq$ CDT. Hence proved. ∎

3.3 Simulation and performance

The simulation work has been carried out using network simulator (ns2.28) [NS205]. In order to evaluate the performances of the spanners CDT, CRNG, and CGG, we have done several simulation experiments and the results are compared with the spanners PLDel, LDel[1], GG, and RNG. In addition to the seven spanners CDT, CRNG, CGG, RNG, GG, LDel, and PLDel, we have simulated six routing protocols, *Grdy*, *Cmp*, *RandCmp*, *MFR*, *NNR*, and *FNR*, to run on these spanners. Please refer to the Section 2.4 for the details of these protocols.

3.3.1 Performance metrics

We have considered the following metrics to evaluate the performance of proposed spanners CDT, CRNG, and CGG. In the simulation results, we have presented average, minimum, and maximum of these metrics.

1. **Hop count:** This metric represent the total number of hops travelled by a data packet from source to destination on the underlying network graph.

2. **Delivery ratio:** The delivery ratio (DR) is the ratio of the total number of packets received successfully to the total packets sent by the source.

3. **Delay:** Delay metric is the total time spent by a packet during the data transmission from source to destination.

4. **Delay variance:** Delay variance is the variance in the delay of a packet from source to destination. The following mathematical formula is used to calculate the delay variance (S_d).

$$S_d = \sqrt{\frac{1}{m} \times \sum_{i=1}^{m} (a_i - \overline{a})}$$

   where a_i denotes delay of i^{th} packet, $\overline{a}$ is average delay, and m is total number of packets.

5. **Throughput:** Throughput is the total number of packets received per unit time by the destination.

3.3.2 Simulation results

All experiments are conducted on five different node scenarios, each containing 100 nodes distributed randomly in a $200 \times 200m^2$ area. The transmission range of each node is 40 m. For each scenario, ten different connection patterns are chosen by randomly selecting the source nodes at one side of grid and the destination nodes at the other side. Data is sent at constant rate. The rate of data transfer is varied by changing the time interval between data packets called constant bit rate (CBR) interval in *ns2.28*.

	Grdy	Cmp	Rcmp	MFR	NNR	FNR
CDT	98.23	100	100	98	82.8	82.6
CGG	95.5	100	100	95.8	78.4	80.4
CRNG	85.8	89.95	89.7	82.12	49.2	57.2
LDel[1]	98	100	100	97.6	88.8	81.6
PLDel	98	100	100	98	75.2	69.2
GG	91.62	100	100	93.4	71.2	69.2
RNG	74	86	85.6	73	34.8	32.6

Table 3.1: Delivery ratio.

The first experiment is conducted to evaluate the delivery ratio of different spanners. The average delivery ratios with six different routing protocols are shown in the Table 1.1. The delivery ratios of CDT, CRNG, and CGG are better than PLDel, RNG, and GG, respectively. The number of transmissions between source and destination becomes fewer in constrained geometric graphs because of the longer constraint edges. So, the channel contention will be less, which leads to less packet loss. On the other hand, fewer transmissions in the network leads to lesser possibility of simultaneous transmissions. Since the simultaneous packet transmissions in the network leads to packet collisions, the constrained geometric graphs reduces packet loss. Hence, the delivery ratio is increased. We can also observe that the delivery ratio of constrained graphs CDT, CGG, and CRNG decreases in the same order because of the inclusion properties CRNG $\subseteq$ CGG $\subseteq$ CDT. That is, the path length between the source and destination in denser graphs is lesser compared to sparse graph, leading to fewer transmissions. However, these fewer transmissions in the network lead to higher delivery ratios. Thus, the order of delivery ratio follows CDT $\geq$ CGG $\geq$ CRNG. From the Table 3.1, we can observe that the delivery ratios of the spanners PLDel, GG, and RNG are in the same order because of the inclusion property PLDel $\supseteq$ GG $\supseteq$ RNG.

The difference in delivery ratio after adding the constraint edges is high in RNG compared to GG and PLDel. This is because the difference in the number of transmissions for CRNG and RNG is higher (3.224763) than CGG and GG (1.947056) (or) CDT and PLDel (1.207223). In otherwords, the larger the number of transmissions higher collisions.

We have calculated the average number of constraint edges for different node density. The average number of constraint edges added is 12.2 in our simulation scenarios.

The next experiment is on the hop count. That is, the number of hops between the source and destination. The source node sends the packets to the destination greedily on different spanners. The average hop count at different CBR intervals is shown in the Figure 3.21. It is clear from the plot that the constrained geometric graphs have lower hop count and the spanners CDT, CRNG, and CGG have less average hop count than their counter parts, PLDel, RNG, and GG, respectively. This is because of the longer constraint edges in CDT, CRNG, and CGG.

We can also observe that the average hop count of constrained graphs CDT, CGG, and CRNG increases in the same order because of the inclusion properties $CRNG \subseteq CGG \subseteq CDT$. In other words, the edge density decreases in the same order.

Similarly, the average hop count of the graphs LDel^1, PLDel, GG, and RNG increases in the same order, because RNG $\subseteq$ GG $\subseteq$ PLDel $\subseteq$ LDel^1.

From the graphs, we observe that the average hop count of the spanners are independent of the CBR interval. This is due to the fact that the data flow rate in the network does not change the number of hops between source and destination.

The third experiment checks the end-to-end packet delay on different network topologies. Here, we consider only the packets that reach the destination. The source node sends the packets greedily to the destination on different spanners at different transfer rates. The Figure 3.22 shows the average delays. From the plot, we can observe that the spanners CRNG, CGG, and CDT contain less average delay compared to the other spanners. The lower hop count of the constrained graphs leads to fewer transmissions and collisions, which reduces the transmission delays and retransmission delays. Hence, lower delay. The inclusion properties CRNG $\subseteq$ CGG $\subseteq$ CDT and RNG $\subseteq$ GG $\subseteq$ PLDel $\subseteq$ LDel^1 lead to the same hierarchy of the number of hops between source and destination. Hence, the same hierarchy of delay.

We can observe from the graph that the average delay of all the spanners decrease with the increase in the CBR intervals. This is because of the higher network congestion at lower CBR intervals. This causes packet loss which inturn increases the retransmission and queuing delays. On the other hand, at higher CBR intervals the network congestion is

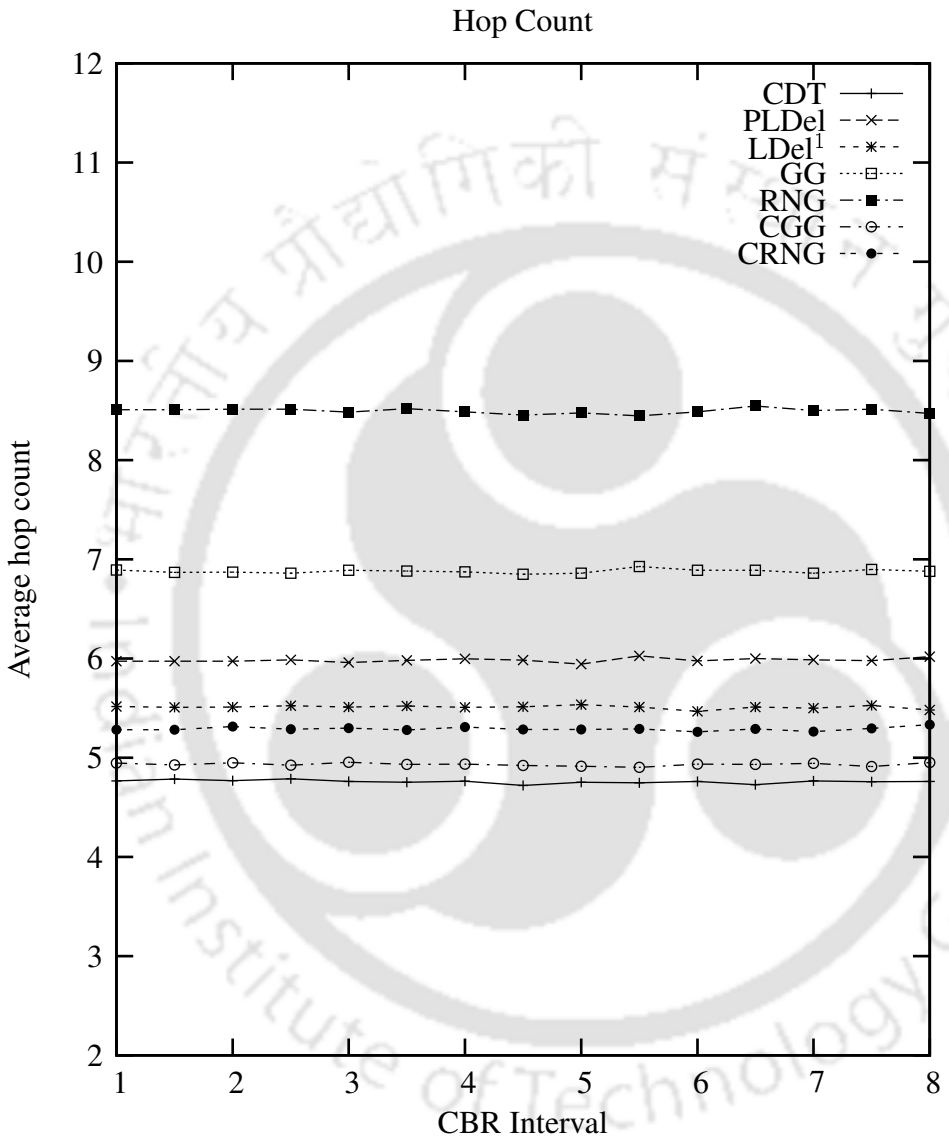


Figure 3.21: Average hop count.

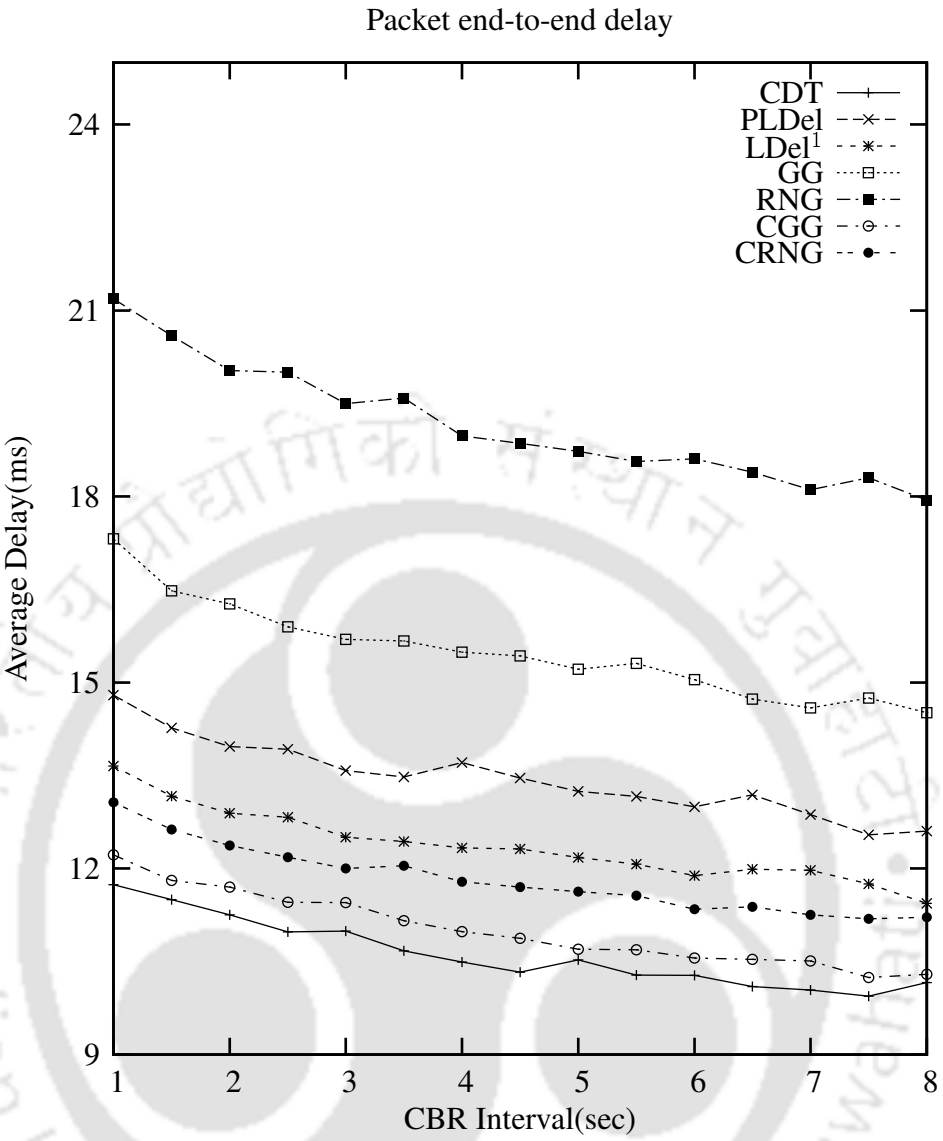


Figure 3.22: Average delay in ms.

low which leads to lower packet delay.

The next experiment is to evaluate the variance in delay, called delay variance (S_d). It is directly related to the QoS parameter called jitter, which is important for many applications such as multimedia, realtime, VOIP, and streaming applications. One can see from the Figure 3.23 and Figure 3.24, the variance of the end-to-end packet delay on the spanners CDT, CRNG, and CGG are lesser than their counter parts PLDel, RNG, and GG, respectively. This could happen, because the participation of transmission delay, retransmission delay, and queuing delays are very small and only propagation delay has the more prominent role in the end-to-end delay of constrained graphs, as the number of transmis-

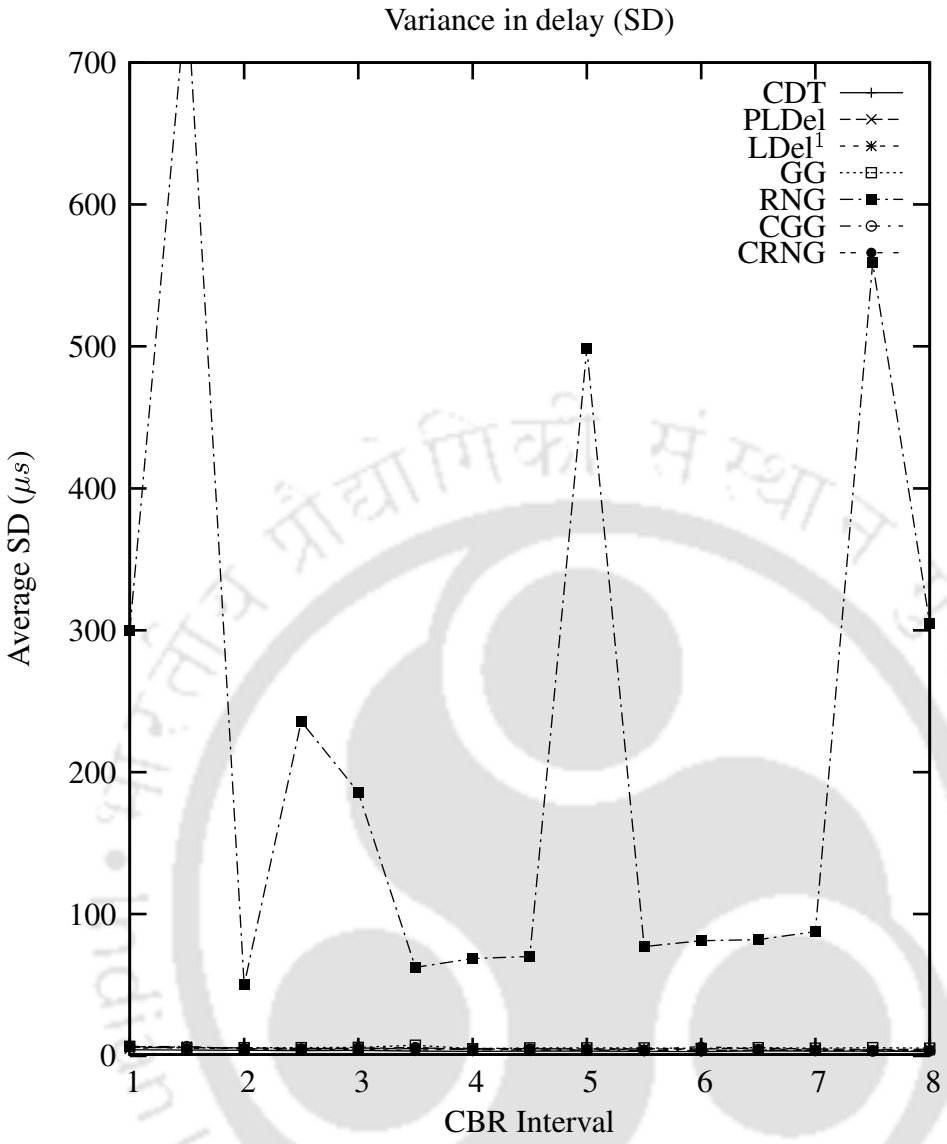


Figure 3.23: Delay variance in μs.

sion are reduced between source and destination. Hence, the delay variation becomes less in CDT, CRNG, and CGG when compared to PLDel, RNG, and GG, respectively. Similarly, the inclusion properties CRNG $\subseteq$ CGG $\subseteq$ CDT lead to the same hierarchy of delay variance. Similarly, the average throughput also follows the same pattern.

Minimum and maximum of hop count, delay, delay variance, and throughput of the constrained graphs are better than their counter parts. Here, the minimum hopcount in CDT represents the least average hop count among the five node scenarios. As we have discussed in the previous experiment, the CDT has lower hop count than PLDel. Thus, even though we consider the minimum hop count among the five node scenarios the CDT has

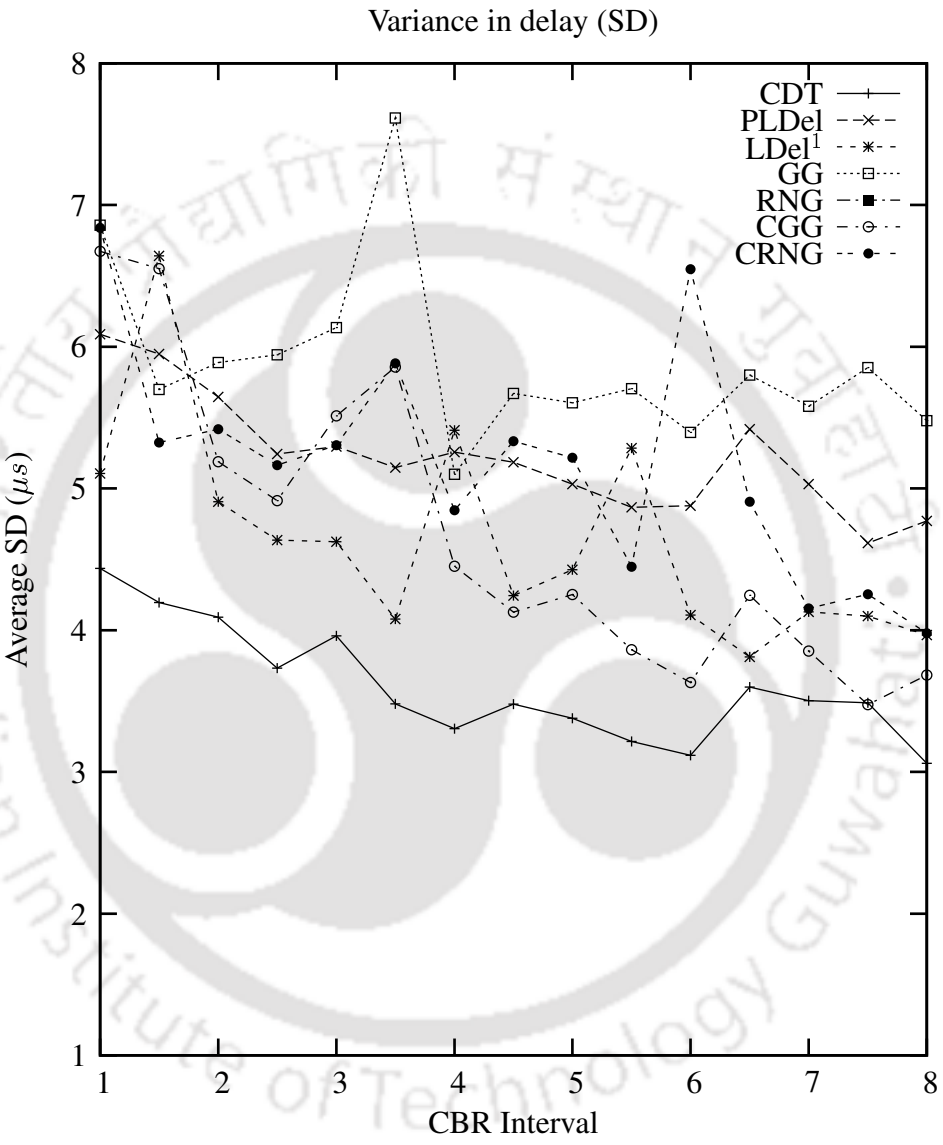


Figure 3.24: Another view of delay variance in μs.

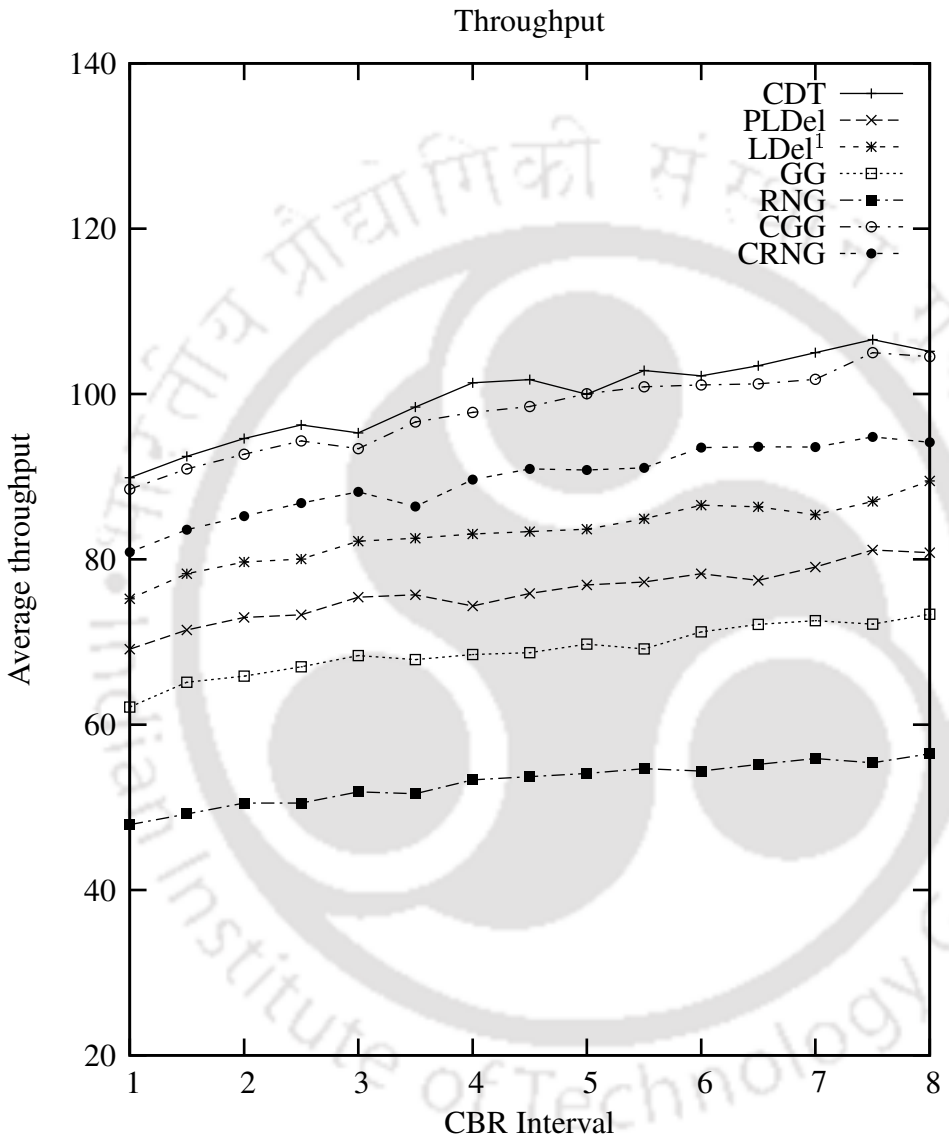


Figure 3.25: Average throughput.

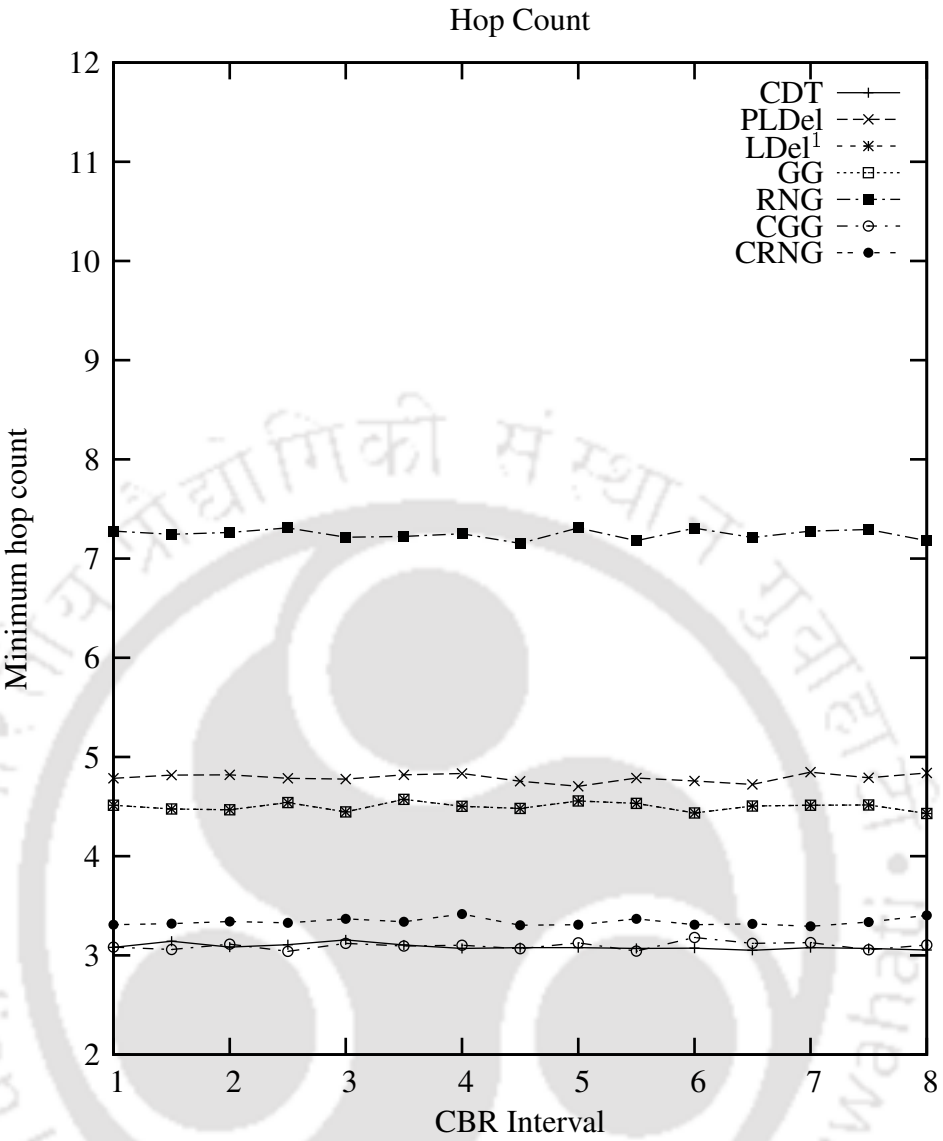


Figure 3.26: Minimum Hop count.

always minimum hopcount than PLDel or LDel, due to the presence of constraint edges in CDT. Similarly, for maximum hopcount, the maximum average hop count among the five node scenarious is considered. From the previous experiment we say that the presence of constraint edges provide better delay values. Hence, irrespective of maximum and minimum delay values among five node scenarios the constrained geometic graphs have better values. Similarly, for the maximum and minimum delay variance and throughput, the constrained geometric graphs have better values.

We have carried experiments to calculate delay and hop count by varying the number of nodes, 100, 150, 200, 250, 300, 350, and 400. The Figure 3.34 and the Figure 3.35 show

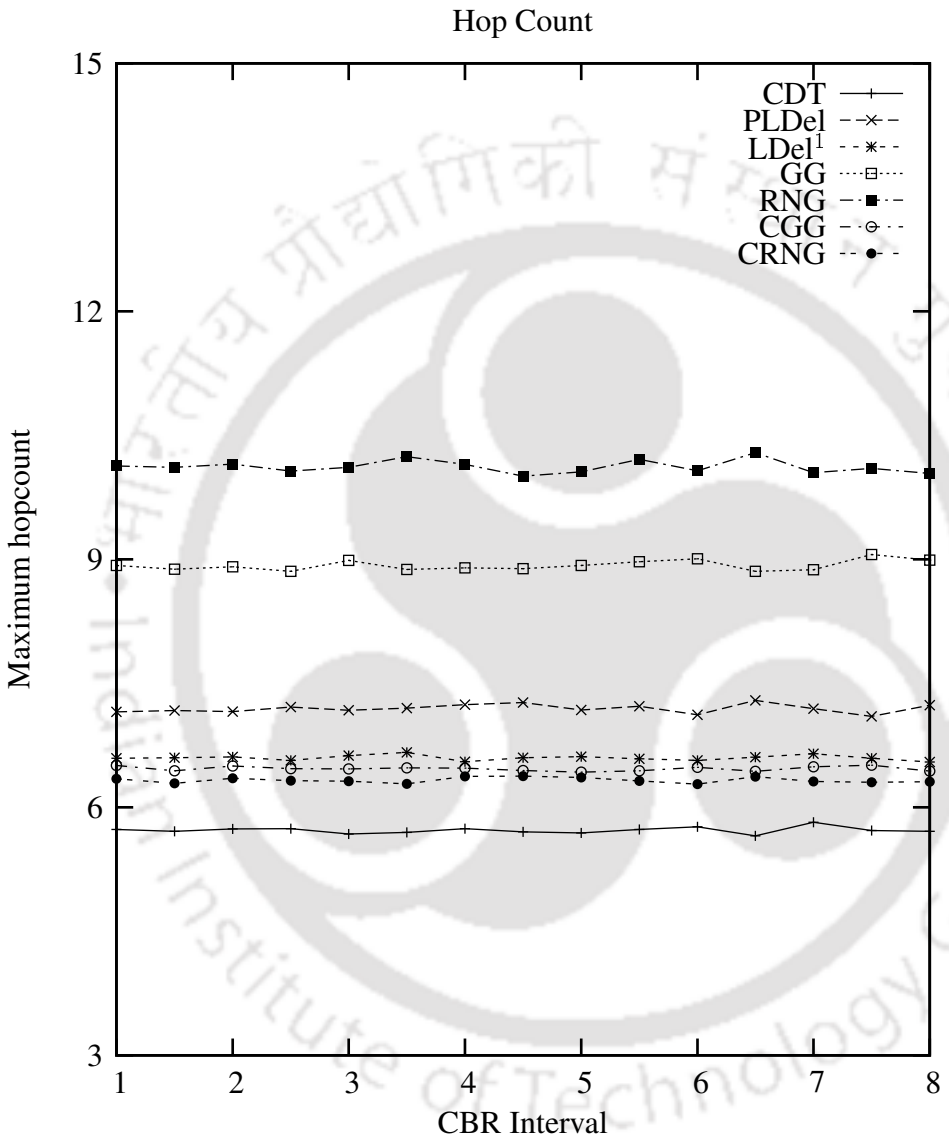


Figure 3.27: Maximum Hop count.

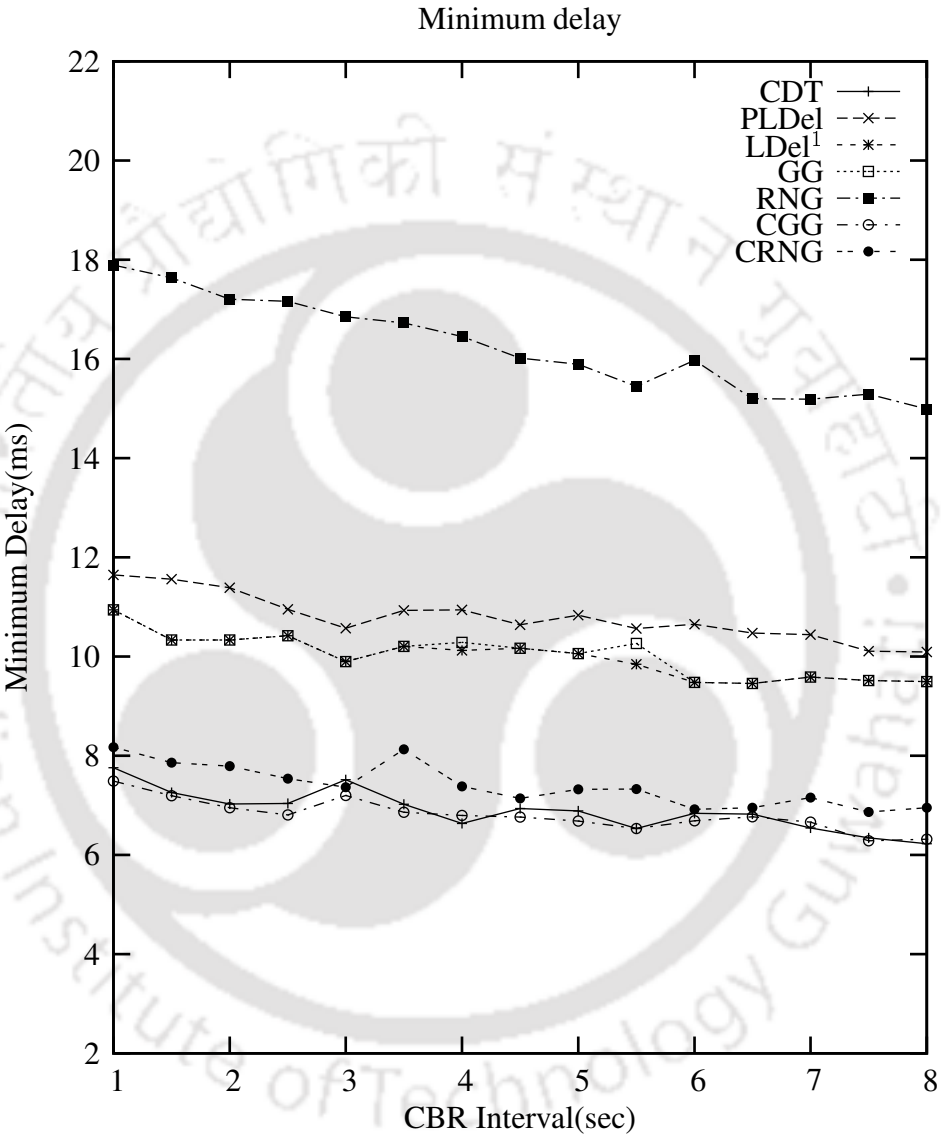


Figure 3.28: Minimum delay in ms.

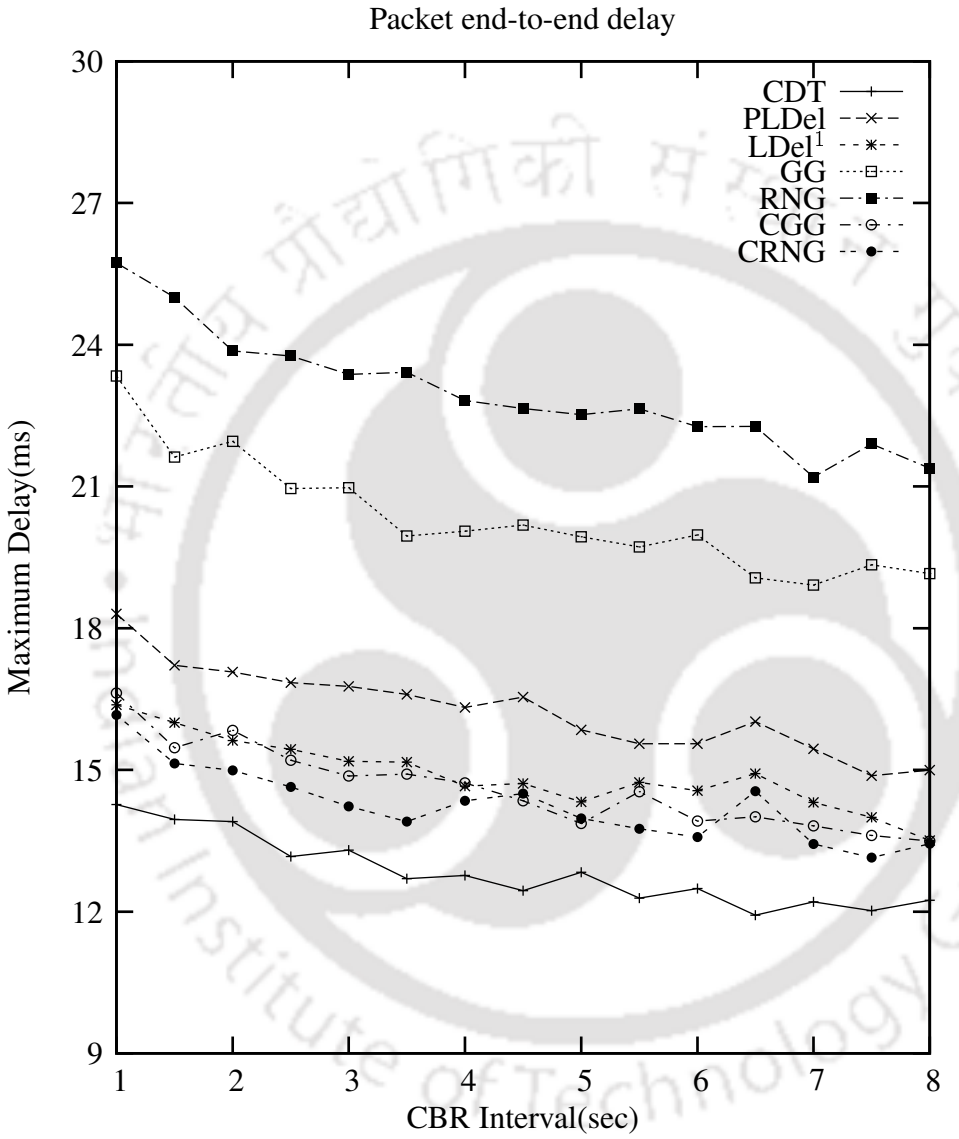


Figure 3.29: Maximum delay in ms.

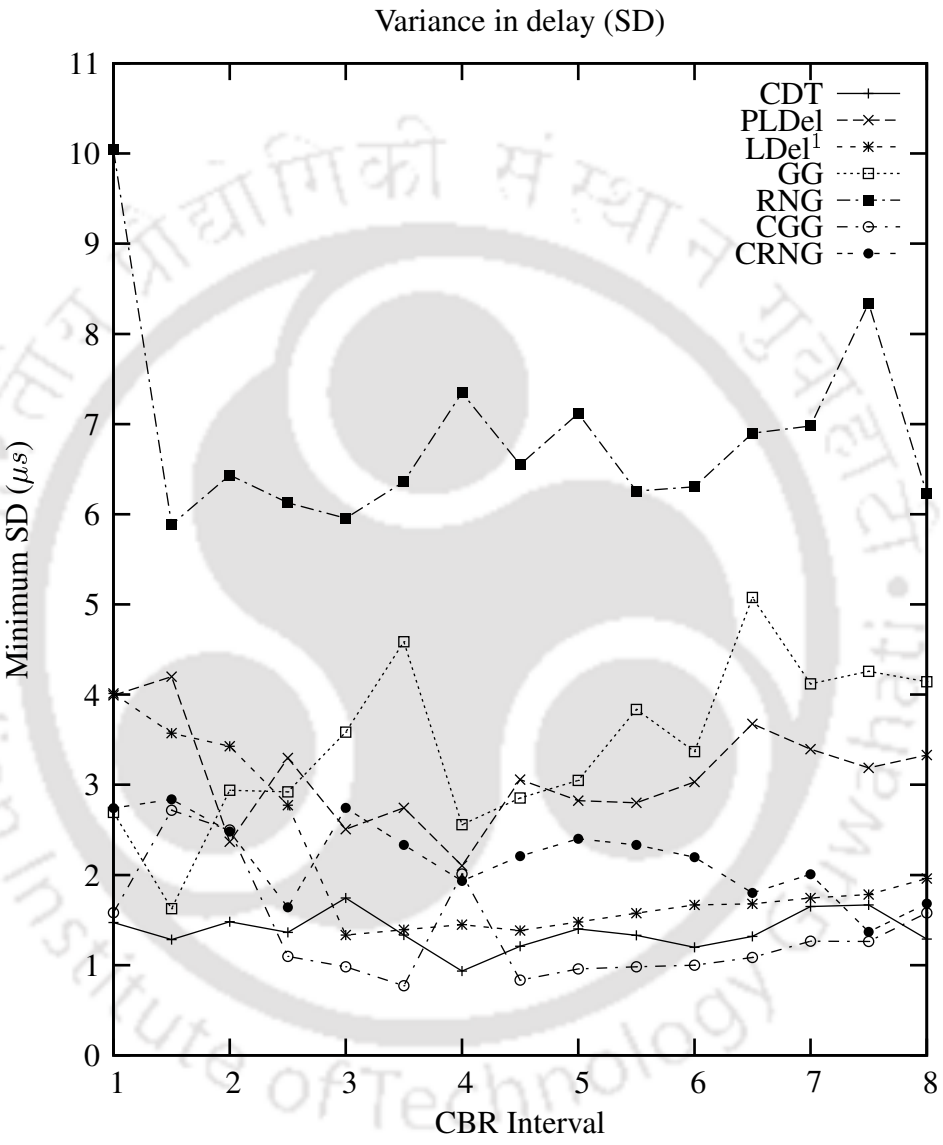


Figure 3.30: Minimum delay variance in μs.

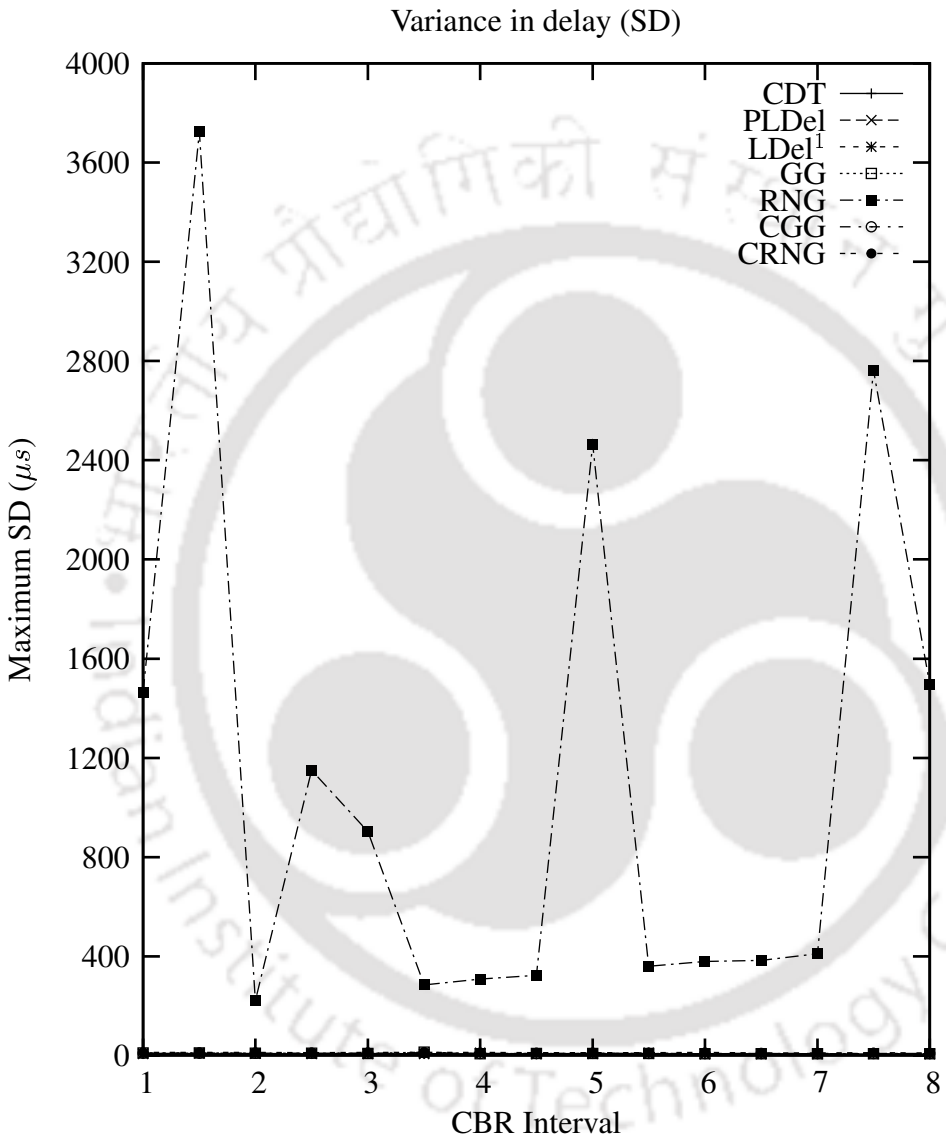


Figure 3.31: Maximum delay variance in μs.

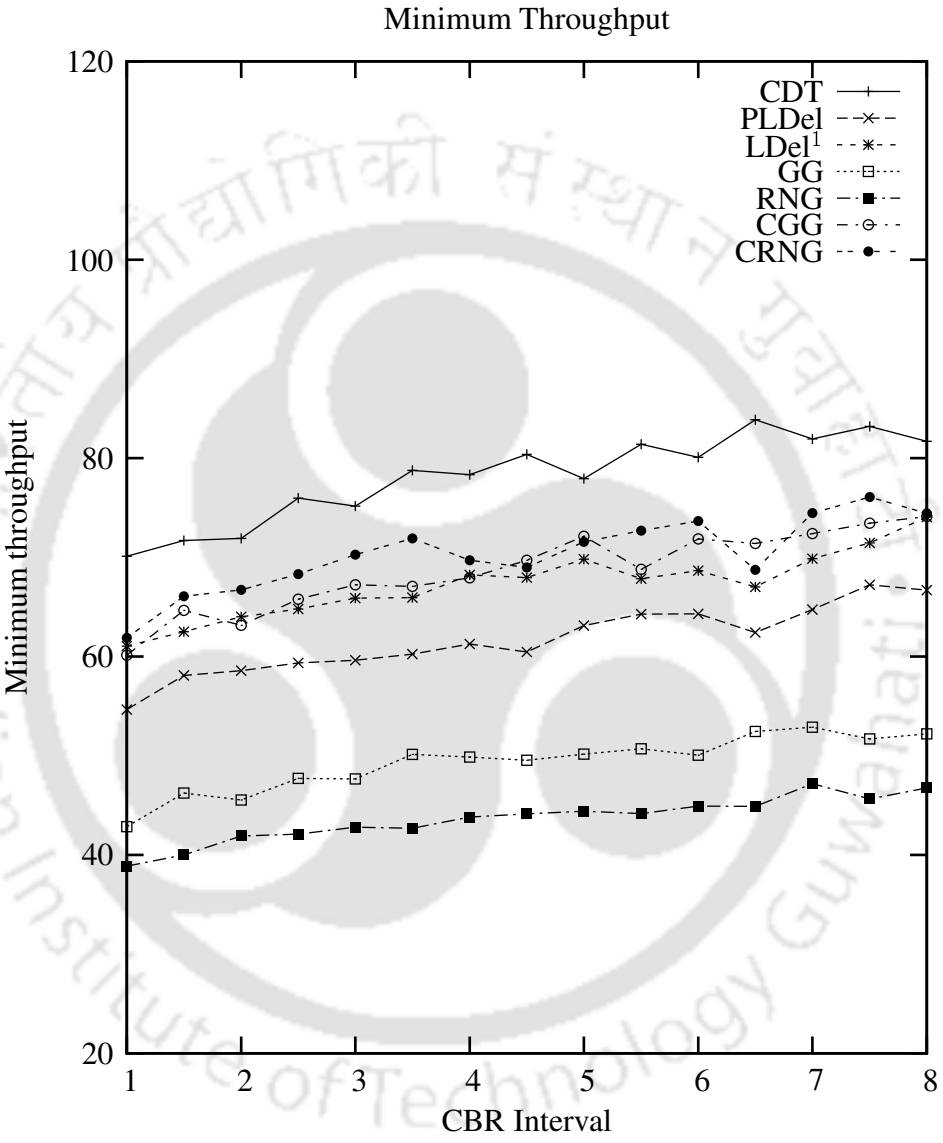


Figure 3.32: Minimum throughput.

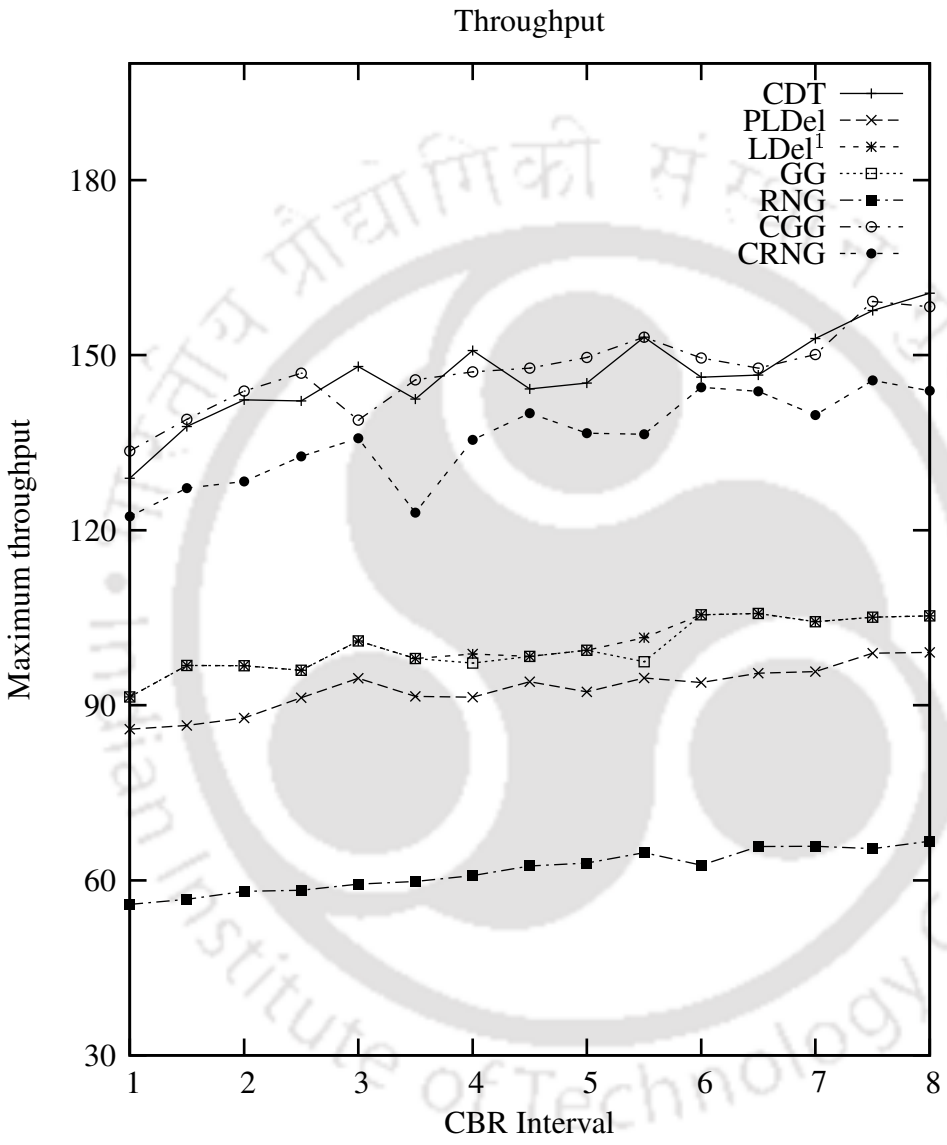


Figure 3.33: Maximum throughput.

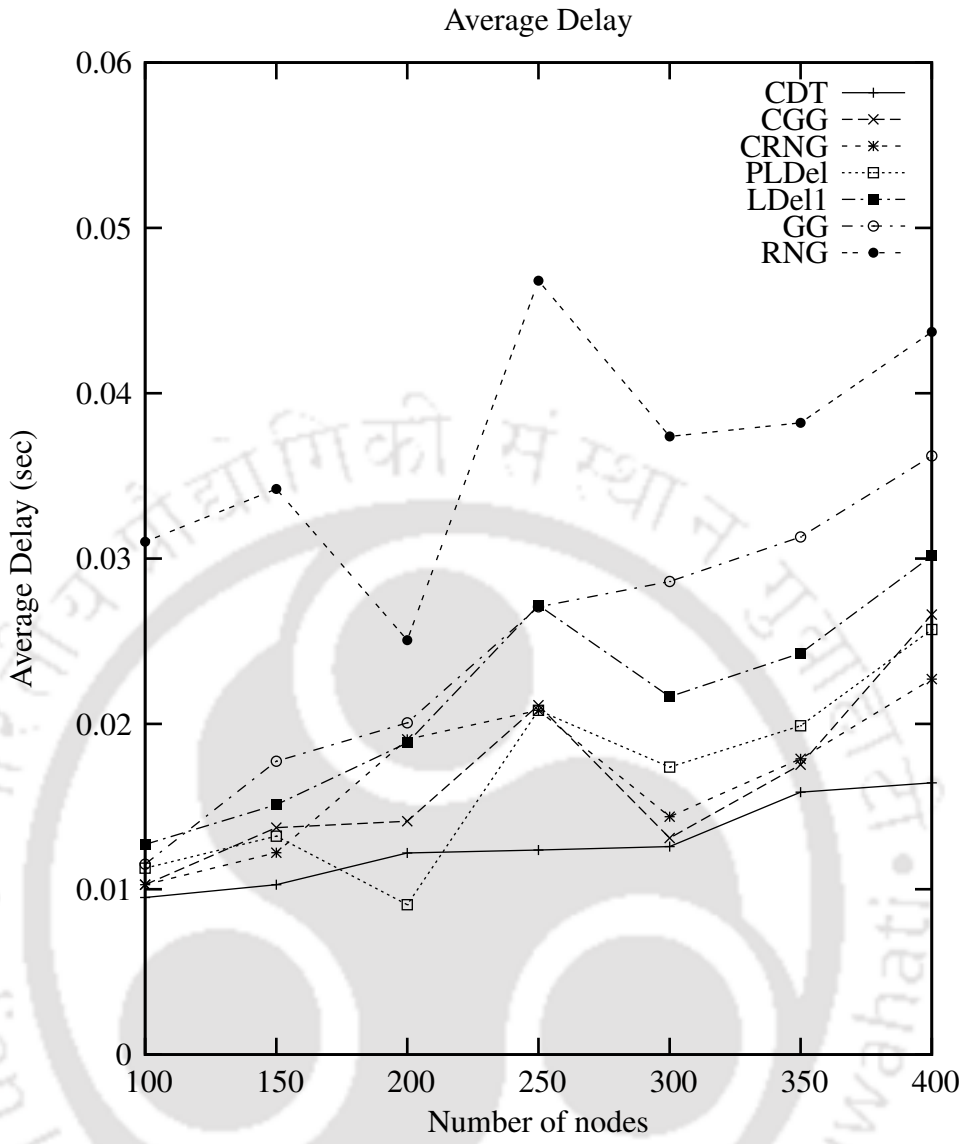


Figure 3.34: Average delay versus number of nodes.

that the average hop count and delay on constrained geometric graphs are better than other graphs. This happens, because the constraint edges on CDT, CRNG, and CGG convey the packets with fewer hops between the nodes. Hence, delay is reduced.

3.4 Conclusions

We have studied the need of constrained based geometric graphs and proposed three spanners, constrained Delaunay triangulation (CDT), constrained relative neighborhood graph (CRNG), and constrained Gabriel graph (CGG) to reduce the number of hops. The sim-

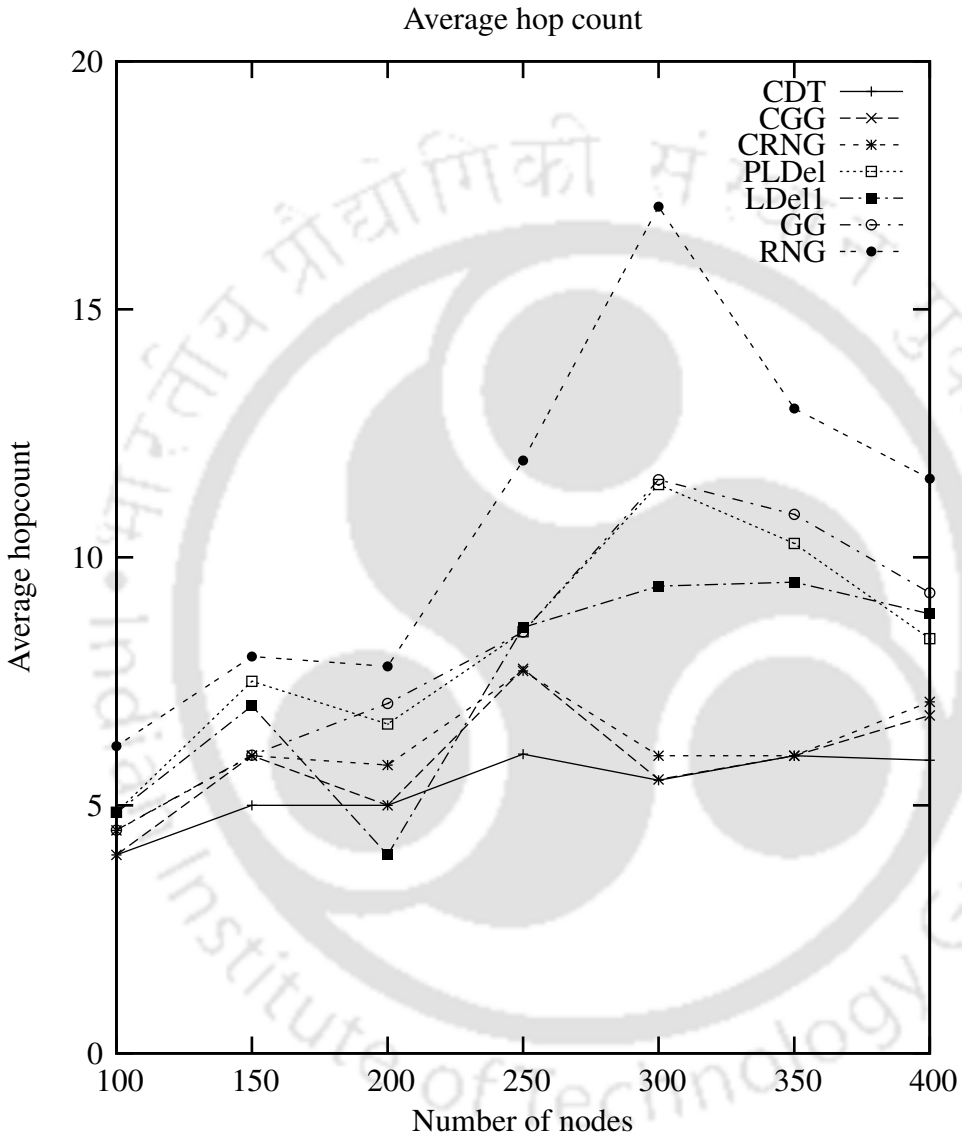


Figure 3.35: Average hop count versus number of nodes.

ulation results in *ns2.28* consolidate our claim of average hop count reduction in these spanners, which in turn reduces the delay and jitter, and increases the throughput. The increasing order of the average hop count, delay, and jitter, in these spanners are CDT, CGG, CRNG, $LDel_1$, PLDel, GG, and RNG, where as the throughput follows the reverse order. The delivery ratios of constrained geometric spanners are better than their non constrained counter parts.

The Algorithm *Constraint edge2* used to find constraint edges are not optimal. It would be interesting to find an algorithm to add more number of constraint edges using minimum number of neighbor information to further reduce various network overhead.

In ideal conditions, the long constraint edges in the network do not affect the link quality, as per the UDG model. But in reality, the quality of signal strength decreases with the increase in distance to the receiving node [ZK07, CWPE05, Rap01]. The constraint edges may have poor link quality. In addition to the distance, link quality also depends on surroundings such as walls, buildings, mountains, and weather conditions. So, one can consider the link quality also as a parameter along with the distance to choose constraint edges. It would be interesting to study these constrained spanners by adding constraint edges based on link quality.

Another future direction is to find the constrained geometric spanners based on partial delaunay triangulation (PDT) and restricted delaunay graph (RDG), and study their properties and performance.

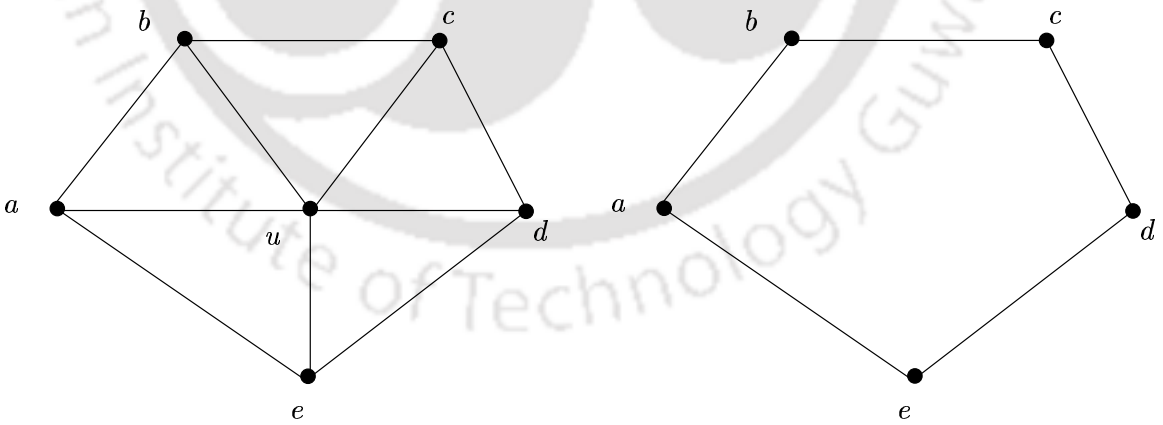
भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 4

Dynamic Spanners

4.1 Introduction

In adhoc networks nodes can go down. That is, nodes do not function as desired, due to various reasons, such as insufficient battery power, environmental effects like eruption of volcano, cyclones, and floods, and accidents like landslides and debris. Moreover, to conserve the energy of network, the nodes can switch off their transmitter or go to the sleep mode. There will be heavy packet loss if these nodes exist in any routing path. Similarly, a new node can join the network or an existing node may wake up from sleep mode.



(i) (ii)

Figure 4.1: The network before and after the node u is down.

In the above cases, the network topology changes are not reflected in the network

graph. To overcome this problem, the underlying network graph need to change dynamically as and when required. In other words, the spanners for adhoc networks need to change dynamically to retain the geometric properties of the spanner for efficient routing.

Consider the network graph in the Figure 4.1(i), if node u fails then the topology of the network changes as shown in the Figure 4.1(ii). Here, the nodes a, b, c, d, and e have to reflect their neighbor list after the node u fails. Otherwise, the nodes assume that the paths through the node u exists resulting in heavy packet loss occur. In addition to this, even after the node lists for the nodes a, b, c, d, and e are updated, the resulting network graph may violate various geometric properties. For example, consider the network graph given in the Figure 4.1(ii), even though the information about node u down is updated in the neighbor lists of the nodes a, b, c, d, and e, the Delaunay triangles are not computed, as shown in the Figure 4.2.

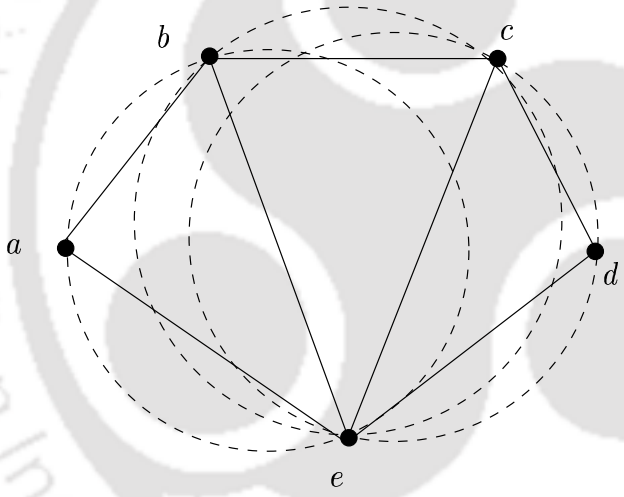


Figure 4.2: The updated network graph after reconstruction of Delaunay triangulation.

In this thesis, we have considered the above problem and proposed three spanners called dynamic local Delaunay triangulation (DLDel), dynamic relative neighborhood graph (DRNG), and dynamic Gabriel graph (DGG). The spanners DLDel, DRNG, and DGG reconfigure the graphs locally, instead of global reconstruction; to recover quickly from any packet loss occurred in PLDel, RNG, and GG, respectively. The performance of dynamic spanners DLDel, DRNG, and DGG are demonstrated by simulation using network simulator (ns2) and the results are compared with the static spanners PLDel, RNG, and GG, respectively. The experimental results show that the spanners DLDel, DRNG, and DGG

perform better than their counter graphs. The system model for dynamic spanners is same as the model used in constrained geometric graphs. In addition, the model should have the facility for each node to measure the remaining energy in the battery.

The remaining part of this chapter is organized as follows: The next section describes the dynamic local Delaunay triangulation (DLDel). In the section 4.3 the spanner dynamic relative neighborhood graph (DRNG) is described. The section 4.4 describes dynamic Gabriel graph (DGG). The simulation work and experimental results are presented in the section 4.5. Finally, the section 4.6 concludes the chapter.

4.2 Dynamic local Delaunay triangulation (DLDel)

The main idea is to reconfigure the PLDel locally, whenever a new node joins the network or existing node goes down, in initially constructed PLDel.

We briefly describe PLDel algorithm [LCWW03, LCW02]. Each node u broadcasts *hello_packet*s, which contain id and position of node u, and listens to the *hello_packet*s broadcasted by other nodes to collect 1-hop neighborhood information $n_1(u)$. Each node u computes the Delaunay triangulation, $Del(n_1(u))$, of the nodes $\{n_1(u) \cup \{u\}\}$. The Gabriel edges are computed from $Del(n_1(u))$.

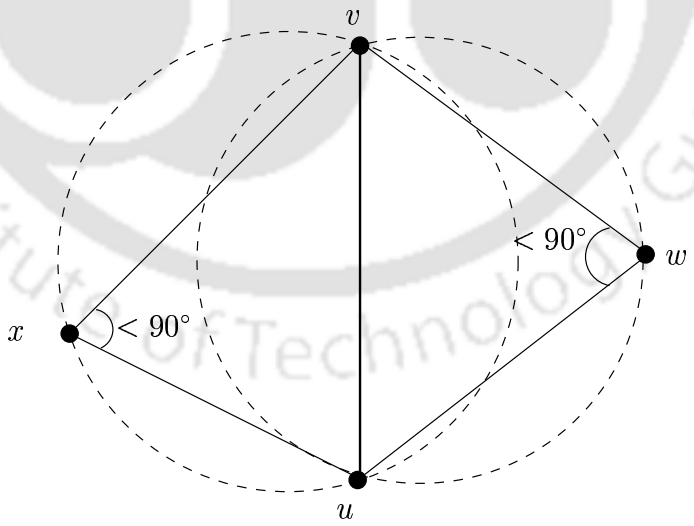


Figure 4.3: The edge $\overline{uv}$ is a Gabriel edge.

An edge $\overline{uv}$ is called Gabriel edge if the edge $\overline{uv}$ is common edge for two triangles $\triangle uvw$ and $\triangle uvx$, and the angles $\angle uxv$ and $\angle uwv$ are less than 90°, as shown in the

Figure 4.3. These edges will never be deleted from PLDel. Next step is to check for consistent triangles. A triangle $\triangle uvw$ is called consistent triangle if $\triangle uvw$ is in $Del(n_1(u))$, $Del(n_1(v))$, and $Del(n_1(w))$, as shown in the Figure 4.4.

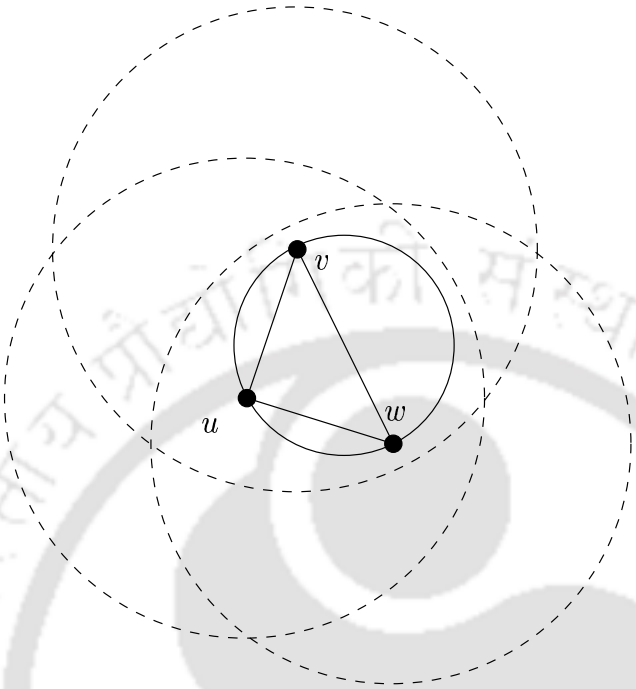


Figure 4.4: The triangle $\triangle uvw$ is a consistent triangle for PLDel.

The graph with the Gabriel edges and consistent triangles is called LDel^1, which may not be a planar graph. A separate planarization algorithm is applied on LDel^1 to extract a planarized graph called planarized local Delaunay triangulation (PLDel).

Definition 4.2.1 Topological neighbors $\aleph_G(v)$ of a node v in the graph G is the set of adjacent nodes in G. That is, $\aleph_G(v) = \{u | \overline{uv} \in G\}$.

Each node u broadcasts $\aleph_{PLDel}(u)$ using message *Tneighbor_Packet* and listens to the messages from its topological neighbors. In other words, each node u maintains the $\aleph_{PLDel}(v)$, where $v \in \aleph_{PLDel}(u)$.

4.2.1 Node Down

A node may go to sleep mode to conserve energy, insufficient battery power, or periodic maintenance. In such cases, the node informs its topological neighbors by broadcasting the

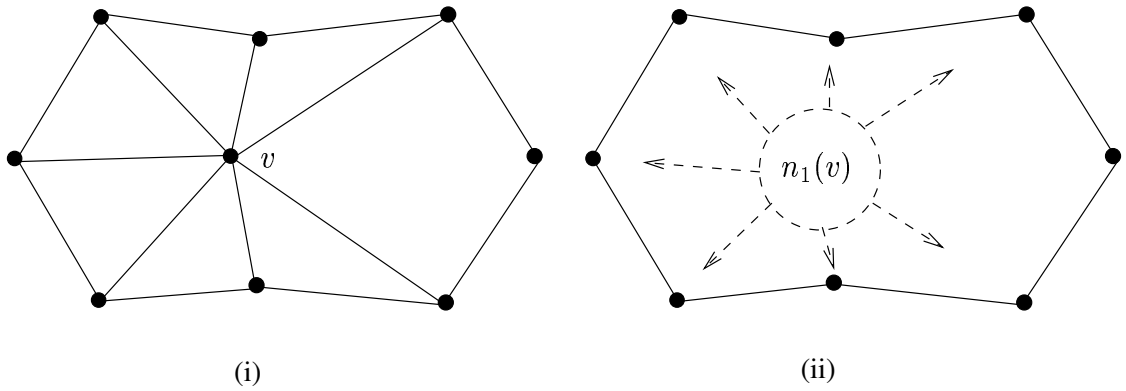


Figure 4.5: Node u informs its down status to its topological neighbors $\aleph_{PLDel}(u)$.

message *Ndown_Packet* before it switches off its transmitter, (see the Figure 4.5). Another possibility of node down is by mean of accidents like system crash or physical damages. In these cases, it can not inform its neighbors. But, each of its neighbor can identify this, due to the absence of periodic *hello_packet* messages from the failed node. In the first case, the node down is known to its topological neighbors immediately by the *Ndown_Packet* message, where as, in the second case it takes a couple of *hello_packet* intervals.

When a node u receives the message *Ndown_Packet* from the node v, it extracts the id from the message and updates the neighbor list $n_1(u)$ by deleting the id from $n_1(u)$. One way of reconstructing the spanner locally is with 1-hop neighbors $n_1(v)$. Each node $u \in n_1(v)$ constructs the $Del(n_1(u))$. Each node removes the inconsistent triangles and forms $LDel'(u)$, where $LDel'(u)$ denotes the set of Gabriel edges incident on u and triangles $\triangle uvw$ of $LDel'$.

However, constructing the PLDel in 1-hop neighbors of node v is expensive in terms of computation and communication, because it may contain large number of nodes. Moreover, we need to maintain 2-hop information for planarization. In order to reduce the time and communication complexities, we reconstruct the spanner using topological neighbors of the failed node. This topological neighbors of failed node v form a part of polygon, as shown in the Figure 4.6.

Each node u, which is a topological neighbor of the failed node v, constructs Delaunay triangulation with the set of nodes $\aleph_{PLDel}(v)$ using any centralized algorithm. Remove the edges which are not in UDG and outside the polygon, see the Figure 4.7. Topological neighbors of node u is updated and broadcasted using the message *Tneighbor_Packet*.

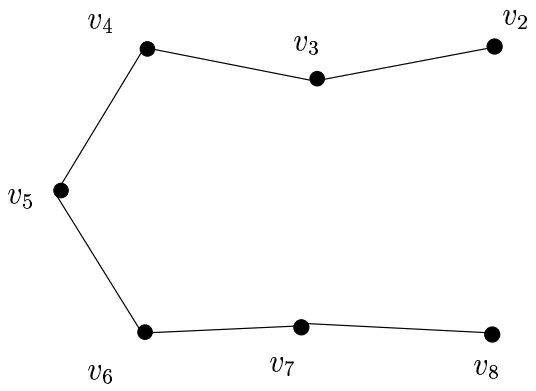


Figure 4.6: Part of polygon with nodes $\aleph_{PLDel}(v)$ of the network in the Figure 4.5.

Similarly, whenever an accidental node failure is detected, same action is taken by its topological neighbors.

This algorithm uses the messages *Ndown_Packet*, *Tneighbor_Packet*, and *hello_packet* in addition to the messages used for constructing PLDel. The message *Ndown_Packet* carries the node id information. The packet *Tneighbor_Packet* contains the list of node ids. The packet *hello_packet* contains the node id and position. A formal description of the algorithm is given below.

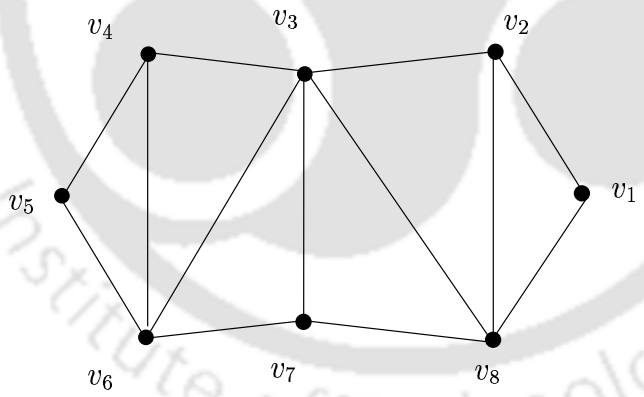


Figure 4.7: Reconfigured Delaunay triangulation after the node v is down in the Figure 4.5.

Algorithm: Node Down in PLDel

Whenever node u receives ***Ndown_Packet*** from node v or detects the accidental node failure of v

{

Remove v from $n_1(u)$.

if($u \in \aleph_{PLDel}(v)$)

{

Construct Delaunay triangulation (DT) with the nodes $\aleph_{PLDel}(v)$.

Remove the edges from DT which are not belongs to UDG.

Remove the edges from DT which are outside the polygon formed by $\aleph_{PLDel}(u)$.

}

}

The following lemma can be verified easily.

Lemma 4.2.1 *The time and message complexity of a node down is in* $O(\Delta \log(\Delta))$ *and* $O(1)$ *respectively, where* Δ *is the maximum node degree in PLDel.*

4.2.2 Node Join

When a new node joins the network or a node wakes up from sleep mode, it broadcasts the message *Njoin_Packet* to inform its neighbors about its arrival into the network, see the Figure 4.8. Each of its neighboring node u updates its neighbor list $n_1(u)$. The reconstruction of the spanner is done only by the nodes within the transmission range of node v. Each node u computes PLDel(u) and broadcast the message *Jnnode_packet* if the edge $\overline{uv} \in PLDel(u)$, where v is the newly joined node. Each joined node v collects this information and update topology neighbor lists $\aleph_{PLDel}(v)$ to maintain the graph, as shown in Figure 4.9.

The node join algorithm uses the following messages apart from the messages used in PLDel. The message *Njoin_Packet* contains a node id. The message *Jnnode_packet* contains a list of node ids. The formal algorithm is given below.

Algorithm: Node Join in PLDel

---

1. Each node u receiving ***Njoin_Packet*** from node v does the following:

   Update the $n_1(u)$.

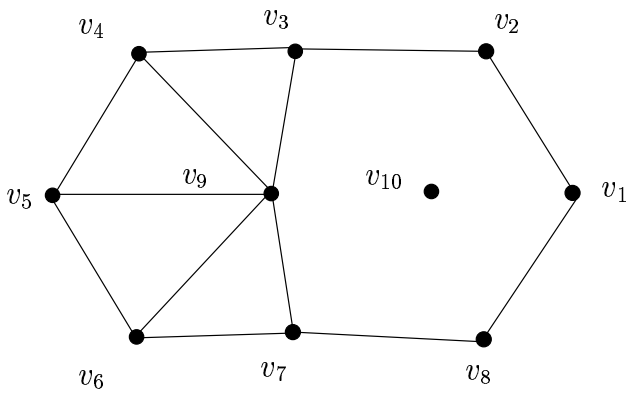


Figure 4.8: New node v_{10} joins the network.

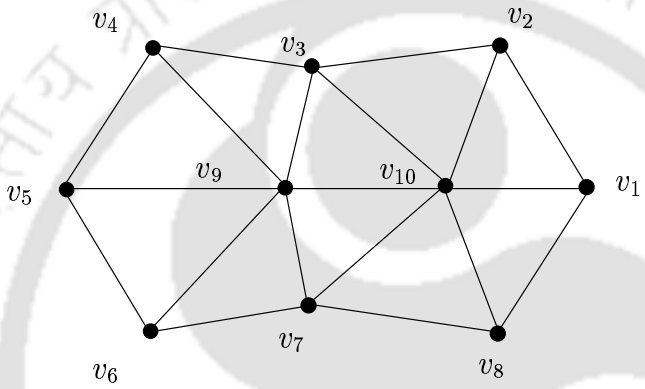


Figure 4.9: Delaunay triangulation of the network in the Figure 4.8 after node v_{10} joins.

Construct PLDel(u).

If edge $\overline{uv} \in PLDel$ then send the message ***Jnnode_Packet***.

2. The joined node v receiving the message ***Jnnode_Packet*** from node u, adds the edge $\overline{vu}$ to $\aleph_{PLDel}(v)$.

One can easily verify the following lemma.

Lemma 4.2.2 *For a node join, the message and time complexity for reconstructing the spanner is in $O(\Delta)$ and $O(\Delta^4)$, where Δ is the maximum node degree in UDG.*

4.3 Dynamic relative neighborhood graph (DRNG)

The existing algorithm for constructing RNG [KK00] discussed in the Section 2, is not amendable for efficient dynamic maintenance. In other words, the time and message complexities of maintaining RNG dynamically are higher. Therefore, we propose a new algorithm for constructing RNG, which has less overhead for dynamic maintenance.

Area around a node u is divided into six equal sections $s_1(u)$, $s_2(u)$, $s_3(u)$, $s_4(u)$, $s_5(u)$, and $s_6(u)$, as shown in Figure 4.10. We denote by $R_i(u)$, the set of nodes in the i^{th} section of a node u for i=1, 2, 3, 4, 5, and 6. That is, $R_i(u) = \{v | v \in s_i(u)\}$. A point v in $R_i(u)$ is said to be geographic neighbor to u in $s_i(u)$ if $d(u, v) = min\{d(u, q) | q \in R_i(u)\}$.

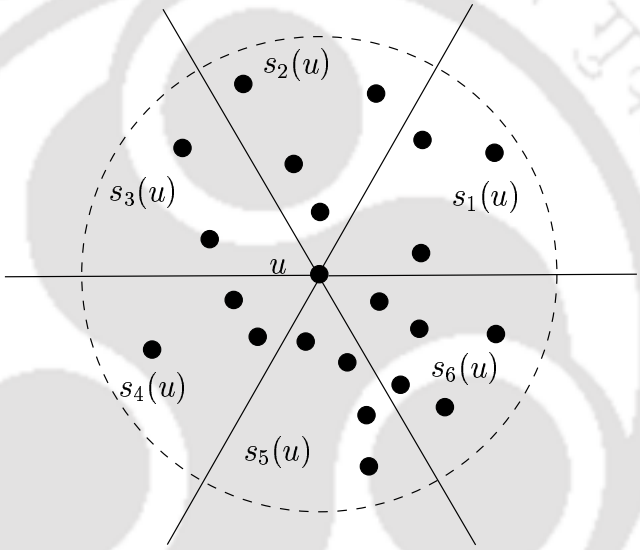


Figure 4.10: Equal sectors around a node.

The geographic neighborhood graph (GNG) of a set of nodes V in $s_i(V)$ is a graph $GNG_i(V) = (V, E_i)$ such that $E_i = \{\overline{uv} | u \in V, d(u, v) = min\{d(u, q) | q \in R_i(u)\}\}$. That is, each node $u \in V$ connect to nearest neighbors in the s_i section. The union of the above graphs for i= 1, 2, $\cdots$, 6 is called the geographic neighborhood graph [KN86] of V, (see the Figure 4.11) and denoted by GNG(V) or simply GNG. That is,

$$\text{GNG}(V) = \bigcup_{i=1}^{6} \text{GNG}_i(V)$$

The relative neighborhood graph is a subgraph of the geographic neighborhood graph, RNG $\subseteq$ GNG [KN86].

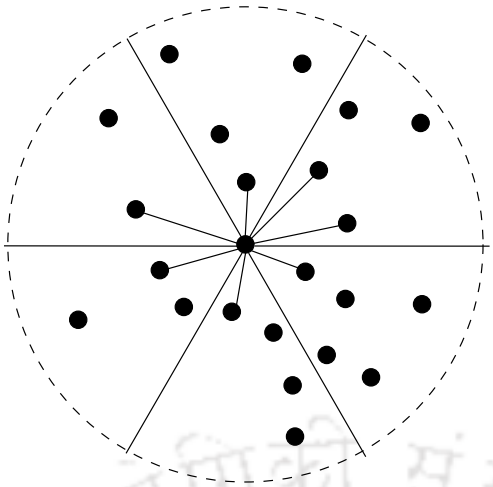


Figure 4.11: Geographic neighborhood graph.

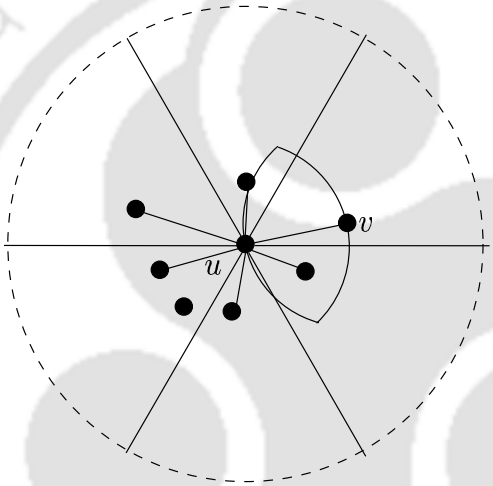


Figure 4.12: Lune testing for GNG edges.

4.3.1 Localized construction of RNG using GNG

Katajainen et. al [KN86] proposed an $O(n^2)$ algorithm to construct RNG with the given set of nodes. This algorithm considers the fact that GNG is a super graph of RNG. But this algorithm is centralized algorithm, which is not suitable for adhoc networks. In this Section, we give a distributed localized algorithm to construct the RNG.

We use the same notation as in centralized RNG construction with the extra localization constraint. $R_i(u)$ to denote the set of nodes in section $s_i(u)$, which are with in 1-unit distance from node u, for i=1, 2, $\cdots$ 6. That is, $R_i(u) = \{v | v \in s_i(u), |\overline{uv}| \leq 1\}$. Similarly, $E_i = \{\overline{uv} | v \in n_1(u), d(u,v) = min\{d(u,q) | q \in R_i(u)\}\}$. The $RNG_i(u)$ denotes the set of nodes, from which the RNG edges are incident on node u in the sector i.

Localized construction of RNG is done in three stages. In the first stage, each node gathers 1-hop neighbor lists from the received *hello_packet* message. In second stage, we construct the GNG. Finally, prone the GNG edges to obtain the RNG edges, see the Figure 4.12. The localized construction of RNG does not use any extra control messages. The formal description of the algorithm is given below.

Algorithm: RNG

---

1. Each node u broadcasts the ***hello_packet*** with id and location information, and waits to receive the ***hello_packet*** from other nodes to collect $n_1(u)$.

2. Each node u constructs GNG.

3. Each node u does the following to find RNG edges.

   For each edge $\overline{uv} \in GNG$

   If ($lune(u, v)$ is empty)

   add $\overline{uv}$ to RNG.

Lemma 4.3.1 *The message complexity for constructing RNG is in $O(n)$.*

Lemma 4.3.2 *The time complexity for constructing RNG is in $O(\Delta)$ for uniformly distributed nodes, where Δ is the maximum node degree in UDG.*

Proof: There are six sectors around each node. For uniformly distributed nodes, the number of GNG edges in a single sector is in $O(1)$ and can be computed in constant time [KN86]. The time complexity to obtain an RNG edges from GNG is in $O(\Delta)$. In other words, each lune test takes $O(\Delta)$ time complexity. Thus, the expected time complexity for constructing RNG is in $O(\Delta)$. ∎

The distributed RNG algorithm described in [KK00] has the time complexity of $O(\Delta^2)$ and message complexity of $O(n)$. The time complexity of the proposed algorithm for localized RNG construction is in $O(\Delta)$ and the message complexity is in $O(n)$. However, most of the times, the cost of local changes for dynamic maintenance of the RNG algorithm given in [KK00] has higher complexity by $O(\Delta)$. The distributed RNG algorithm by Borbash and Jennings [BJ02] has the time complexity of $O(\Delta)$. However, the

number of messages in this algorithm is k times higher than the messages in the proposed algorithm. Moreover, their algorithm is not suitable for unit disk graph model. Hence, our algorithm is better than the existing algorithms in terms of time and message complexities. In addition, it is more suitable for dynamic maintenance of RNG.

We make use of the property $RNG_i(u) \subseteq GNG_i(u)$ for efficient dynamic maintenance of RNG. Thus, we need to maintain GNG also.

4.3.2 Node Down

When a node is down in the network, it informs its neighbors by broadcasting the message *Ndown_Packet* before it switches off its transmitter. Each node u receiving the packet *Ndown_Packet* updates its neighbor lists $n_1(u)$ and reconstructs the edges of GNG and RNG. In case of the node failures due to the accidents, the RNG reconstruction is carried out after a couple of beacon intervals.

Let v be the down node such that $v \in R_i(u)$. Depending on number of vertices in $RNG_i(u)$ and $GNG_i(u)$, we have different cases for RNG updation. First we discuss the updation required in the sector i.

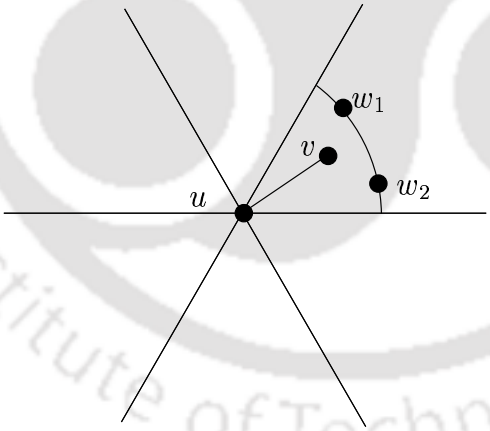


Figure 4.13: Case 1: $RNG_i(u) = \{v\}$ and $GNG_i(u) = \{v\}$.

Case 1 ($RNG_i(u) = \{v\}$ **and** $GNG_i(u) = \{v\}$)**:**. Find the nodes which are nearest neighbors of u in the sector i to form $GNG_i(u)$. For each $w \in GNG_i(u)$ do the lune test. That is, if $lune(u, w)$ is empty, add w to $RNG_i(u)$ or $\overline{uw}$ to RNG (see the Figure 4.13).

Case 2 ($RNG_i(u) = \{v\}$ **and** $GNG_i(u) - \{v\} \neq \emptyset$)**:**. This is because $lune(u, v)$ is empty and for all $w \in GNG_i(u) - \{v\}$, the $lune(u, w)$ is not empty. The node v down does not

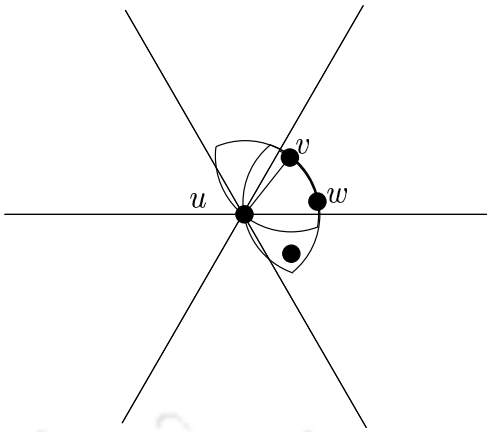


Figure 4.14: Case 2: $RNG_i(u) = \{v\}$ and $GNG_i(u) - \{v\} \neq \emptyset$.

effect the emptiness of the $lune(u, w)$. Hence, no new edge is added in the i^{th} sector, see the Figure 4.14.

Case 3 ($RNG_i(u)$ = $\{v, v_1, v_2, \cdots\}$):. If node v is down, there is no change in $GNG_i(u)$, since $v, v_1, v_2, \cdots$ are also nearest neighbors in the sector i. $RNG_i(u)$ becomes $\{v_1, v_2, \cdots\}$, because the node v down does not effect the emptiness of the $lune(u, v_j)$, for all $j = 1, 2, \cdots$. For all nodes $w \in GNG_i(u) - RNG_i(u)$, the emptiness of the $lune(u, w)$ is not going to effect.

Now we consider the updation required in the other sectors $R_j(u)$, where $j \neq i$, and $v \in R_i(u)$. If v is in $lune(u, w)$ for some $w \in GNG_j(u) - RNG_j(u), j \neq i$, then the $lune(u, w)$ may become empty if the node v is down. We have to add all such nodes to $RNG_j(u)$. Therefore, for each $w \in GNG_j(u) - RNG_j(u)$ for $j \neq i$, if the $lune(u, w)$ is empty, add w to $RNG_j(u)$, see the Figure 4.15. The node down maintenance algorithm uses only the message *Ndown_Packet* which carries a node id. The formal algorithm is given below.

Algorithm: Node Down in RNG

If a node v is down in the sector i of node u

(a)If($RNG_i(u) - \{v\} = \emptyset$ and $GNG_i(u) = \{v\}$)

{

Find $GNG_i(u)$.

For each node $w \in GNG_i(u)$

Add $\overline{uw}$ to RNG if $lune(u, w)$ is empty.

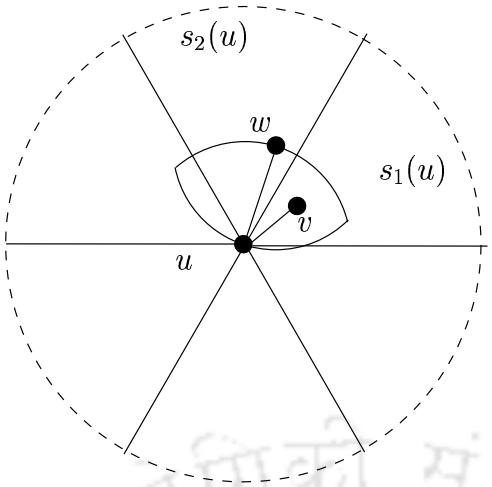


Figure 4.15: Updations in other sectors.

}

If($RNG_i(u) - \{v\} \neq \emptyset$)

Delete v from $GNG_i(u)$ and $RNG_i(u)$.

(b) For each sector j of u, where $j \neq i$

For each $w \in GNG_j(u) - RNG_j(u)$

If($lune(u, w) = \emptyset$) add w to $RNG_j(u)$.

Lemma 4.3.3 *The time and message complexity of updating RNG for a node down is in* $O(\Delta)$ *and* $O(1)$ *respectively, where* Δ *is maximum node degree in UDG.*

4.3.3 Node Join

The message *Njoin_Packet* is broadcasted when a node joins the network. Each node receiving the packet *Njoin_Packet* updates its neighbor lists and computes the RNG.

Let v be the node joining the network in the sector i of node u, that is $v \in R_i(u)$, and $\overline{uv} \leq 1$. When a new node joins the network, the following are the various cases for updating the RNG and GNG edges in the sector i.

Case 1 ($v_1 \in RNG_i(u)$ and $|\overline{uv_1}| < |\overline{uv}|$): Since the newly joined node v is not a nearest neighbor of node u, no changes are required in $RNG_i(u)$ and $GNG_i(u)$, see the Figure 4.16.

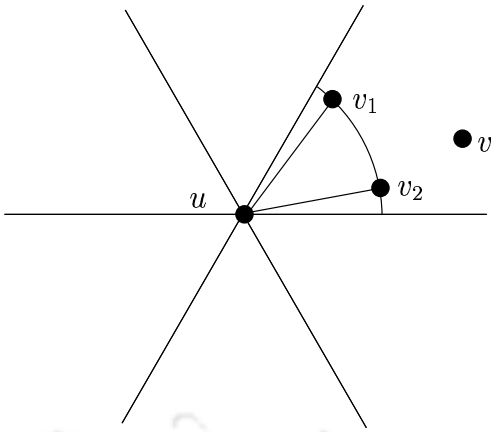


Figure 4.16: Case 1: Node v joins the network and $v_1 \in RNG_i(u)$.

Case 2 ($v_1 \in RNG_i(u)$ and $|\overline{uv_1}| = |\overline{uv}|$): In this case, update the $GNG_i(u)$ by $GNG_i(u) = GNG_i(u) \cup \{v\}$. Add the node v to $RNG_i(u)$ if the $lune(u, v)$ is empty, see the Figure 4.17.

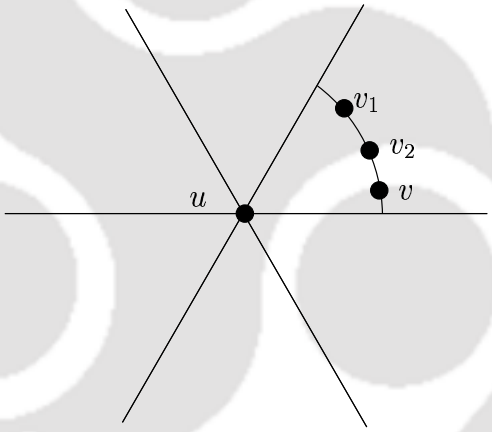


Figure 4.17: Case 2: Node v joins the network and $|\overline{uv_1}| = |\overline{uv}|$.

Case 3 ($v_1 \in RNG_i(u)$ and $|\overline{uv_1}| > |\overline{uv}|$): Delete all the nodes in $GNG_i(u)$ and $RNG_i(u)$, as v becomes the nearest neighbor. Add the newly joined node v to GNG and add v to $RNG_i(u)$, if $lune(u, v)$ is empty, see the Figure 4.18. For each other sector $j \neq i$, check the lune test for all the RNG edges. That is, for each $w \in RNG_j(u)$, if $v \in lune(u, w)$ then remove w from $RNG_j(u)$.

The Node Join algorithm uses the following messages: The message *Njoin_Packet* and *Nnode_Packet* carries the node id. The message *hello_packet* contains node id and position information. The formal algorithm is given below.

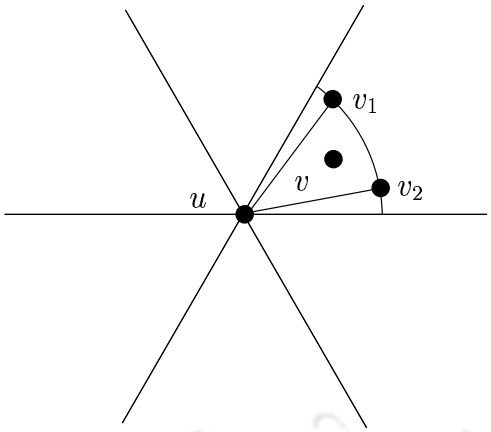


Figure 4.18: Case 3: Node v joins the network and $|\overline{uv_1}| > |\overline{uv}|$.

Algorithm: Node Join in RNG

If a node v joins in the sector i of node u

(a)If($v_1 \in RNG_i(u)$ and $|\overline{uv_1}| = |\overline{uv}|$)

Add v to $GNG_i(u)$.

Add v to $RNG_i(u)$ if $lune(u, v)$ is empty.

If($v_1 \in RNG_i(u)$ and $|\overline{uv_1}| > |\overline{uv}|$)

Delete all the nodes in $GNG_i(u)$ and $RNG_i(u)$.

Add v to $GNG_i(u)$.

Add v to $RNG_i(u)$ if $lune(u, v)$ is empty.

(b) For each sector $j \neq i$

For each $w \in RNG_j(u)$

If($v \in lune(u, w)$)

Delete w from $RNG_j(u)$.

Each node u, which has edge $\overline{uv}$ to the newly joined node v, broadcasts this information using the message *Jnnode_packet*. Node v collects this information and updates the topology neighbor lists $\aleph_{DRNG}(v)$ to maintain the graph.

Node v can participate in routing immediately, since it is connected to network by RNG edges without much delay. However, it can not participate in dynamic maintenance immediately, since it does not have neighbors $n_1(v)$ and GNG. Node v keeps updating

$n_1(v)$ and GNG(v) from received *hello_packet*s. Node v completes the updation of $n_1(v)$ within a beacon interval and computes GNG.

Lemma 4.3.4 *The time complexity for a node join in RNG is in $O(\Delta)$ for uniformly distributed nodes, where Δ is the maximum node degree of UDG.*

Proof: The time complexity for a single lune test is in $O(\Delta)$. Since each node has six sectors, there can be constant number of lune tests to update RNG. Thus the time complexity for RNG is in $O(\Delta)$. ∎

Lemma 4.3.5 *The message complexity for a node join in RNG is in $O(1)$.*

Lemma 4.3.6 *The DRNG is a planar graph.*

Proof: In both the cases, node down and join, the reconstruction is done using RNG algorithm. Since RNG is a planar graph, DRNG maintains the planarity. ∎

Lemma 4.3.7 *The spanning ratio of DRNG is in $O(n)$.*

Proof: The spanning ratio of RNG is in $O(n)$. In case of either node down or join, the reconstruction is done using RNG algorithm. Hence, the spanning ratio of DRNG is in $O(n)$. ∎

4.4 Dynamic Gabriel graph (DGG)

Similar to the DLDel and DRNG, DGG first constructs the spanner Gabriel graph (GG) and follows the maintenance of the constructed spanner. The maintenance is done locally, which avoids the global reconstruction.

Each node u collects the 1-hop neighbors $n_1(u)$ and finds Gabriel edges incident on u using $n_1(u)$. An edge $\overline{uv}$ is present in GG if and only if $disk(u, v)$ does not contain any other node from $n_1(u)$, where the $disk(u, v)$ represents the circle with diameter as the distance $|\overline{uv}|$ and centered at $\frac{u+v}{2}$, as shown in Figure 4.19. Dynamic maintenance of GG for a node down and join into the network is discussed in the following subsections.

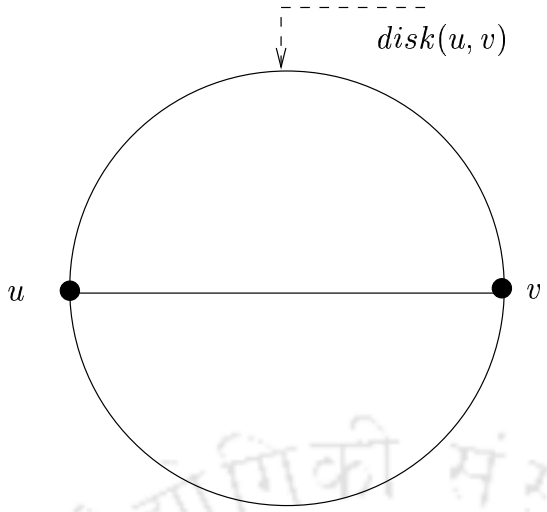


Figure 4.19: Gabriel Rule.

4.4.1 Node Down

Each node v broadcasts the packet *Ndown_Packet* before it goes down and switching off its transmitter. The node u receiving the packet *Ndown_Packet* from the failed node v, reconstruct the GG with 1-hop neighbors $n_1(u)$. If a node does not inform its neighbors before it goes down due to some sudden events, information about the failed nodes is obtained due to the lack of *hello_packet*s from them.

This algorithm uses the following messages: The message *Ndown_Packet* contains a node id. The message *hello_packet* contains a node id and position information. The formal algorithm is given below.

Algorithm: Node Down in GG

---

1. Each node u, after receiving ***Ndown_Packet*** or detects a node down, do the following:

   Extract id from ***Ndown_Packet*** and remove it from $n_1(u)$.

   Construct Gabriel graph with nodes $n_1(u)$.

Lemma 4.4.1 *The message and time complexity for a node down in DGG is in* $O(1)$ *and* $O(\Delta^2)$ *respectively, where* Δ *is the maximum node degree.*

4.4.2 Node Join

When a new node v joins the network or wakes up from sleep mode, it informs its neighbor by broadcasting the packet *Njoin_Packet*. Each node u receiving the message *Njoin_Packet* from node v, reconstructs the spanner GG with its 1-hop neighbors $n_1(u)$. Each node u having the edge $\overline{uv}$ to the joined node v, sends a message *Jnnode_Packet*. When a node v receives the message *Jnnode_Packet*, it adds the edge $\overline{vu}$ to GG.

The node join algorithm uses the messages *Njoin_Packet* and *Jnnode_Packet* that contain the node id. The formal algorithm for node join is as follows.

Algorithm: Node Join in GG

---

1. Each node u, after receiving the ***Njoin_Packet*** from node v, follow the step :
   Extract id and position information from ***Njoin_Packet*** and adds it to $n_1(u)$.
   Reconstruct the GG with the nodes $n_1(u)$.
   If the edge $\overline{uv} \in GG$, broadcast the message ***Jnnode_Packet***.
2. The joined node v, receiving the message ***Jnnode_Packet*** from node u, adds the edge $\overline{vu}$ to $\aleph_{GG}(v)$.

Lemma 4.4.2 *The time and message complexity for a node join is in $O(\Delta^2)$ and $O(\Delta)$ respectively, where Δ is the maximum node degree.*

The node down and up procedures are dynamically maintaining GG without violating any localized GG properties. Hence, follows the lemmas.

Lemma 4.4.3 *The DGG is a planar graph.*

Lemma 4.4.4 *The spanning ratio of DGG is in $O(\sqrt{n})$.*

Lemma 4.4.5 *DRNG $\subseteq$ DGG.*

Lemma 4.4.6 *$\Pi_{DRNG} \geq \Pi_{DGG}$, where Π_G denotes the spanning ratio of graph G.*

Proof: This follows from the fact that $|\Pi_{DRNG(u,v)}| \geq |\Pi_{DGG(u,v)}|$, where $|\Pi_{G(u,v)}|$ denotes the length of shortest path between two nodes u and v in the graph G. ∎

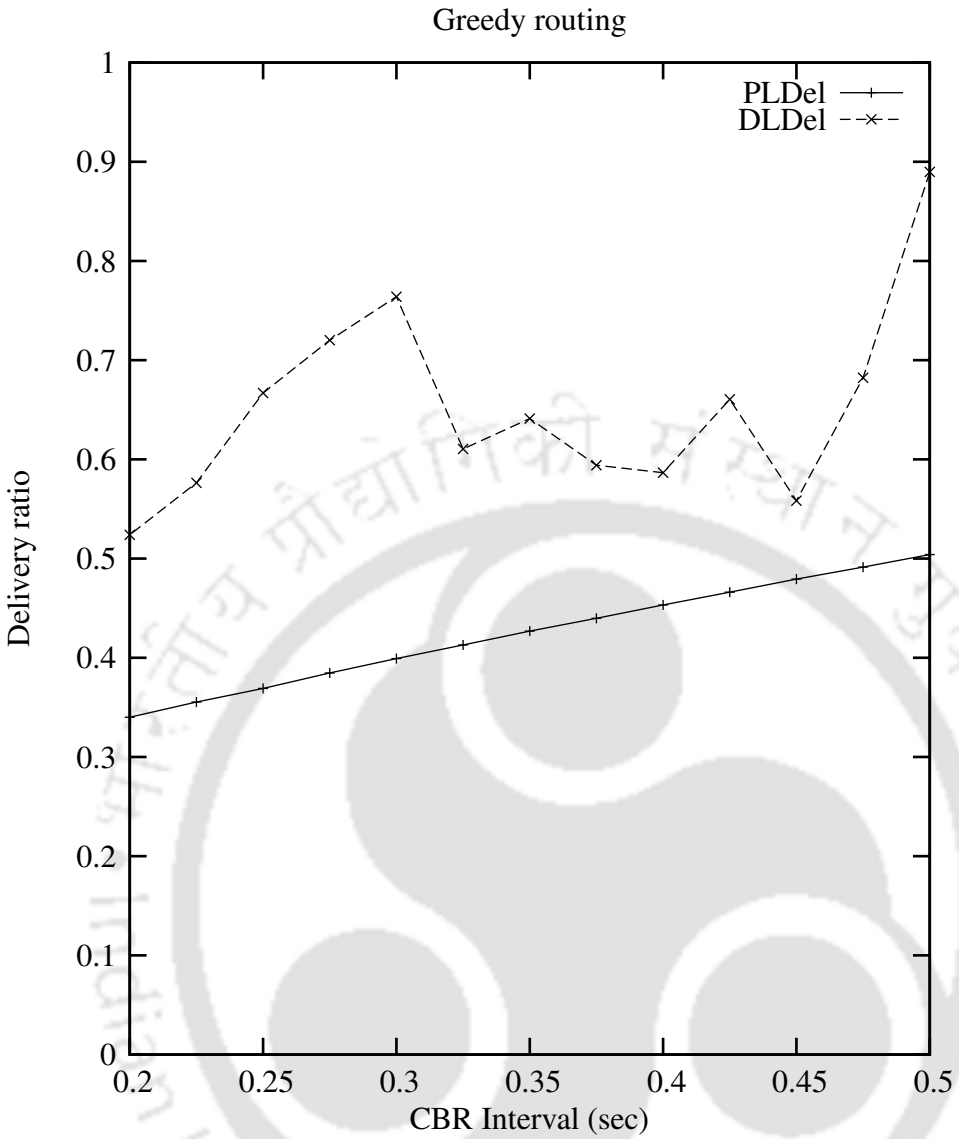


Figure 4.20: Greedy routing.

4.5 Simulation and performance

We have considered only node down, since its impact on delivery ratio is higher than node join. We have simulated the node down by exhausting battery power. That is, a node goes down when its battery power is exhausted. In addition to spanners DLDel, DRNG, and DGG, we have simulated the spanners PLDel, RNG, and GG, in network simulator (ns2.28) [NS205] for comparison.

We have taken 50 nodes, randomly distributed in a 600×600 m^2 square grid. The transmission range of each node is $200m$. The initial energy given is 30 joules. The energy

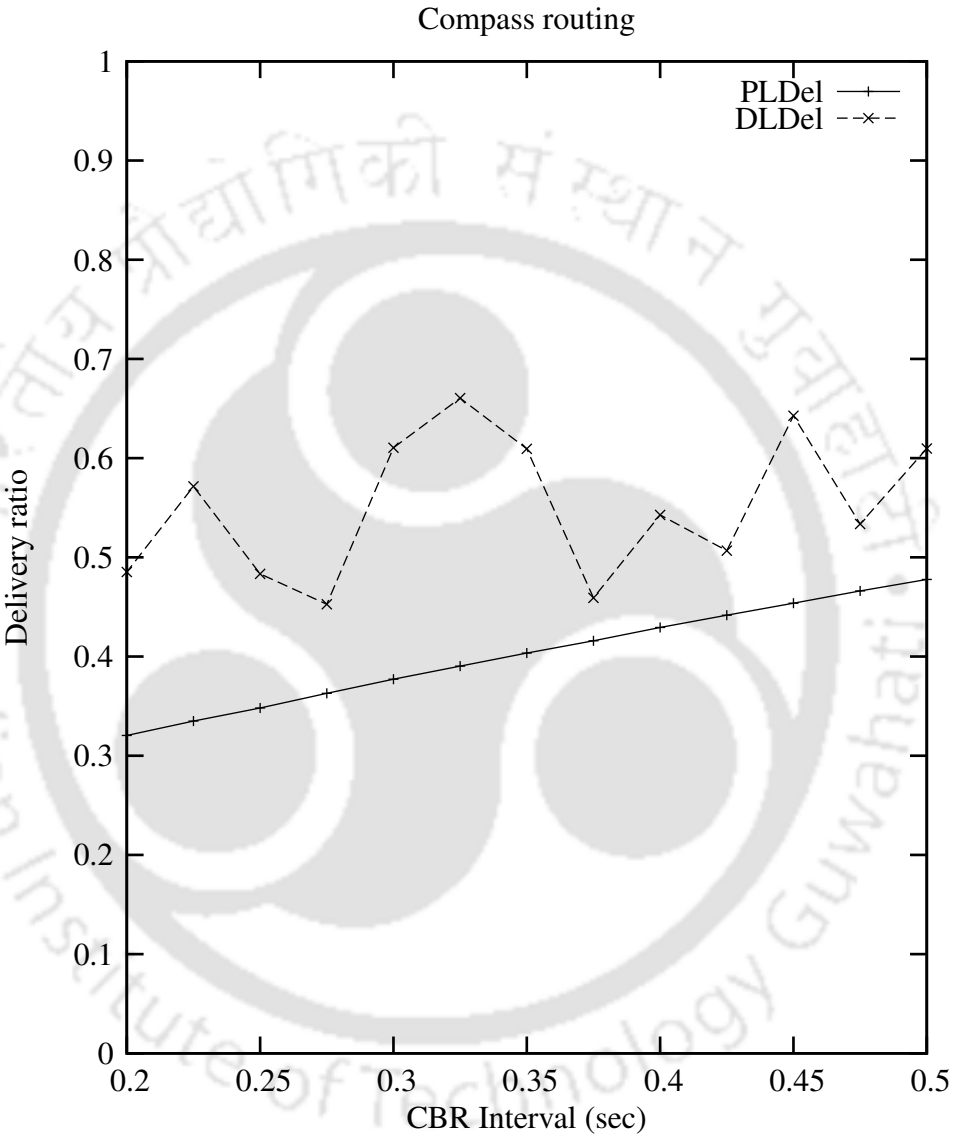


Figure 4.21: Compass routing.

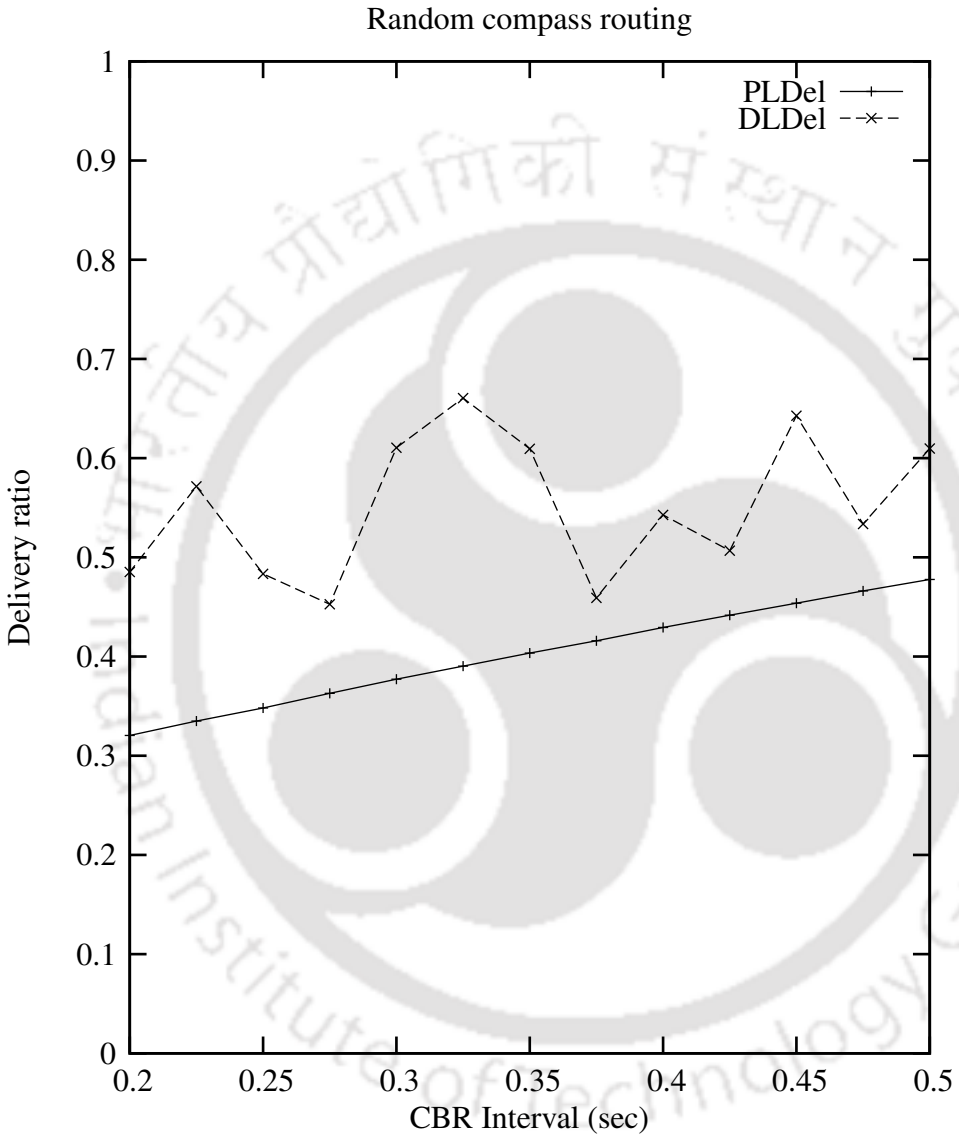


Figure 4.22: Random compass routing.

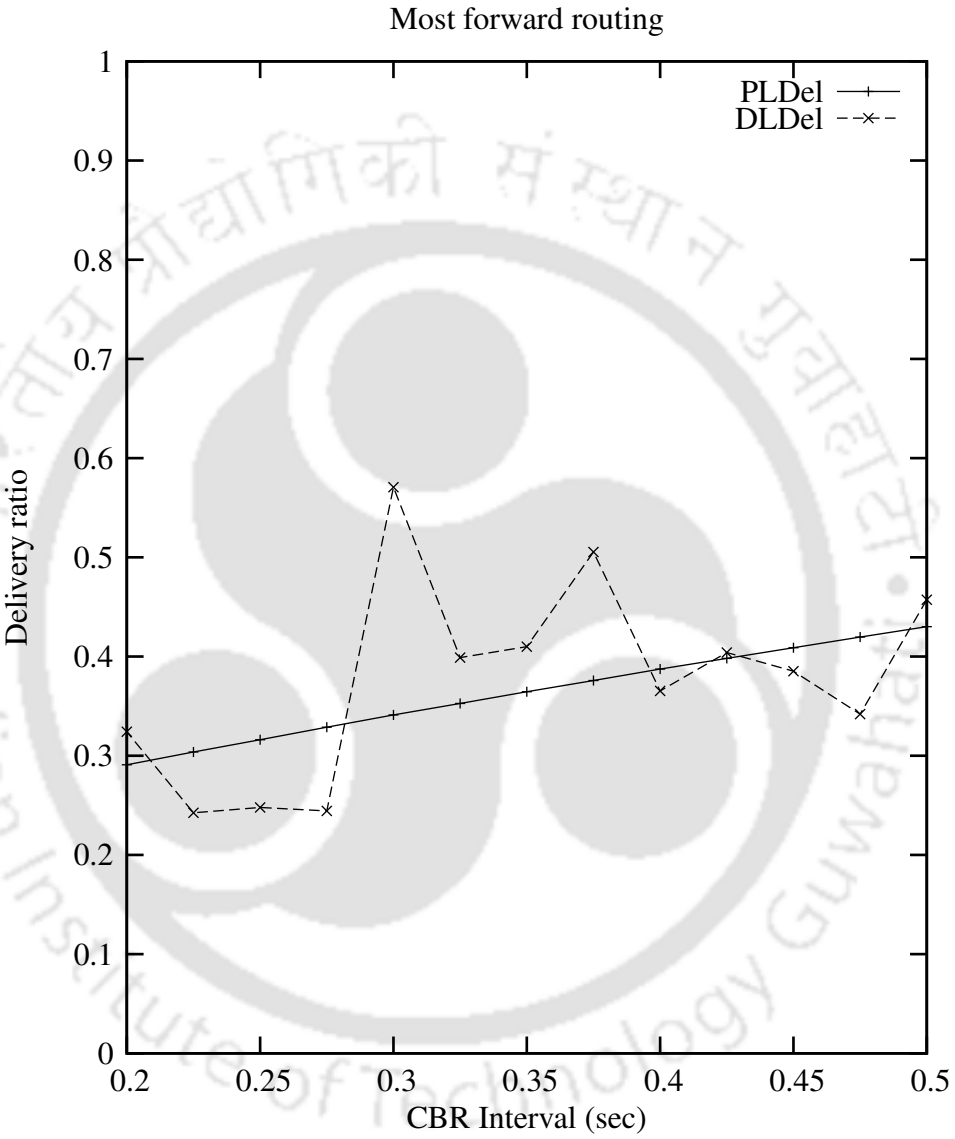


Figure 4.23: Most Forward routing.

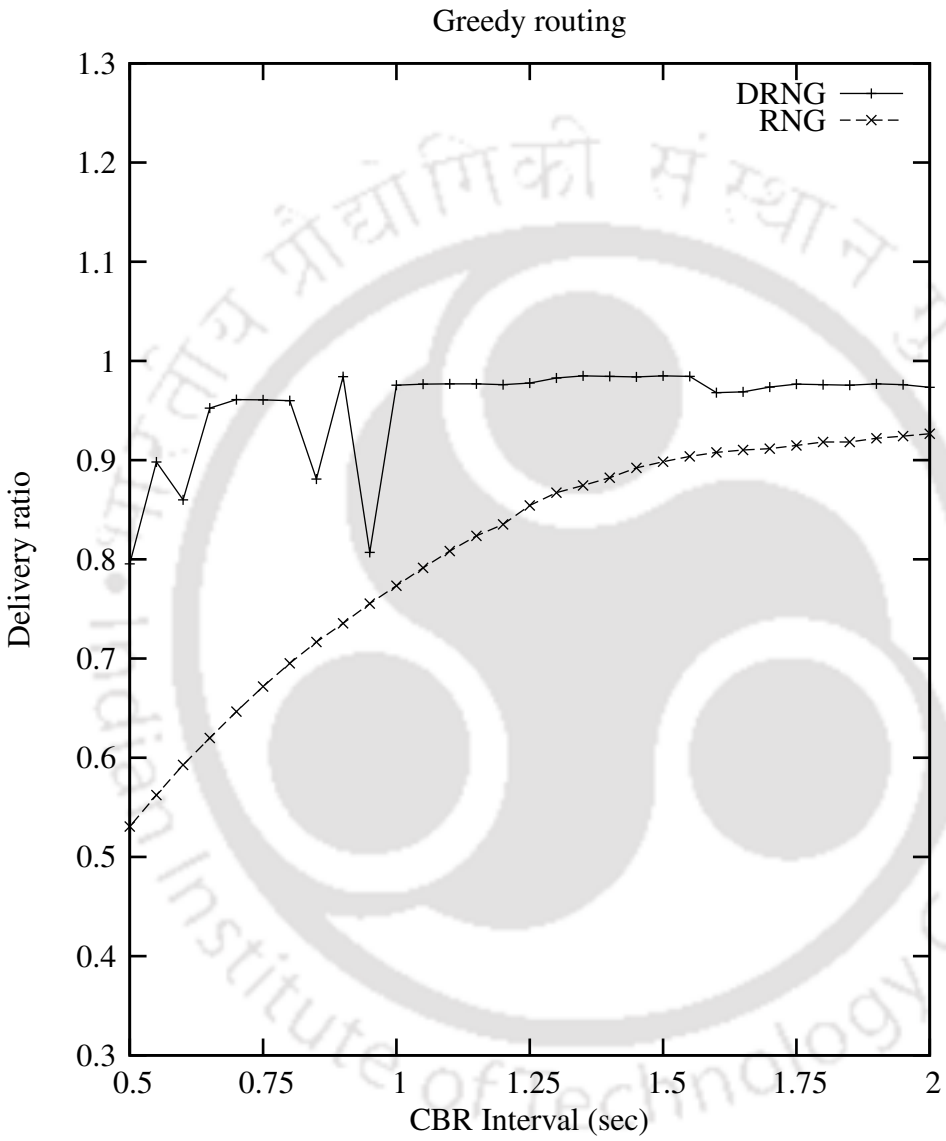


Figure 4.24: Greedy routing.

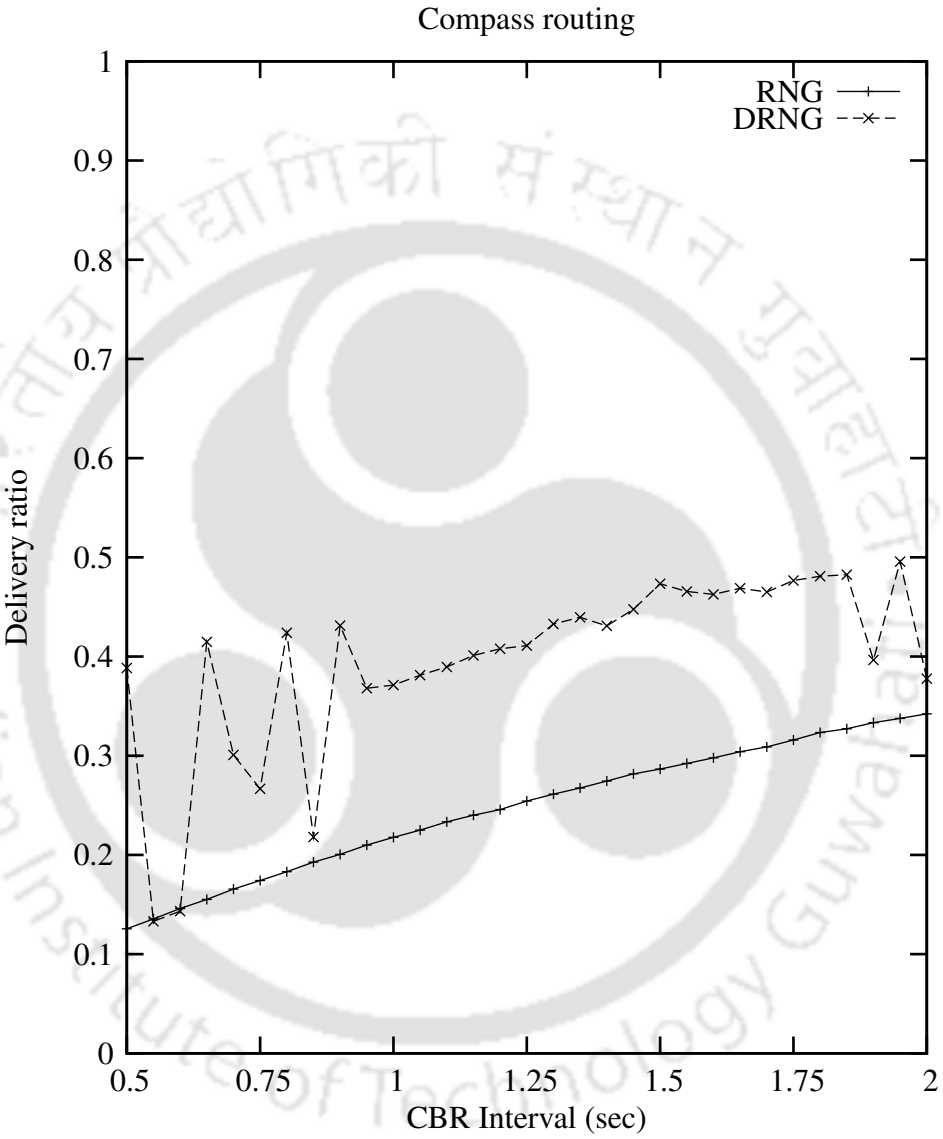


Figure 4.25: Compass routing.

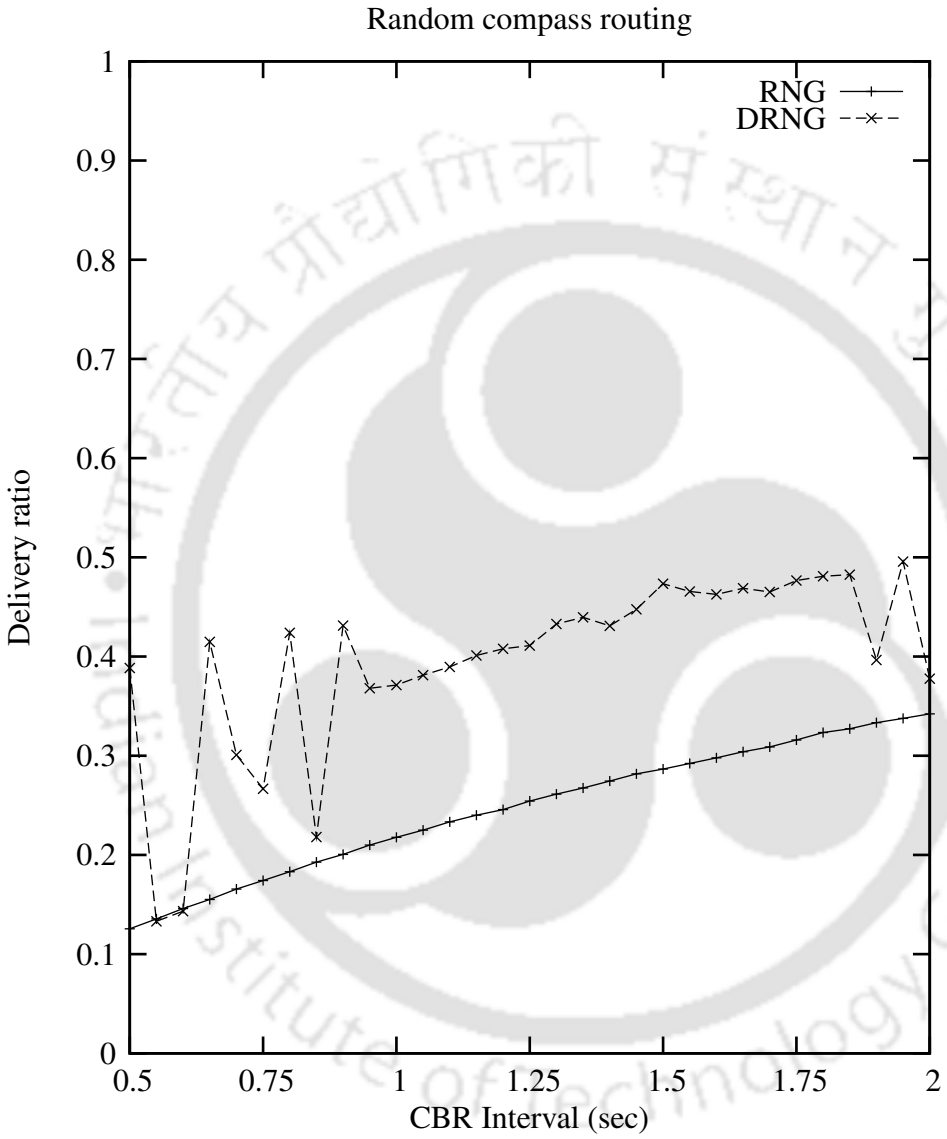


Figure 4.26: Random compass routing.

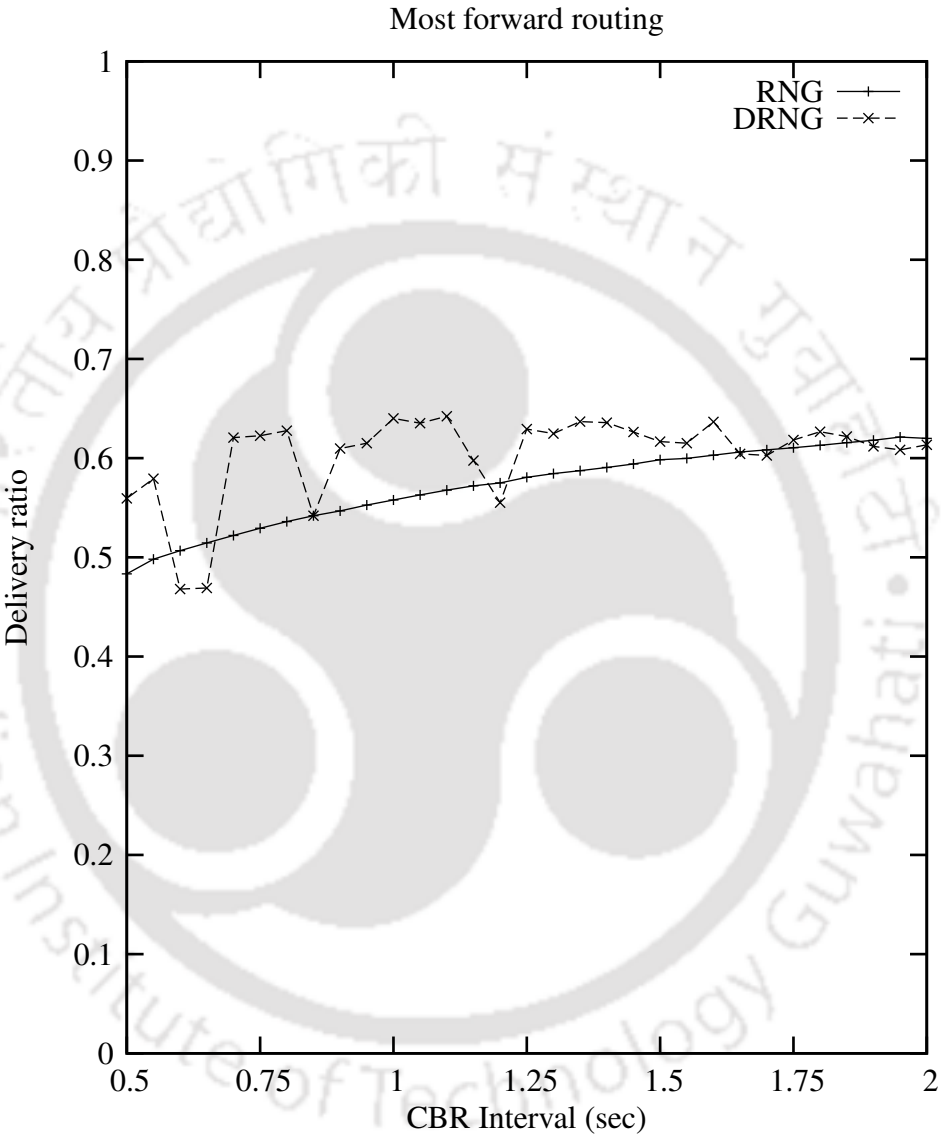


Figure 4.27: Most forward routing.

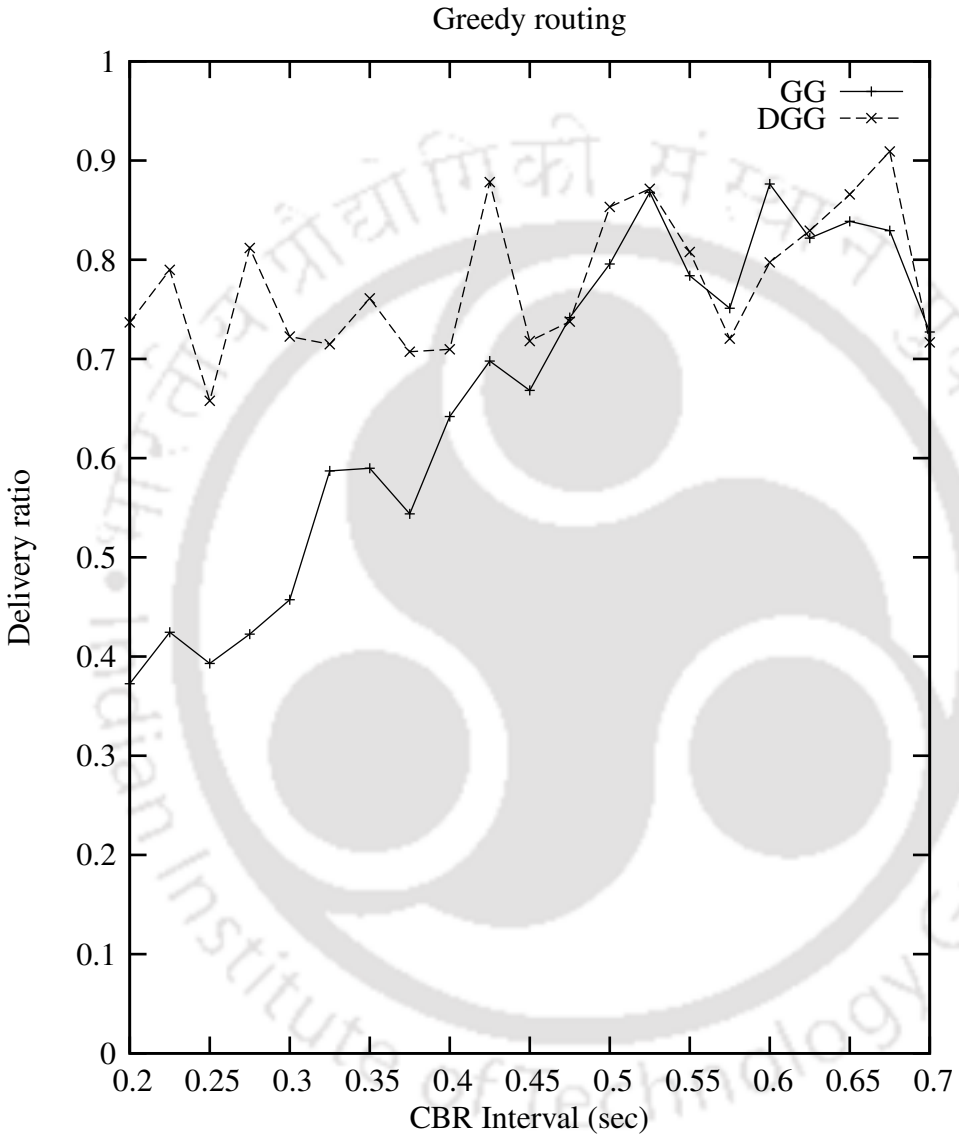


Figure 4.28: Greedy routing.

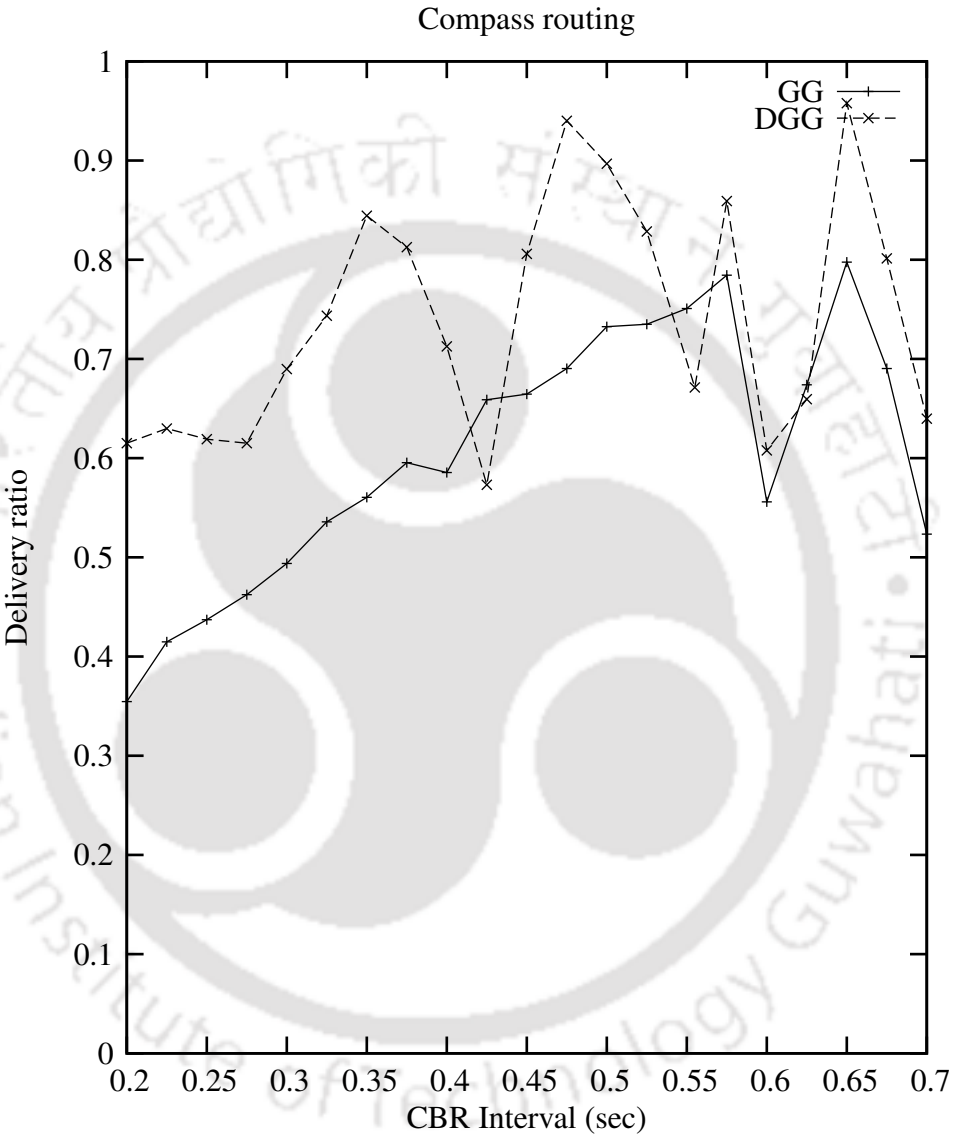


Figure 4.29: Compass routing.

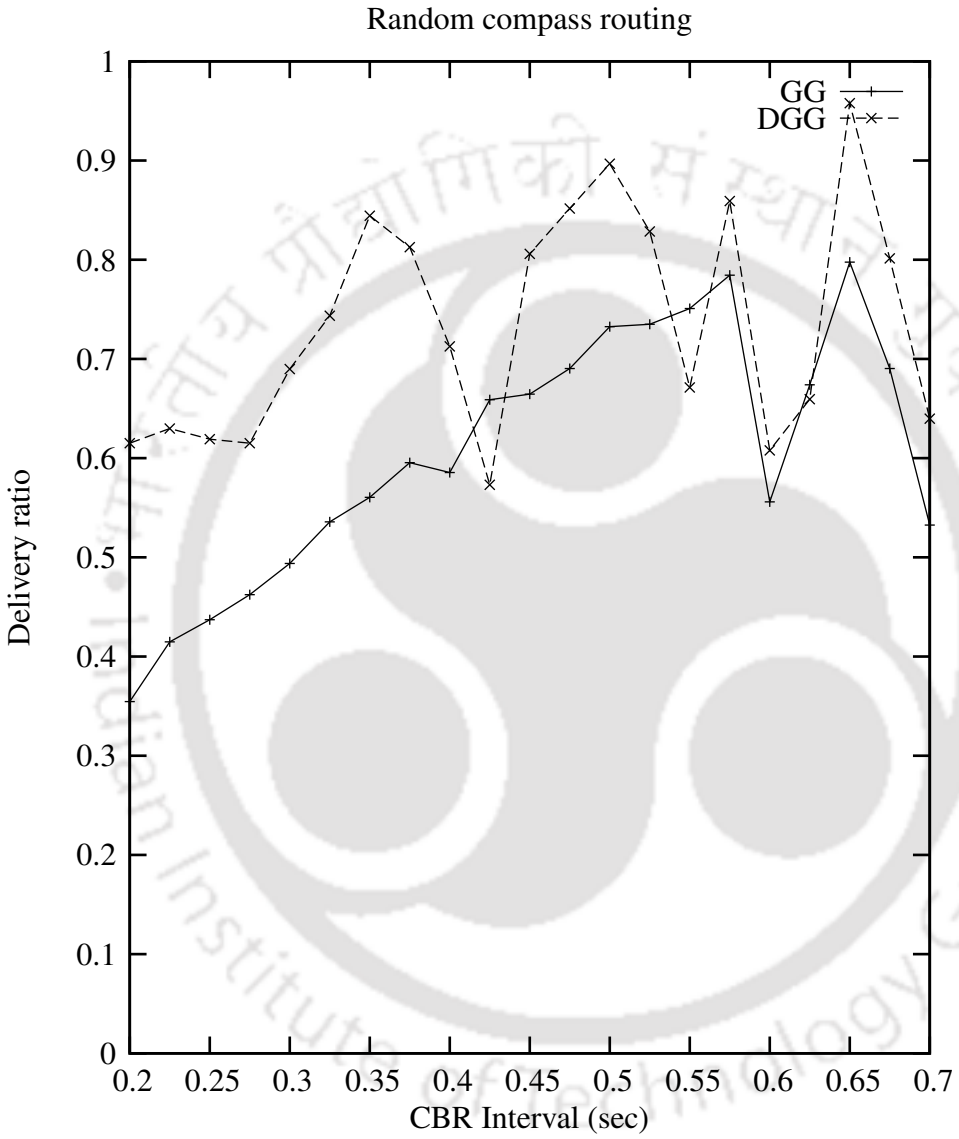


Figure 4.30: Random compass routing.

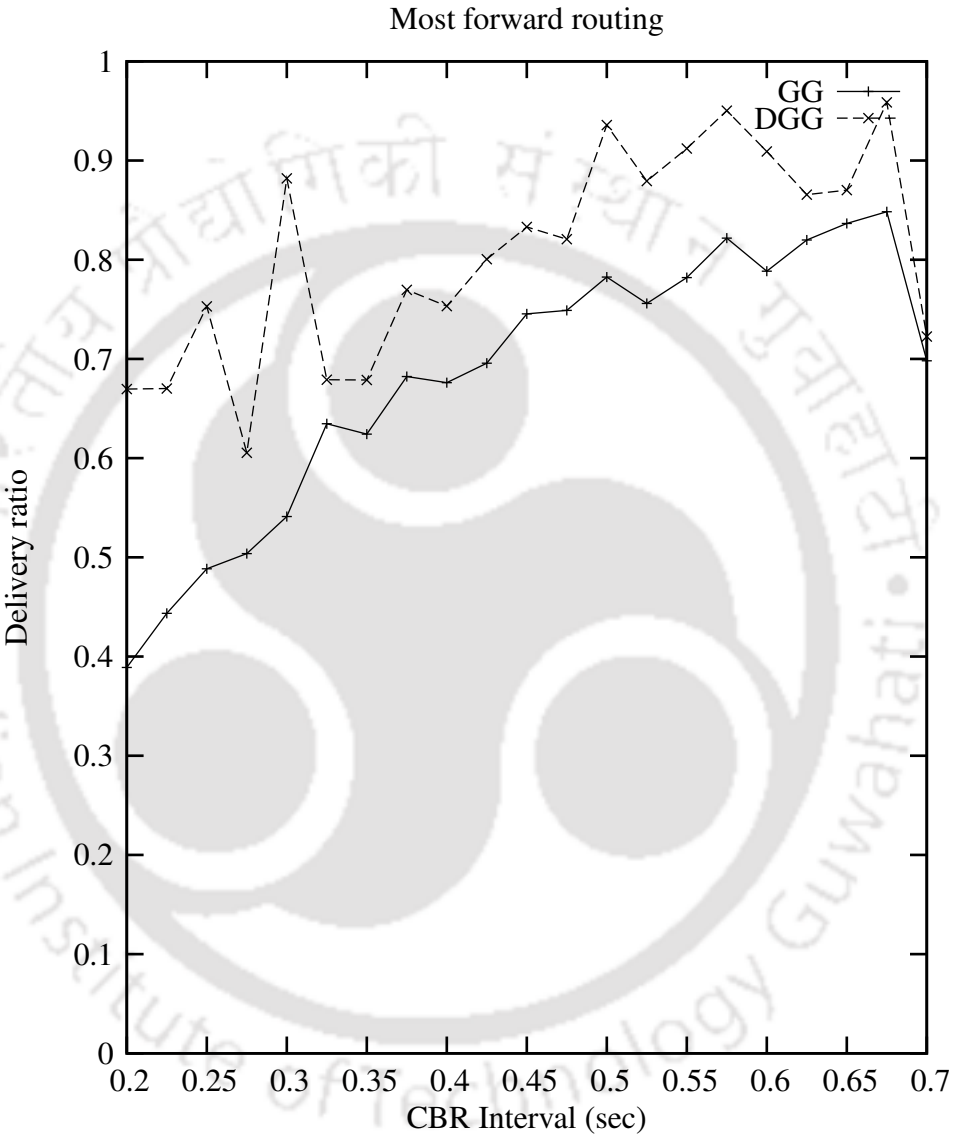


Figure 4.31: Most forward routing.

is consumed for each packet transmission with the transmission power ($txPower\_$) of 0.665 watts. For each packet reception, the energy is consumed with the receiving power ($rxPower\_$) of 0.395 watts. The idle power is 0.035 watts. The total simulation time is 600 seconds. We have taken three different connection patterns where the sources at one side of the grid and the destination nodes are at the other side. The packets of size 512 bytes are sent from source to destination at constant rate.

4.5.1 Simulation results

In the first set of experiments, we evaluate the performance of dynamic local Delaunay triangulation (DLDel). We run four routing protocols Grdy, Cmp, RandCmp, and MFR on two different spanners DLDel and PLDel. We have calculated the average delivery ratio for each of these four routing protocols. As we know that the length stretch factor of PLDel is much higher than RNG and GG, the total number of hops in RNG and GG will be higher. So, the less power consumption occurs in PLDel compared to RNG and GG spanners. Hence, we have considered higher transfer rates of the CBR packets in DLDel than in DRNG, and DGG. From the figures Figure 4.20, Figure 4.21, Figure 4.22, and Figure 4.23, we can say that the delivery ratio of DLDel is higher than PLDel. This happens because when a CBR packet is transfered from a source node to a destination node, all the relay nodes in the routing path need to forward the packets. Thus, for each packet transmission and reception the energy model decreases the remaining energy in the battery of all the relay nodes. After several CBR packet transfers, the energy in the battery drains and node failures occur. These node failures are detected in the dynamic spanners before the node dies and the protocol reconstructs the network graph locally. Thereafter, the routing algorithm forward the data packets through another route and recovers the packet loss. On the other hand, PLDel does not reconstruct the network graph and hence packet loss occurs.

In the second set of experiments, we evaluate the performance of DRNG. In this experiment, we run these four different routing protocols on DRNG and RNG. We have computed the average delivery ratio of the above routing protocols. Since the spanning ratio of RNG is smaller than GG, the total number of hops in RNG will be higher. So, more power consumption occurs in RNG spanner. Hence, we have considered smaller

transfer rates of the CBR packets in DRNG than in DGG. From the figures Figure 4.24, Figure 4.25, Figure 4.26, and Figure 4.27 we can say that the delivery ratio of DRNG is higher than RNG, because more packet loss occurs when the nodes are failed in the RNG, where as DRNG recovers it dynamically.

Finally, we evaluate the performance of dynamic Gabriel graph (DGG). In this process, we run the four routing protocols Grdy, Cmp, RandCmp, and MFR on two different spanners DGG and GG. In each simulation, the average delivery ratio is calculated. From the Figure 4.28, Figure 4.29, Figure 4.30, and Figure 4.31 we say that the delivery ratio of the routing protocols on the spanner DGG is much better than GG. This happens because, once the battery of a node is down, the edges in GG will no longer exists. Hence, heavy packet loss occur in GG, where as it is recovered in DGG because of its dynamic nature.

Though the delivery ratio of dynamic spanners is higher than their respective static spanners, the delivery ratio values in the dynamic spanners are irregular. This happens because, after reconstructing the spanner due to node failure, the hop count in the new routing path is different from the hop count prior to the node failure. Thus, the variation in the number of transmissions and routing path lead to variation in the delivery ratio. In addition, the reconstruction process due to the node failure requires additional time. During this period, the data packets sent may be lost which leads to the irregularity in delivery ratio.

In the next experiment, we have calculated the number of node failures occurred during the data transmission. The Table 4.1 shows the number of node failures for various routing protocols. From the table, we can observe that the number of node failures for all protocols is in the order of DLDel $\leq$ DGG $\leq$ DRNG. Since PLDel $\supseteq$ GG $\supseteq$ RNG, the path length in PLDel is smaller and has fewer intermediate relay nodes. When data packets are transmitted through these relay nodes, the energy is drained from the battery and the node failures occur. In other words, for the spanners RNG and GG, the number of relay nodes in the routing path are high due to their lower edge density. Thus, the order of node failures are in DLDel $\leq$ DGG $\leq$ DRNG.

In the next experiment, we have computed the end-to-end packet delay. The end-to-end packet delay of a single data packet is the time duration between the packet start time at the source node and the packet received time at the destination node. We carried the simulation for greedy routing algorithm on spanners DLDel and PLDel. The Figure 4.32

	DLDel	DGG	DRNG
Gdr	6	8	8
Cmp	8	9	20
Rcmp	8	9	20
Mfr	6	8	8

Table 4.1: Nodes failures.

shows the average packet delay at various CBR intervals. From the plot, we can observe that the average packet delay in DLDel does not have substantial improvement over the spanner PLDel. This is because we consider only the packets that are successfully transmitted to the destination node. Due to the node failures, the packets that are dropped in between the source and destination are neglected. However, for a successfully received packet, the delay difference in both the spanners is not high as the packets traverse through the Delaunay edges in both the spanners. Similarly, from the Figure 4.33, Figure 4.34, the delay difference between DRNG and RNG (or) DGG and GG is not much.

4.6 Conclusions

In this chapter, we have studied the dynamic maintenance of PLDel, RNG, and GG. Simulation results consolidate our claim that the spanners DLDel, DRNG, and DGG recover quickly and prevent packet loss occur in PLDel, RNG, and GG, respectively, due to frequent node failures. Our approach preserves their geometric properties by dynamically reconstructing the graph. Similarly, one can study dynamic maintenance of other geometric graphs like Yao, PDT, and RDG.

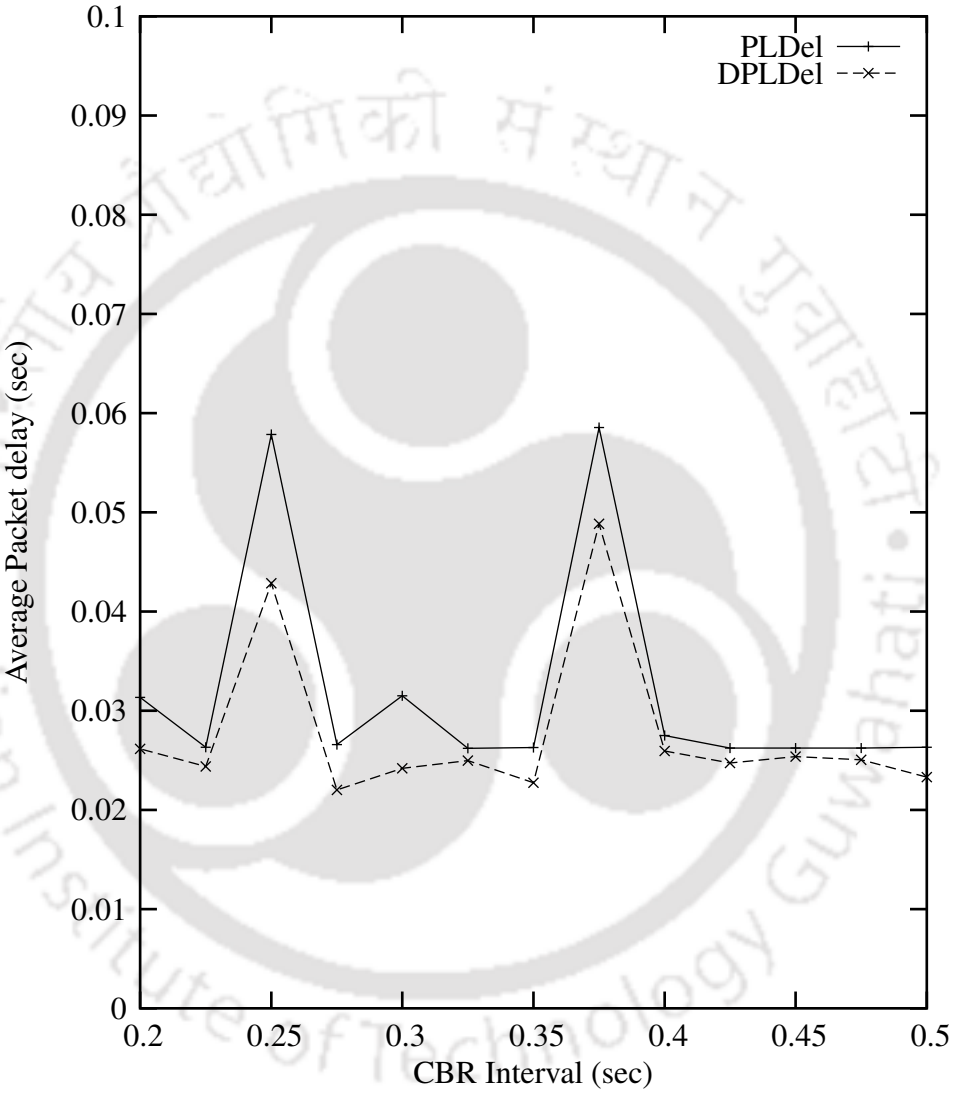


Figure 4.32: Delay in DLDel.

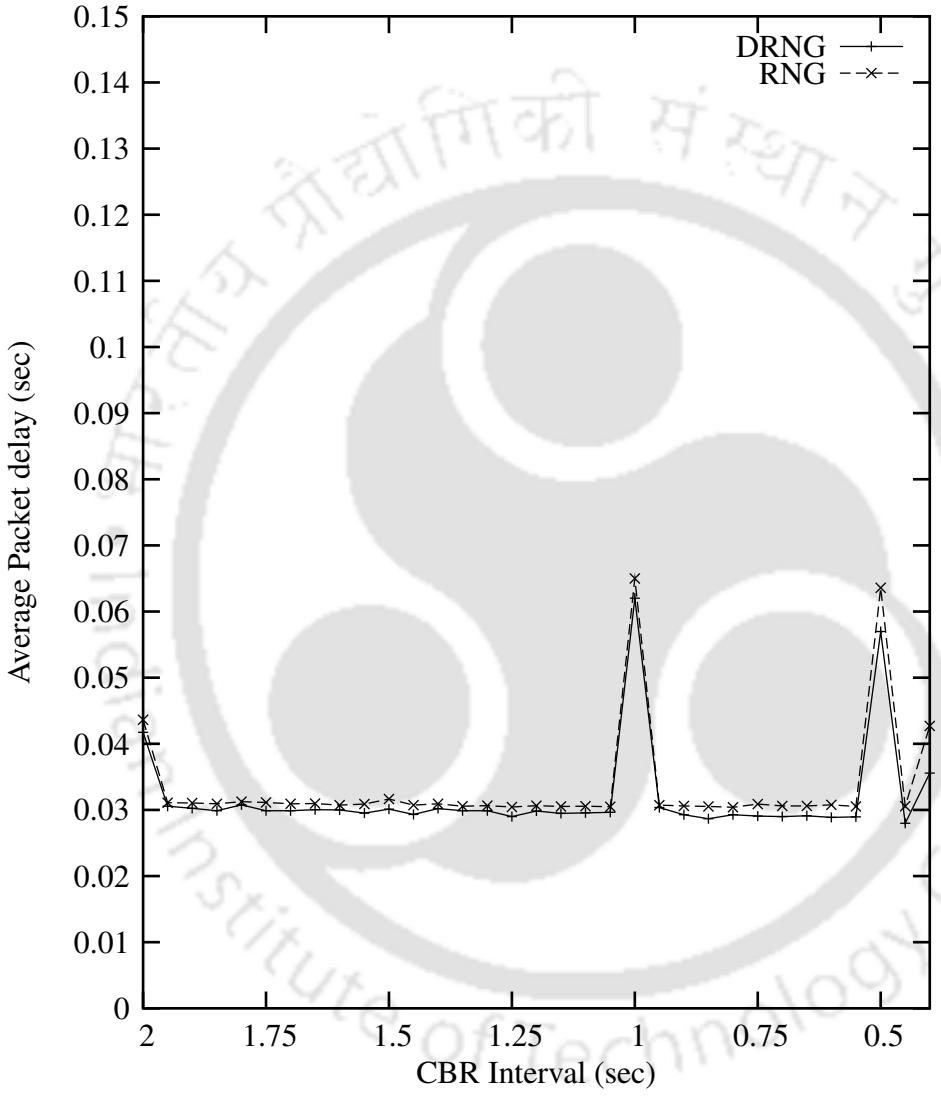


Figure 4.33: Delay in DRNG.

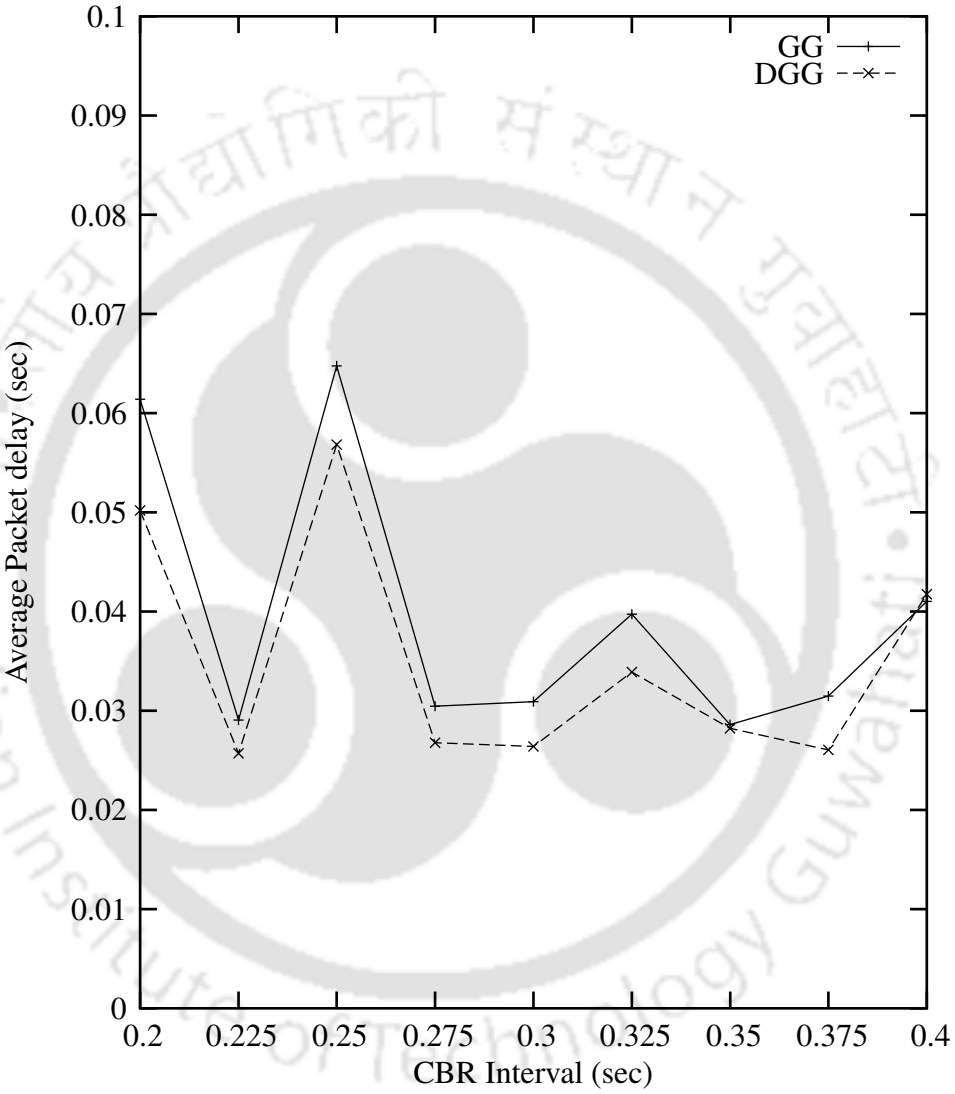


Figure 4.34: Delay in DGG.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 5

Fault Tolerant Spanners

5.1 Introduction

Various resource limitations and environmental constraints make frequent link and node failures in adhoc networks, which make the networks unreliable. For example, the edge disconnections occur due to buildings, walls, mountains, and obstacles between the wireless nodes. Similarly, the node failures occur due to the exhausted battery power, accidents, landslides, debris, eruption of volcano, and cyclones. So, network topology should be fault tolerant to take care of these failures. In the literature, two types of fault tolerant spanners are considered [LNS02, CZ03, Luk99], namely, k-node fault tolerant spanners (k-NFTS) and k-link fault tolerant spanners (k-LFTS). The definitions of the k-fault tolerant spanners are as follows. A graph G=(V, E) is called a k-node fault tolerant t-spanner for the given set of nodes V, if for any subset V' of V of size at most k, the graph G/V' is a t-spanner for the node set V/V'. Similarly, the graph G=(V, E) is called k-link fault tolerant t-spanner for V, if for any subset E' of E of size at most k and for any pair of nodes a and b in V, the shortest path length between a and b in the graph G/E' is at most t times the length of the shortest path between a and b in graph K_V/E'. Here, K_V denotes the complete graph with the node set V. The notation G/V' denotes the graph with the vertex set V/V', and edge set E that have both end points in V/V'. In adhoc networks, one can use UDG(V) instead of K_V. However, the computational complexities for constructing these spanners are in $O(n \log n)$. But these are centralized algorithms, which are not suitable for adhoc networks.

Another way of overcoming the frequent node failure is to consider more stable nodes in the network graph. So, we consider a fault tolerant spanner for adhoc network as a stable spanner, which depends on the stability of nodes, also called as stability based spanners.

In this chapter, we have considered the node failure which is frequent in battlefield and disaster relief applications and have proposed various stable spanners called fault tolerant spanners. One way of overcoming this problem due to the node failures is to update the spanners dynamically as discussed in the previous chapter. However, these frequent updates result in communication and computational overhead. If we know the nodes which are going to be failed or unstable before hand, we can avoid such nodes as a part of spanners to overcome this problem. The node stability depends on various parameters such as power level, mobility, its environment, and usability. This chapter considers more stable nodes for constructing the geometric spanners, called fault tolerant spanners.

The proposed fault tolerant spanners are called fault tolerant local Delaunay triangulation (FTLDel), fault tolerant relative neighborhood graph (FTRNG), and fault tolerant Gabriel graph (FTGG). The spanners, FTLDel, FTRNG, and FTGG are simulated using network simulator, *ns-2.28* [NS205], and shown better performance than their counter parts PLDel, RNG, and GG, respectively.

The rest of the chapter is organized as follows: In the next section, the fault tolerant spanner FTLDel is described. In the Section 5.3, the spanner FTRNG and its properties are presented. The spanner FTGG is described in the Section 5.4. In the Section 5.5, the simulation details and results are given. Finally, the chapter is concluded in the Section 5.6.

5.2 Fault tolerant local Delaunay triangulation (FTLDel)

The FTLDel construction algorithm has two stages. In the first stage, we check the node's stability factor and select a subset of nodes which are stable. In the second stage of the algorithm, the planarized local Delaunay triangulation is constructed with the nodes chosen in the first stage.

The system model considered for fault tolerant spanners is asynchronous wireless ad hoc networks. Each node in the network has unique identification number and is aware of its location information through any positioning techniques. The omni-directional antenna

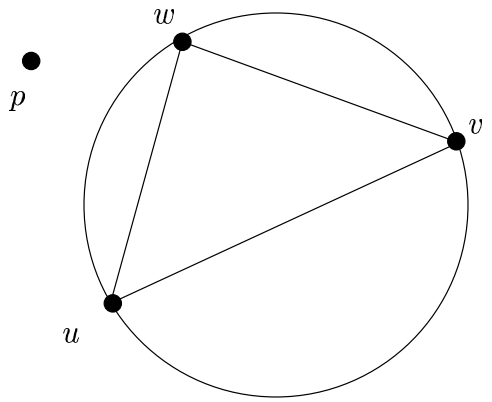


Figure 5.1: Delaunay triangle.

is used by the transceiver. All nodes in the network has same transmission range. The network has reliable communication, where a single packet transmitted is received by all the nodes in its vicinity. Each node has the facility to measure the remaining energy in its battery.

5.2.1 Stability factor

In order to maintain stable spanner under high topological changes, the stability factor is employed to mark the stability of each node in the network. The stability of a node represents the node's robustness. The node stability is measured with the help of various parameters depending on the application and environment. For example, the node's stability can be computed with the parameters: the node's battery power level, node mobility, location of the node, and usability of the node.

In the literature, the stability factor is computed in various ways depending on the need. For example, Yao Yu et al. [YZD09] computed the stability factor using the parameters link quality and mobility of a node. Charu et al. [UG09] considers the parameters mobility and energy for calculating the stability factor, where as Meenu et al. [CSJS08] has considered only mobility. Liu et al. [Liu09] has taken the mobility for calculating the stability factor, where as Punde et al. [PPM03] has considered both mobility and packet processing ratio.

However, in this thesis, we have considered the remaining battery power for calculating the stability factor. The probability of node failure is high if its battery power is low.

For a node with high percentage of remaining power (PRP), the probability of node failure is low. Hence, the stability factor increases exponentially with the remaining battery power as shown in the Figure 5.2. The normalized stability factor on a 10 point scale is given by the formula:

$$\text{Stability factor (SF)} = \frac{10}{\log_2 100} * \log_2(PRP)$$

where PRP is the percentage of remaining battery power.

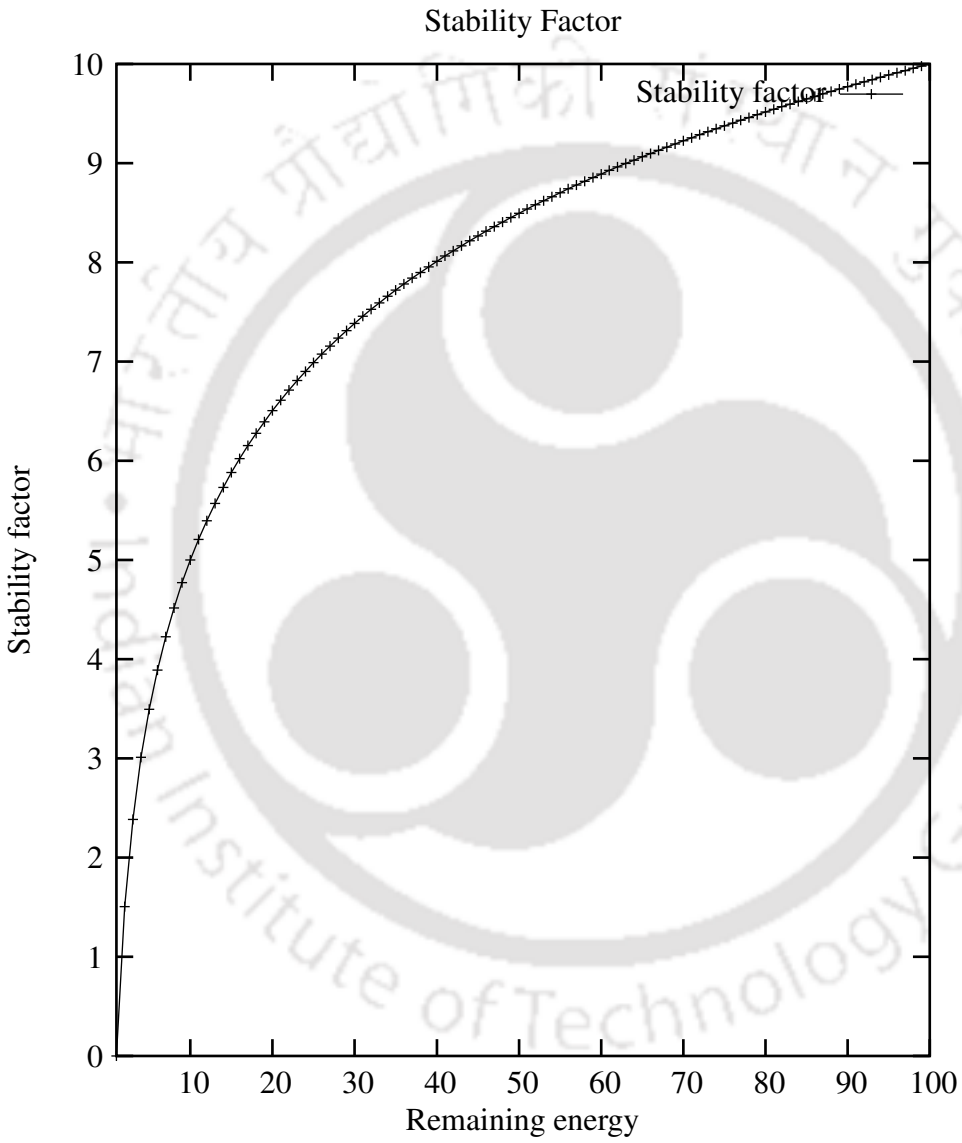


Figure 5.2: Stability Factor.

Each node finds its stability factor using the above formula and checks for its suitability to participate in the spanner. The suitability depends on the threshold value. If the stability factor is more than the threshold value then the node participates in the spanner

construction otherwise it does not participate. However, the fixing of the threshold value depends on the actual communication pattern and applications. For example, for large data transfer the threshold should be high. Nodes, which are not suitable, do not participate in spanner by switching off their transmitter. Each node u gathers one hop stable neighborhood information $n_1(u)$ from the broadcasted *hello_packet*s. Construct the PLDel with the stable nodes as follows: first each node u computes $Del(n_1(u))$. For any nodes u, v, and w belong to $n_1(u)$, the triangle $\triangle uvw \in Del(n_1(u))$ if and only if the circumcircle of the triangle $\triangle uvw$ does not contain any other node p from $n_1(u)$, as shown in the Figure 5.1. To test whether a node p is inside the circumcircle of three nodes u, v, and w or not, we can use the following determinant, where u_x and u_y represent x and y coordinates of the node u.

$$\begin{vmatrix} u_x & u_y & u_x^2+u_y^2 & 1 \\ v_x & v_y & v_x^2+v_y^2 & 1 \\ w_x & w_y & w_x^2+w_y^2 & 1 \\ p_x & p_y & p_x^2+p_y^2 & 1 \end{vmatrix}$$

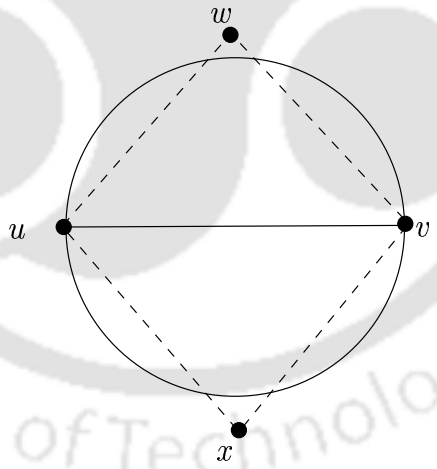


Figure 5.3: $\overline{uv}$ is Gabriel edge.

Find all the Gabriel edges from $Del(n_1(u))$, which will never be deleted. An edge $\overline{uv}$ is called Gabriel edge, if the edge $\overline{uv}$ is common edge for two triangles $\triangle uvw$ and $\triangle uvx$, and the angles $\angle uwv$ and $\angle uxv$ must be less than or equal to 90°, as shown in Figure 5.3. Find all the consistent triangles. A triangle $\triangle uvw$ is called consistent if the $\triangle uvw$ is in $Del(n_1(u))$, $Del(n_1(v))$, and $Del(n_1(w))$. The resulting graph is called LDel[1], which is

not a planar graph. For each consistent triangle $\triangle uvw$, compute the incircle test with all LDel[1] nodes of its 1-hop neighbors. If the circumcircle of the nodes is not empty then remove the triangle $\triangle uvw$ from the graph and the resulting graph is PLDel. The algorithm FTLDel requires only the control messages used for constructing the PLDel.

Algorithm: FTLDel

---

1. Each node u checks, if the *stability* is less than the *threshold* then the node u switches off its transmitter.
2. Each node u broadcasts its ID and location information through ***hello_packet***.
3. Each node u collects 1-hop neighborhood information, $n_1(u)$, through the ***hello_packets*** sent in the step2.
4. Each node u computes Delaunay triangulation of $n_1(u)$, $Del(n_1(u))$. Find all the Gabriel edges which will never be removed.
5. Each node u finds the consistent triangles. A triangle $\triangle abc$ is consistent if $\triangle abc \in Del(n_1(a))$, $\triangle abc \in Del(n_1(b))$, and $\triangle abc \in Del(n_1(c))$.
6. Each node u broadcasts all the edges of consistent triangles and Gabriel edges.
7. Remove the consistent triangles which have the edge crossings with the edges sent in the previous step.

One can easily verify the following properties of FTLDel.

Lemma 5.2.1 *FTLDel* $\subseteq$ *PLDel*

Lemma 5.2.2 *The FTLDel is a planar graph.*

Proof: This follows from FTLDel $\subseteq$ PLDel and PLDel is a planar graph. ∎

Lemma 5.2.3 *The message complexity for constructing FTLDel is in* $O(n)$.

Lemma 5.2.4 *The spanning ratio of FTLDel is* 2.5.

Proof: The length of the shortest path between two nodes p and q in FTLDel is t times the length of the shortest path between the same pair of nodes in UDG of the same vertices, where t is the spanning ratio, which is 2.5 in case of PLDel. Thus the spanning ratio of FTLDel becomes 2.5. ∎

5.3 Fault tolerant relative neighborhood graph (FTRNG)

The spanner FTRNG is constructed in two phases. In the first phase, a subset of the given nodes, which are stable, are selected. In the second phase, the relative neighborhood graph (RNG) [BDEK02] is constructed on nodes computed in the first phase. The resulting graph is called fault tolerant relative neighborhood graph (FTRNG). The formal algorithm is given below.

Initially each node checks its stability factor. If the stability factor is below the critical threshold, it switches off its transmitter and does not participate in the spanner. Hence, only a subset of nodes exits in the spanner construction. Each node u gathers one hop neighborhood information $n_1(u)$ from the broadcasted *hello_packet*s, computes the RNG edges, and add them to FTRNG. Edge $\overline{ua_i}$ is added to FTRNG if the $lune(u, a_i)$ is empty, as shown in Figure 5.4. The algorithm FTRNG does not require any extra control messages except the messages required for constructing the RNG. Formal algorithm is given below.

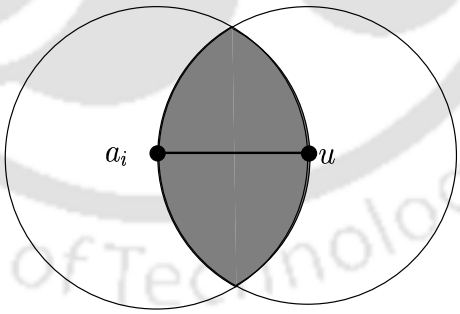


Figure 5.4: RNG Neighborhood relation.

Algorithm: FTRNG

1. Each node u switch off its transmitter if($stability \leq threshold$).

2. Broadcast ***hello_packet*** with ID and location information.

3. Node u receiving ***hello_packet*** stores the information in $n_1(u)$.

4. Each node u follow the steps below

   for each node $v \in n_1(u)$

   if ($lune(u, v)$ does not contain any node in $n_1(u)$)

   add edge $\overline{uv}$ to FTRNG.

One can easily verify the following properties of FTRNG.

Lemma 5.3.1 *FTRNG $\subseteq$ RNG.*

Lemma 5.3.2 *The FTRNG is a planar graph.*

Proof: This follows from FTRNG $\subseteq$ RNG and RNG is planar graph. ∎

Lemma 5.3.3 *The spanning ratio of FTRNG is in $O(n)$.*

Proof: The length of the shortest path between two nodes in FTRNG is t times the length of the shortest path between the same nodes in UDG, where t is the spanning ratio. In worst case, it becomes $O(n)$. Thus, FTRNG spanning ratio is in $O(n)$. ∎

Lemma 5.3.4 *The message complexity for constructing the FTRNG is in $O(n)$.*

5.4 Fault tolerant Gabriel graph (FTGG)

The similar procedure is followed in FTGG construction. In the first phase of the FTGG spanner construction, a subset of the given nodes is selected based on their stability factor. These subsets of nodes are strong enough to function for a longer period. In the second phase of the algorithm, the Gabriel graph (GG) [BDEK02] is constructed on these subset of nodes. The resulting graph is called fault tolerant Gabriel graph (FTGG). The formal algorithm is given below. In this algorithm, initially the unstable nodes are switched off and will not be participated in the network graph. The Gabriel graph edges are calculated and

added to FTGG. The step 4 of FTGG algorithm describes the GG construction procedure with one hop neighborhood information. Please note that the $disk(u, v)$ represents the circle with diameter $|\overline{uv}|$ centered at $\frac{u+v}{2}$, as shown in the Figure 5.5. The algorithm FTGG requires only the control messages used in GG algorithm.

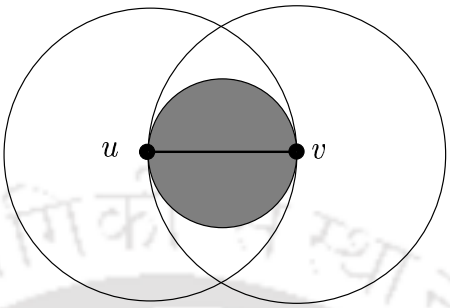


Figure 5.5: GG Neighborhood relation.

Algorithm: FTGG

1. Each node u switch off its transmitter if($stability \leq threshold$).
2. Broadcast ***hello_packet*** with ID and location information.
3. Node u receiving ***hello_packet*** stores the information in $n_1(u)$.
4. Each node u follow the steps below

   for each node $v \in n_1(u)$

   if ($disk(u, v)$ does not contain any node in $n_1(u)$)

   add edge $\overline{uv}$ to FTGG.

One can easily verify the following properties.

Lemma 5.4.1 *FTGG $\subseteq$ GG.*

Lemma 5.4.2 *The FTGG is a planar graph.*

Proof: This follows from FTGG $\subseteq$ GG and GG is planar graph. ∎

Lemma 5.4.3 *The message complexity for FTGG is in $O(n)$.*

Lemma 5.4.4 *The spanning ratio of FTGG is in* $O(\sqrt{n})$.

Proof: The length of the shortest path between two nodes p and q in FTGG is t times the length of the shortest path between the same nodes in UDG, where t is called the spanning ratio, which is $O(\sqrt{n})$ in case of GG. Thus, the spanning ratio of FTGG is in $O(\sqrt{n})$. ∎

The fault tolerant spanners FTLDel, FTRNG, and FTGG preserve the properties of the spanners PLDel, RNG, and GG respectively. We use the notation FTLDel(V') to represent FTLDel of a set V' of fault tolerant nodes.

Lemma 5.4.5 *FTRNG(V′) ⊆ FTGG(V′) ⊆ FTLDel(V′).*

Lemma 5.4.6 $\Pi_{FTRNG(V')} \geq \Pi_{FTGG(V')} \geq \Pi_{FTLDel(V')}$

Proof: This follows from the Lemma 5.4.5 and the fact that $|\Pi_{FTRNG(u,v)}| \geq |\Pi_{FTGG(u,v)}| \geq |\Pi_{FTLDel(u,v)}|$, where $|\Pi_{G(u,v)}|$ is the length of shortest path between two nodes u and v in the graph G. ∎

5.5 Simulation and performance

We have used network simulator (*ns-2.28*) [NS205] in the simulation work. Apart from the fault tolerant spanners FTLDel, FTRNG, and FTGG, we have simulated PLDel, RNG, and GG for the sake of comparison. To evaluate the performance of proposed spanners, we run six routing protocols on these spanners. These routing protocols are greedy routing (Grdy), compass routing (Cmp), random compass routing (RandCmp), most forward routing (MFR), nearest neighbor routing (NNR), and farthest neighbor routing (FNR). In the simulation, we have considered the power level of a node as the stability factor.

We have taken five different node scenarios and each containing 50 nodes distributed randomly on $600 \times 600m^2$ grid. The transmission range of each node is 250m. For each node, a random initial energy between 15 and 45 joules is given. The transmission power of each packet transmitted is 0.665 watts. The receiving power is 0.395 watts and the idle power is 0.035 watts. The value of stability factor is 7. The simulation time is 600 seconds. For the connection patterns, we have chosen randomly three source nodes at one side of the grid and three destination nodes at the other side.

5.5.1 Simulation results

With the above simulation model, we run several simulations at different packet transfer rates. To evaluate the performance of FTLDel, we run six geometric routing protocols *Grdy* [BMSU01], *Cmp* [BCSW98, KSU99], *RandCmp* [KSU99], *MFR* [TK84], *NNR* [HL86], and *FNR* [LCW02] on two spanners FTLDel and PLDel. For each spanner, we have computed the delivery ratios for the above six routing protocols at different CBR intervals. The figures Figure 5.6, Figure 5.7, Figure 5.8, Figure 5.9, Figure 5.10, and Figure 5.11 show that the spanner FTLDel has better delivery ratios compared to PLDel. This happens because the spanner FTLDel considers stable nodes for fault tolerance and avoids heavy packet loss, where as PLDel does not.

Similarly, next two experiments are performed to evaluate the performance of FTRNG and FTGG. The delivery ratios are computed on each spanner for the above four routing protocols at different CBR intervals. From the figures Figure 5.12, Figure 5.13, Figure 5.14, Figure 5.15, Figure 5.16, Figure 5.17, and Figure 5.18, we can say that the spanners FTRNG and FTGG show better delivery ratios than RNG and GG, respectively. This happens because these spanners, FTRNG and FTGG, avoided the packet loss by considering stable nodes in fault tolerant spanner, where as their counter part spanners suffer from the packet loss.

From all the plots, we can observe that the delivery ratio of the spanners at lower CBR intervals are lower and are increased gradually with the increase in CBR intervals. This happens because, at lower CBR intervals large number of data packets are sent due to the lower inter-packet arrival time. This causes the problems of network congestion which in turn causes heavy packet loss. The network congestion is lower at high CBR intervals and thus provides higher delivery ratios.

The number of hops between source and destinations in FTLDel is lower than FTRNG and FTGG, since FTLDel $\supseteq$ FTGG $\supseteq$ FTRNG (Lemma 5.4.5). This causes more energy consumption in FTRNG and FTGG, which in turn causes node failures. On the other hand, FTLDel consumes less power and exhausts the battery slowly compared to FTRNG and FTGG. In order to provide uniformity in CBR intervals, we have plotted the graphs for the spanners FTRNG and FTGG at higher CBR intervals whereas the lower CBR intervals are taken for the spanner FTLDel.

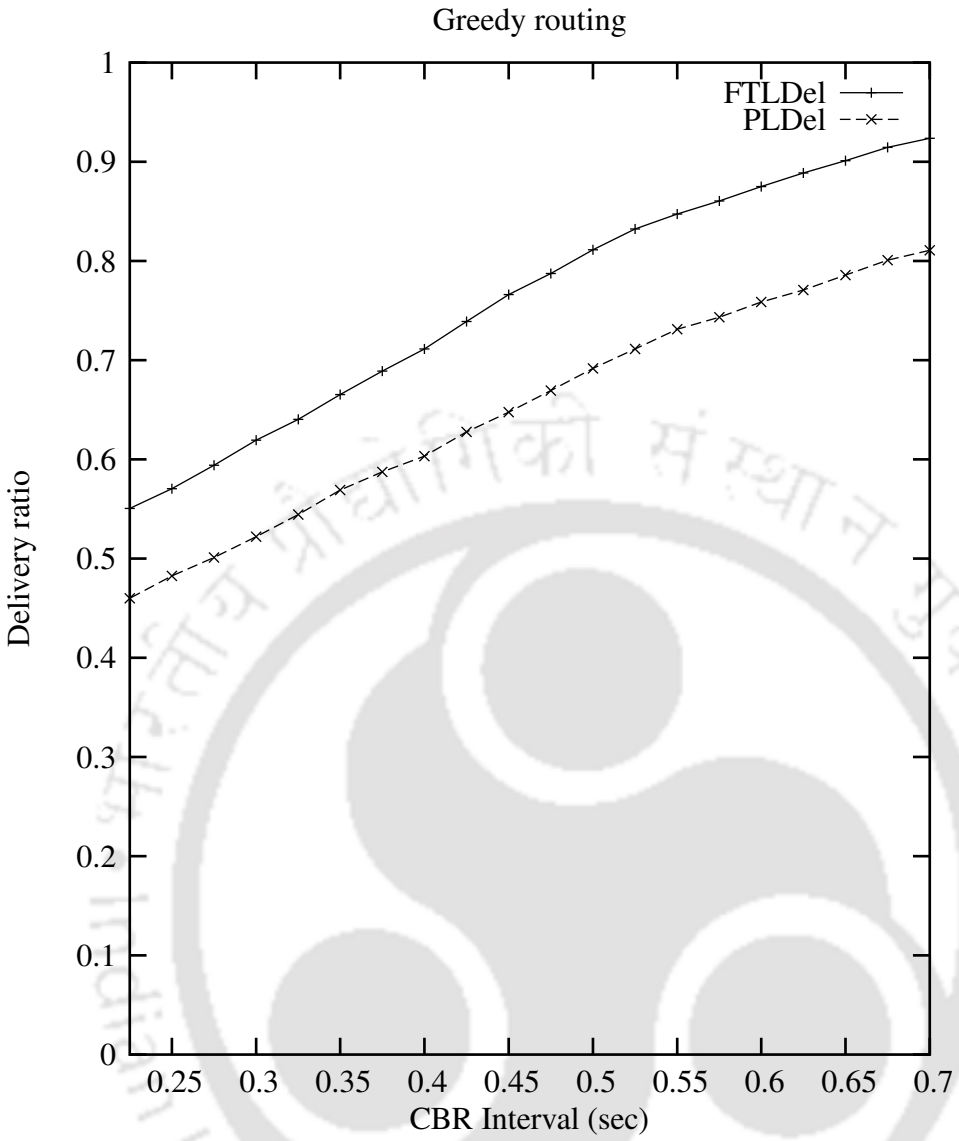


Figure 5.6: Greedy forward routing.

In the next experiment, we compute the packet delay for the spanners. We run the greedy routing algorithm on the spanner FTLDel and PLDel. From the Figure 5.19, we observe that the packet delay for both the spanners are almost the same. Even though heavy packet loss occurs during data transmission, we consider the end-to-end packet delay for only the successfully received packets. Thus, we do not find much improvement in the packet delay for the proposed spanner FTLDel. The same rule is applicable for other fault tolerant spanners FTRNG and FTGG as shown in the figures Figure 5.20 and Figure 5.21, respectively.

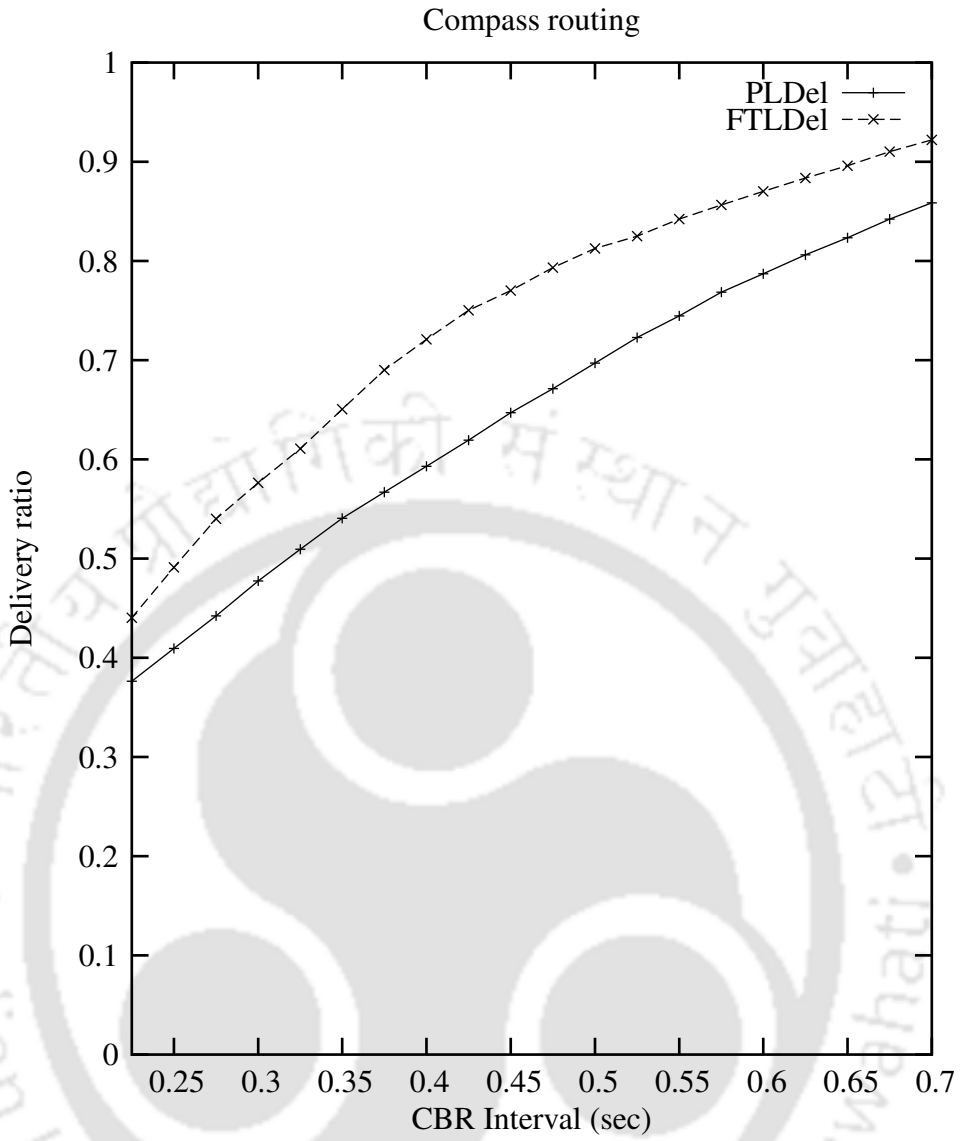


Figure 5.7: Compass routing.

5.6 Conclusions

In this chapter, for fault tolerant routing, we have proposed fault tolerant spanners FTLDel, FTRNG, and FTGG by choosing more stable nodes in the network. These spanners show better performance at frequent node failure conditions.

As we mentioned earlier, the k-node fault tolerant spanners (k-NFTS) and k-link fault tolerant spanner (k-LFTS) are studied in the literature and proposed efficient centralized algorithms [LNS02, CZ03, Luk99]. It would be interesting to design efficient localized distributed algorithm for k-NFTS and k-LFTS and test them for suitable adhoc networks.

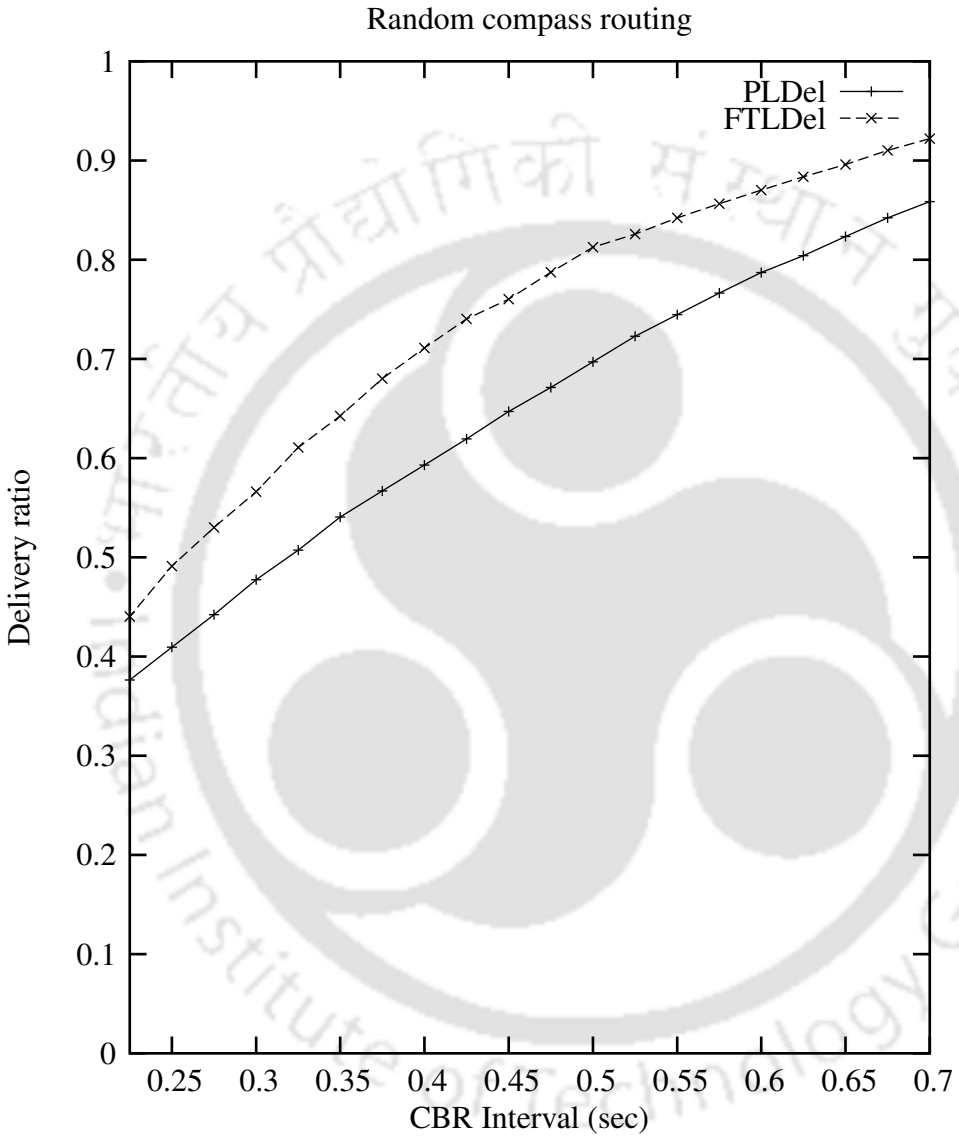


Figure 5.8: Random compass routing.

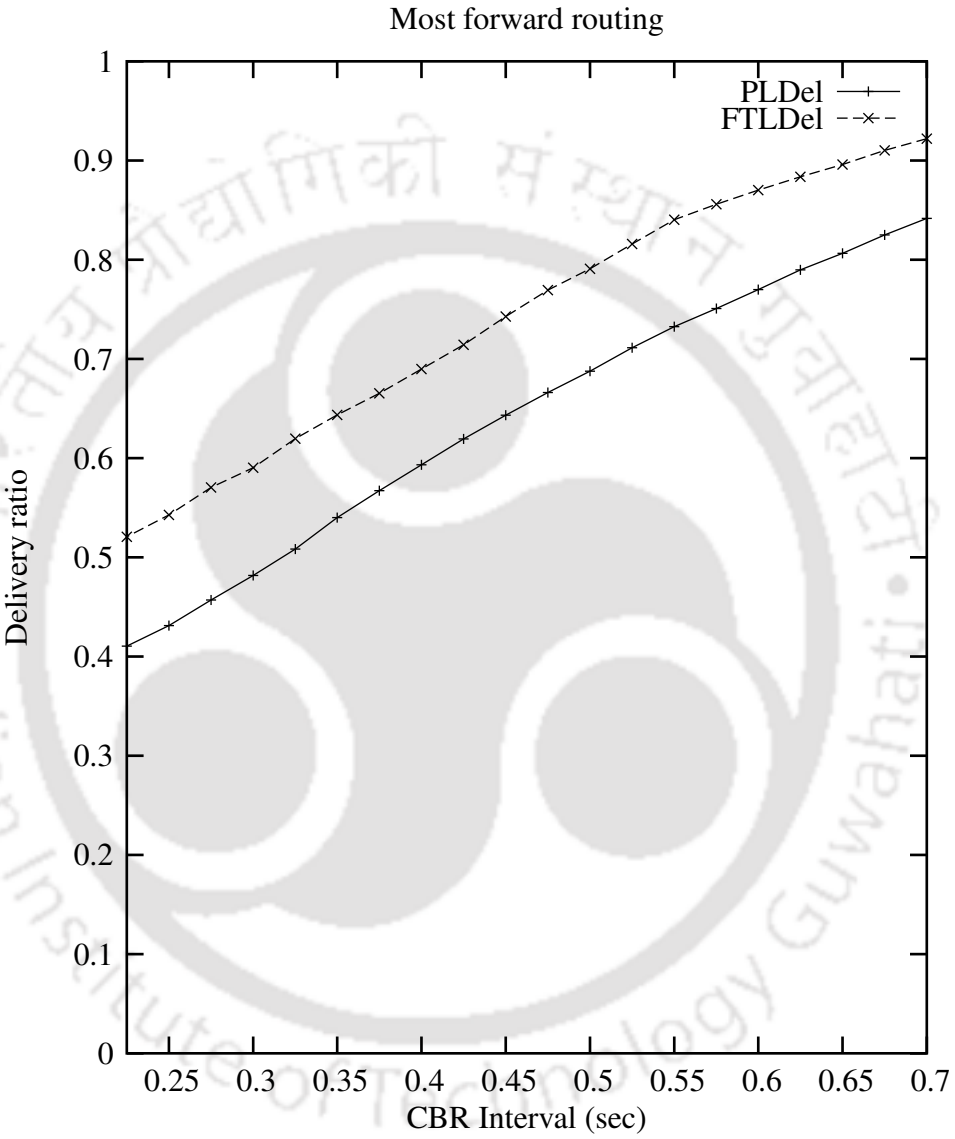


Figure 5.9: Most forward routing.

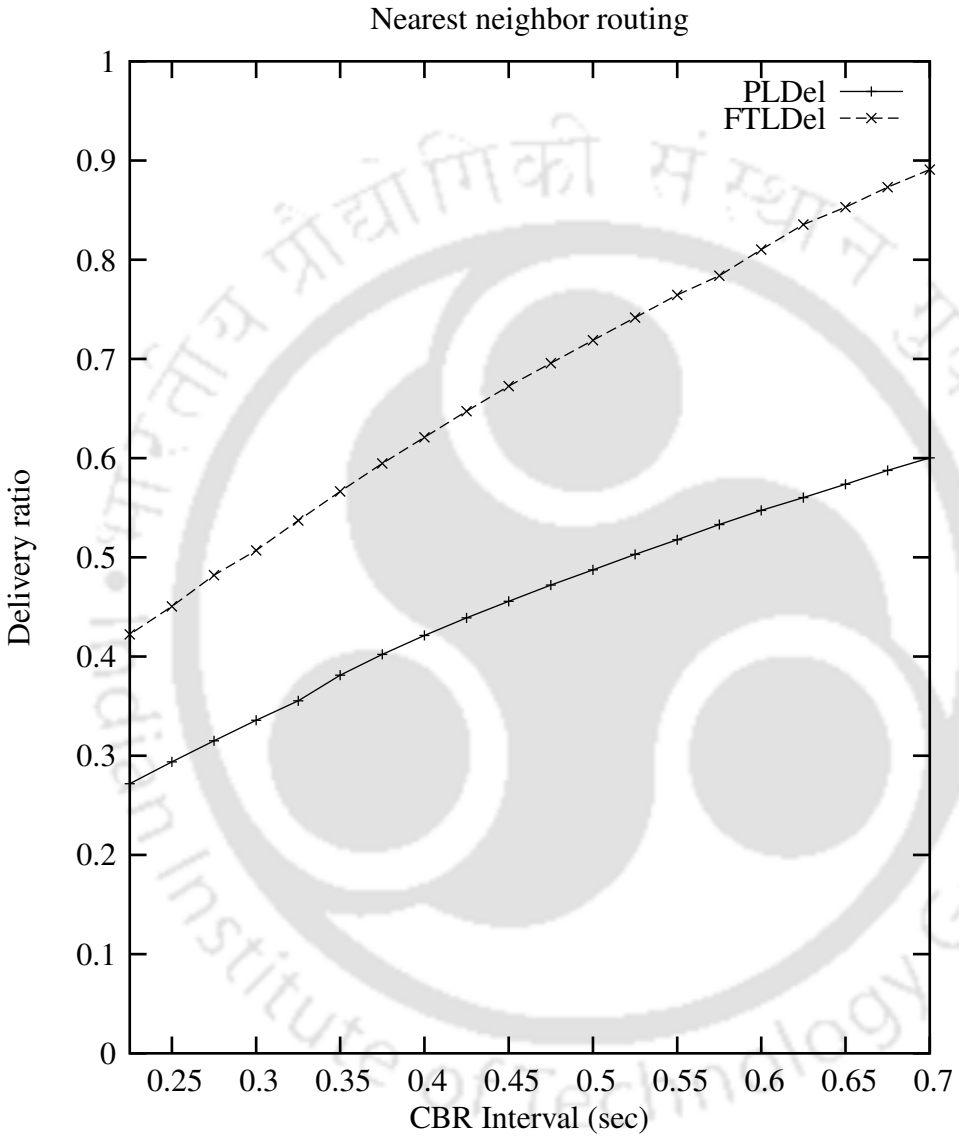


Figure 5.10: Nearest neighbor routing.

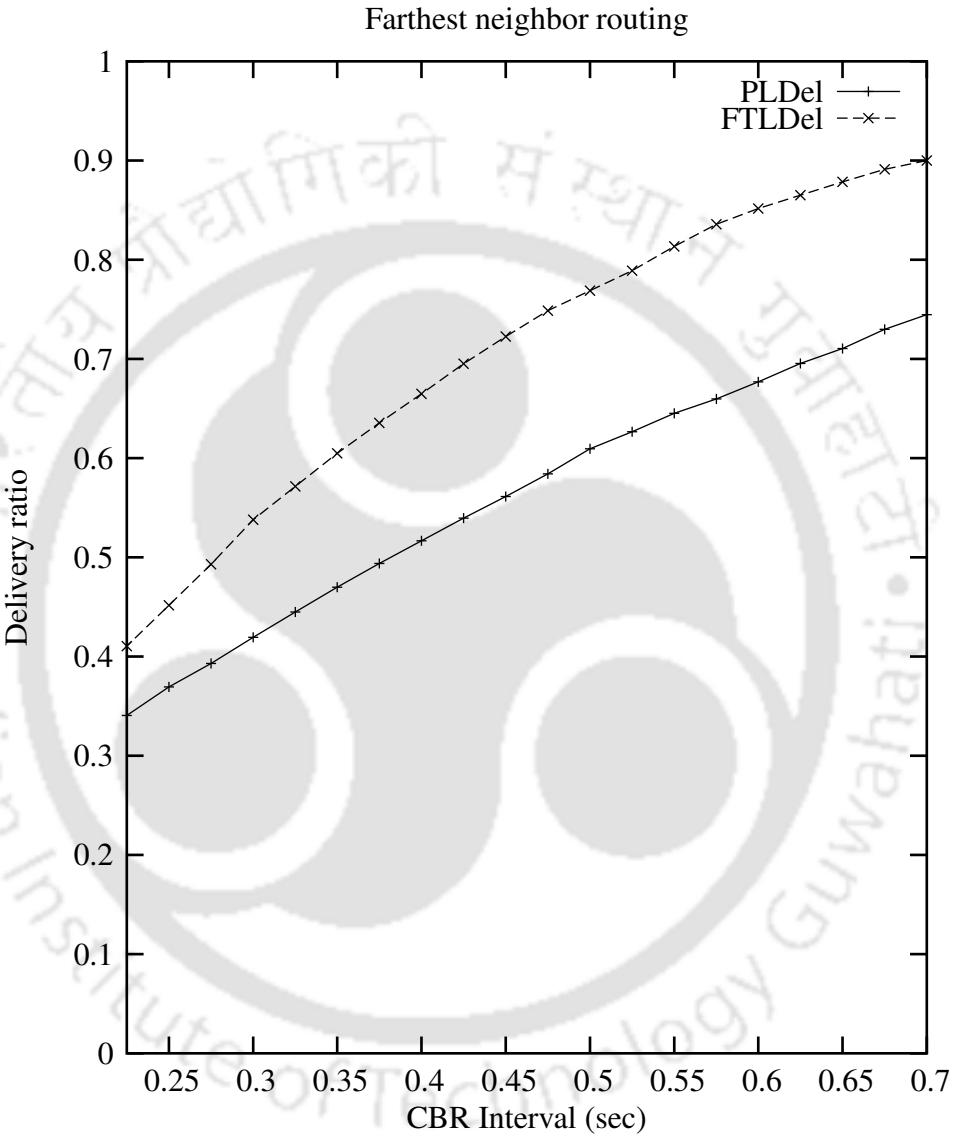


Figure 5.11: Farthest neighbor routing.

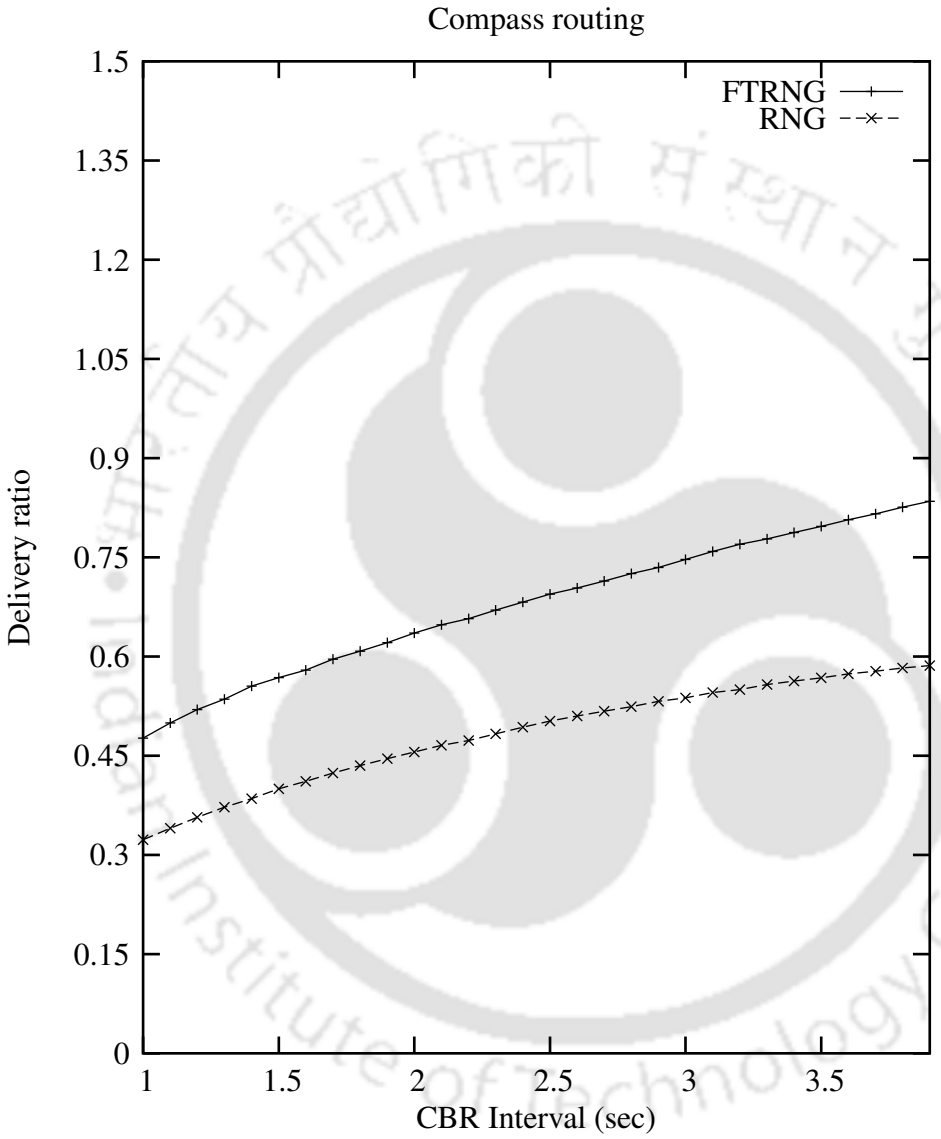


Figure 5.12: Compass routing.

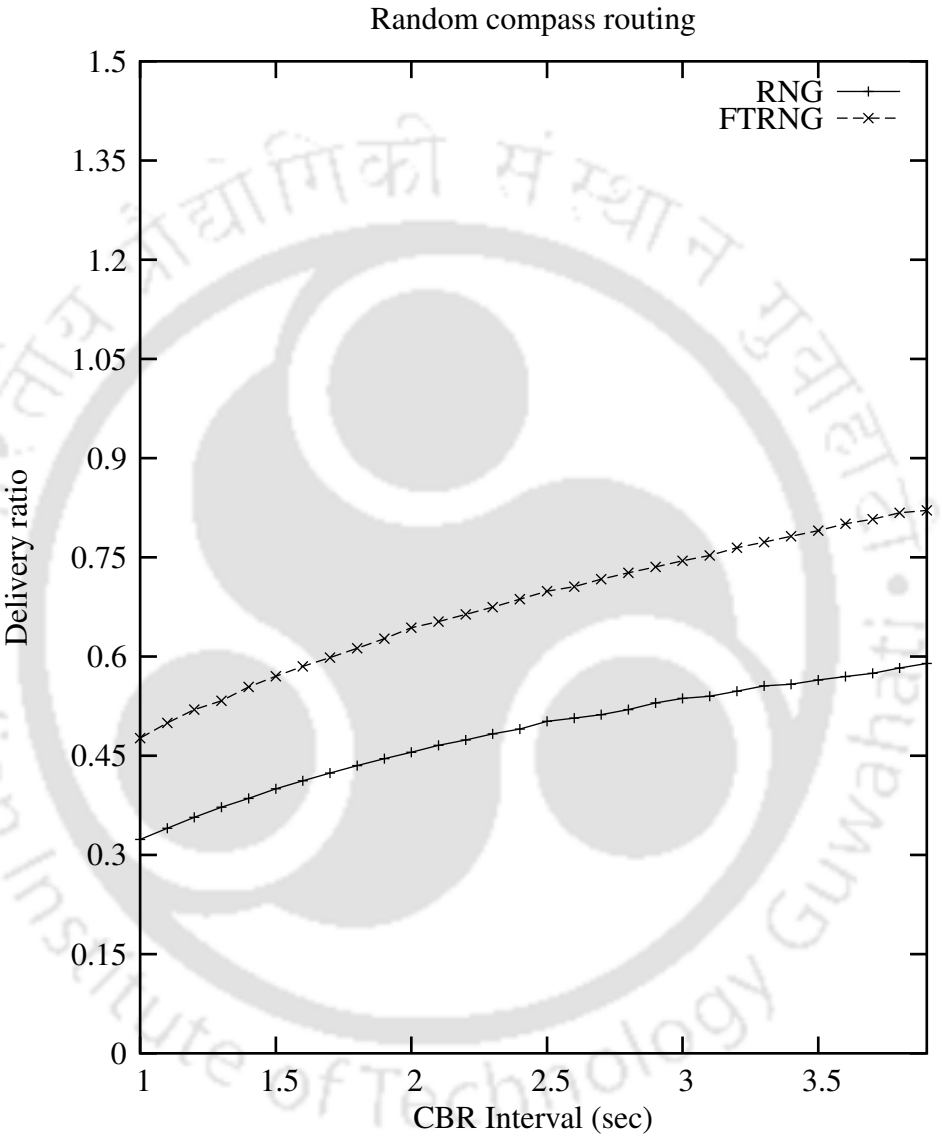


Figure 5.13: Random compass routing.

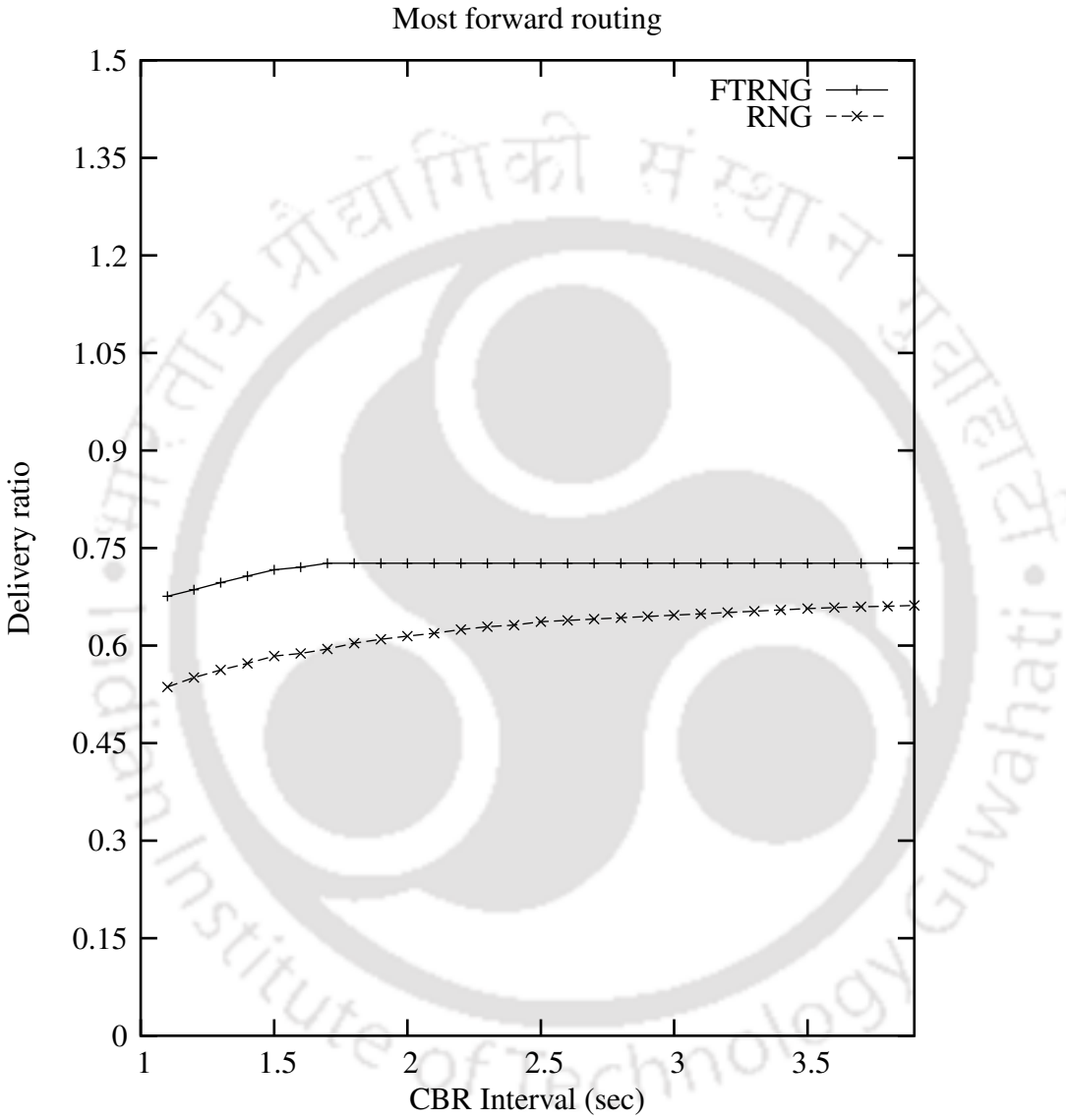


Figure 5.14: Most forward routing.

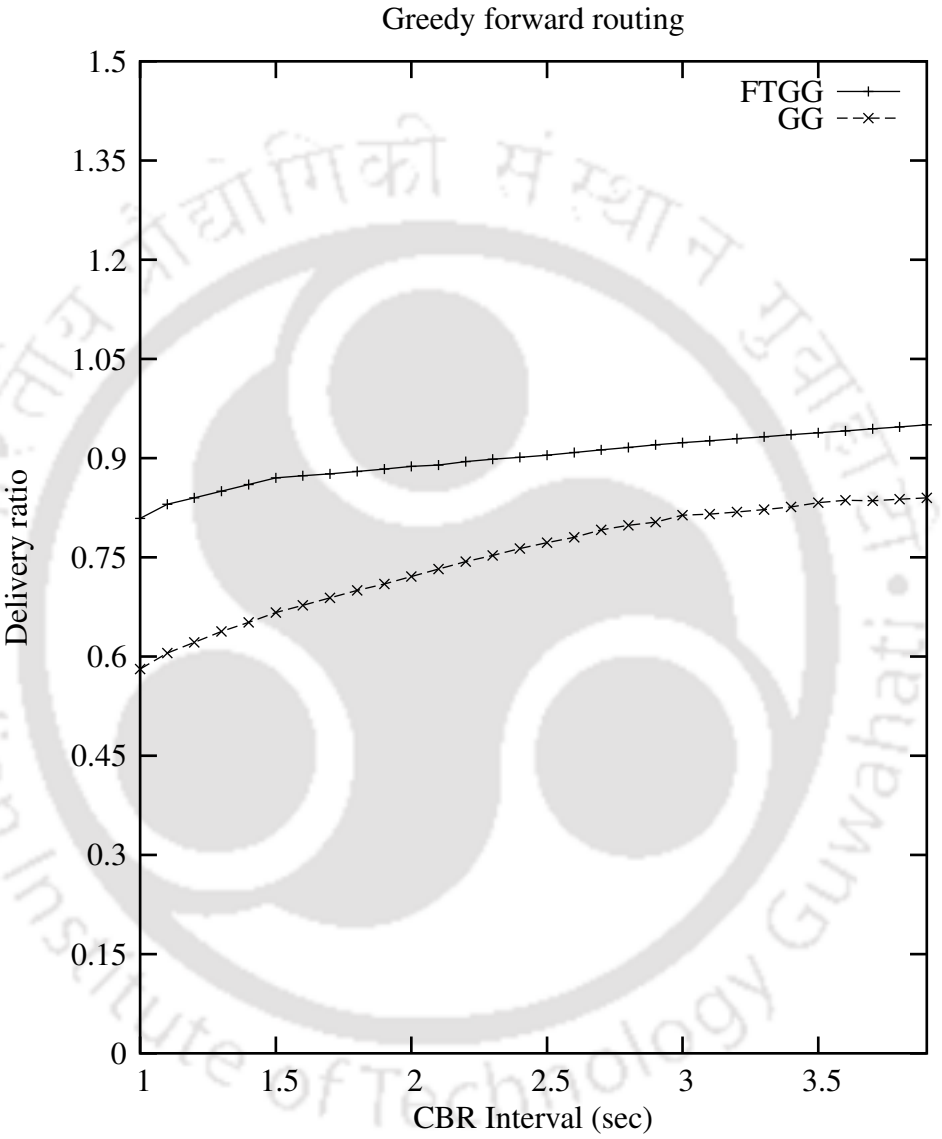


Figure 5.15: Greedy routing.

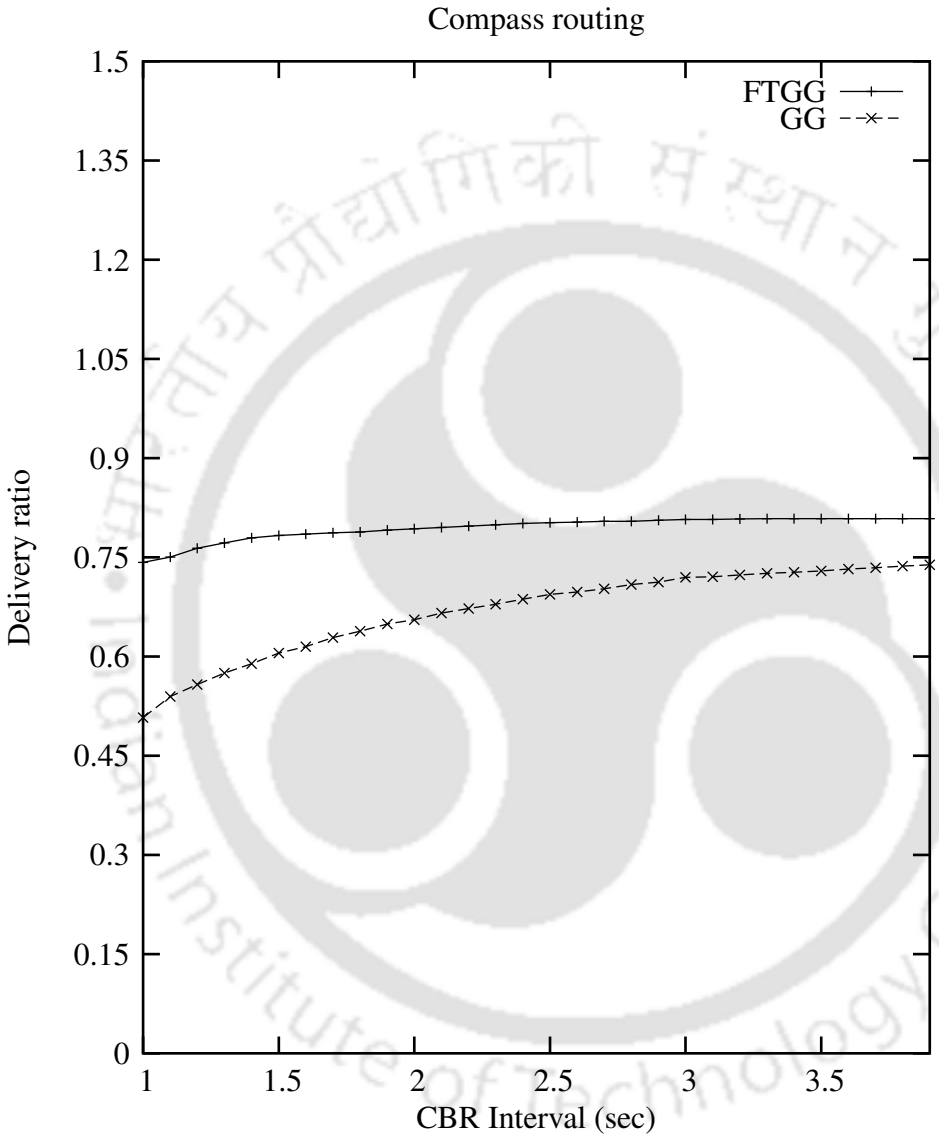


Figure 5.16: Compass routing.

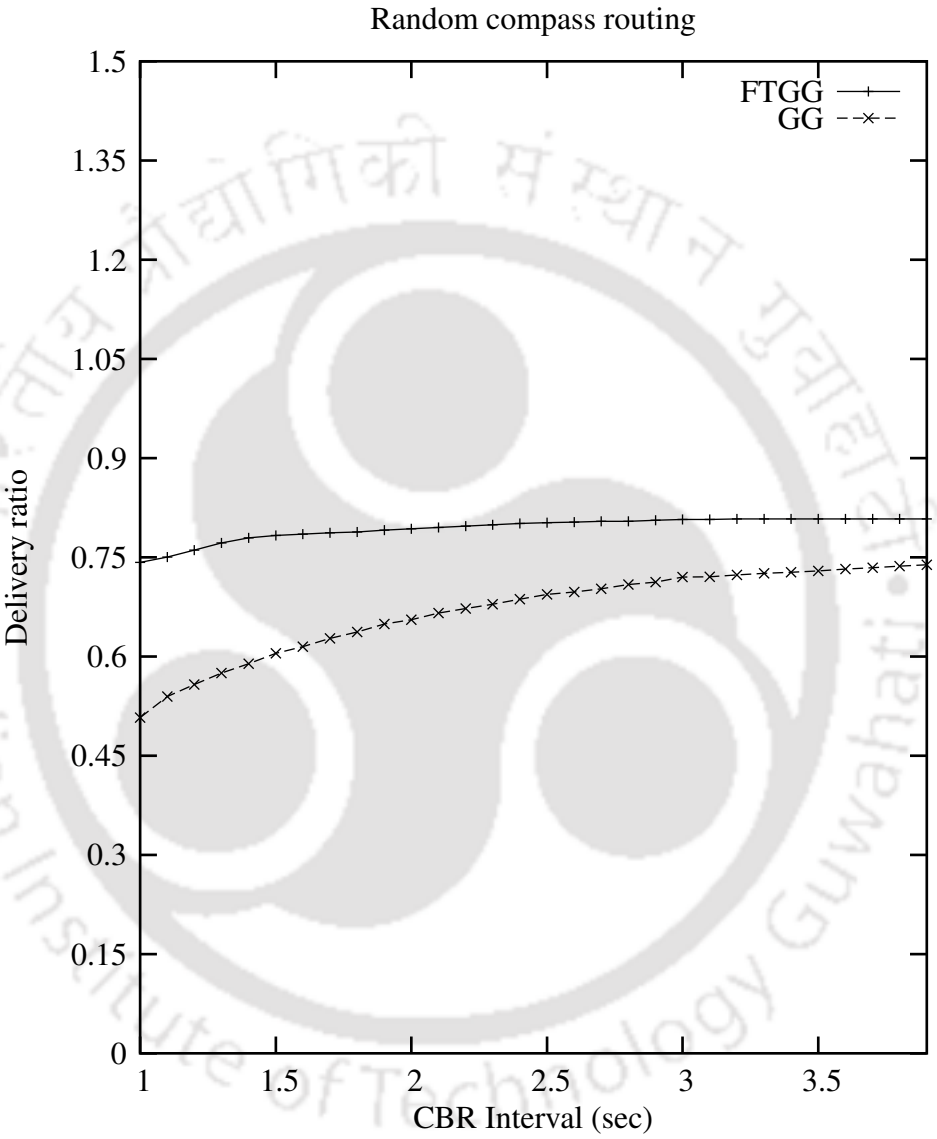


Figure 5.17: Random compass routing.

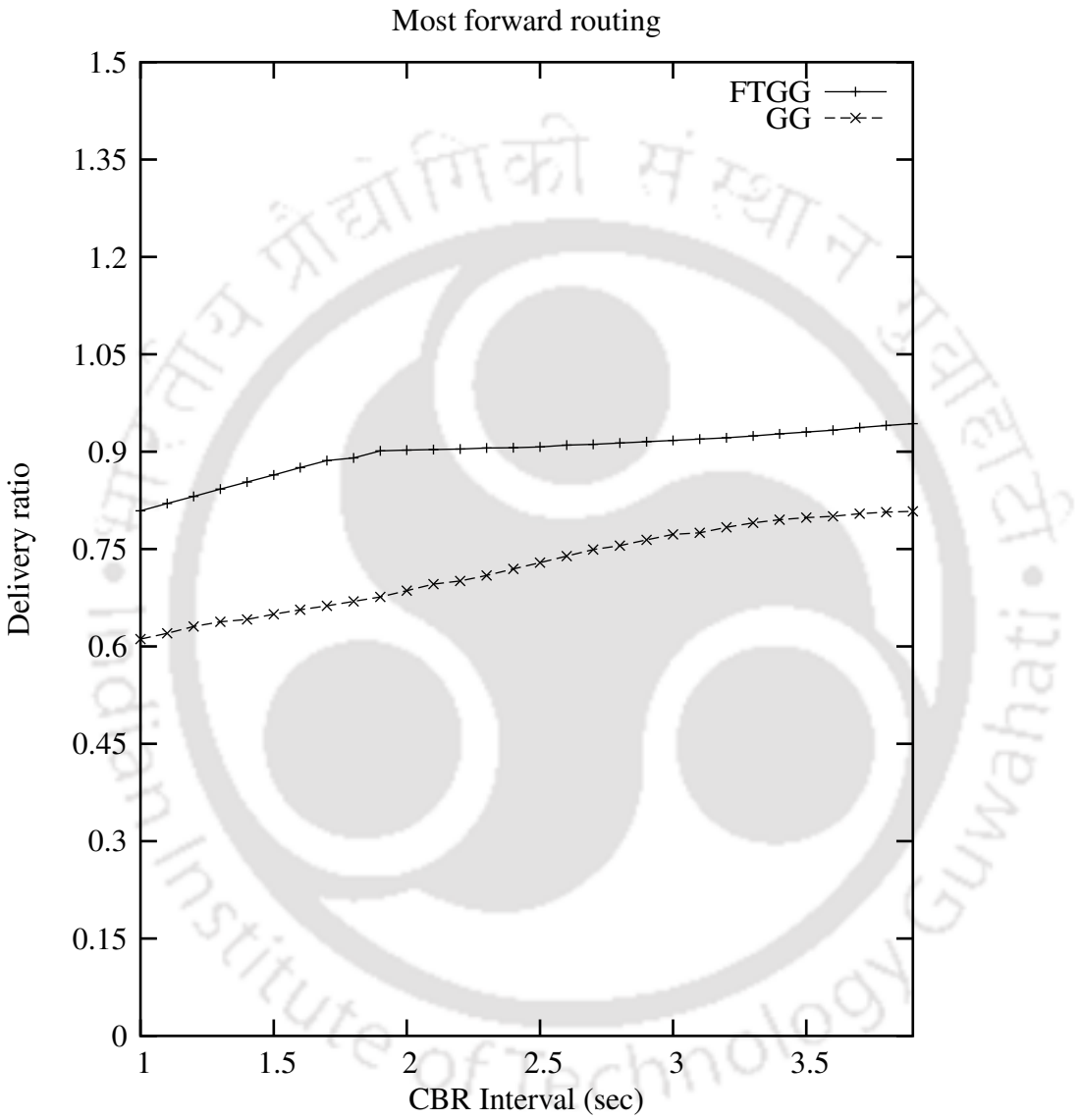


Figure 5.18: Most forward routing.

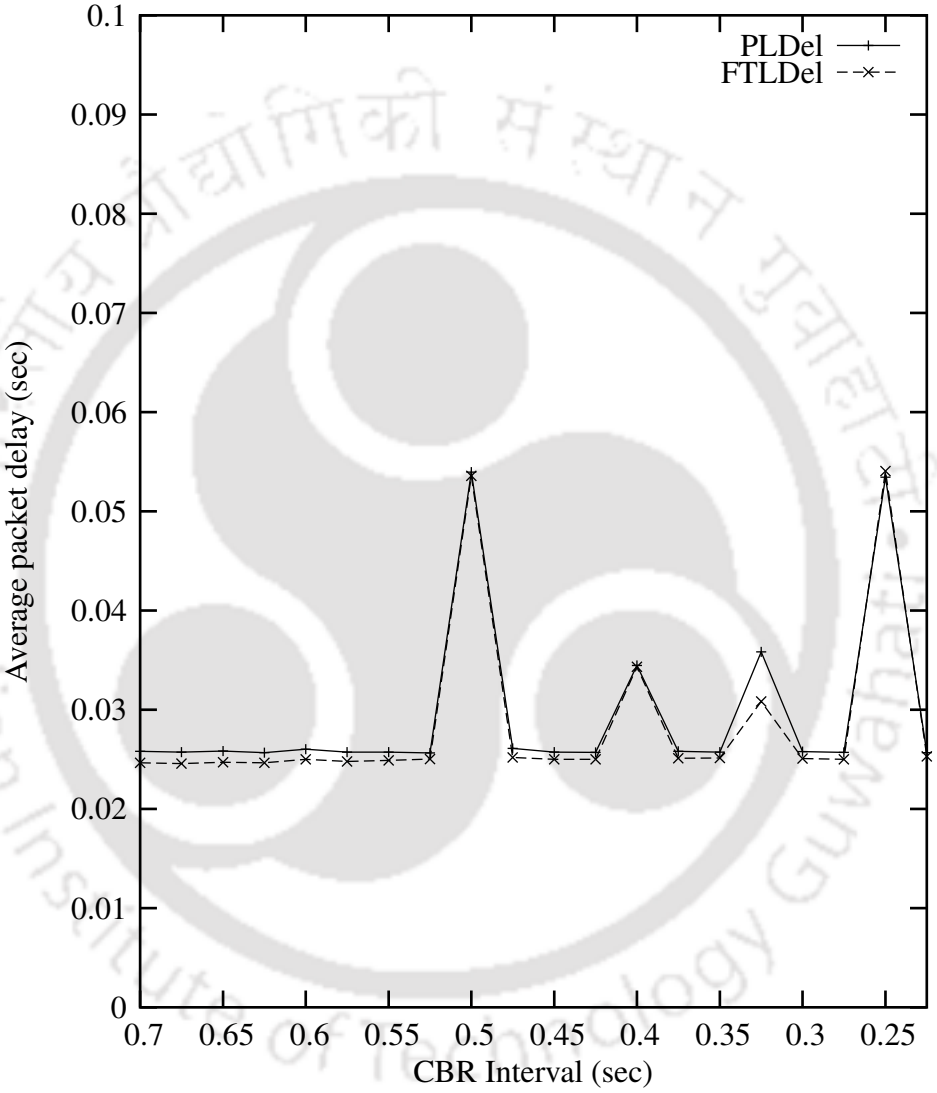


Figure 5.19: Delay in FTLDel.

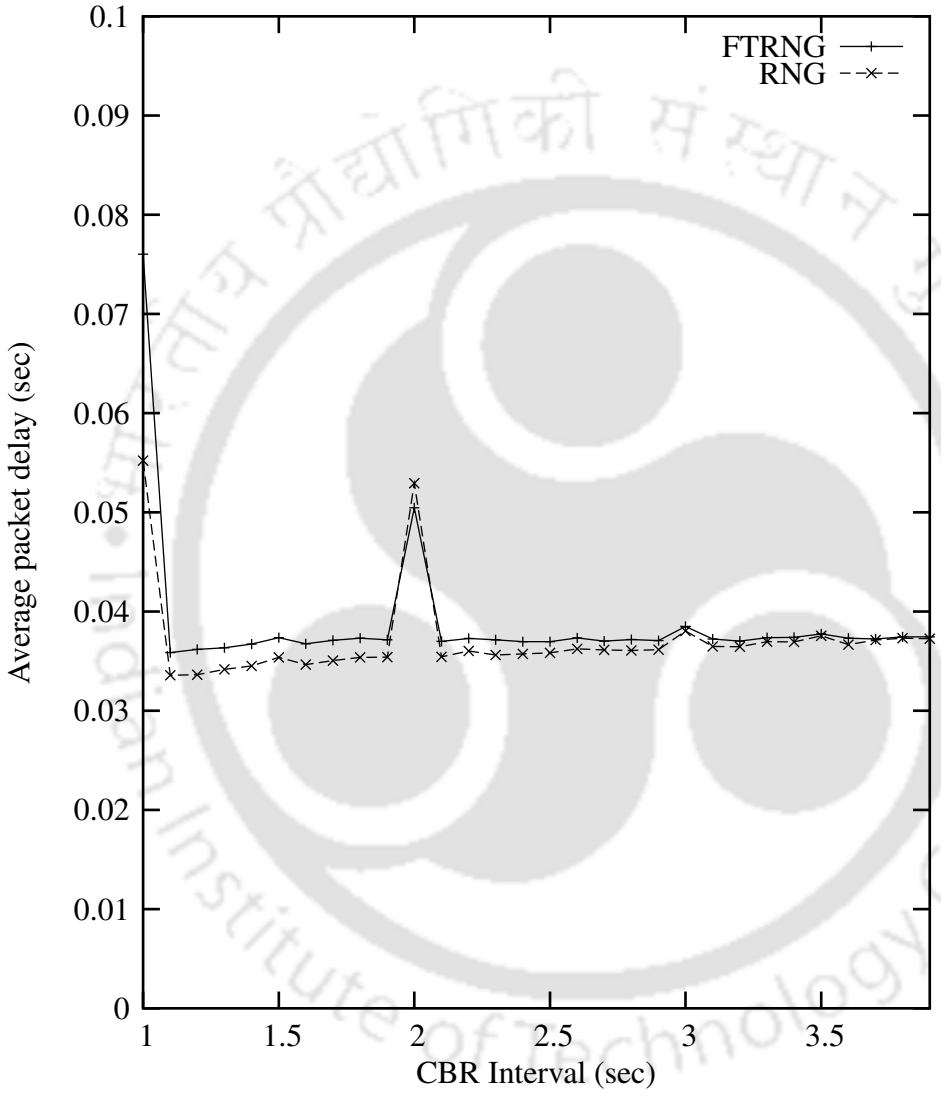


Figure 5.20: Delay in FTRNG.

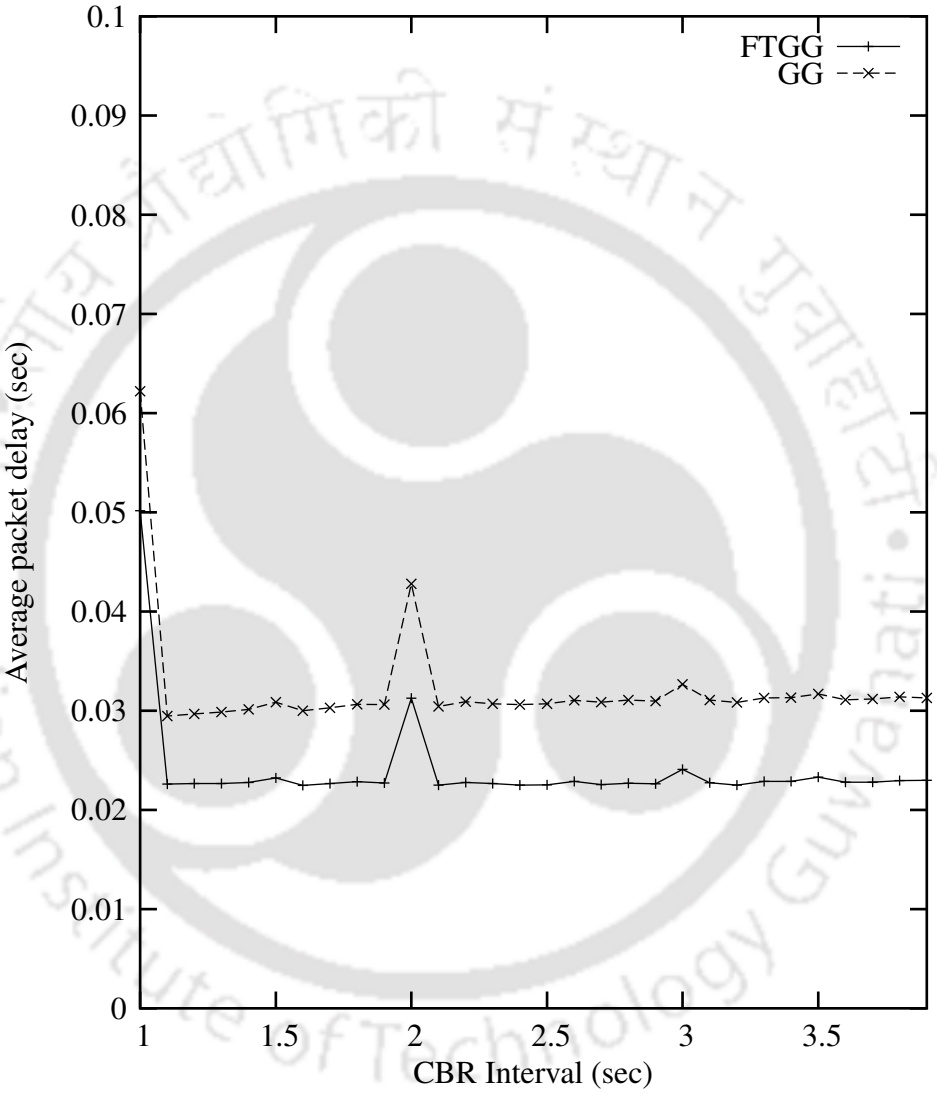


Figure 5.21: Delay in FTGG.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 6

Geometric Spanners under Mobility

6.1 Introduction

Mobility in MANET is a significant consideration when designing a routing algorithm. The existing spanners assume that the nodes in the network are static, that is zero mobility. When these nodes move around in the network, the spanner looses its geometric properties like neighborhood relation, spanning ratio, and planarity. Moreover, some of the edges may become invalid links and may lead to disconnected network. This leads to degradation of network parameters like delivery ratio, delay, and throughput, when these spanners are used as an underlying network graph by routing protocols. For example, consider the triangle $\triangle uvw$ of PLDel, in Figure 6.1, at time t_1. As the node v moves out of the transmission range of the node u at time t_2, $\triangle uvw$ becomes invalid triangle in PLDel even though $\triangle uvw$ satisfies the in-circle test. Similarly, $\triangle uvw$ becomes invalid in PLDel as the node x comes into the circumcircle of $\triangle uvw$, as shown in the Figure 6.2.

The effect of node mobility can be modeled with the combination of one or more node downs and node joins, discussed in the fourth chapter *Dynamic spanners*. However, this procedure has more computational and communication overhead. In this chapter, we have considered this as a problem and proposed localized algorithms to maintain the geometric spanners under mobility. These algorithms maintain the mobility management in multiple invocations to reduce the computational and communication overhead. Precisely, the mobility management for the spanners is maintained in three phases. In Phase 1, the basic functions required for mobility management are taken care. The Phase 2 involves

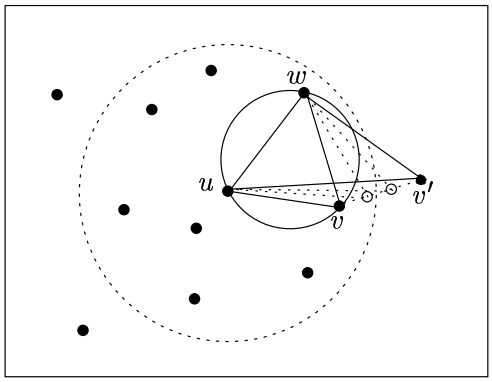


Figure 6.1: $\triangle uvw$ becomes invalid in PLDel as the node v goes out of the transmission range of u.

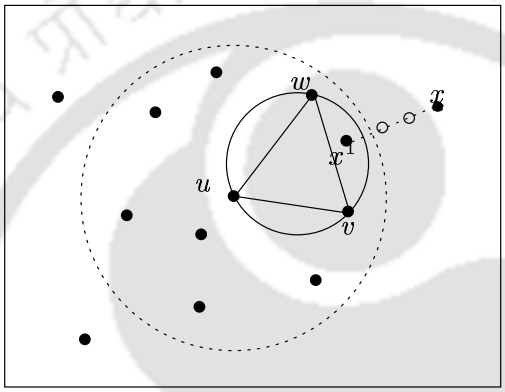


Figure 6.2: $\triangle uvw$ becomes invalid in PLDel as the node x comes inside the in-circle of $\triangle uvw$.

in protecting the network parameters, the delivery ratio and packet delay. The final phase is for maintaining the geometric properties of the spanner. The proposed algorithms for maintaining PLDel, RNG, and GG under mobility are called mobile local Delaunay triangulation (MLDel), mobile relative neighborhood graph (MRNG), mobile Gabriel graph (MGG), respectively. These spanners are simulated in *ns-2.28* [NS205] and compared with their counter parts.

The system model considered for this chapter is an asynchronous wireless mobile adhoc networks, where the network is an unit disk graph model. Here, each node has the capability of measuring its velocity at any given time. Let (x_1, y_1) and (x_2, y_2) be the positions of a node at time t_1 and t_2, respectively. The velocity V of the node at time t_2 can be measured as follows:

$$V = \frac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{|t_2-t_1|}$$

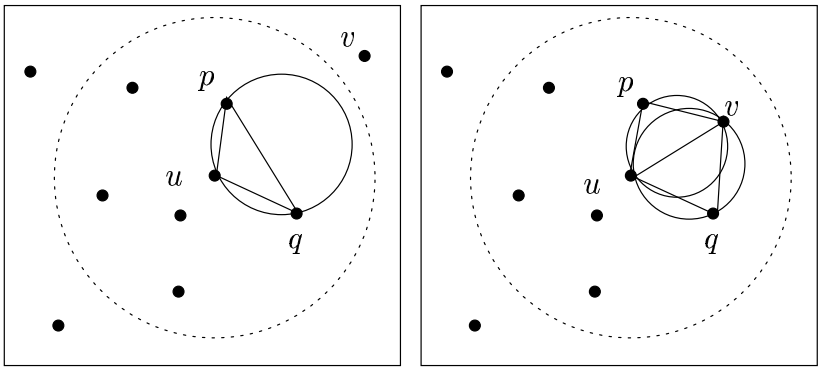


Figure 6.3: New edge $\overline{uv}$ is added to PLDel when v comes into transmission range of u.

The remaining part of this chapter is organized as follows. The Section 6.2 describes the proposed local Delaunay triangulation under mobility (MLDel). The Section 6.3 presents mobile relative neighborhood graph (MRNG) and followed by mobile Gabriel graph (MGG) in the Section 6.4. In the Section 6.5, the simulation work and results are presented. The Section 6.6 concludes the chapter.

6.2 Mobile local Delaunay triangulation (MLDel)

The main objective of our approach is to protect the packet loss in the spanner PLDel under mobility. One can use the existing static spanner for routing in MANETs. But, due to the mobility, the topology of the network changes, which in turn degrades the packet delivery ratio. However, this approach does not involve any extra cost.

Another way of maintaining the PLDel under mobility is by periodically reconstructing the entire PLDel. The accuracy of PLDel increases as the time period for reconstruction decreases. But constructing the entire PLDel from the scratch involves large communication and computational costs. Hence, our approach is in between these two, which consists of more frequent less cost local updates and less frequent more cost global updates.

The algorithm is broadly divided into three phases. In the first phase, each node u broadcasts *hello_packet* periodically, that is at every t_0 time units, and listens for the same that are broadcasted by other nodes. After collecting 1-hop neighbors $n_1(u)$, each node u constructs PLDel [LCWW03, LCW02].

The major objective of the second phase is to protect the delivery ratio, delay, and throughput. Packet loss increases drastically when a failed link is on the routing path. This can happen when next forwarding node moves out of the transmission range of the current node. In such cases, it is necessary to update the spanner to avoid the packet loss. Hence, the second phase of the algorithm focuses on updating the PLDel locally. That is, if a node $v \in n_1(u)$ goes out of the transmission range, the node u deletes the edge $\overline{uv}$ from the spanner, if it exists.

Similarly, when a node v comes into the transmission range of the node u, it receives the *hello_packet* from node v and updates $n_1(u)$. Node u finds two nodes p and q such that the nodes p and q are either side of the node v in PLDel, as shown in the Figure 6.3. Node u checks the in-circle test of the triangles $\triangle uvp$ and $\triangle uvq$. If these two triangles satisfy the in-circle tests then the edge $\overline{uv}$ is added to the spanner.

These periodic local updates makes the spanner violates the geometric properties. So, in order to maintain geometric properties, each node u reconstructs PLDel if there is any change in its 1-hop neighbor lists or their positions, in the third phase of the algorithm. The objective of the third phase of the algorithm is to preserve the PLDel properties.

The time periods of second and third phase can be fixed with respective to beacon interval t_0, such that $t_0 < t_1 < t_2$, where t_1 and t_2 are the time intervals of second and third phases respectively. This algorithm uses the message *hello_packet*, in addition to the messages used on PLDel. This packet carries a node id and position information.

Algorithm : MLDel

---

1. Each node Broadcast ***hello_packet*** periodically, say at time t_0 with ID and location information.

2. Each node u updates their neighbor list $n_1(u)$, from received ***hello_packet***.

3. Each node u constructs PLDel(u).

4. Each node u do the following for every t_1 units time,

   For each $v \in$ PLDel(u) and $v \notin n_1(u)$

   /* node v goes out of transmission range of u */

   PLDel(u) = PLDel(u)$-\{v\}$.

For every new neighbor $v \in n_1(u)$

/* new node v comes into transmission range of u */

Find $p \in PLDel(u)$, such that p is immediately counter clockwise side of $\overline{uv}$.

Find $q \in PLDel(u)$, such that q is immediately clockwise side of $\overline{uv}$.

if (incircle(u,v,p) is empty and incircle(u,v,q) is empty)

$PLDel(u) = PLDel(u) \cup \{v\}$.

5. Each node u does the following at every t_2 units time

/* construct PLDel*/

if there is any change in $n_1(u)$ or their positions

Construct PLDel(u).

The intervals t_0, t_1, and t_2 provide the trade off between the accuracy and overhead of the spanner. The accuracy of the spanner increases by decreasing the intervals, but the communication and computational overhead increases. If the intervals are increased, the overhead is reduced, but the accuracy of the spanner decreases. One way of choosing the intervals is based on mobility. Smaller intervals are required in high mobility conditions whereas a longer interval works fine in low mobility.

6.3 Mobile relative neighborhood graph (MRNG)

The main objective of MRNG is to maintain the relative neighborhood graph under mobility conditions.

Each node u broadcasts the message *hello_packet* with its id and location information periodically, that is at every t_0 time units. Each node u collects one hop neighborhood information, $n_1(u)$, by receiving the *hello_packet* from the nodes in its transmission range. Construct the RNG with $n_1(u)$ using the procedure given in the Section 4.3.1 or Section 2.

Similar to PLDel, RNG is also maintained in three phases. The second phase makes local changes to preserve network parameters and the third phase preserves the spanner properties. The second phase of the algorithm is executed at every time interval t_1, which is discussed below.

We use the notation $\text{RNG}_i(u)$ to denote the set of RNG neighbors in the sector i as used in the Section 4.3.1.

Case 1 (Existing node moves out of transmission range): For each node $v \in \text{RNG}_i(u)$, if the node v does not exists in the $n_1(u)$, remove it from RNG. In other words, if the node v moves out of the transmission range of node u then $\text{RNG}_i(u) = \text{RNG}_i(u) - \{v\}$. The node u selects a node w in the sector i, such that w is the nearest neighbor of node u, if $\text{RNG}_i(u) = \emptyset$. Conduct the lune test $lune(u, w)$, and add the edge $\overline{uw}$ to RNG if it is empty, as shown in the Figure 6.4. There can be more than one nearest neighbor in the sector i. We can add all these nodes to $\text{RNG}_i(u)$ if their respective lunes are empty. This takes time proportional to the square of number of 1-hop neighbors of u. But it is sufficient to have a constant number of edges in each sector to preserve network parameters. So, we choose only one nearest neighbor in the sector i. The time complexity is proportional to the number of 1-hop neighbors of u.

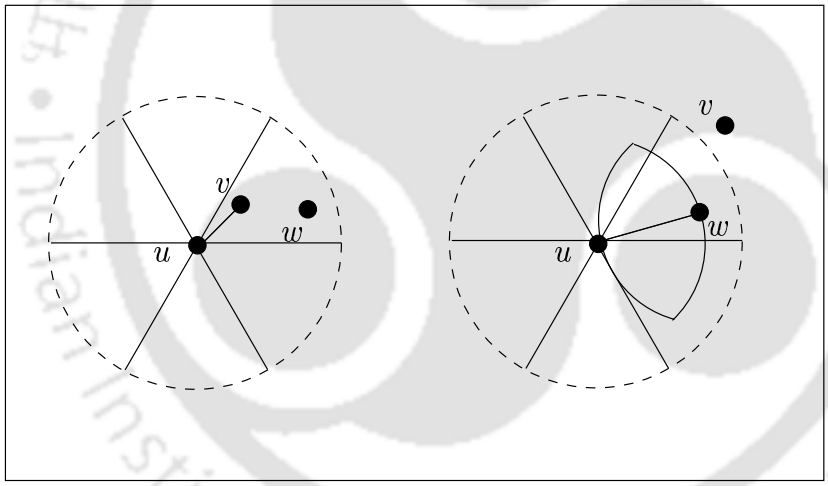


Figure 6.4: Case 1: Node v is moved out of the transmission range of u.

Case 2 (New node enters into transmission range): If a node v enters into the transmission range of node u, find the sector i, where the node v exist. Check if the distance between nodes u and v is less than the distance between the nodes u and w, where $w \in \text{RNG}_i(u)$. If so, add the edge $\overline{uv}$ to RNG if $lune(u, v)$ is empty and remove $\overline{uw}$ from RNG (see the Figure 6.5).

There are four possible cases when a node v moves from the sector i to the sector j of the node u. These are discussed below.

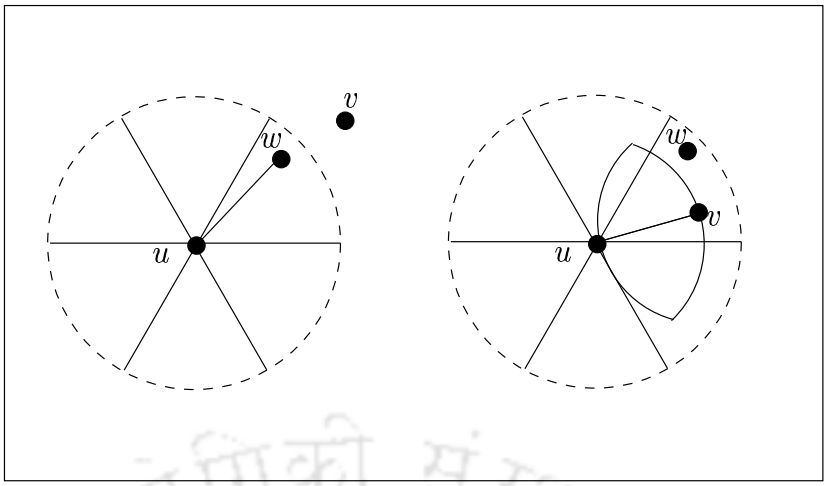


Figure 6.5: Case 2: Node v moves into the transmission range of u.

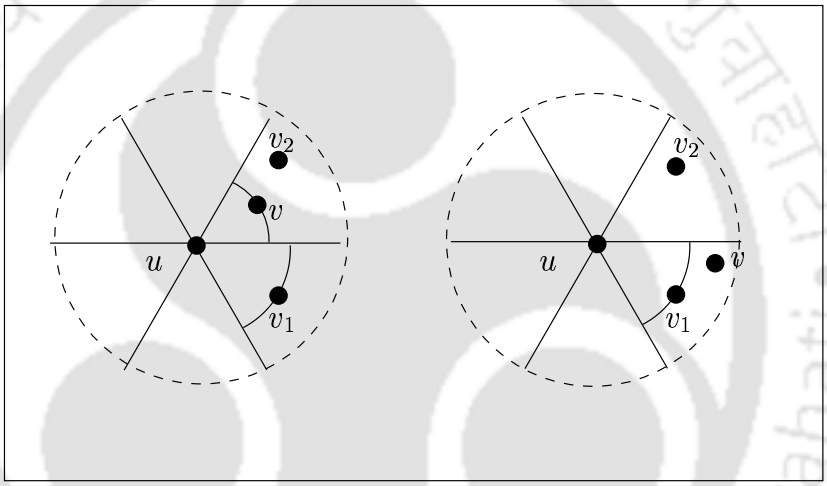


Figure 6.6: Case 3a: Node v moves into the sector j and $|\overline{uv}| > |\overline{uv_1}|$.

Case 3a (Node v moves to another sector, $|\overline{uv}| > |\overline{uv_1}|$ and $v_1 \in$ RNG$_j(u)$): If a node v moves from the sector i to the sector j and $|\overline{uv}| > |\overline{uv_1}|$, where $v_1 \in$ RNG$_j(u)$, then no action is necessary because v is not nearest neighbor of the node u in the sector j. If RNG$_i(u) - \{v\} = \emptyset$ then update RNG edges in the sector i as in the Case 1.

Case 3b (Node moves to another sector, $|\overline{uv}| = |\overline{uv_1}|$ and $v_1 \in$ RNG$_j(u)$): If a node v moves from sector i to sector j and $|\overline{uv}| = |\overline{uv_1}|$ then do nothing, since there are edges in the sector j. In the sector i, new RNG edges are computed as in the Case 1.

Case 3c (Node moves to another sector, $|\overline{uv}| < |\overline{uv_1}|$ and $v_1 \in$ RNG$_j(u)$): RNG$_j(u) = \{v\}$ if $lune(u, v)$ is empty, otherwise RNG$_j(u) = \emptyset$. For the sector i, find the new RNG edges as in the Case 1.

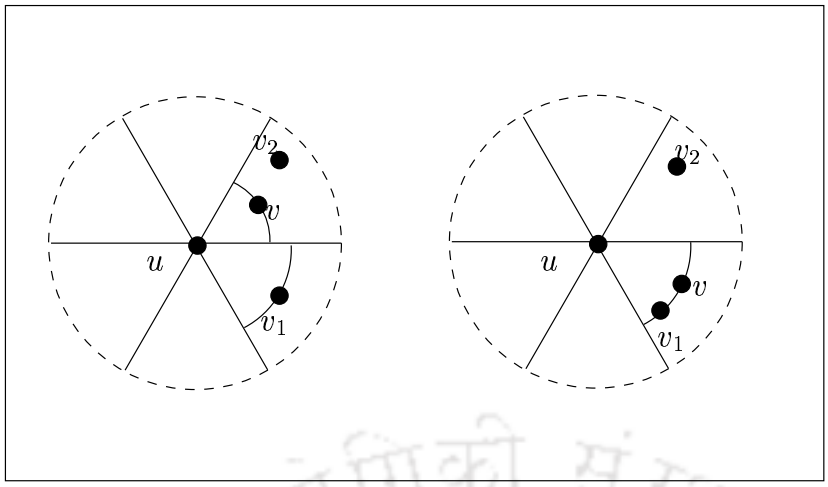


Figure 6.7: Case 3b: Node v moves into the sector j and $|\overline{uv}| = |\overline{uv_1}|$.

Case 3d (Node moves to another sector and RNG$_j(u) = \emptyset$): If $lune(u, v)$ is empty, RNG$_j(u) = \{v\}$. Updation in the sector i is same as other cases.

These local adjustments to the spanner due to mobility of the nodes can violate various geometric properties. Hence, the third phase reconstructs the RNG locally by using either the algorithm described in the Section 4.3.1 or the algorithm in Chapter 3, at every time interval t_2. That is, each node u computes RNG if there is any changes in 1-hop neighbor list or their position from previously constructed RNG.

The accuracy of the spanner depends on the time intervals t_0, t_1, and t_2. That is, the spanner is more accurate if these intervals are small, but it requires frequent updates.

In this method, the third phase does not require any additional communication. So, one can reconstruct the RNG in the second phase of the algorithm at every time interval t_1. However, the time complexity of update is proportion to the square of number of 1-hop neighbors of node u. That is, the maximum node degree in UDG. The algorithm MRNG does not use any extra control messages apart from the messages used in RNG.

Algorithm : MRNG

1. Each node u broadcast the message ***hello_packet***, which contain id and location information, at every t_0 time.

2. Each node u gathers 1-hop neighborhood information $n_1(u)$ by receiving the ***hello_packet***s.

3. Each node u constructs $RNG(u)$ with $n_1(u)$.

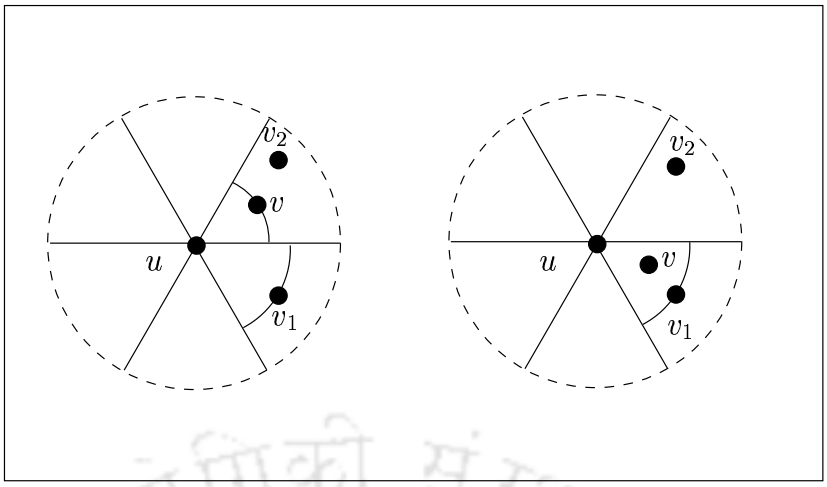


Figure 6.8: Case 3c: Node v moves to another sector and $|\overline{uv}| < |\overline{uv_1}|$.

4. After every t_1 units time, each node u follows the steps:

   (a). For each $v \in \text{RNG}_i(u)$ and $v \notin n_1(u)$

   /* node v goes out of transmission range of u */

   $\text{RNG}_i(u) = \text{RNG}_i(u) - \{v\}$.

   if ($\text{RNG}_i(u) = \emptyset$) then w=nearest neighbors of u in the sector i.

   if ($lune(u, w)$ is empty) then add w to $\text{RNG}_i(u)$.

   (b). if v enters into the transmission range of u

   Find the sector i, such that $v \in \text{RNG}_i(u)$.

   Let $w \in \text{RNG}_i(u)$.

   if ($d(u, v) < d(u, w)$)

   Remove w from $\text{RNG}_i(u)$.

   if ($lune(u, v)$ is empty) then add v to $\text{RNG}_i(u)$.

   (c). if v moves from sector i to j and $d(u, v) > d(u, v_1)$, where $v_1 \in \text{RNG}_j(u)$

   /* Case 3a */

   if ($\text{RNG}_i(u) = \emptyset$) then compute new $\text{RNG}_i(u)$.

   (d). if v moves from sector i to j and $d(u, v) = d(u, v_1)$, where $v_1 \in \text{RNG}_j(u)$

   /* Case 3b */

   if ($\text{RNG}_i(u) = \emptyset$) then compute new $\text{RNG}_i(u)$.

   if ($lune(u, v)$ is empty) then add $v \in \text{RNG}_j(u)$.

   (e). if v moves from sector i to j and $d(u, v) < d(u, v_1)$, where $v_1 \in \text{RNG}_j(u)$

/* Case 3c */

if (RNG$_i(u) = \emptyset$) then compute new RNG$_i(u)$.

if ($lune(u, v)$ is empty) then add RNG$_j(u)$ = $\{v\}$.

else RNG$_j(u) = \emptyset$.

(f). if v moves from sector i to j and RNG$_j = \emptyset$

/* Case 3d */

if (RNG$_i(u) = \emptyset$) then compute new RNG$_i(u)$.

if ($lune(u, v)$ is empty) then add RNG$_j(u)$ = $\{v\}$.

5. Each node u does the following at every t_2 unit time

/* construct RNG*/

if there is any change in $n_1(u)$ or their position

Construct RNG(u).

6.4 Mobile Gabriel graph (MGG)

Initially, the Gabriel graph is constructed with all the nodes, later changes to the network graph are made locally when the nodes move around.

In this algorithm, each node broadcasts the message *hello_packet* containing id and its location information, and listens for the same broadcasted by other nodes. Each node u, after collecting the neighborhood information $n_1(u)$, constructs the GG with $n_1(u)$. For constructing GG, each node u checks, whether the $disk(u, v)$ contain any node w inside or not, where $v \in n_1(u)$ and $w \in n_1(u) - \{v\}$. If it does not contain, add the edge $\overline{uv}$ to GG. Note that the $disk(u, v)$ represents the circle with the diameter $|\overline{uv}|$ centered at $\frac{u+v}{2}$.

Each node u checks at every t_1 time units, where $t_1 > t_0$, whether a node v is in GG(u) and not in $n_1(u)$. Then, node u removes the node v from its GG(u). In other words, if a node v is moved out of u's transmission range then node u updates its neighbors by removing the node v. That is $GG(u) = GG(u) - \{v\}$. When a node v enters into the u's transmission range, then add the edge $\overline{uv}$, if the $disk(u, v)$ does not contain any other node $w \in n_1(u)$ (see the Figure 6.9).

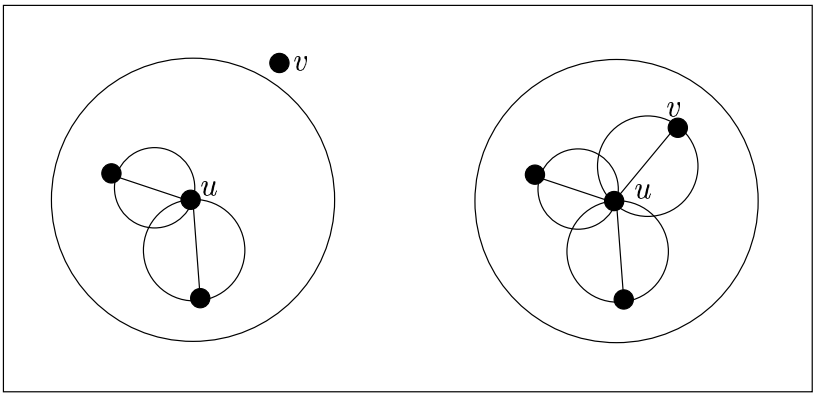


Figure 6.9: Node v enters into the transmission range of u.

Similar to the third phase of MRNG, each node u reconstructs the spanner with its one hop neighbors $n_1(u)$ at every t_2 time units, if there is any change in neighbor lists and their position. The algorithm MGG does not use any additional messages other than the messages used in GG.

Algorithm : MGG

---

1. Each node u broadcasts its id and position information through the ***hello_packet***.

2. Each node u collects 1-hop neighborhood information $n_1(u)$ by receiving the ***hello_packet***s.

3. Each node u constructs $GG(u)$ with $n_1(u)$.

4. Each node u do the following at every t_1 units time

   (a). For each node $v \in$GG(u) and $v \notin n_1(u)$

   GG(u) = GG(u) - $\{v\}$.

   (b). For every new neighbor $v \in n_1(u)$

   if ($disk(u, v)$ is empty) then add $\overline{uv}$ to GG.

5. Each node u do the following at every t_2 units time

   if there is any change in $n_1(u)$ or their position

   Construct GG with $n_1(u)$.

6.5 Simulation and performance

The performance of MLDel, MRNG, and MGG are evaluated using network simulator ($ns2.28$) [NS205]. Apart from the proposed spanner, we have simulated PLDel, RNG, and GG for the purpose of comparison. The six routing protocols greedy routing (Grdy) [BMSU01], compass routing (Cmp) [BCSW98, KSU99], random compass routing (RandCmp) [KSU99], most forward routing (MFR) [TK84], nearest neighbor routing (NNR) [HL86], and farthest neighbor routing (FNR) [LCW02] use these spanners as underlying network graph.

We have taken five different node scenarios, each containing 100 nodes randomly distributed in a $1000 \times 1000\ m^2$ area. The transmission range of each node is $250m$. Packets of size 512 bytes are sent from source at constant rate. The total simulation time is 400 seconds. We have taken three different connection patterns with sources at one side of the grid and the destination nodes at the other side. We run several simulations at different transfer rates.

The mobility model used in our simulation is random way point (RWP) model [CBD02]. The speed of each node is chosen randomly between $0\ m/s$ and $10\ m/s$. The time intervals t_0, t_1, and t_2 are set to 10, 40, and 80 seconds respectively.

6.5.1 Simulation results

We have calculated delivery ratio of mobile spanners using the routing protocols Grdy, Cmp, RandCmp, MFR, NNR, and FNR and compared with their counter static spanners. The performances of the mobile spanners are better than their counter parts. This is because, the topology changes due to the node mobility are reflected in mobile spanners. A node with maximum speed $(10m/s)$ takes 25 seconds to cross the transmission range of a node. But the Phase 2, which is responsible to update the edge list, runs in every 40 seconds. Thus, in the MLDel spanner, there is a 37 percent possibility of packet loss if a relay node has maximum speed. If we assume that the average speed of a node is $5m/s$ then the node moves out of the transmission range in 50 seconds. Thus, the packet drop is protected. However, in the next iteration of Phase 2, the node moves out of the transmission range in 10 seconds and there is 75 percent of packet loss. However, we need to consider the relative velocity of a node instead of only the velocity of neighboring nodes.

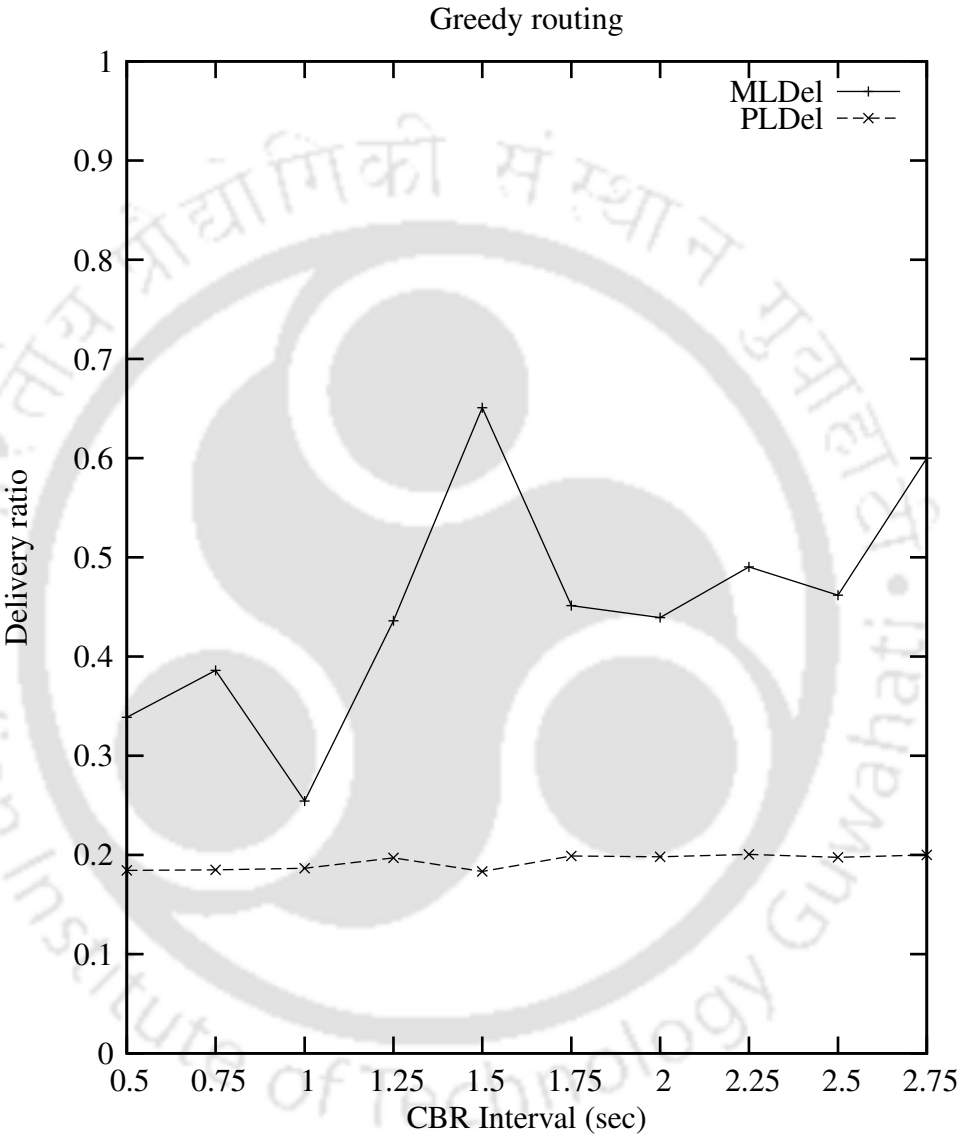


Figure 6.10: Greedy routing.

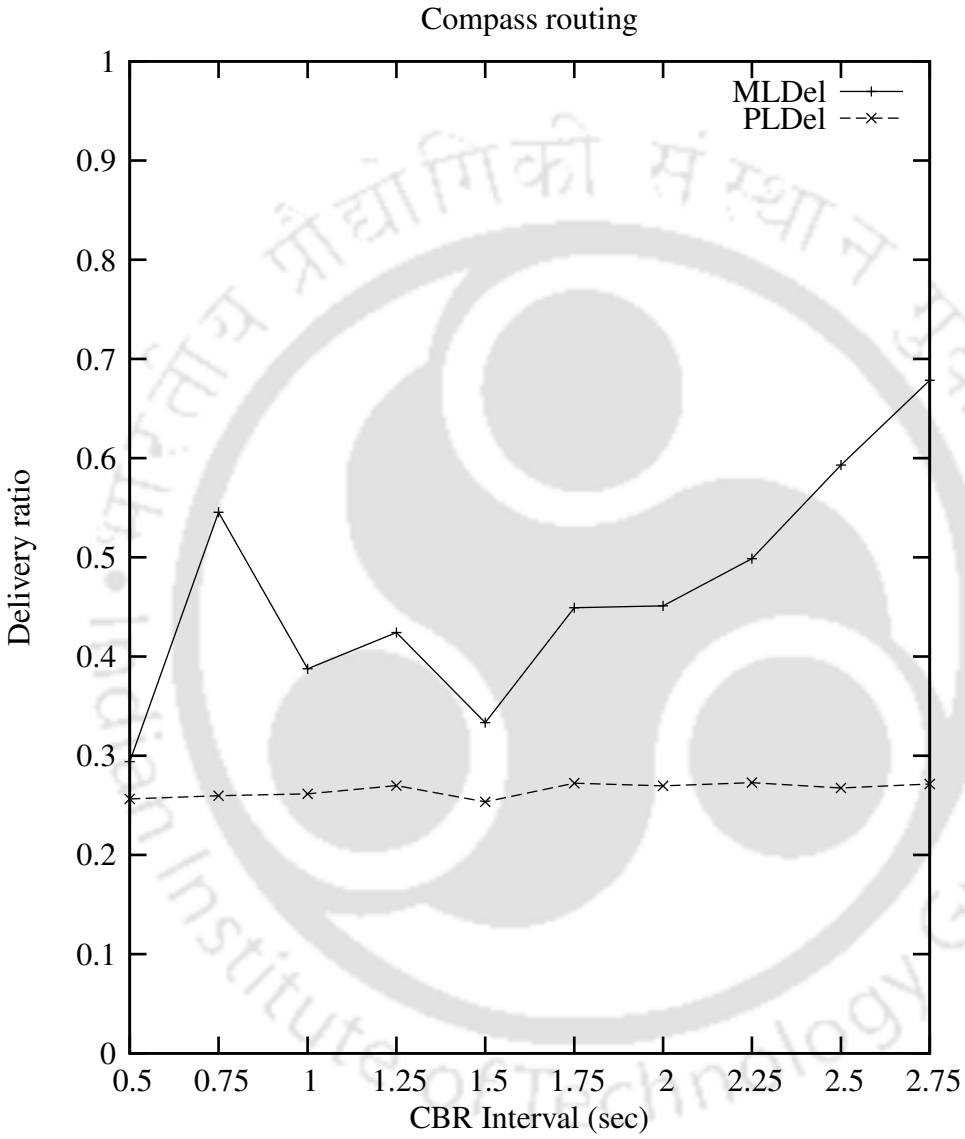


Figure 6.11: Compass routing.

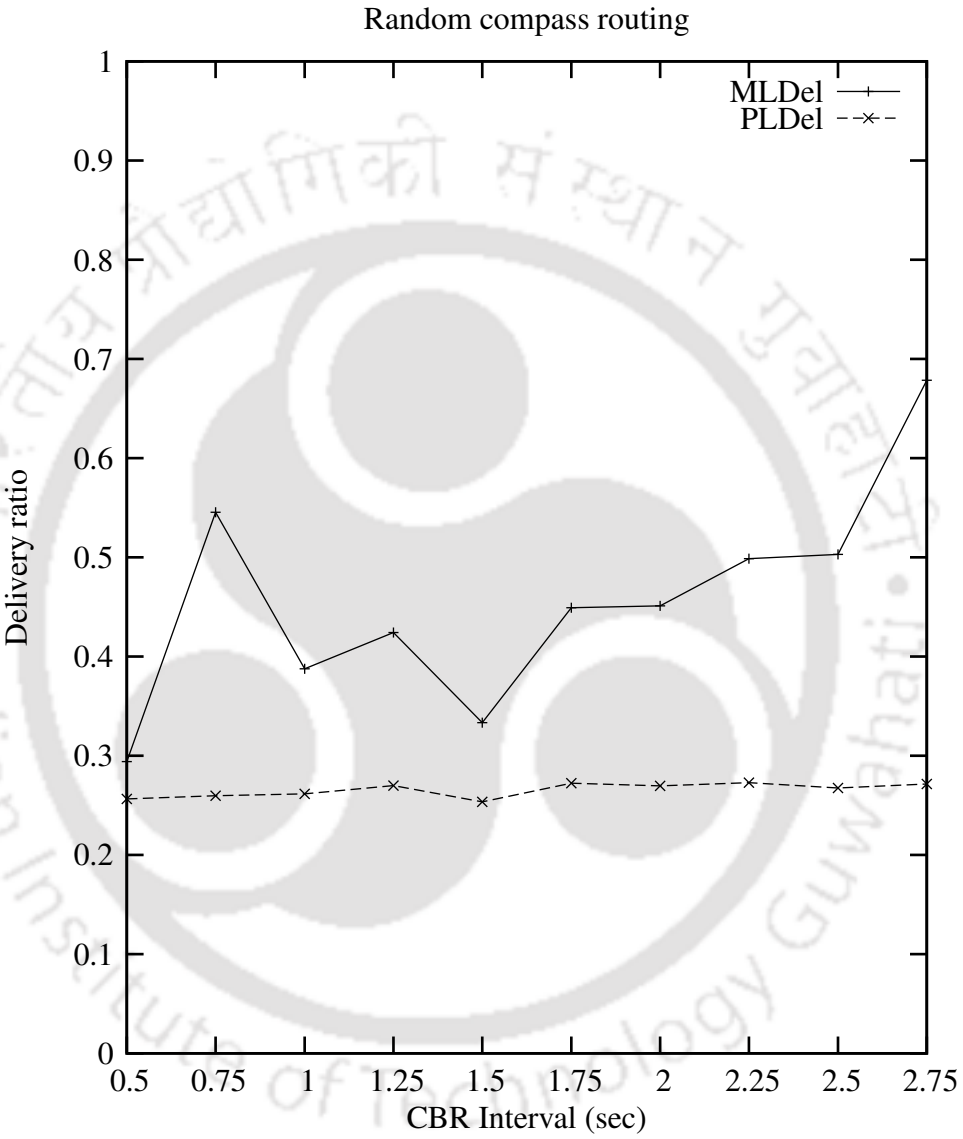


Figure 6.12: Random compass routing.

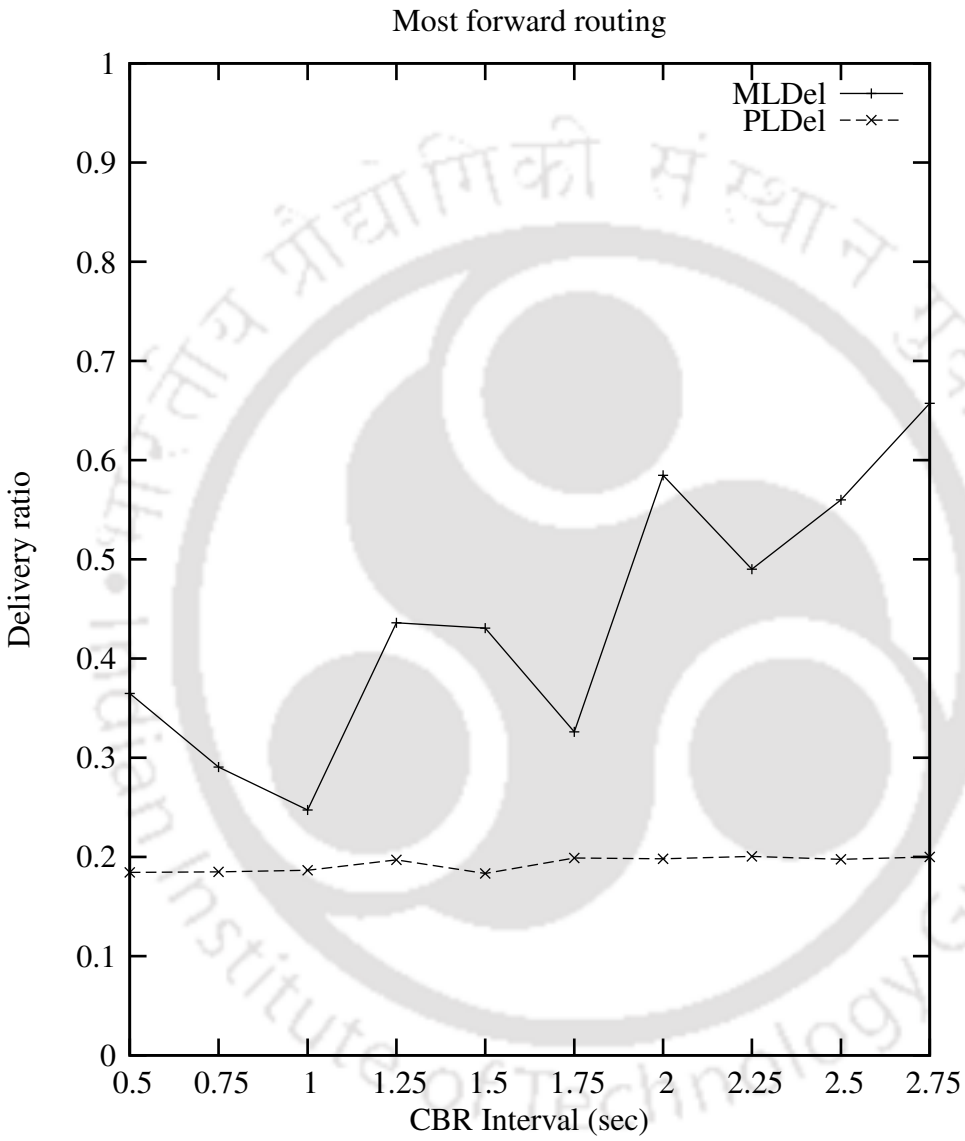


Figure 6.13: Most forward routing.

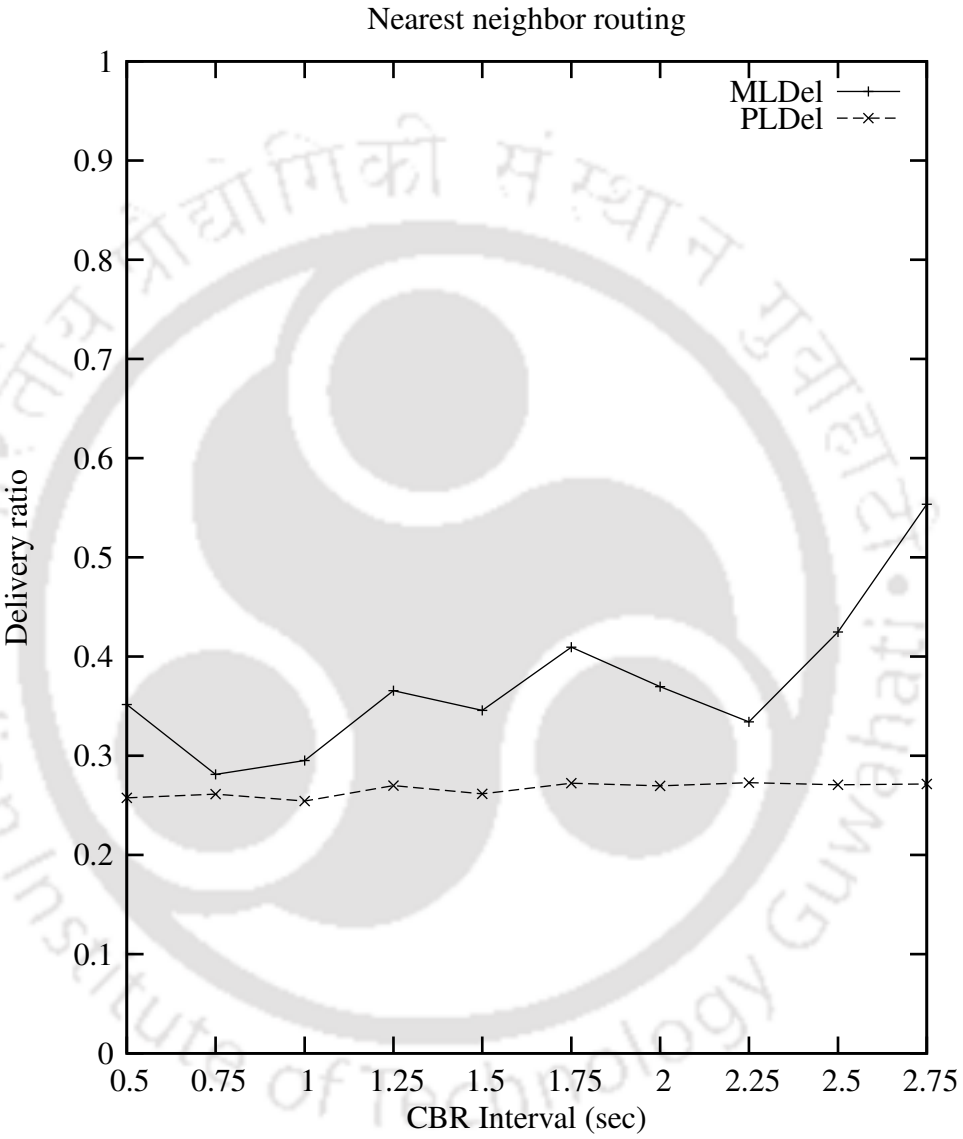


Figure 6.14: Nearest Neighbor routing.

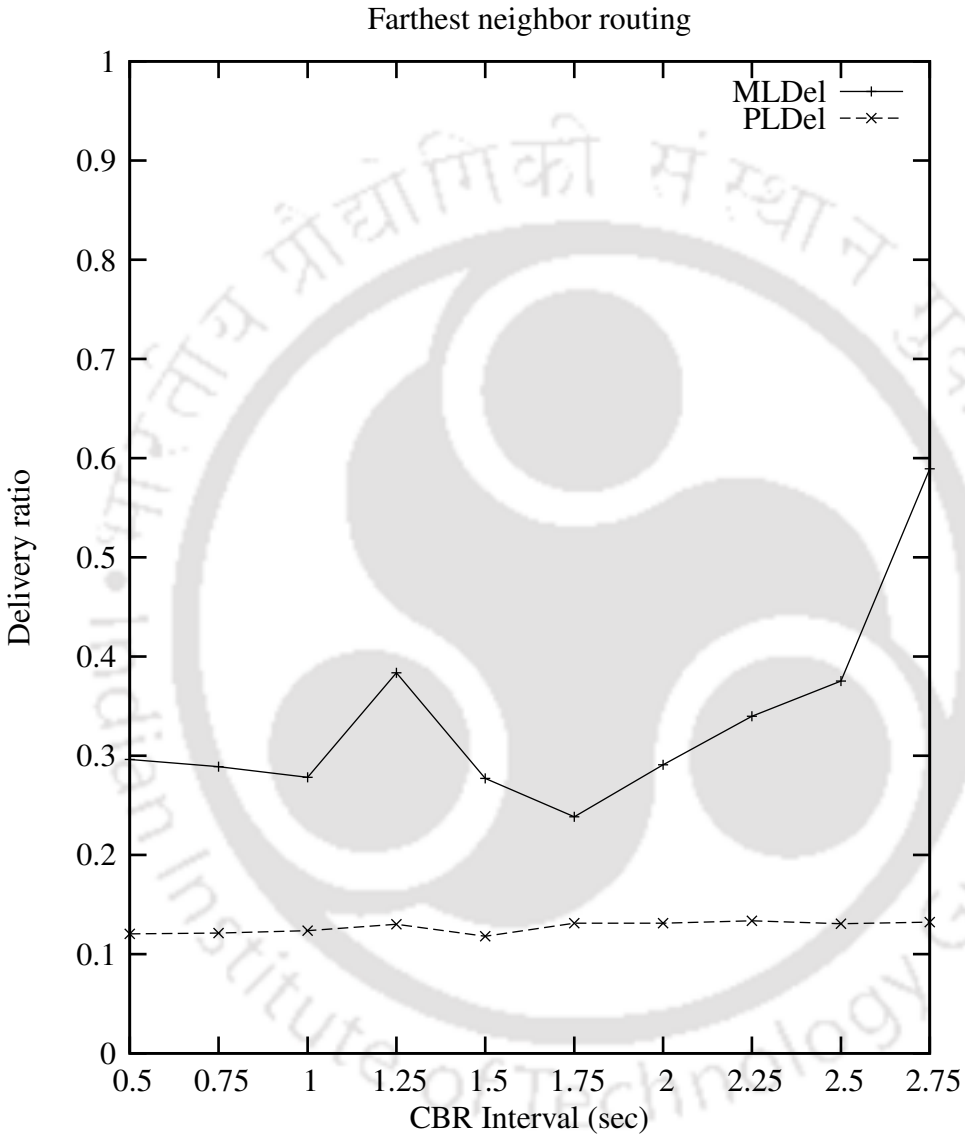


Figure 6.15: Farthest Neighbor routing.

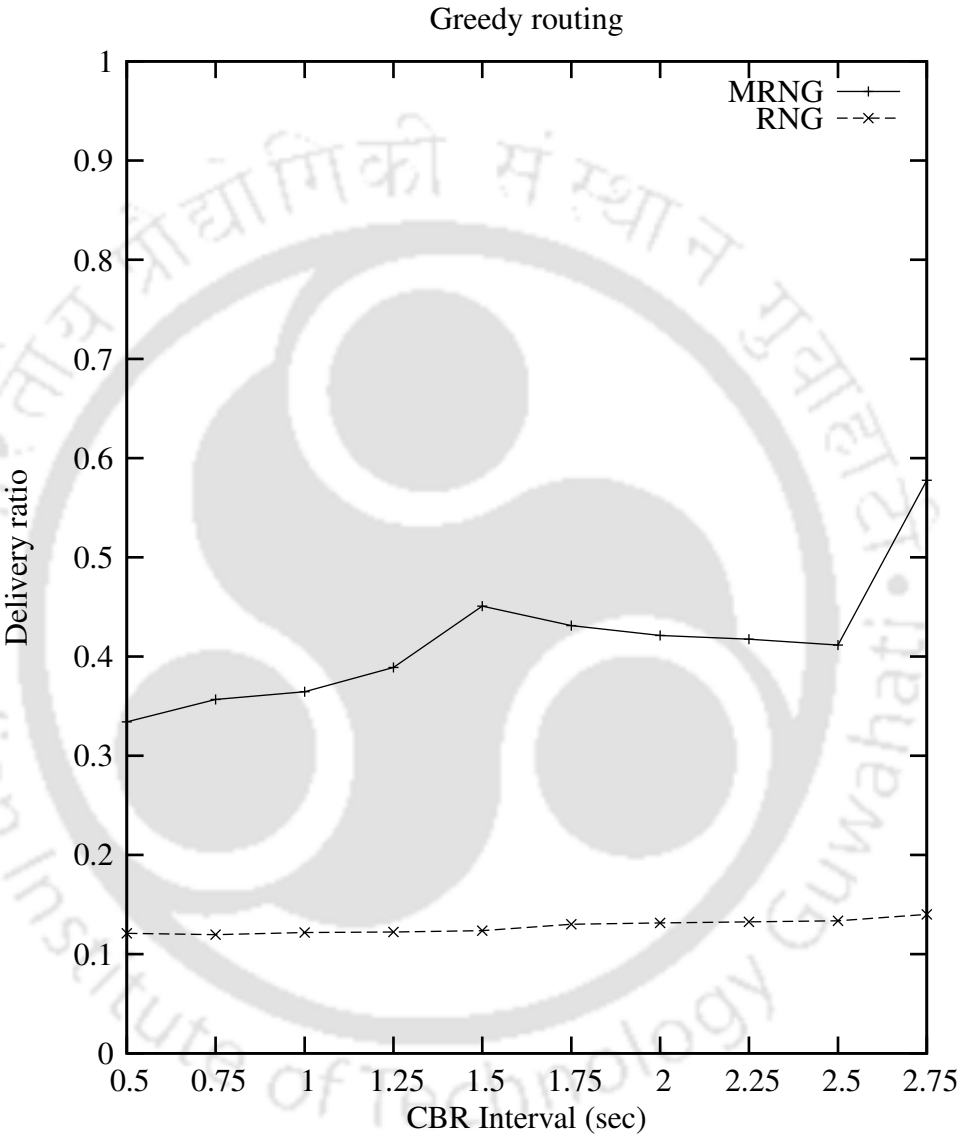


Figure 6.16: Greedy routing.

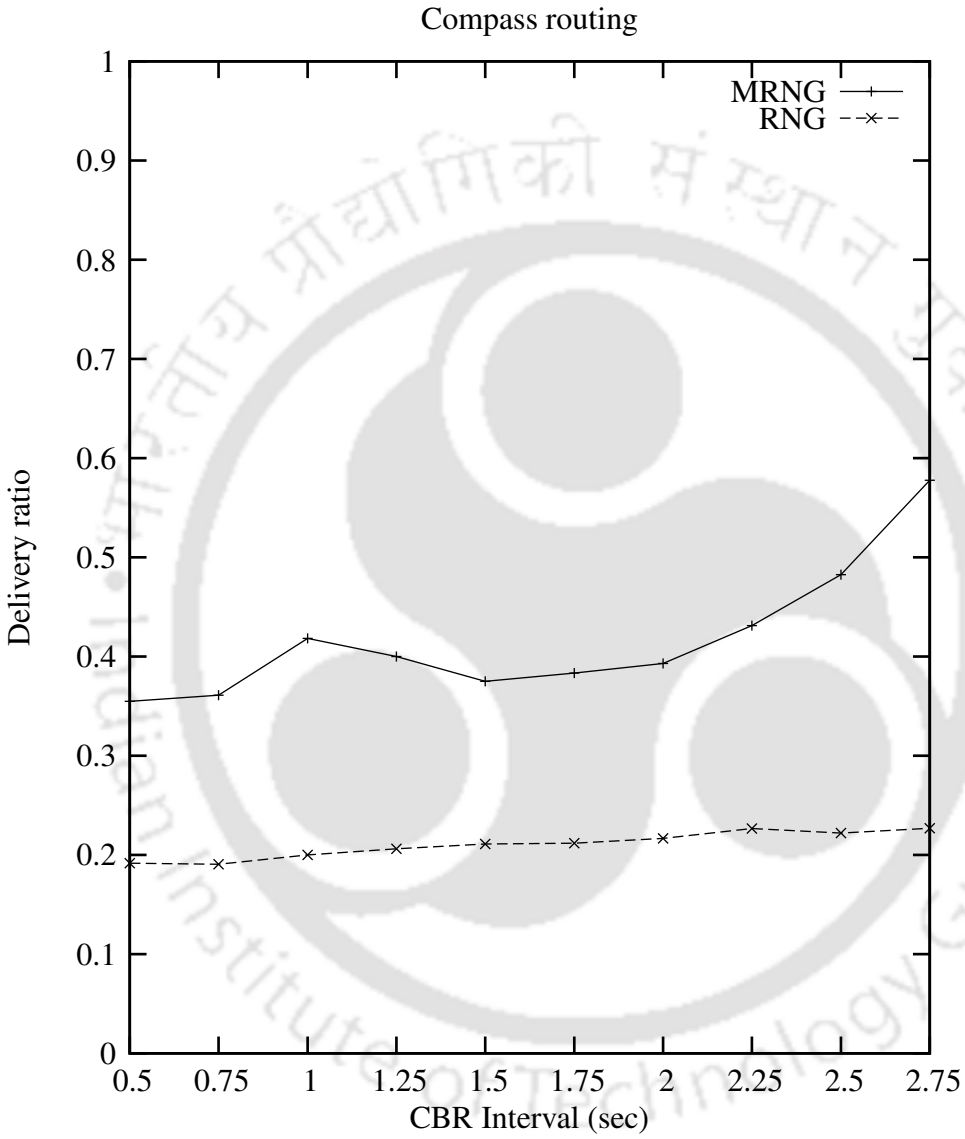


Figure 6.17: Compass routing.

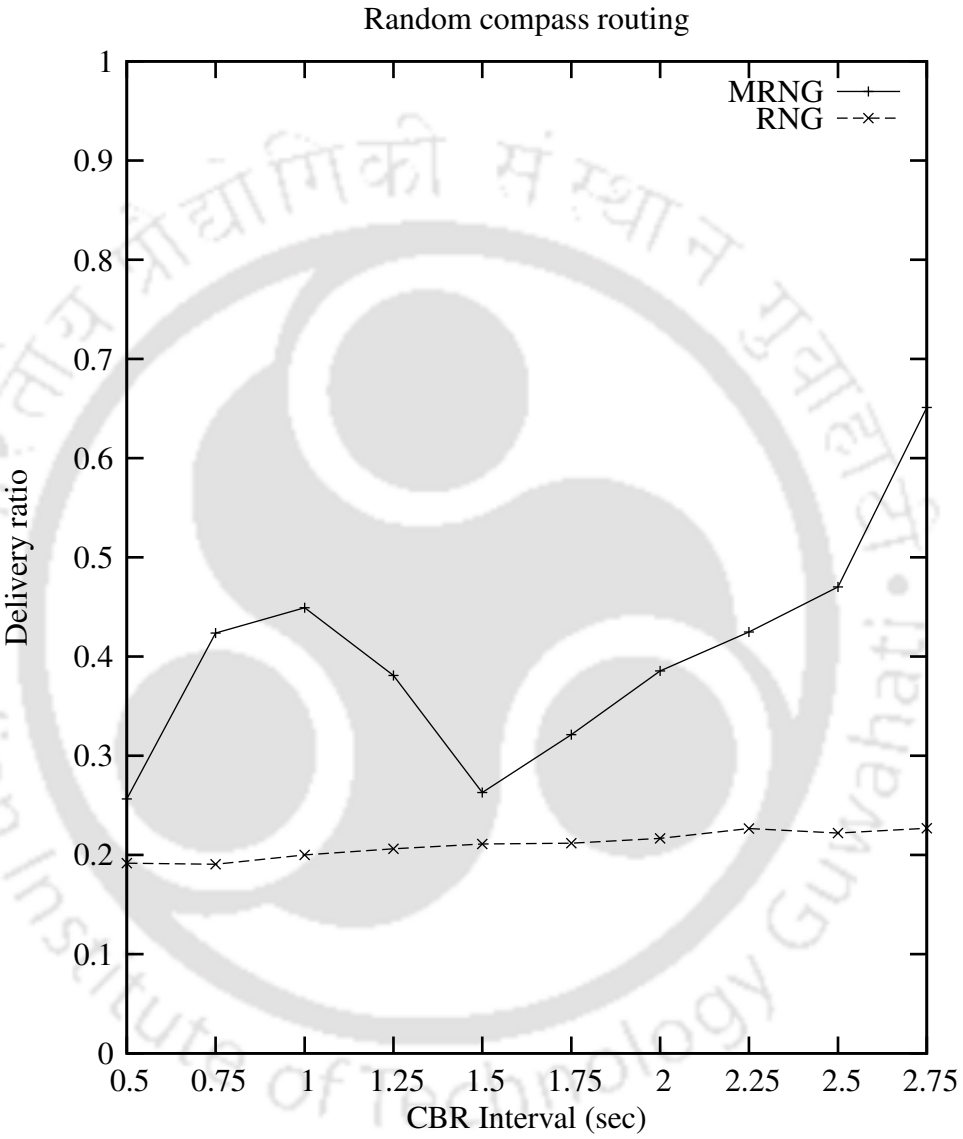


Figure 6.18: Random compass routing.

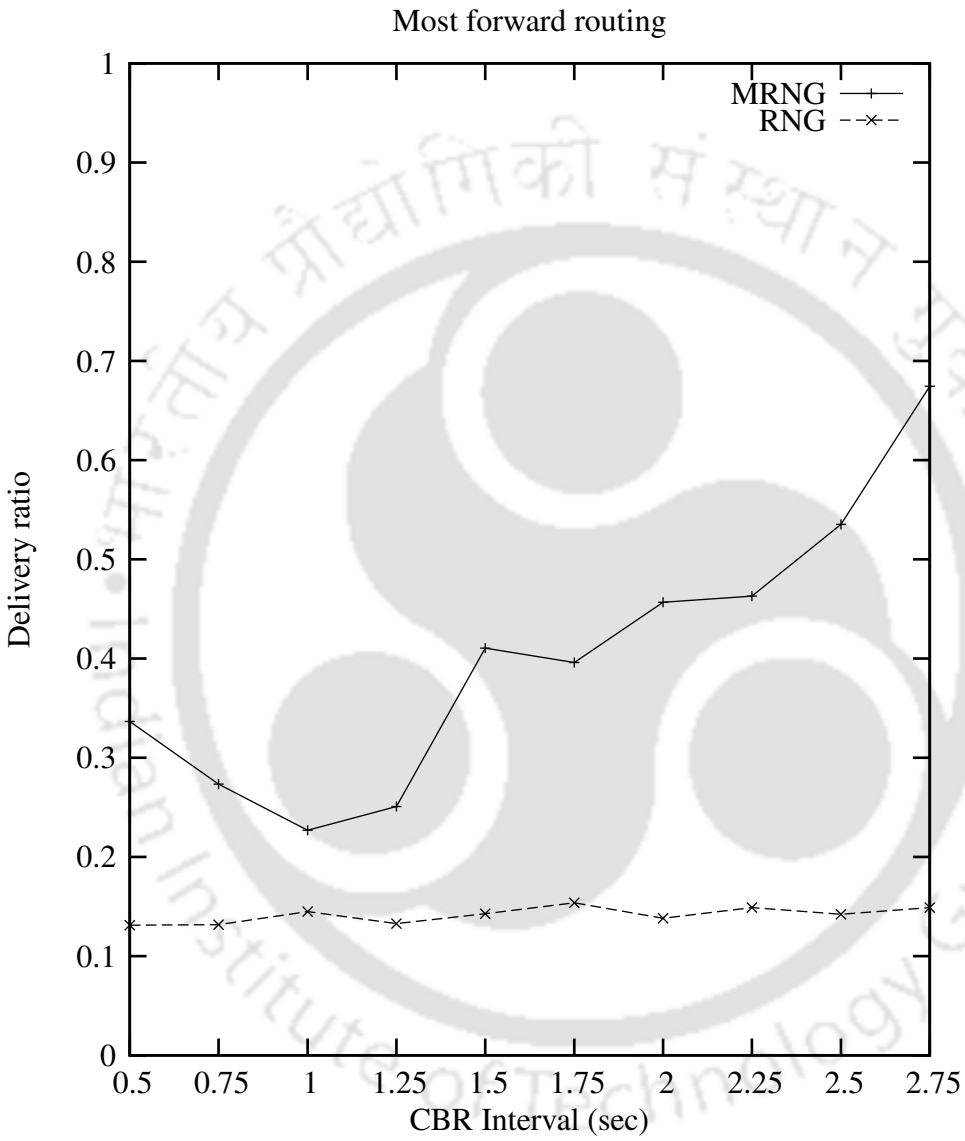


Figure 6.19: Most forward routing.

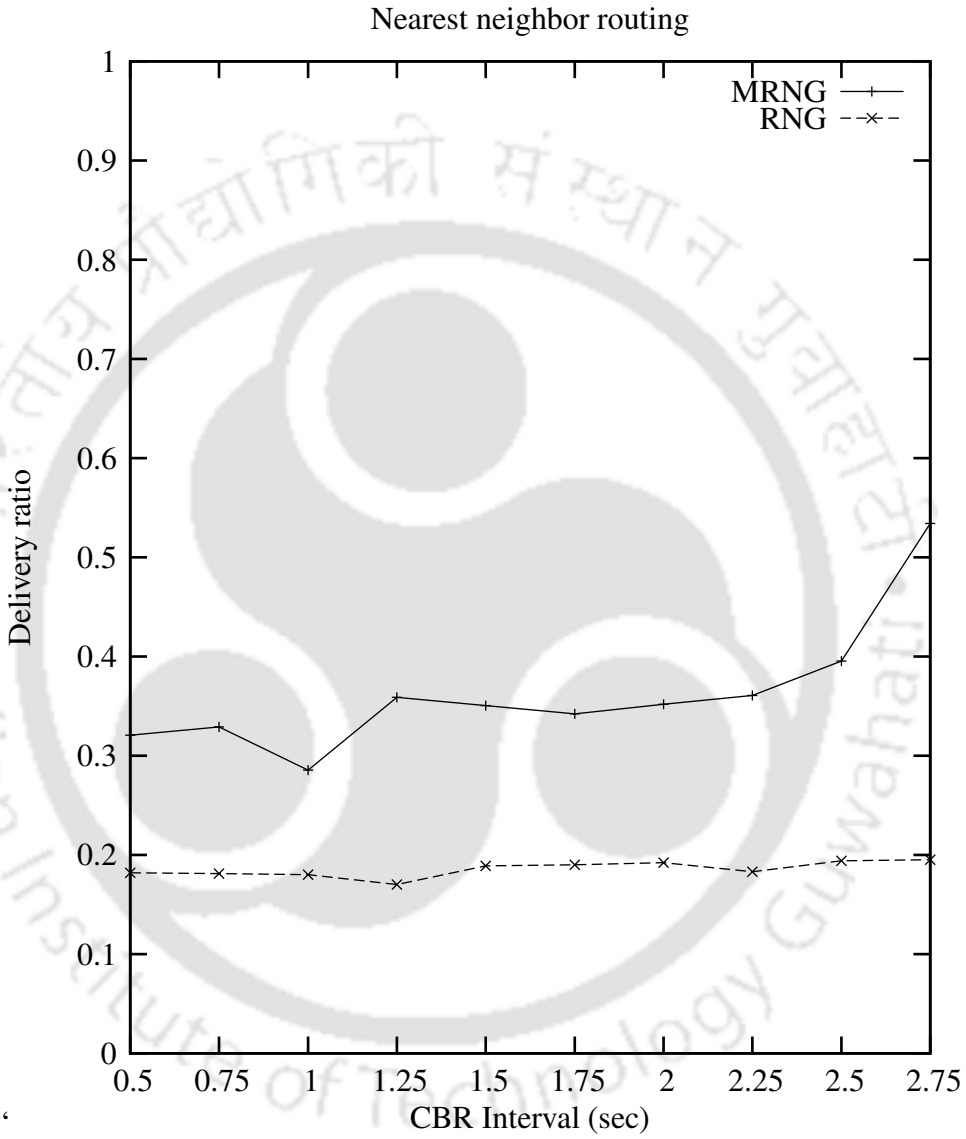


Figure 6.20: Nearest Neighbor routing.

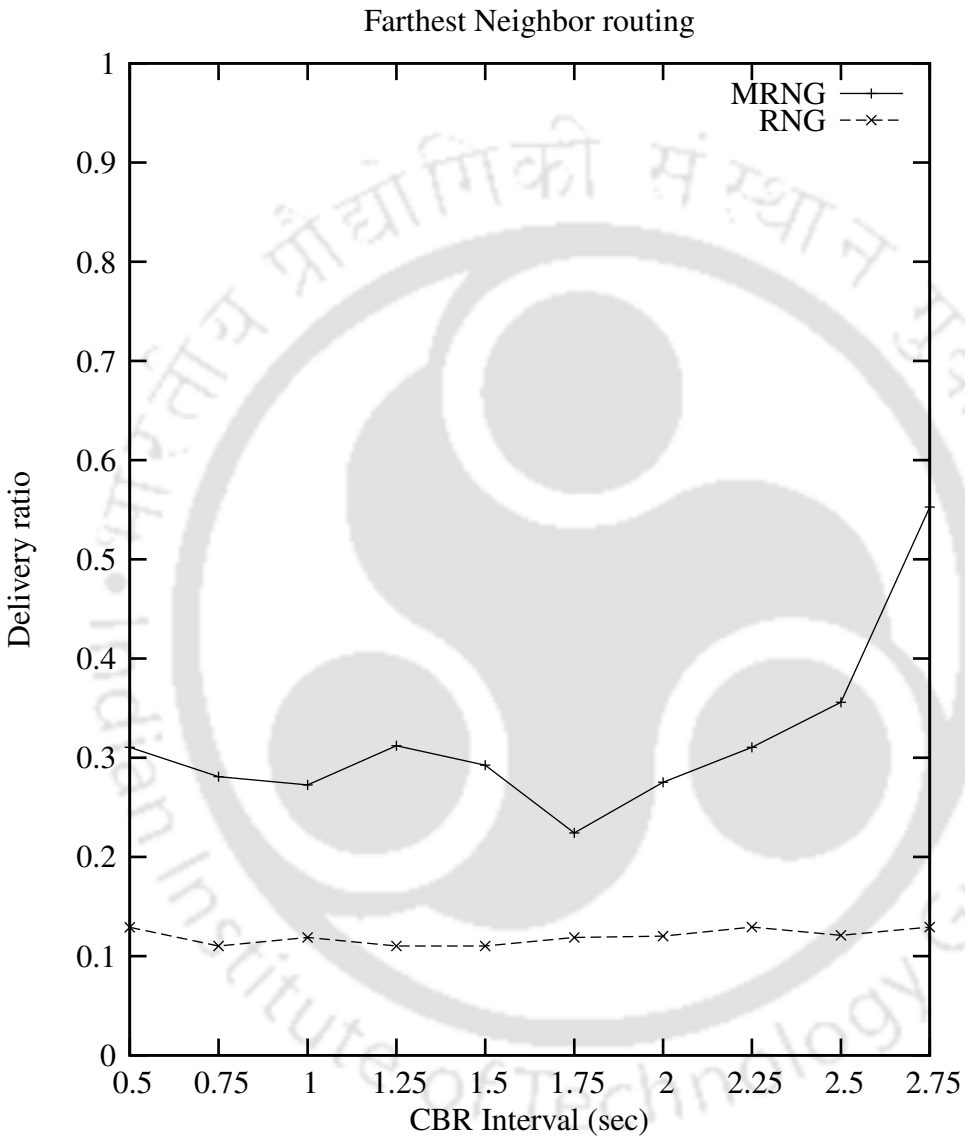


Figure 6.21: Farthest Neighbor routing.

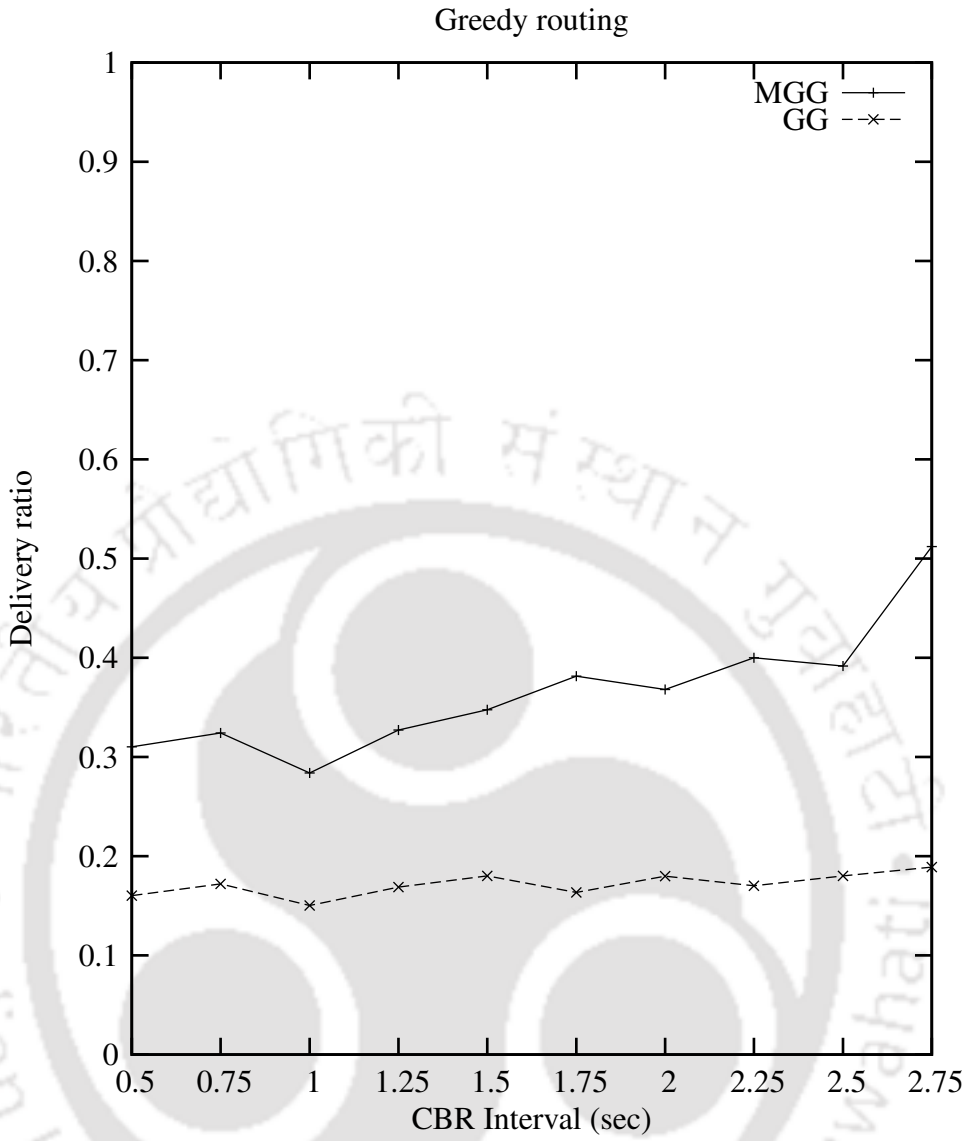


Figure 6.22: Greedy routing.

The Phase 3, which is responsible to protect the geometric properties, runs 8 times in the entire simulation. The static spanner PLDel suffers from the packet loss as the updations are not considered when the nodes move around. The same is applicable for algorithms MRNG & RNG and MGG & GG.

In the next experiment, we compute the average packet delay for only the successfully received packets. We run the greedy (Grdy) routing protocol on two spanners MLDel and PLDel at various CBR intervals. From the Figure 6.28, we say that the average packet delay for the spanner MLDel is lower than PLDel. This is because, even though the number of received packets are few, the end-to-end packet delay computed is high due to node

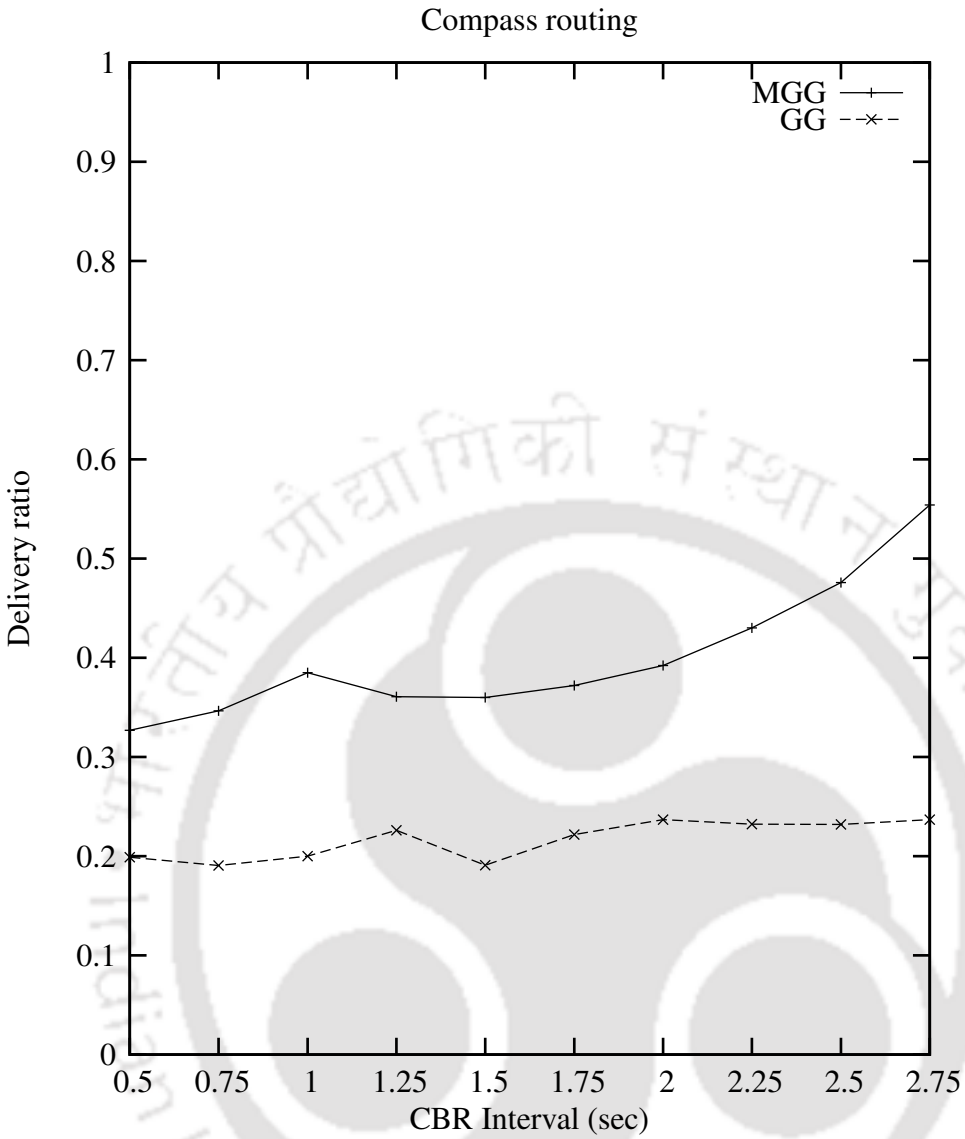


Figure 6.23: Compass routing.

mobility. The major delays encountered due to the node mobility are transmission delays, propagation delays, and retransmission delays. The same is applicable for MRNG & RNG and MGG & GG, as shown in the Figure 6.29 and Figure 6.30, respectively. The other observation from the plots is that the average packet delay is in the order of MLDel $\leq$ MGG $\leq$ MRNG and PLDel $\leq$ GG $\leq$ RNG. This is because the path length and hop count is in the order of PLDel $\leq$ GG $\leq$ RNG. That is, the propagation delays and transmission delays for the spanners is in the same order and hence the average packet delay is also in the same order.

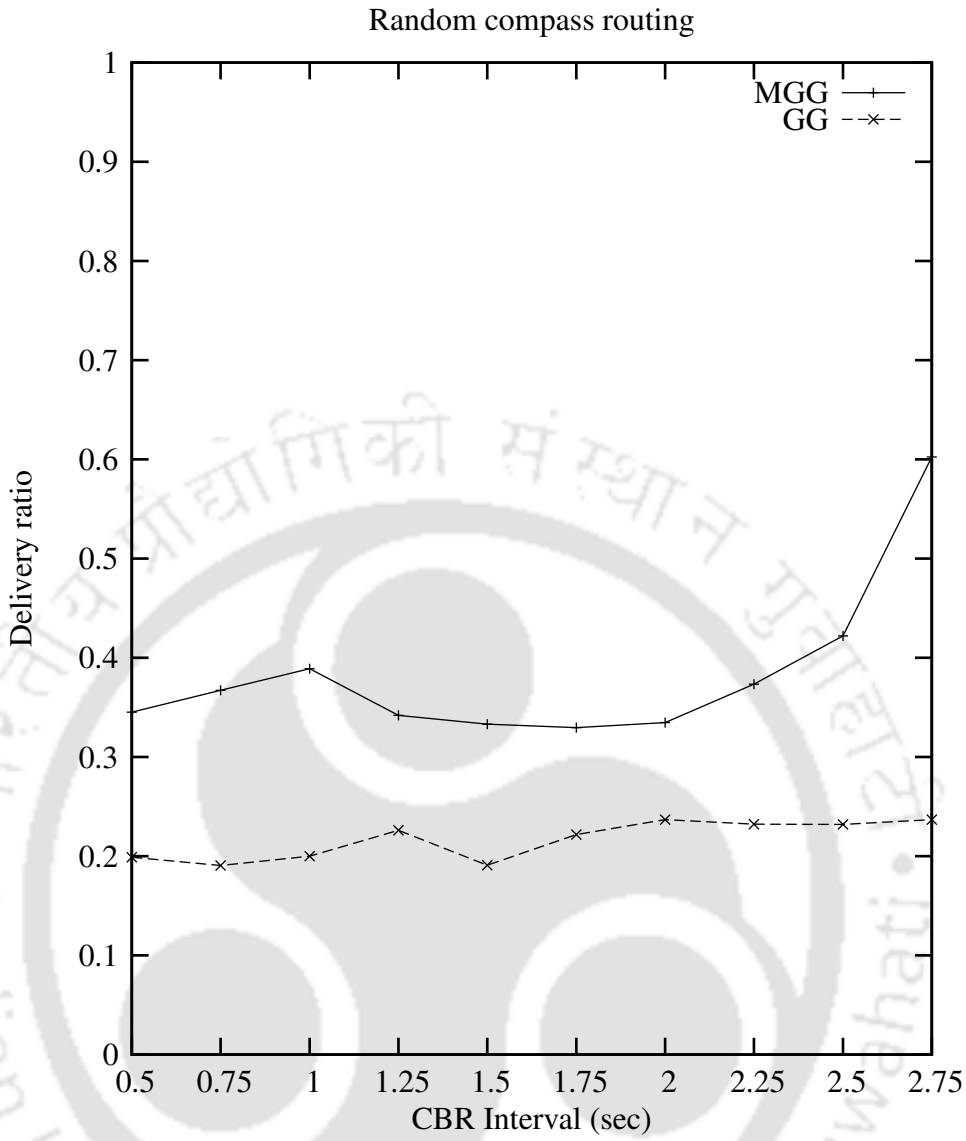


Figure 6.24: Random compass routing.

6.6 Conclusions

In this chapter, we have proposed methods to maintain PLDel, RNG, and GG under node mobility. These methods are simulated in *ns-2.28* and compared with their static counter parts. The simulation results show that the performance of mobile spanners is better than their static counter parts.

We have considered random way point (RWP) mobility model [JM96] in our simulation. It would be interesting to study the mobile spanners under different mobility models, like Gauss-Markov mobility model [LH99], random direction mobility [RMSM01]. An-

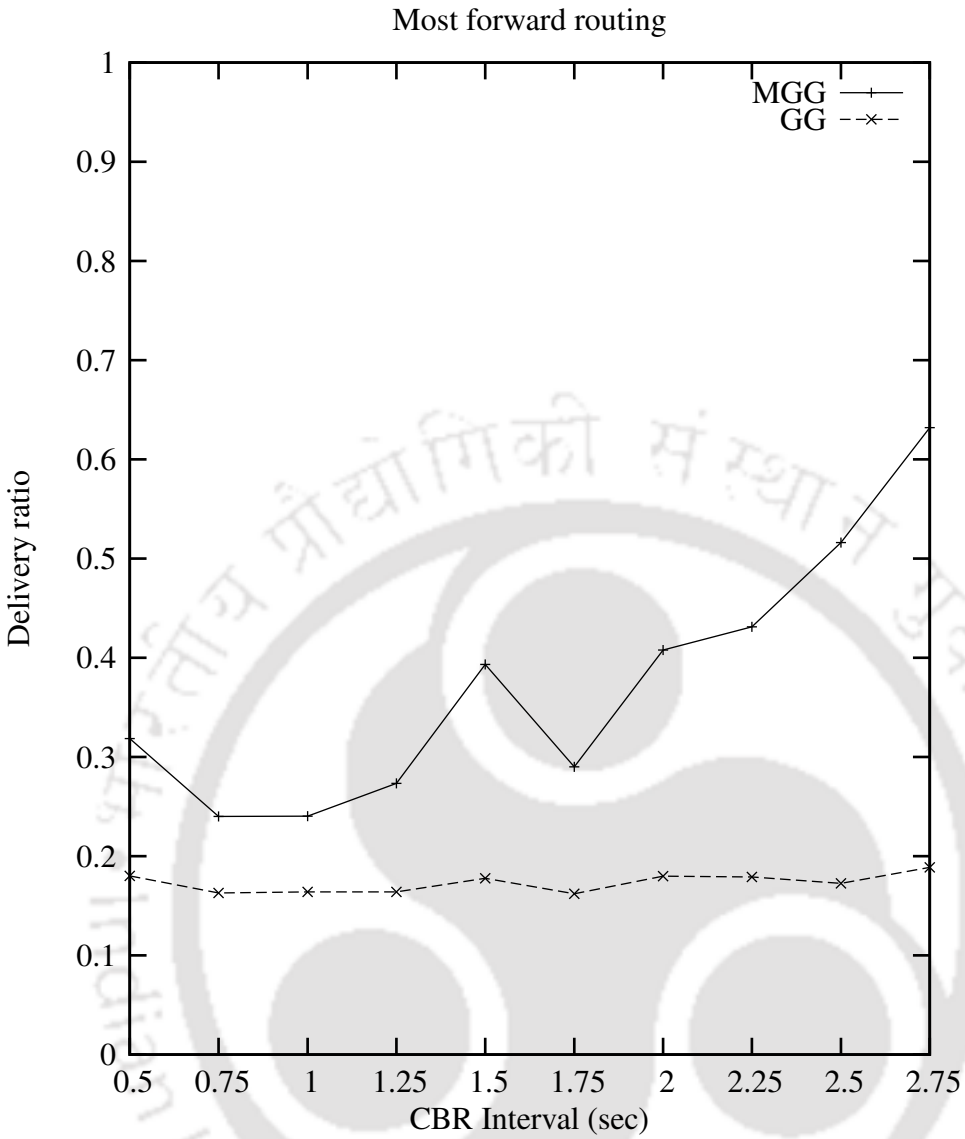


Figure 6.25: Most forward routing.

other interesting future direction is to use various node position prediction methods [DJS04] to maintain geometric spanners efficiently to further improve various network parameters.

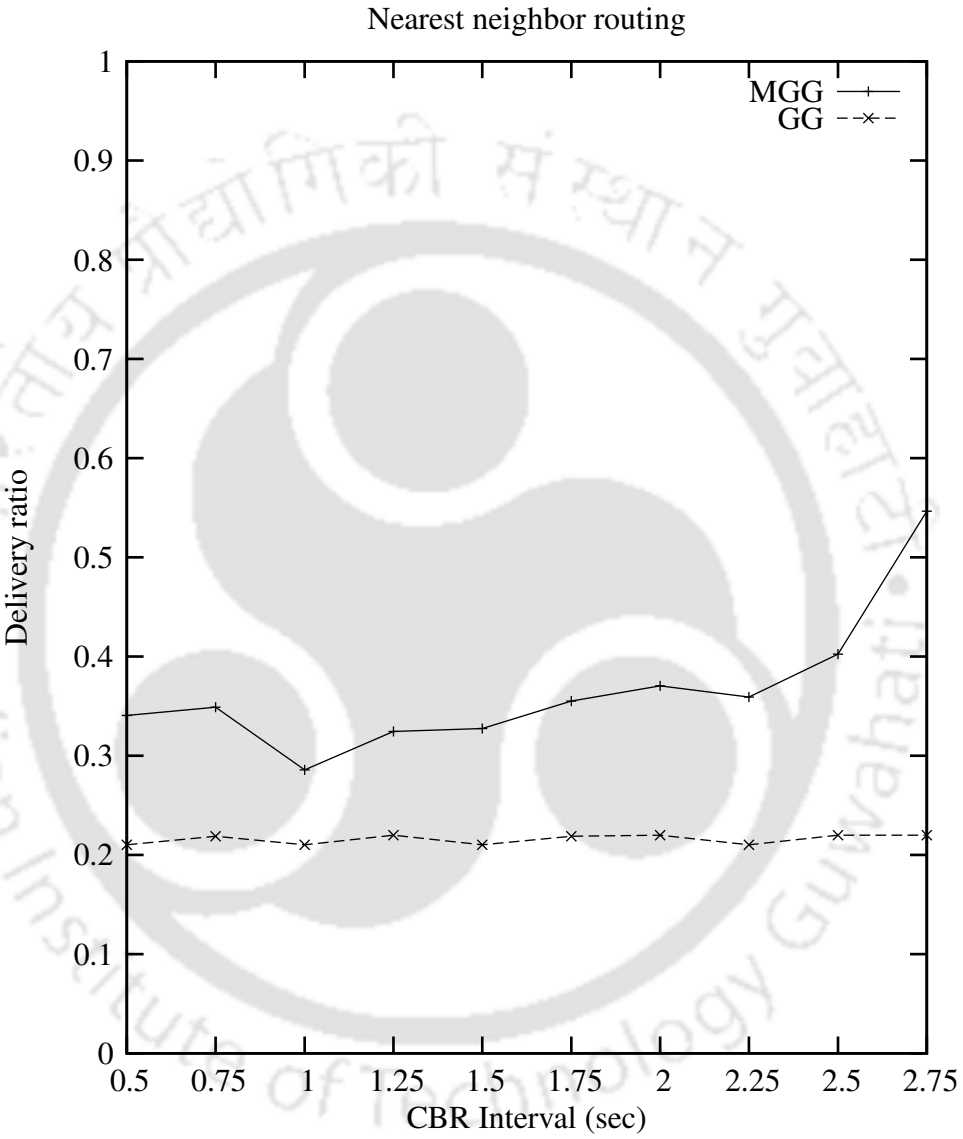


Figure 6.26: Nearest Neighbor routing.

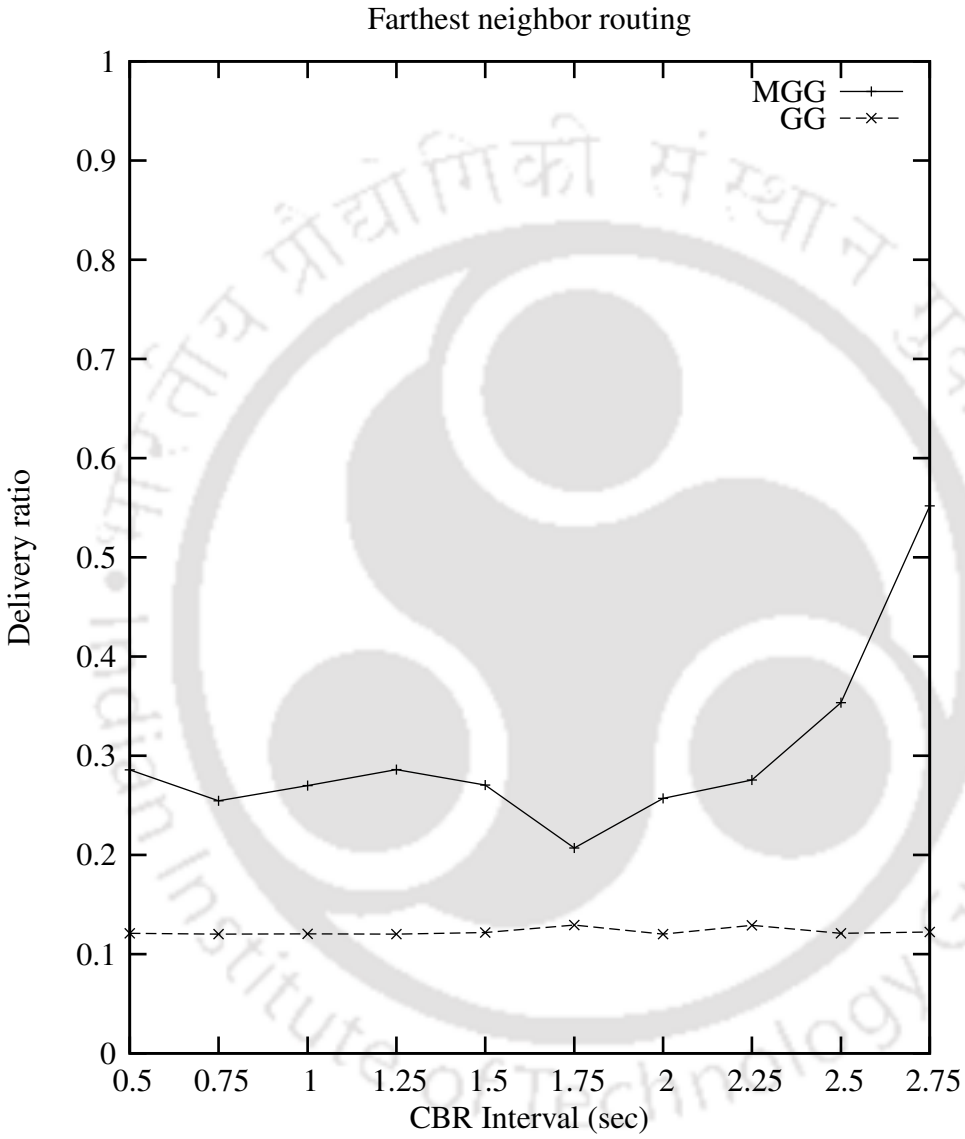


Figure 6.27: Farthest Neighbor routing.

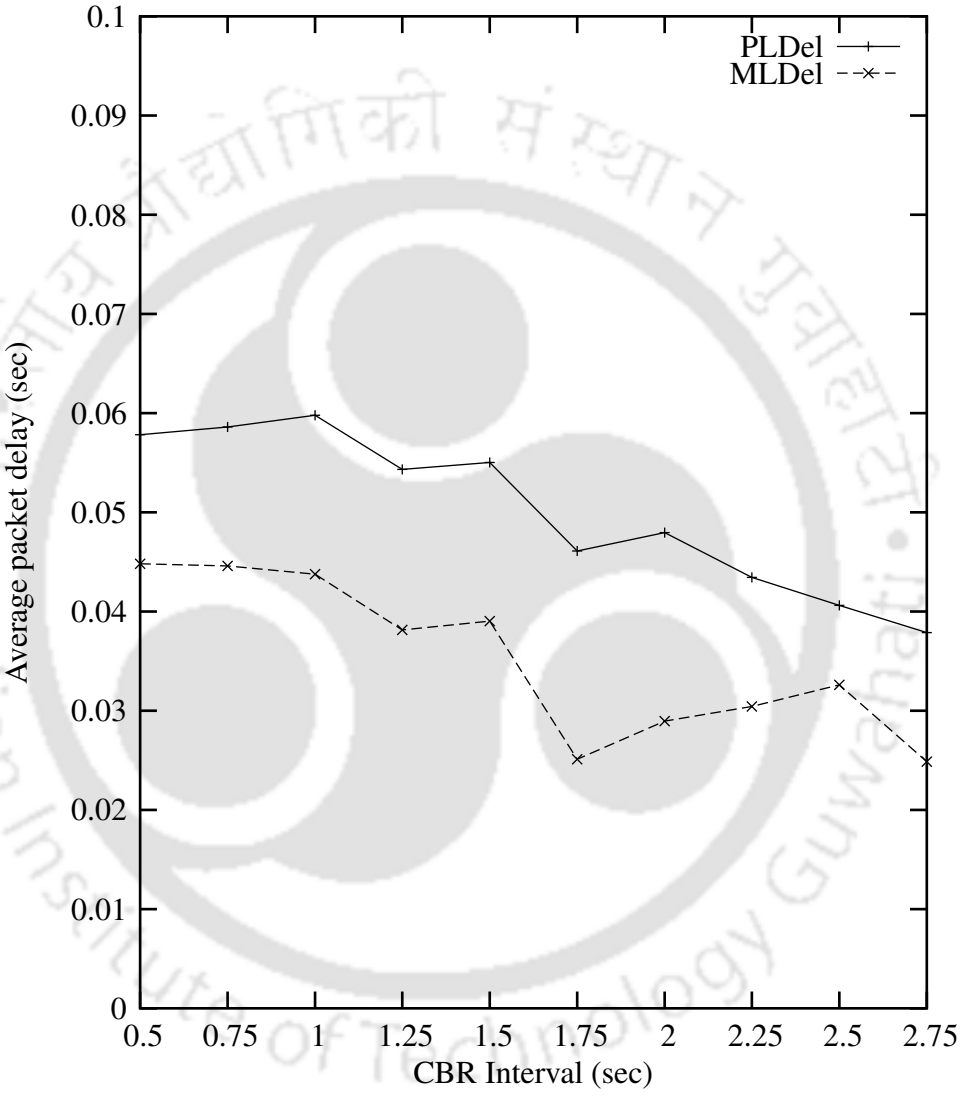


Figure 6.28: Delay in MLDel.

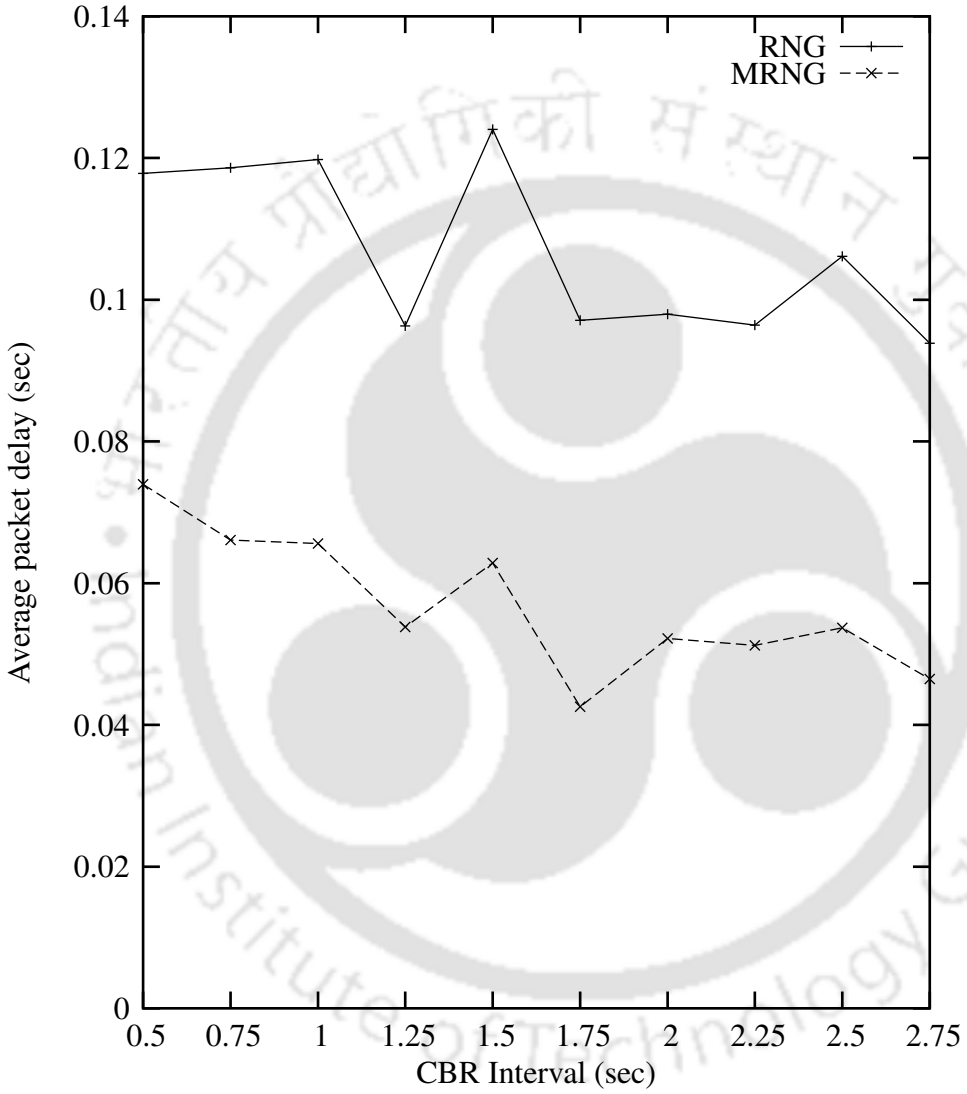


Figure 6.29: Delay in MRNG.

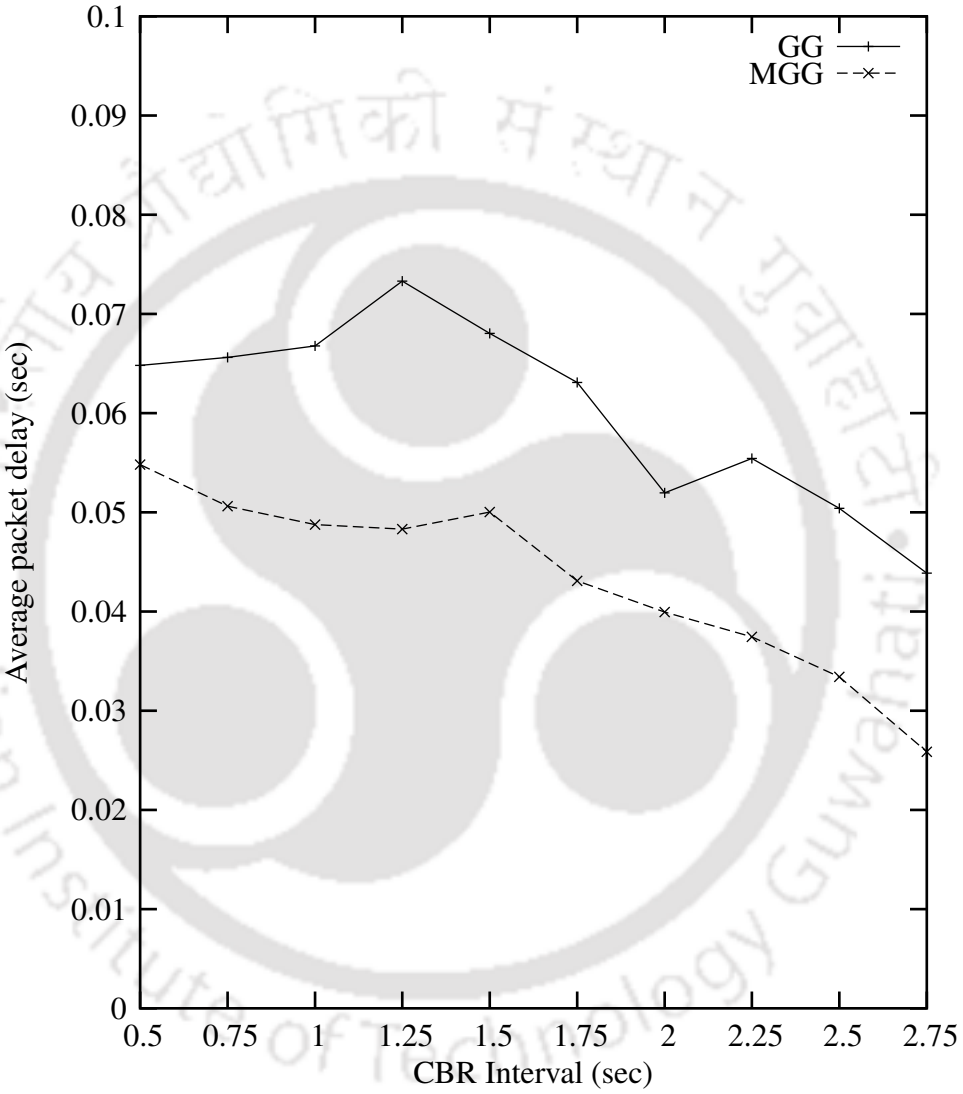


Figure 6.30: Delay in MGG.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 7

Conclusions

In this thesis, we have attempted to study the Delaunay triangulation based spanners, planarized local Delaunay triangulation (PLDel), relative neighborhood graph (RNG), and Gabriel graph (GG) at various network requirements. The highlights of our contributions, in chapter wise order, are as follows.

In the second chapter, we have studied the need of constrained based geometric graphs and proposed three spanners, constrained Delaunay triangulation (CDT), constrained relative neighborhood graph (CRNG), and constrained Gabriel graph (CGG) to reduce the number of hops. The simulation results in *ns2.28* consolidates our claim of average hop count reduction in these spanners, which in turn reduces the delay and jitter, and increases the throughput. The increasing order of the average hop count, delay, and jitter, in these spanners are CDT, CGG, CRNG, LDel^1, PLDel, GG, and RNG, where as the throughput follows the reverse order. The delivery ratio of constrained geometric spanners are better than their non constrained counter parts.

We have proposed efficient algorithms to maintain the geometric spanners PLDel, RNG, and GG under frequent node down and join to improve delivery ratio. The proposed algorithms preserve the geometric properties by dynamically reconstructing the graphs. Simulation results consolidate our claim that the spanners DLDel, DRNG, and DGG recover quickly and prevent packet loss occur in PLDel, RNG, and GG, respectively, due to frequent node failures.

In the fourth chapter, for fault tolerant routing, we have proposed fault tolerant spanners FTLDel, FTRNG, and FTGG by choosing more stable nodes in the network. These

spanners show better performance than their counter part spanners at frequent node failure conditions.

We have proposed efficient algorithms to maintain the spanners PLDel, RNG, and GG under node mobility. Our approach follows low cost frequent local updates and high cost less frequent global updates to improve the delivery ratio and restore the geometric properties. Simulation results show improvement in delivery ratio.

7.1 Future work

In ideal conditions, the long constraint edges in the network do not affect the link quality, as per the UDG model. But in reality, the quality of signal strength decreases with the increase in distance to the receiving node.The constraint edges may have poor link quality. In addition to the distance, link quality also depends on surroundings such as walls, buildings, mountains, and weather conditions. So, one can consider the link quality also as a parameter along with the distance to choose constraint edges. It would be interesting to study these constrained spanners by adding constraint edges based on link quality.

The Algorithm *Constraint edge2* used to find constraint edges are not optimal. It would be interesting to find an algorithm to add more number of constraint edges using minimum number of neighbor information to further reduce various networks overhead.

The k-node fault tolerant spanners (k-NFTS) and k-link fault tolerant spanner (k-LFTS) are studied in the literature and proposed efficient centralized algorithms.It would be interesting to design efficient localized distributed algorithm for k-NFTS and k-LFTS and test them for suitable adhoc networks.

The methods to maintain PLDel, RNG, and GG are simulated under random way point mobility model. It would be interesting to study the mobile spanners under different mobility models, like Gauss-Markov mobility model and random direction mobility. Another interesting future direction is to use various node position prediction methods to maintain geometric spanners efficiently to further improve various network parameters.

In this thesis, we have studied PLDel, RNG, and GG under various network requirements. It would be interesting to study other spanners like PDT, RDG, CDG, and Yao under these network requirements.

Spanners in three-dimension is an unexplored area. One can extend various spanners available in literature to the higher dimension.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Bibliography

[ADM+95] S. Arya, G. Das, D. Mount, J. Salowe, and M. Smid. Euclidean spanners: Short, thin, and lanky. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 489–498. ACM Press, 1995.

[ALW+03] K. M. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, vol. 14(4):pages 408–421, April 2003.

[Alz02] K. M. Alzoubi. *Virtual backbones in wireless ad hoc networks*. PhD thesis, Chicago, IL, USA, 2002. Adviser-Peng-Jun Wan.

[AN06] L. M. Arboleda and N. Nasser. Cluster-based routing protocol for mobile sensor networks. In *QShine '06: Proceedings of the 3rd International conference on Quality of service in heterogeneous wired/wireless networks*, page 24, New York, USA, 2006. ACM Press.

[AWD04] M. Abolhasan, T. A. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, vol. 2(1):pages 1–22, 2004.

[AWF02] K. M. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM International symposium on Mobile ad hoc networking & computing*, pages 157–164, New York, USA, 2002. ACM Press.

[BCSW98] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (dream). In *MOBICOM*, pages 76–84, 1998.

[BDEK02] P. Bose, L. Devroye, W. S. Evans, and D. G. Kirkpatrick. On the spanning ratio of gabriel graphs and beta-skeletons. In *Latin American Theoretical INformatics*, pages 479–493, 2002.

[BJ02] S. A. Borbash and E. H. Jennings. Distributed topology control algorithm for multihop wireless networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2002)*, volume 1, pages 355–360, 2002.

[BM99] P. Bose and P. Morin. Online routing in triangulations. In *Proceedings of the 10th International Symposium on Algorithms and Computation*, pages 113–122, London, UK, 1999. Springer-Verlag.

[BMJ+98] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Mobile Computing and Networking*, pages 85–97, 1998.

[BMSU01] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, vol. 7(6):pages 609–616, 2001.

[BR03] D. Bertsimas and G. J. V. Ryzin. An asymptotic determination of the minimum spanning tree and minimum matching constants in geometrical probability. Working papers 3058-89, Massachusetts Institute of Technology (MIT), Sloan School of Management, April 2003.

[BSC02] A. Boukerche, V. Sheetal, and M. Choe. A route discovery optimization scheme using gps system. *Proceedings of 35th Annual Simulation Symposium*, pages 20–26, April 2002.

[BVGLA99] D. A. Beyer, M. D. Vestrich, and J. J. Garcia-Luna-Aceves. The rooftop community network: free, high-speed network access for communities. pages 75–91, 1999.

[Caf99] J. J. Caffery. *Wireless Location in CDMA Cellular Radio Systems*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[CBD02] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2(5):pages 483–502, 2002.

[CE95] M. S. Corson and A. Ephremides. A distributed routing algorithm for mobile wireless networks. *Journal of Wireless Networks*, vol. 1(1):pages 61–81, 1995.

[CG98] T.-W. Chen and M. Gerla. Global state routing: a new routing scheme for ad-hoc wireless networks. *IEEE International Conference on Communications*, vol. 1:pages 171–175, Jun 1998.

[Che87] L. P. Chew. Constrained delaunay triangulations. In *Proceedings of the third annual symposium on Computational geometry*, pages 215–222, New York, USA, 1987. ACM Press.

[CHH01] S. Capkun, M. Hamdi, and J.-P. Hubaux. Gps-free positioning in mobile ad-hoc networks. *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, page 10, Jan 2001.

[CM99] S. Corson and J. Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. Request for Comments 2501, IETF, January 1999.

[CR90] A. K. Cline and R. J. Renka. A constrained two-dimensional triangulation and the solution of closest node problems in the presence of barriers. *SIAM J. Numerical Analysis*, vol. 27:pages 1305–1321, 1990.

[CSJS08] M. Chawla, J. Singhai, S. Jain, and A. Srivatsava. Node stability based clustering algorith for mobile ad hoc networks. In *Proceedings of the Fourth International conference on Wireless Communication and Sensor Networks, WCSN-2008*, pages 31–35, 2008.

[CSRL01] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[CW99] F. Chin and C. A. Wang. Finding the constrained delaunay triangulation and constrained voronoi diagram of a simple polygon in linear time. *SIAM J. Computing*, vol. 28(2):pages 471–486, 1999.

[CWPE05] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *MobiHoc '05: Proceedings of the 6th ACM International symposium on Mobile ad hoc networking and computing*, pages 414–425, New York, USA, 2005. ACM Press.

[CZ03] A. Czumaj and H. Zhao. Fault-tolerant geometric spanners. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 1–10, New York,USA, 2003. ACM Press.

[DFS90] D. B. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete and Computational Geometry*, vol. 5(4):pages 399–407, 1990.

[DJS04] R. C. Doss, A. Jennings, and N. Shenoy. A review of current mobility prediction techniques for ad hoc networks. *WNET 2004: Proceedings of the Wireless Networks and Emerging Technologies - 2004*, pages 536–542, July 2004.

[DRWT97] R. Dube, C. D. Rais, K.-Y. Wang, and S. K. Tripathi. Signal stability-based adaptive routing (ssa) for ad hoc mobile networks. *IEEE Personal Communicationsi*, vol. 4(1):pages 36–45, Feb 1997.

[EFK07] J. Eriksson, M. Faloutsos, and S. V. Krishnamurthy. Dart: dynamic address routing for scalable ad hoc and mesh networks. *IEEE/ACM Transactions on Networking*, vol. 15(1):pages 119–132, 2007.

[Epp96] D. Eppstein. Beta-skeletons have unbounded dilation. Technical Report ICS-TR-96-15, 1996.

[EPY97] D. Eppstein, M. Paterson, and F. F. Yao. On nearest neighbor graphs. *GEOMETRY: Discrete and Computational Geometry*, vol. 17:pages 263–282, 1997.

[GGH+01] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanner for routing in mobile networks. In *MobiHoc '01: Proceedings of the 2nd ACM International symposium on Mobile ad hoc networking & computing*, pages 45–55, New York, USA, 2001. ACM Press.

[GGH+05] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanners for routing in mobile networks. *IEEE Journal on Selected Areas in Communications*, vol. 23(1):pages 174–185, Jan. 2005.

[GLAS99] J. J. Garcia-Luna-Aceves and M. Spohn. Source-tree routing in wireless networks. *Proceedings of the Seventh International Conference on Network Protocols, (ICNP '99)*, pages 273–282, Oct.-3 Nov. 1999.

[GYS05] S. Guo, O. Yang, and Y. Shu. Improving source routing reliability in mobile ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, vol. 16(4):pages 362–373, April 2005.

[HHCW05] E. Huang, W. Hu, J. Crowcroft, and I. Wassell. Towards commercial mobile ad hoc network applications: a radio dispatch system. In *MobiHoc '05: Proceedings of the 6th ACM International symposium on Mobile ad hoc networking and computing*, pages 355–365, New York, USA, 2005. ACM Press.

[HL86] T.-C. Hou and V. Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, vol. 34(1):pages 38–44, Jan 1986.

[HPS02] Z. J. Haas, M. R. Pearlman, and P. Samar. The zone routing protocol (zrp) for ad hoc networks. Internet-draft, IETF MANET Working Group, July 2002. Expiration: January, 2003.

[ID03] M. Ilyas and R. C. Dorf, editors. *The handbook of ad hoc wireless networks*. CRC Press, Inc., Boca Raton, FL, USA, 2003.

[JM96] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[JNL99] M. Jao-Ng and I-T. Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, vol. 17(8):pages 1415–1425, Aug 1999.

[JPS01] R. Jain, A. Puri, and R. Sengupta. Geographical routing using partial information for wireless ad hoc networks. *IEEE Personal Communications*, vol. 8(1):pages 48–57, Feb 2001.

[KFWM04] W. Kiess, H. Fussler, J. Widmer, and M. Mauve. Hierarchical location service for mobile ad-hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8(4):pages 47–58, 2004.

[KG89] J. M. Keil and C. A. Gutwin. The delauney triangulation closely approximates the complete euclidean graph. In *Proceedings of the Workshop on Algorithms and Data Structures*, pages 47–56, London, UK, 1989. Springer-Verlag.

[KG92] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete Computational Geometry*, vol. 7(1):pages 13–28, 1992.

[KH06] E. Kaplan and C. J. Hegarty. *Understanding GPS: Principles and Applications*. Artech House, Norwood, 2006.

[KK00] B. Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual International conference on Mobile computing and networking*, pages 243–254, New York, USA, 2000. ACM Press.

[KN86] J. Katajainen and O. Nevalainen. Computing relative neighbourhood graphs in the plane. *Pattern Recognition*, vol. 19(3):pages 221–228, 1986.

[KN97] Y.-B. Ko and V. Nitin. Using location information to improve routing in ad hoc networks. Technical Report TR97-013, 11, 1997.

[KSU99] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. 11 th Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, August 1999.

[KV98] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 66–75, 1998.

[KV99] Y.-B. Ko and N. H. Vaidya. Geocasting in mobile ad hoc networks: location-based multicast algorithms. *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications, 1999. WMCSA '99*, pages 101–110, Feb 1999.

[KVCP97] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. *SIGCOMM Computer Communication Review*, vol. 27(2):pages 49–64, 1997.

[KWZ02] F. Kuhn, R. Wattenhofer, and A. Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *DIALM '02: Proceedings of the 6th International workshop on Discrete algorithms and methods for mobile computing and communications*, pages 24–33, New York, USA, 2002. ACM Press.

[KWZZ03] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: of theory and practice. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 63–72, New York, USA, 2003. ACM Press.

[KZ03] F. Kuhn and A. Zollinger. Ad-hoc networks beyond unit disk graphs. In *DIALM-POMC '03: Proceedings of the 2003 joint workshop on Foundations of mobile computing*, pages 69–78, New York, USA, 2003. ACM Press.

[LC05] L. Layuan and L. Chunlin. A qos multicast routing protocol for mobile ad-hoc networks. *International Conference on Information Technology: Coding and Computing, 2005. ITCC 2005*, vol. 2:pages 609–614, April 2005.

[LCN05] X. Luo, T. Camp, and W. Navidi. Predictive methods for location services in mobile ad hoc networks. *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005*, page 6, April 2005.

[LCW02] X.-Y. Li, G. Calinescu, and P.-J. Wan. Distributed construction of a planar spanner and routing for ad hoc wireless networks. *Proceedings of the Twenty-First IEEE Annual Joint Conference on Computer and Communications Societies. INFOCOM 2002*, vol. 3:pages 1268–1277, 2002.

[LCWW03] X.-Y. Li, G. Calinescu, P.-J. Wan, and Y. Wang. Localized delaunay triangulation with application in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, vol. 14(10):pages 1035–1047, Oct. 2003.

[Lee97] F. Lee. Constructing the constrained delaunay triangulation on the intel paragon. In *SAC '97: Proceedings of the ACM symposium on Applied computing*, pages 464–467, New York, USA, 1997. ACM Press.

[LH99] B. Liang and Z. J. Haas. Predictive distance-based mobility management for pcs networks. In *INFOCOM*, pages 1377–1384, 1999.

[Liu09] Y. Liu. Advanced dynamic source routing with qos guarantee. In *Proceedings of the Second Symposium International Computer Science and Computational Technology(ISCSCT '09)*, pages 504–506, 2009.

[LJD+00] J. Li, J. Jannotti, D. S. J. DeCouto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *MobiCom '00: Proceedings of the 6th annual International conference on Mobile computing and networking*, pages 120–130, New York, USA, 2000. ACM Press.

[LNS02] C. Levcopoulos, G. Narasimhan, and M. Smid. Improved algorithms for constructing fault-tolerant spanners. *Algorithmica*, vol. 32, 2002.

[LSG02] S. J. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *Mobility Network Applications*, vol. 7(6):pages 441–453, 2002.

[LSW04] X.-Y. Li, I. Stojmenovic, and Y. Wang. Partial delaunay triangulation and degree limited localized bluetooth scatternet formation. *IEEE Transactions on Parallel and Distributed Systems*, vol. 15(4):pages 350–361, April 2004.

[Luk99] T. Lukovszki. New results of fault tolerant geometric spanners. In *WADS '99: Proceedings of the 6th International Workshop on Algorithms and Data Structures*, pages 193–204, London, UK, 1999. Springer-Verlag.

[LWW01] X. Y. Li, P. J. Wan, and Y. Wang. Power efficient and sparse spanner for wireless ad hoc networks. *Proceedings of the Tenth International Conference on Computer Communications and Networks*, pages 564–567, 2001.

[LWWF02] X.-Y. Li, P.-J. Wan, Y. Wang, and O. Frieder. Sparse power efficient topology for wireless networks. *HICSS. Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 3839–3848, Jan. 2002.

[MGLA96] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Network Applications Journal, A special issue on Routing in wireless communication networks*, vol. 1(2):pages 183–197, 1996.

[MWH01] M. Mauve, A. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE Network*, vol. 15(6):pages 30–39, Nov/Dec 2001.

[MXP05] L. Ming, L. XiCheng, and W. Peng. *Chapter "Dynamic Delaunay Triangulation for Wireless Ad Hoc Network"*, volume 3756, pages 283–289. LNCS, SpringerLink, 2005.

[NS205] The Network Simulator ns-2.28. http://www.isi.edu/nsnam/ns/, October 2005.

[PB94] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 234–244, New York, USA, 1994. ACM Press.

[PC97] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *INFOCOM '97. Proceedings of the Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3:pages 1405–1413, Apr 1997.

[Pet] Samuel Peterson. Computing constrained delaunay triangulations in the plane, university of minnesota undergraduate.

[PGHC99] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang. A wireless hierarchical routing protocol with group mobility. *Wireless Communications and Networking Conference, 1999. WCNC. 1999. IEEE*, vol. 3:pages 1538–1542, 1999.

[PL00] W. Peng and X.-C. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. *First Annual Workshop on Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000*, pages 129–130, 2000.

[PL02] W. Peng and X. Lu. Ahbp: An efficient broadcast protocol for mobile ad hoc networks. *Journal of Sceince and Technology*, 2002.

[Poo94] H. V. Poor. *An introduction to signal detection and estimation (2nd ed.)*. Springer-Verlag New York, USA, 1994.

[PPM03] J. Punde, N. Pissinou, and K Makki. On quality of service routing in ad-hoc networks. In *Proceedings of the 28th Annual IEEE International conference on Local Computer Networks, 2003, LCN-03*, pages 276–278, 2003.

[PR99] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. *Second IEEE workshop on Mobile Computing Systems and Applications, WMCSA '99*, pages 90–100, Feb 1999.

[PS90] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 3rd edition, October 1990.

[QHM+06] B. Qazi, Y. Harikrishnan, R. Mehmood, J.M.H. Elmirghani, and H. Dass. A mobile diverse heterogeneous ad hoc network for city management. *Proc. Intelligent Transport Systems and Services ITS'06*, pages 8–12, October 2006.

[QKS06] Y. Qi, H. Kobayashii, and H. Suda. Analysis of wireless geolocation in a non-line-of-sight environment. *IEEE Transactions on Wireless Communications*, vol. 5(3):pages 672–681, 2006.

[Rap01] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[RE04] P. L. Royds and J. M. H. Elmirghani. Delay characteristics of diverse ad hoc networks. *Electronics Letters*, vol. 40(7):pages 439–440, April 2004.

[RMSM01] E. Royer, P. M. Melliar-Smith, and L. E. Moser. An analysis of the optimum node density for ad hoc mobile networks. *IEEE International Conference on Communications, 2001. ICC 2001*, vol. 3:pages 857–861, 2001.

[RT99] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, vol. 6(2):pages 46–55, Apr 1999.

[She00] J. R. Shewchuk. Sweep algorithms for constructing higher-dimensional constrained delaunay triangulations. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 350–359, New York,USA, 2000. ACM Press.

[SL01] I. Stojmenovic and X. Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, vol. 12(10):pages 1023–1032, Oct 2001.

[SLJ08] I. Stojmenovic, D. Liu, and X. Jia. A scalable quorum based location service in ad hoc and sensor networks. *Int. J. Commun. Netw. Distrib. Syst.*, vol. 1(1):pages 71–94, 2008.

[SM00] J. Sucec and I. Marsic. An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks, 2000.

[SSB99] P. Sinha, R. Sivakumar, and V. Bharghavan. Cedar: Core extraction distributed ad hoc routing. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '99, New York, USA*, pages 202–209, March 1999.

[Sto02] I. Stojmenovic. Position-based routing in ad hoc networks. *IEEE Communications Magazine*, vol. 40(7):pages 128–134, Jul 2002.

[Sup83] K. J. Supowit. The relative neighborhood graph, with an application to minimum spanning trees. *J. ACM*, vol. 30(3):pages 428–448, 1983.

[TK84] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, vol. 32(3):pages 246–257, Mar 1984.

[TLW02] Y.-C. Tseng, W.-H. Liao, and S.-L. Wu. Mobile ad hoc networks and routing protocols. pages 371–392, 2002.

[Toh97] C.-K. Toh. Associativity-based routing for ad hoc mobile networks. *Wireless Personal Communications*, vol. 4(2):pages 103–139, 1997.

[UG09] S. Upadhayaya and C. Gandhi. Qos routing using link and node stability in mobile ad hoc networks. *Journal of Theoritical and Applied Information Technology*, vol. 8(2):pages 117–122, October 2009.

[WL02] Y. Wang and X.-Y. Li. Geometric spanners for wireless ad hoc networks. *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 171–178, 2002.

[WLBW01] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. *Proceedings of the twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2001*, vol. 3:pages 1388–1397, 2001.

[WT99] C. W. Wu and Y. C. Tay. Amris: a multicast protocol for ad hoc wireless networks. *Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE*, vol. 1:pages 25–29, 1999.

[XC06] P. Xue and S. Chandra. Revisiting multimedia streaming in mobile ad hoc networks. In *NOSSDAV '06: Proceedings of the 2006 International workshop on Network and operating systems support for digital audio and video*, pages 1–7, New York, USA, 2006. ACM Press.

[XTML02] J. Xie, R. R. Talpade, A. Mcauley, and M. Liu. Amroute: ad hoc multicast routing protocol. *Mob. Netw. Appl.*, vol. 7(6):pages 429–439, 2002.

[YZD09] Y. Yu, Q. Zhang, and S. Du. Mobility-aware multicast over mobile ad hoc networks. In *2009 ISECS International Colloquim on Computing, Communication, Control, and Management*, volume 2, pages 119–122, 2009.

[ZK07] M. Z. Zamalloa and B. Krishnamachari. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Trans. Sen. Netw.*, vol. 3(2):pages 7, 2007.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Publications based on the Thesis

1. D. Satyanarayana and S. V. Rao, Constrained Delaunay Triangulation for Ad Hoc Networks, Journal of Computer Systems, Networks, and Communications, Article ID 160453, 10 pages, 2008. doi:10.1155/2008/160453.

2. D. Satyanarayana, S. V. Rao, Constraint based Geometric graphs for ad hoc networks submitted to *Journal of Adhoc Networks, Elsevier*.

3. D. Satyanarayana, S. V. Rao, Fault Tolerant Spanners for Ad hoc Sensor Networks, In *Proceedings of the 7th Asia Pacific Symposium on Information and Telecommunication Technologies (APSITT-2008)*, Pages 235-240, Maldives, April, 2008 (IEEE Press).

4. D. Satyanarayana, S. V. Rao, Fault Tolerant Local Delaunay Trinagulation for Ad hoc Sensor Networks, In *Proceedings of the Third International Conference on Wireless Communication and Sensor Networks(WCSN-07)*, Pages 36-40, Allahabad, December, 2007 (IEEE Press).

5. D. Satyanarayana, S. V. Rao, Local Delaunay triangulation for mobile nodes, In *Proceedings of the International Conference on Emerging Trends in Engineering and Technology (ICETET-2008)*, Pages 282-287, Nagpur, July, 2008 (IEEE Press).

6. D. Satyanarayana, S. V. Rao, A Spanner for Multimedia Applications in Ad hoc Networks, In *Proceedings of the IET International Conference on Wireless, Mobile and Multimedia Networks*, Pages 231-234, Mumbai, 2008.

7. D. Satyanarayana, S. V. Rao, Dynamic Spanners for Ad hoc Sensor Networks, In *Proceedings of the International Conference on Sensors and Related Networks (SENNET-07)*, Pages 415-420, Vellore, 2007.