Quality of Service Issues in Mobile Ad Hoc Networks

Thesis submitted

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

by

Nityananda Sarma

Under the guidance of

Prof. Sukumar Nandi

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Guwahati - 781 039, INDIA

January 2009

Certificate

This is to certify that the thesis entitled **"Quality of Service Issues in Mobile Ad Hoc Networks"**, submitted by **Nityananda Sarma**, a research student 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 and in my opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Date: 23rd Jan 2009
Guwahati

Sukumar Nandi,
Professor,
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati- 781 039, India.

Acknowledgement

This thesis is the result of almost four years of hard work whereby I have been accompanied and supported by many people. It is a pleasant moment that I have now the opportunity to express my gratitude for all of them.

First of all, I am deeply indebted to my supervisor Prof. Sukumar Nandi, for being a most supportive supervisor, whose help, stimulating suggestions and encouragement helped me all the time during this period of research. His in-depth knowledge on the area of my research and his logical way of thinking have been of great value for me. His understanding, encouraging and personal guidance have provided a good basis for completion of my Ph.D thesis.

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I am grateful to Prof. D. K. Bhattacharyya and Prof. D. K. Saikia, current and previous Head of Department of Computer Science and Engineering, Tezpur University, who have supported me for getting leaves to complete my Ph.D thesis.

My thanks and appreciation goes to members of doctoral committee, Dr. A. Mitra, Dr. P. Bhaduri and Dr. S. V. Rao for their valuable comments. I would like to thank Dr. G. Sajith, the Head of the Department of Computer Science and Engineering, and other faculty members for their support and help.

I am very grateful to my wife Manu and my daughter Jisha, for their love and patience during my research. Without their encouragement and understanding it would have been impossible for me to finish this work. I feel a deep sense of gratitude for my late father and mother whom I lost in my early part of life.

Finally, I must acknowledge

- my sisters, brother and their families for their loving support
- my colleague at Tezpur University, Dr. U. Sharma, Dr. S. M. Hazarika and Mr. S. Saharia for their continuous encouragement and support
- my friends at IIT Guwahati - Arup, Suresh, Mausumi, Tejmani, Rajendra, Bidyut, Satyanarayan, Neminath, Sraban, Godfrey, Guru Prasad, Amrita, Lipika, Alka, Mamata and many more for their friendship and encouragement
- the laboratory staff and office staff for assisting me with different resources at various times whenever required

Nityananda Sarma

Abstract

Multihop wireless network has emerged recently as an evolution of wireless network technology and are likely to be the integral part of future communication environment. In this context, support for Quality of Service (QoS) is becoming an inherent necessity rather than an additional feature of the network. Due to lack of a centralized control, highly dynamic nature of topology and existence of variable and limited shared resources, traditional approaches for supporting QoS in the Internet can not be applicable to mobile ad hoc networks (MANETs). Given that quality of service provisioning in MANETs is extremely challenging and is a multi-layer problem, this thesis takes a holistic view to this QoS issue by identifying the key components of an overall MANET QoS framework. We take minimum throughput and maximum delay as the applications' QoS requirements to be supported by the QoS mechanisms developed in this dissertation.

We begin by developing two dynamic priority based QoS-aware MAC protocols which are based on legacy IEEE 802.11 DCF and hence can easily be integrated into existing systems without much difficulty. Our first scheme, called Priority based QoS-aware MAC protocol (PQAMP) is designed for achieving enhanced level of service differentiation to provide QoS for real-time traffic along with maximizing network utilization in a multihop environment. By assigning highest priority to relay traffic, PQAMP reduces packet's end-to-end delay for real-time traffic and at the same time improves network utilization. Further, it supports a minimum level of service for best-effort traffic. The second scheme called Strict Priority based QoS-aware MAC Protocol (SPQAMP) is designed to achieve a strict service differentiation to support QoS for real-time traffic without any service guarantee for best-effort traffic and with network utilization as compatible to IEEE 802.11 DCF. SPQAMP uses a resetting of backoff timer mechanism for best-effort traffic as an extra measure to provide strict prioritized channel access for real-time traffic. Due to strict service differentiation in SPQAMP, the amount of best-effort traffic present in the network can not affect the QoS provided to real-time traffic. Using both these schemes substantially improve the QoS support at MAC layer of MANETs.

Our investigation on existing QoS routing algorithms show that most of them choose the shortest path among the feasible paths and none of them take any measure to select QoS routes that endure longer time. In this direction, we develop a route stability model (RSM) to estimate stability of link and end-to-end path in MANETs, which has less control overhead, requires no extra hardware and has the potential to be applicable in most of the real-life mobility scenarios. By incorporating RSM through some route stability related fields in route request/reply messages, we propose a Route Stability based QoS Routing (RSQR) for MANETs. RSQR incorporates a soft resource reservation, a hop by hop admission control and applies route stability metric to improve the durability of established QoS route such that probability of the resource reservation to remain valid for duration of

data transmission is high. RSQR reduces QoS disruption due to expensive route recoveries and gives higher performance in terms of packet delivery ratio, average end-to-end delay and control overhead.

To further reduce the QoS disruption due to frequent and expensive route recoveries, we propose a Stability based Multipath QoS Routing (SMQR) protocol for MANETs. Reliability of multiple QoS-aware routes is improved through node-disjointness and stability properties of the discovered routes. Detection of potential route failures and switching to an available stable route before actual route breaks saves route recovery time as well as reduces possibility of packet loss during active communication using SMQR. Significant QoS performance improvements are observed in terms of average end-to-end delay, packet delivery ratio and maximum delay jitter.

Finally, we investigate QoS provisioning from multilayer perspective. In this direction, we propose a cross-layer QoS framework which maps QoS parameters across three layers of the protocol stack to support a stronger notion of per-class service guarantees in ad hoc networks. QoS mapping across protocol layers are realized through adaptation of relevant protocols based on both QoS parameters received from the upper layer and prevailing network condition. Per-class differentiated service is obtained through class-based scheduling in network layer and allocation of non-overlapping contention window at MAC layer. Further, MAC layer and network layer are made to adapt their behaviour based on both dynamic network conditions and QoS requirements of the admitted flows. Accordingly, MAC layer adaptation is carried out in the form of dynamic contention window adjustment per class, whereas network layer adaptation takes place in the form of rerouting or termination of existing flows.

Keywords: Mobile Ad Hoc Networks (MANETs), Quality of Service (QoS), QoS-aware Medium Access Control, Route stability, QoS routing, Multipath QoS Routing, Cross-layer QoS framework

Contents

List of Figures

List of Tables

List of Algorithms

Glossary of Terms

ABR	Associativity Based Routing
AC	Access Category
ACK	Acknowledgment
ADELAY	Accumulated DELAY
ADQR	Adaptive Dispersity QoS Routing
AF	Assured Forwarding
AIFS	Arbitrary Inter Frame Space
AIMD	Additive Increase Multiplicative Decrease
AODV	Ad hoc On-demand Distance Vector
AODV-BR	AODV Backup Routing
AODVM	Ad hoc On-demand Distance Vector Multipath
AOMDV	Ad Hoc On-demand Multipath Distance Vector
AP	Access Point
APS	Accumulated Path Stability
APU	Accumulated Path Uncertainty
AQOR	Ad hoc QoS On-demand Routing
ARF	Auto Rate Fallback
ASBM	Advanced Signal Strength Based link stability Model
BB	Black Burst
BK	Background traffic
CACP	Contention-aware Admission Control Protocol
CBR	Constant Bit Rate
CBWFQ	Class Based Weighted Fair Queuing
CC	Contention Count
CDMA	Code Division Multiple Access
CE	Congestion Enabled
CEDAR	Core Extraction Distributed Ad hoc Routing
CFP	Contention Free Period
CL	Controlled Load
CLQM	Cross-layer QoS framework for MANETs
CP	Contention Period

CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CSR	Carrier Sensing Range
CTS	Clear-To-Send
CW	Contention Window
CWT	Contention Window Table
DARPA	Defense Advanced Research Projects Agency
DB-DFS	Deadline Bursting Distributed Coordination Function
DCF	Distributed Coordination Function
DCF-PC	DCF with Priority Class
DDCF	Differentiated Distributed Coordination Function
DFS	Distributed Fair Scheduling
DiffServ	Differentiated Services
DIFS	Distributed Coordination Function IFS
DP	Dropped Packets
dRSVP	Dynamic Resource Reservation Protocol
DSCP	Differentiated Service Code Point
DSR	Dynamic Source Routing
DSS	Differentiated Signal Strength
DSSS	Direct Sequence Spread Spectrum
E2ED	End-to-End Delay
EBLLD	Entropy-based Long-Life Distributed QoS Routing
ECN	Explicit Congestion Notification
EDCA	Enhanced Distributed Channel Access
EDCF	Enhanced DCF
EF	Expedited Forwarding
ES-DFS	Elimination by Sieving Distributed Coordination Function
ETSI	European Telecommunications Standards Institute
EWMA	Exponentially Weighted Moving Average
FAR	Full Auto Rate
FDMA	Frequency Division Multiple Access
FID	First Hop ID
FQMM	Flexible Quality of service Model for MANETs
FT	Flow Table
FTP	File Transfer Protocol
GPS	Global Positioning System
GS	Guaranteed Service

HCF	Hybrid Coordination Function
HELLO	Hello Packet
HIPERLAN	High PERformance Local Area Network
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IFS	Inter Frame Space
IMRP	Interference-aware Multipath Routing Protocol
INORA	INSIGNIA + TORA
INSIGNIA	IN-band SIGNaling In Ad hoc network
IntServ	Integrated Services
IP	Internet Protocol
ITU-T	Telecommunication standardization unit of International Telecommunication Union
LBR	Link Life Based Routing
LID	Last Hop ID
LS	Link Stability
LU	Link Uncertainty
MAC	Medium Access Control
MACA	Multiple Access with Collision Avoidance
MACA/PR	Multiple Access Collision Avoidance with Piggyback Reservation
MANET	Mobile Ad Hoc Network
MJ	Maximum Jitter
MP-AOMDV	Mobility Prediction based AOMDV
MP-DSR	Multipath Dynamic Source Routing
MQRD	Multipath QoS Routing protocol of supporting DiffServ
MRSVP	Mobile Resource Reservation Protocol
MSS	Minimum Signal Strength
NAV	Network Allocation Vector
NCO	Network Control Overhead
NDMR	Node-Disjoint Multipath Routing
NIT	Neighbor Information Table
NOAH	NO Ad Hoc routing
NS-2	Network Simulator Version 2
OAR	Opportunistic Auto Rate
P-DCF	Persistent factor DCF
PAC	Perceptive Admission Control
PANDA	Positional Attributes based Next hop Determination Algorithm
PC	Point Coordinator

PCF	Point Coordination Function
PD	Packet Dropped
PDR	Packet Delivery Ratio
PHB	Per-Hop Behaviour
PHY	Physical Layer
PIFS	Point Coordination Function IFS
PLCP	Physical Layer Convergence Protocol
PPTT	Path Predicted Transmission Time
PQAMP	Priority based QoS-Aware MAC Protocol
PRNet	Packet Radio Network
PS	Path Stability
PU	Path Uncertainty
QAODV	QoS-aware AODV
QASP	QoS Aware Stable Path
QDMR	QoS-aware Disjoint Multipath Routing protocol
QoS	Quality of Service
QPART	QoS Protocol for Ad hoc Realtime Traffic
QRREP	Quality of Service RREP
QRREQ	Quality of Service RREQ
RBAR	Receiver Based Auto Rate
RDP	Route Discovery Packet
RED	Random Early Detection
RERR	Route ERRor
RFC	Route Fragility Coefficient
RFT	Route Request Forwarded Table
RID	Receiver Interference Distance
RLT	Route List Table
RouteM	Route Maintenance
RREP	Route REPly
RREQ	Route REQuest
RRL	Route Reply Latency
RSM	Route Stability Model
RSQR	Route Stability based QoS Routing
RSVP	Resource Reservation Protocol
RT	Routing Table
RT-MAC	Real-Time Medium Access Control
RTS	Request-To-Send
RUPD	Route UPDate
RxR	Reception Range
SCFQ	Self-Clocked Fair Queuing

SD	Saturation Delay
SHARC	Stability and Hop-count based Algorithm for Route Computation
SID	Sender Interference Distance
SIFS	Short Inter Frame Space
SLA	Service Level Agreement
SMQR	Route Stability based Multipath QoS Routing
SNR	Signal-to-Noise Ratio
SPQAMP	Strict Priority based QoS-Aware MAC Protocol
SS	Signal Strength
SSA	Signal Stability based Adaptive routing
STP	Saturation Throughput
SURAN	Survivable Adaptive Radio Network
SWAN	Service differentiation in Wireless Ad hoc Networks
TAR	Traffic Admission Ratio
TBP	Ticket Based Probing
TC	Traffic Category
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TORA	Temporally Ordered Routing Algorithm
TOS	Type of Service
TP	Throughput
TTL	Time to Live
TXOP	Transmission Opportunity
TxR	Transmission Range
UDP	User Datagram Protocol
VI	Video
VO	Voice
VoIP	Voice over IP
WFQ	Weighted Fair Queuing

Chapter 1

Introduction

The Mobile Ad Hoc Network has its origin as the packet switched radio project of DARPA, which was initiated in the early 70's. Since that time, the complexity of mobile ad hoc network designs has challenged generations of researchers. With the development of novel and inexpensive wireless technologies such as IEEE 802.11, research in this field has gained more and more attention. Multihop Wireless network has emerged recently as an evolution of wireless network technology having the potential to deliver real-time services such as video, audio and Voice over Internet Protocol (VoIP) besides traditional data service. This type of networks can provide a competitive alternative to cellular networks, especially in areas where the latter are not available. Mobile Ad Hoc Networks consist of wireless mobile hosts that communicate with each other without the presence of a fixed infrastructure, forming dynamically self-organized and self-configured autonomous networks. Due to its non-dependency on a centralized infrastructure, it can be rapidly deployed and can dynamically be reconfigured over time [37].

MANETs have many potential applications in a variety of fields like - military tactical communication, disaster rescue and recovery, and collaborative group meetings. With the increase in Quality of Service (QoS) needs in evolving applications and the wide spread use of wireless and mobile devices, it is also desirable to support QoS in MANETs. The resource limitations and variability further add to the need for QoS support in such networks. The dynamic nature of MANETs is attributed to the time-varying channel condition, node movements, changing network topology, and variations in application's demands. Providing hard QoS (in terms of guaranteed services) in such dynamic environment is not an easy task. Therefore, almost all technologies to provide QoS for MANETs concentrate on Soft QoS. It means after the connection setup, there may exist transient periods of time when the assured QoS specification is not honored. However, even targeting for a soft QoS, QoS support remains as a very complex process [15, 73, 112, 121].

Quality of Service (QoS) is usually defined as the set of service requirements that need to be met by the network while transporting a packet stream between a given source-destination pair [24]. The network is expected to guarantee a set of measurable pre-defined end-to-end service attributes to the users. The common end-to-end service attributes are - delay, bandwidth, probability of packet loss, delay variance (jitter), etc. Power consumption and service coverage area are two other QoS attributes that are more specific to MANETs. Furthermore, the QoS metrics could be defined in terms of a single parameter or a set of parameters in varied proportions. The multi-constraints QoS support aims at optimizing multiple QoS metrics while provisioning network resources, and is an admittedly complex problem.

Although a lot of work has been done in supporting QoS in the Internet, but none of them can directly be extended to work in MANETs [15, 73, 112]. Supporting QoS in a network requires the measurements of link state information such as delay, bandwidth, cost, loss rate, and error rate in the network. However, getting and managing link state information in MANETs are very difficult, because the quality of a wireless link is apt to change with the surrounding circumstances. Furthermore, the resource limitations and the mobility of hosts make things more complicated.

1.1 Issues and difficulties for supporting QoS in MANETs

Some of the issues and difficulties for supporting QoS in MANETs [73, 89, 121] are:

- *Unreliable and Unpredictable Link Properties:* Wireless media is highly unreliable and its capacity may vary dramatically. Signal propagation faces difficulties such as signal fading, interference, and multi-path cancellation. Therefore, QoS-aware protocols should not rely on exact knowledge of channel capacity.

- *Contention on wireless medium:* Due to shared nature of the wireless medium, transmission from a node not only uses local available bandwidth but also consumes bandwidth of its contending neighbors. Therefore, resource allocation should also consider impact on neighboring nodes. Further, intra-flow contention impacts the available bandwidth of a node.

- *Node Mobility:* Mobility of the nodes creates a dynamic network topology. Links are dynamically formed when two nodes come into transmission range of each other and are torn down when they move out of range.

- *Limited Battery Life:* Mobile devices are generally run on finite battery resources. Thus, the techniques for QoS provisioning should be power-aware and power-efficient.

- *Hidden and Exposed Terminal Problem:* Traditional carrier sense multiple access (CSMA) protocol when applied in multi-hop packet relay introduces the hidden and exposed terminal problems. Due to Hidden terminal problem, some nodes in the network will have smaller contention probability than others, thus introduces local specific contention. Exposed terminal problem blocks the spatial reuse of channel capacity. Both these problems should be handled to improve the overall performance.

- *Route Maintenance:* The dynamic nature of the network topology and changing behaviour of the communication medium make the precise maintenance of network state information very difficult.

- *No centralized control:* Due to lack of a centralized control, QoS-aware protocols must use distributed algorithm and rely only on local information.

- *Security:* Security can be considered as an important QoS attribute. Without adequate security, unauthorized access and usage may violate the QoS negotiations.

1.2 Motivation

MANETs are likely to be the integral part of future communication environment, and support for QoS is becoming an inherent necessity rather than an additional feature of the network. The entire issues and difficulties as we have discussed above make QoS provisioning in MANETs an extremely challenging task which needs effective/efficient solutions at multiple layers of the protocol stack. Given that quality of service (QoS) provisioning in MANETs is extremely challenging and is a multi-layer problem, this thesis work takes a holistic view to this issue by identifying the required components of an overall MANET QoS framework. By virtue of their importance, we take minimum throughput and maximum delay as the applications' QoS requirements to be supported by the QoS mechanisms developed in this dissertation.

1.3 Aim of the Thesis

The broad aim of this dissertation is to consider issues of Quality of Service in mobile ad hoc networks and to devise techniques that can support an adequate level of QoS for real-time traffic in MANETs.

1. We first work on techniques that can provide QoS support in MANETs from the single-layer perspectives with little interaction between two adjacent layers. In this direction, we mainly work on MAC layer and Routing layer of the protocol stack.

2. We adopt a bottom-up approach, by looking first at the underlying MAC layer. It is important that any QoS mechanism for MANETs must necessarily be supported at the MAC layer. We observe that IEEE 802.11 Distributed Coordination Function (DCF), which is the most widely used MAC protocol in MANETs does not provide QoS support due to its inherent problems. We propose two different schemes for resolving MAC layer contention in the form of priority based variants of the basic DCF. The first scheme is aimed at achieving enhanced service differentiation along with maximizing network resource utilization in multihop environment. The second one aims at providing a strict level of service differentiation such that the presence of best-effort traffic in the network can not affect the service quality achieved by real-time traffic. Because, most of the real-time applications can tolerate some amount of packet loss, we assign lower retry limit to real-time traffic in both the schemes for reducing channel access delay due to retransmission.

3. QoS-aware routing plays an important role for supporting QoS in a multihop ad hoc network. Most of the existing QoS routing algorithms proposed are based on reactive routing. In these routing protocols, to cope up with frequent route failures (due to link failure or QoS violation), route recovery procedures are initiated. These procedures consumes substantial amount of resources and introduce extra delays in QoS flows. Existing QoS routing protocols do not take any measure to select QoS routes that endure longer time. Further, most of them choose shortest path route among the feasible paths, which has short lifetime due to selection of longer hops (also called edge effects [64]). From the best of our knowledge, no stability based QoS routing with both throughput and delay constraints is available in the literature. Our objective is to develop a route stability based QoS routing with delay and throughput constraints, which

can improve the durability of established QoS route and hence can reduce QoS disruption due to expensive route recoveries. The related objective is that the route stability model to be incorporated in the QoS routing should have less computational overhead with minimum control overhead and without any extra hardware support.

4. Our next aim is to extend the route stability based QoS routing (RSQR) to be developed with proactive route maintenance feature for detection of potential link breaks or QoS violations before actual link breaks or QoS violation takes place. This will save route recovery time and reduce possibility of packet loss during active communication. In this direction, our objective is to extend route discovery mechanism in RSQR, utilizing the existing information to discover a node-disjoint QoS-aware multipath routing which will reduce expensive route recoveries through proactive route maintenance. We found that, most of the multipath routing protocols that are available in literature provide only enhanced best-effort service. Although some of them have claimed to be QoS-aware, they either provide multiple reliable paths without support for real-time traffic or the QoS metric they consider are very restrictive.

5. Finally, our aim is to view the QoS provisioning problem from a multilayer perspective with mapping of user specified QoS parameters across different layers including application layer, network layer and MAC layer. It is known that QoS support in MANET not only depends on application requirements, it should also rely on the underlying network condition/quality. Therefore, each layer should function in an adaptive manner based on the applications' requirements and network status. We investigated the different possibilities in adapting each layer and proposed an overall Cross-layer framework, where network layer and MAC layer work in an adaptive way based on underlying network condition and the QoS requirements of the upper layer. We found that most of the existing QoS approaches for MANETs are based on conventional layering concepts with little or no cross-layering interactions. While traditional protocol layering is an important abstraction to reduce complexity of network design in wired network, it is not well suited to wireless networks to provide complex functionalities like QoS due to inter-dependencies of different layers.

1.4 Dissertation Contributions

In this section we briefly outlines the major contributions of this dissertation.

1.4.1 PQAMP: A Priority based QoS-aware MAC Protocol for Mobile Ad Hoc Networks

The design goal of the first QoS-aware MAC scheme, called Priority based QoS-aware MAC Protocol (PQAMP), is to achieve service differentiation for providing QoS for real-time traffic along with maximizing utilization of network resource without starving the best-effort traffic in multihop environment. PQAMP is a dynamic priority based QoS-aware MAC protocol, where nodes are assigned priority based on the type of traffic (either Real-Time or Best-Effort) to handle and their roles (relay or transmit). Although it appears that there are only two classes of traffic, real-time and best-effort, present in the network, whereas a node can have further derived priority classes to achieve

QoS support for end-to-end flow. With a non-overlapping range of contention window assigned to nodes with different priority classes (but in the same backoff stage), PQAMP achieves a level of traffic differentiation for providing QoS to real-time traffic. Due to possibility of some priority inversion cases, PQAMP can not support strict level of QoS for real-time traffic in the presence of heavy best-effort traffic load conditions. PQAMP can ensure a minimum level of service for best-effort traffic and increases overall network utilization due to higher priority assigned to relayed frames. We evaluate PQAMP through a Markov chain model for single hop scenarios, and demonstrate its validity through simulation studies. Finally, we have shown the superiority of PQAMP to provide QoS support in different multihop scenarios as compared to IEEE 802.11 DCF.

1.4.2 SPQAMP: A Strict Priority based QoS-aware MAC Protocol for Mobile Ad Hoc Networks

The design goals of the second scheme, called Strict Priority based QoS-aware MAC Protocol (SPQAMP), are - to achieve a strict service differentiation to support QoS for real-time traffic without any delivery guarantee for best-effort traffic, and to achieve overall network utilization as compatible with IEEE 802.11 DCF. SPQAMP is also a dynamic priority based QoS-aware MAC protocol, where nodes are assigned priority based on the type of traffic to handle. In SPQAMP, contention window for real-time traffic is always smaller than that of best-effort traffic and they remain non-overlapped irrespective of the their backoff stage. Further, freezing of backoff counter for best-effort traffic is replaced with a resetting mechanism to eliminate the possibility of priority inversion. This ensures SPQAMP to maintain a strict or stronger level of traffic differentiation as compared to PQAMP.

We have evaluated the performance of SPQAMP through Markov chain analysis and demonstrated its validity through simulations. Finally, we have demonstrated the suitability of the model in multihop scenarios using extensive simulation studies. We found that the proposed protocols are efficient in supporting QoS to real-time traffic in terms of end-to-end delay and throughput with some added advantages.

1.4.3 RSQR: A Route Stability based QoS Routing for Mobile Ad Hoc Networks

We first develop a Route Stability Model (RSM) to estimate stability of link/path in MANETs. The RSM is based on simple calculations on measured received signal strengths obtained from link layer, has less control overhead, require no extra hardware, and has the potential to give good approximation on link stability in most of the real-life mobility scenarios. Using RSM, we propose a Route Stability based QoS Routing (RSQR) for wireless ad hoc networks. RSQR incorporates a soft resource reservation, an admission control based on throughput and delay, and applies route stability metric to improve the durability of established QoS route such that the probability of the reservation to remain valid for duration of data transmission is high. Further, to maintain QoS assurance to admitted flows, it incorporates an efficient route recovery due to QoS violation and route failures. RSQR reduces QoS disruption due to frequent route recoveries and gives higher performance in terms of packet delivery ratio, average end-to-end delay and control overhead as compared to the QoS routing protocol, AQOR [115], proposed by Q Xue and A Ganz.

1.4.4 SMQR: A Route Stability based Multipath QoS Routing for Mobile Ad Hoc Networks

To further reduce the QoS disruption due to frequent route recoveries, we have proposed a Stability based Multipath QoS Routing (SMQR) protocol for MANETs. SMQR uses a number of special route request forwarding rules to compute a maximum of three node-disjoint QoS routes which reduces path diminution problems without imposing excessive control overheads. Like the unipath protocol, RSQR, SMQR is also designed to support throughput and delay sensitive real-time applications in MANETs. We incorporate a hop by hop admission control and resource reservation scheme during route discovery process to ensure QoS assurance to real-time applications. Further, RSM is utilized during both route discovery and maintenance phases for selecting QoS routes with higher stability. Reliability of the multiple QoS-aware routes are improved through node-disjointness and stability properties of the discovered routes. Following an effective QoS violations detection and route maintenance mechanism, the protocol reduces the frequency of route recovery leading to significant reduction in QoS disruption. Simulation studies show that SMQR can support higher level of QoS in terms of packet delivery ratio, average end-to-end delay, maximum jitter, and overall network throughput. Detection of potential route failures and switching to an available stable route before actual route breaks saves route recovery time as well as reduces possibility of packet loss during active communication using SMQR.

1.4.5 A Cross-Layer QoS Framework for Mobile Ad Hoc Networks (CLQM)

In this context, we are looking at the problem of QoS provisioning from the perspective of three layers - application layer, network layer and MAC layer. Our goal is to design a QoS framework which maps QoS parameters across three layers of the protocol stack to support a stronger notion of per-class service guarantees in terms of throughput and delay in ad hoc networks. Applications' QoS requirements are specified as service classes - Class I, Class II, Class III and Class IV which represent delay sensitive, delay and throughput sensitive, throughput sensitive and best-effort traffic respectively. QoS metrics at application layer are accordingly represented as delay, throughput and best-effort. Application layer metrics are mapped to corresponding path metric in network layer and finally to the individual link layer metrics at the MAC layer. QoS mapping at network layer is realized through a class based packet queuing and scheduling policy, whereas MAC layer QoS mapping takes place in terms of non-overlapping contention windows assigned to different classes of traffic. Further, to cope up with network dynamics, MAC layer and network layer are made to adapt their behaviour based on both dynamic network conditions and QoS requirements of admitted flows. MAC layer adaptation takes place in terms of dynamic contention window adjustment per class, whereas network layer adaptation takes place in the form of rerouting or termination of existing flows. We have proposed a detailed QoS mapping scheme with adaptive protocols using a TCP/IP protocol suite.

1.5 Dissertation Organization

The rest of the dissertation is organized as follows.

- Before going to discuss our contributions, the literature survey about the current approaches for supporting QoS in MANETs and their effectiveness are presented in **Chapter 2**.

- **Chapter 3** is about the Priority-based QoS-aware MAC Protocols that are developed in this dissertation. The Designs of these protocols and their analysis through Markov chain models are presented. To validate the analytical results obtained through Markov Chain models of the proposed MAC protocols, detailed simulation studies are reported in this chapter.

- **Chapter 4** is about the Route stability based QoS Routing Protocol developed in this thesis work. In this chapter, we first discuss about the route stability model (RSM) developed and then discuss about the route stability based QoS routing (RSQR) which incorporate the RSM. Further, we report the detailed simulations carried out to evaluate the QoS performance of RSQR.

- In **Chapter 5**, we discuss about the Route stability based Multipath QoS Routing (SMQR) Protocol developed in this thesis work along with its performance evaluation through simulation studies.

- **Chapter 6** is about the Cross-Layer QoS Framework that we have proposed in this thesis. In this chapter, we discuss about the mapping of QoS parameters across three protocols layers namely - Application layer, Network layer and MAC layer. Further, we discuss about the different components to form an overall QoS framework.

- Finally, **Chapter 7** summarizes the overall contribution of this thesis and identifies some future research directions.

Chapter 2

Literature Survey of QoS Approaches

2.1 Introduction

In recent years, wireless networks are becoming more and more popular. Among the wireless networks, mobile ad-hoc networks have attracted considerable research interest, as they are easy to deploy and maintain. Research in ad hoc networking has been going on for some time. The history of wireless ad hoc networks can be traced back to the Defense Advanced Research Projects Agency (DARPA) packet radio network (PRNet) [43], which evolved into the survivable adaptive radio network (SURAN) program [31]. Ad hoc networks have played an important role in military applications and related research efforts. Recent years have seen a new spate of industrial and commercial applications for wireless ad hoc networks, as many portable computers and personal digital assistants (PDAs) equipped with wireless interfaces are becoming more compact and inexpensive. Ad hoc networks have numerous potential applications. For example, the IEEE 802.11 wireless LAN standard family and the European Telecommunications Standards Institute (ETSI) HIPERLAN/2 standard, or even Bluetooth, support an ad hoc mode of operation for building simple ad hoc networks. Ad hoc networks are very useful in battle-field, disasters (such as flood, fire and earthquake) recovery, emergency search-and-rescue operations, home networking, meetings or conventions in which people wish to quickly share information.

The Internet Engineering Task Force (IETF), the body responsible for guiding the evolution of the Internet, provides the definition of mobile ad hoc network as follows.

"A *mobile ad-hoc network (MANET)* is a kind of wireless ad-hoc network, and is a self-configuring network of mobile routers (and associated hosts) connected by wireless links - the union of which form an arbitrary topology. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet. Minimal configuration and quick deployment make ad hoc networks suitable for emergency situations like natural or human-induced disasters, military conflicts, emergency medical situations etc."

A MANET is an autonomous system of mobile nodes. The system may operate in isolation, or may have gateways to interface with a fixed network. Nodes in MANET are equipped with wireless transmitters and receivers using antennas which may be omni directional (broadcast), highly-directional (point-to-point), possibly steerable, or some combination thereof. At a given point in time, depending on the nodes' positions and their transmitter and receiver coverage patterns, transmission power levels and co-channel interference levels, a wireless connectivity in the form of a random, multihop graph or "ad hoc" network exists between the nodes. This ad hoc topology may change with time as the nodes move or adjust their transmission and reception parameters. Figure 2.1 shows a typical MANET with four nodes, where the solid lines are used to represent wireless links and circles with dashed lines are used to show the communication ranges of the nodes. In the figure, a multihop path between nodes S and D is shown with two thick solid lines.

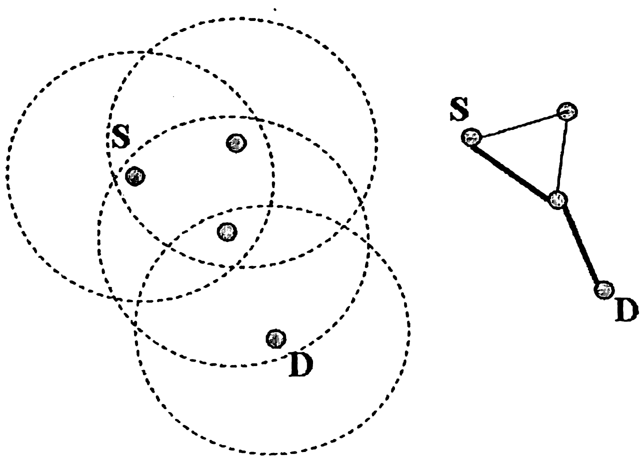


Figure 2.1: A Mobile Ad Hoc Network

MANETs have several salient characteristics [74]. These are

(i) Dynamic topologies: Nodes are free to move arbitrarily; thus, the network topology may change randomly and rapidly at unpredictable times.

(ii) Bandwidth-constrained, variable capacity links: Wireless links will continue to have significantly lower capacity than their wired counterparts. In addition, the realizable throughput of wireless communications, after accounting for the effects of multiple access, fading, noise, and interference conditions, etc. is often much less than a radio's maximum transmission rate.

(iii) Energy-constrained operation: Some or all of the nodes in a MANET may rely on batteries for their energy. For these nodes, the most important system design criteria for optimization may be energy conservation.

(iv) Limited physical security: Mobile wireless networks are generally more prone to physical security threats than the wired networks. Therefore, increased possibility of eavesdropping, spoofing, and denial-of-service attacks should be carefully considered in deployment of MANETs.

With the increase in Quality of Service (QoS) needs in evolving applications and the widespread use of wireless and mobile devices, it is also desirable to support QoS in MANETs. The resource limitations and variability further add to the need for QoS support in such networks. However, the characteristics of these networks make the QoS support a very complex process [15, 26, 112].

Quality of Service is usually defined as the set of service requirements that need to be met by the network while transporting a packet stream from a source to a destination [24]. Intuitively, QoS means that there is an agreement or a guarantee by the network to provide a set of measurable pre-defined service attributes to the users in terms of end-to-end performance, such as delay, throughput, probability of packet loss, delay variance (jitter), etc [15]. Power consumption and service coverage area are two other QoS attributes that are more specific to MANETs. The Telecommunication standardization unit of the International Telecommunication Union (ITU-T) defines quality of service in recommendation E.800 as "the collective effect of service performance which determines the degree of satisfaction of a user of the service". This is the notion of *hard* or *absolute* QoS guarantee. Since it is extremely difficult to archive *hard* QoS in MANETs due to their inherent properties, *soft* QoS comes in the domain of MANETs. This *soft* QOS means there may exist transient periods of time during which the required QoS guarantee is not possible due to non-availability of suitable forwarding nodes or network partition or wireless channel impairments [15, 121]. The main technology driver for quality of service to date has been converged networking. Converged networking is based on the idea that one data network should be used to transport voice, video, and data where voice and video applications are delay, throughput and jitter sensitive. Hence, QoS provisioning can be understood as a way of prioritizing certain types of traffic over the other.

The QoS metrics could be defined in terms of one of the parameters or as a set of parameters in varied proportions. The multi-constraints QoS aim at optimizing multiple QoS metrics while provisioning network resources, and is an admittedly complex problem. As different MANET applications have different requirements, the services required by them and the associated QoS parameters differ from application to application. Following are some of the potential application areas of MANETs and their associated QoS parameters.

- *Multimedia Applications*: Guaranteed bandwidth, delay and delay jitter;
- *Search-and-rescue operations*: Accessibility or availability to the network;
- *Military networking*: Stringent security requirements;
- *Sensing networks*: Power consumption efficiency;
- *Last mile provider*: Fluent bandwidth management;

In the literature, the research on QoS support in MANETs spans over all the layers of the protocol stack. The major approaches for providing QoS in MANETs are QoS Models, QoS aware physical layer, QoS-aware Medium Access Control (MAC) protocols and QoS-aware routing. In the following sections we briefly discuss about some of the available schemes under these approaches.

2.2 QoS Models

The QoS model [112] specifies the architecture for providing some kind of services in the network. A QoS model for MANETs should consider the challenges of MANETs, e.g. dynamic topology and time-varying link capacity etc. In addition, the potential commercial applications of MANETs require connection to the Internet. Thus QoS model for MANETs should also consider the existing QoS characteristics in the Internet. In the following subsections we briefly discuss about some of existing QoS models and their effectiveness to apply in MANETs.

2.2.1 Integrated Service with Resource Reservation Protocol (IntServ/RSVP)

IntServ [11] architecture allows sources to communicate their QoS requirements to routers and destinations on the data path by means of a signaling protocol such as RSVP [120]. In this model the flow-specific states are kept in every IntServ-enabled router. IntServ proposes two service classes in addition to Best-Effort Service. One is Guaranteed Service; the other is Controlled Load Service. The Guaranteed Service is provided for applications requiring strict delay bound. The Controlled Load Service is for applications requiring reliable and enhanced best effort service. IntServ/RSVP is not suitable for MANETs due to the following factors:

- Huge amount of storage and processing overhead, so the problem of scalability.
- Signaling packets will consume a substantial amount of bandwidth in MANETs.
- Performing of admission control, classification and scheduling by every node will put a heavy burden for the resource-limited mobile hosts.

The Dynamic Resource Reservation Protocol (dRSVP) [51, 72] extends the basic RSVP to support dynamic QoS in MANETs, but has similar problem like RSVP due to extensive use of out-of-band signaling. The mobile RSVP (MRSVP) [103] was designed for data communication in cellular network which needs an infrastructure for controlling of resource reservation, hence it can not be applicable in MANETs.

2.2.2 Differentiated Service (DiffServ) Model

Differential Service (DiffServ) [9, 77] architecture avoids the problem of scalability by defining a small number of per-hop behaviors (PHBs) at the boundary routers and associating a different DiffServ Code Point (DSCP) in the IP header of packets belonging to each class of PHBs. The interior routers along the forwarding path use DSCP to differentiate between different QoS classes on per-hop basis. Thus, DiffServ is scalable but it does not guarantee services on end-to-end basis. This hinders DiffServ deployment in the Internet as well as in MANETs. In DiffServ, we can identify three different service classes: expedited forwarding, assured forwarding, and best effort. DiffServ may be a possible solution to the MANET QoS model, because it is light-weighted in interior routers. But it also has some weaknesses as follows:

- It is difficult to predefine boundary and interior routers.
- The concept of Service Level Agreement (SLA) in the Internet does not exist in MANETs which is indispensable for the functioning of DiffServ.

2.2.3 Flexible QoS Model for MANETs (FQMM)

Flexible QoS Model for MANETs (FQMM) [113] tries to take the advantage of both the per-flow service granularity in IntServ and the service differentiation in DiffServ. This model defines three types of nodes: an ingress node which sends data, an interior node which forwards data to other nodes, and an egress node (destination). Obviously, each node may have multiple roles. FQMM selectively uses the per-flow QoS guarantees of IntServ for applications with high priority, and the service differentiation of DiffServ for applications with lower priorities. FQMM is the first attempt at proposing a QoS model for MANETs with the following problems:

- Without an explicit control on the number of services with per-flow granularity, the scalability problem still exists.
- To make a dynamically negotiated traffic profile is a very difficult problem.
- It is difficult to code the PHB in the DS field of IP, if the PHB includes per-flow granularity considering the DS field is at most 8 bits without extension [112].

2.2.4 Service Differentiation in Stateless Wireless Ad hoc Networks (SWAN)

Service Differentiation in Stateless Wireless Ad hoc Networks (SWAN) [3] is a stateless network model which uses distributed control algorithms to deliver service differentiation in mobile wireless ad hoc networks in a simple, scalable and robust manner. The SWAN model [3] considers TCP traffic as best-effort traffic and UDP traffic as real-time traffic requiring QoS assurances. It tries to maintain delay and bandwidth requirements of real-time traffic by admission control of UDP traffic and rate control of TCP and UDP traffic. SWAN differentiates traffic into two classes, best-effort and real-time traffic [3, 4]. It treats all UDP (real-time) traffic with equal priority so they will be treated similarly during network congestion. Although SWAN is claimed to be stateless, intermediate nodes have to remember about the traffic flow to distinguish between new and old flows for regulating flows. SWAN does not consider best-effort traffic for calculating available bandwidth and source-based admission control using probing packets is ineffective to cope up with dynamic environment in MANETs.

2.2.5 INORA

INORA [28] is a QoS support mechanism that makes use of the INSIGNIA (In-band Signaling in Ad hoc Network) in-band signaling and TORA (Temporally Ordered Routing Algorithm) routing protocol [83] for MANETs. INORA represents a QoS-signaling approach in a loosely coupled kind of manner. The idea is based upon the property of TORA to provide multiple routes between a given source-destination pair. While INSIGNIA [55] does not take any help from the network with regard to redirecting the flow along routes which are able to provide the required QoS guarantees, INORA gives feedback to the routing protocol (TORA) on a per-hop basis to direct the flow along the route that is able to satisfy the QoS requirements of the flow. For resource reservation, a soft state reservation mechanism is employed. Beyond doubt the concept of loosely coupled QoS signaling and routing is a very promising approach and the shortcomings of INORA mostly are the shortcomings of INSIGNIA. However, the interface for signaling to access routing should be as generic as possible in order to guarantee portability. INORA is better than INSIGNIA in that it can search multiple paths with lesser QoS guarantees. Since no resources are reserved before the actual data transmission starts and since data packets have to be transmitted as best-effort packets in case of admission control failure at the intermediate nodes, this model may not be able to provide better service guarantees for real-time applications.

2.3 QoS at Physical layer

QoS at physical layer refers to data rate and packet loss rate on wireless channel. Due to unstable channel quality, maintaining a constant data rate and low packet loss is very difficult. Physical layer of IEEE 802.11 a/b/g can support multiple data rates that are achieved by different modulation schemes in the PLCP (Physical Layer Convergence Protocol) header of the physical layer. Since transmission rate can vary depending on the channel quality, an appropriate link adaptation mechanism is desirable to maximize the throughput under dynamically varying channel conditions. Link adaptation mechanism may be modified to implement QoS requirements of higher layers in terms of throughput and packet loss rate. For efficient utilization of a multi-rate physical layer, there has been several adaptation algorithms proposed at the physical layer, where most of them are closely tied to the MAC layer as well. These algorithms impact the observed link throughput on hop by hop basis which leads to change in end-to-end throughput of a selected path. An algorithm by Kamerman et al, called Auto Rate Fallback (ARF) [45], is a sender based channel estimation and rate adaptation mechanism originally designed for Lucent's WaveLAN II devices running IEEE 802.11b standard. ARF uses the ACKs of transmitted frames as a measure of channel condition. When an ACK is not received after transmission of a frame, the first retransmission is done at the same data rate. If no ACK is received after the first retransmission, the data rate is lowered to the next level for subsequent transmissions and retransmissions. If ten consecutive ACKs are received correctly or if a timer expires, then the device attempts to upgrade the data rate. If the first transmission at the higher data rate fails, it goes back to the lower data rate. A similar but more adaptive approach was proposed by Chevillat et al uses two thresholds S and F for estimating channel condition. In their approach [21], when the number of consecutive successful transmission exceeds S, the data rate is increased to the next level. Similarly, when the number of consecutive failed transmission exceeds F, then transmission rate is decreased to the next level. Holland et al observed that the channel condition can fluctuate very frequently and ARF is not capable of adapting the data rate accordingly. In their method [39], called Receiver Based Auto Rate (RBAR) adaptive protocol, channel estimation is done at the receiver based on SNR (Signal-to-Noise Ratio) of RTS. Accordingly receiver informs the sender about the maximum data transmission rate possible for the upcoming data transmission. Since channel estimation is done just before the actual data transmission, link adaptation in RBAR is better than ARF. Another mechanism called Opportunistic Auto Rate (OAR) proposed by Sadeghi et al try to improve throughput in ad hoc networks with multi-rate links. OAR [91] is based on the idea that using higher data rate it is possible to transmit multiple packets. Based on channel estimation either at receiver or at the source, the algorithm ensures that all nodes are granted channel access for the same time shares as achieved by single rate IEEE 802.11. The Full Auto Rate (FAR) [61] adapt the transmission rate of all the frames (both control and Data frames) to further improve the performance of IEEE 802.11.

2.4 QoS-aware MAC protocols

MAC protocols [7, 22, 27, 33, 60, 61, 65, 69, 107] are mostly for solving the problems of medium contention, supporting reliable unicast communication, and providing resource reservation for real-time traffic in a distributed wireless environment. While the main design goal of most of the MAC protocols is usually to solve medium contention and hidden/exposed terminal problems and to improve throughput, a QoS-based MAC protocol must also provide a channel access mechanism that considers the QoS constrains of real-time flows [85, 89]. Since it is not possible to provide upper bounds on channel access delay when using random access mechanisms, the goal is simply to reduce the channel access delay of real-time flows. A related problem to QoS is providing the fairness among different traffic flows inside a traffic class [60]. The overall problem of providing QoS and fairness becomes more difficult due to dynamic topology and limited resources in the MANETs. The QoS-based MAC protocols can be broadly classified as either priority-based or reservation-based. Due to their simplicity in design, here we consider only random access MAC protocols. In the following subsections, we briefly discuss about some of the available QoS-aware MAC schemes for MANETs.

2.4.1 Real-Time Medium Access Control (RT-MAC)

Baldwin et al [7] proposed a variation of the IEEE 802.11 protocol called RT-MAC that supports real-time traffic by avoiding packet collisions and the transmission of the packets that have already missed their deadlines. To achieve this, RT-MAC scheme uses a packet transmission deadline and an enhanced collision avoidance scheme to determine the transmission station's next backoff value. When a real-time packet is queued for transmission, a time-stamp is recorded locally in the node indicating the time by when the packet should be transmitted. The sending node checks whether a packet has expired at three points: before sending the packet, when its backoff timer expires and when a transmission goes unacknowledged. An expired packet is immediately dropped from the transmission queue. When the packet is actually about to be sent out, the sending node chooses the next backoff value and records it in the packet header. Any node that overhears this packet then ensures that it chooses a different backoff value. This eliminates the possibility of collision. The range of values (i.e., contention window, CW) from which the backoff value is chosen, is made a function of the number of nodes in the system. Therefore, the number of nodes should be known or at least estimated in this scheme. RT-MAC scheme has been shown to achieve drastic reductions in mean packet delay, missed deadlines, and packet collisions as compared to IEEE 802.11. However, the contention window may typically become quite large in a network with large number of nodes. This will result in wasted bandwidth when the network load is light.

2.4.2 DCF with Priority Classes (DCF-PC)

Deng et al [27] proposed another variation of the IEEE 802.11 protocol that supports priority based access for different classes of data. The basic idea is to use a combination of shorter IFS or waiting times and shorter backoff time values (i.e., maximum allowable size of contention window) for higher priority data (i.e., real-time traffic). While normal nodes wait for the channel to remain idle after DIFS interval before they transmit data, a higher priority node waits for only PIFS. However, if the chosen backoff value happens to be longer, the higher priority node can still lose out to another node that has a larger IFS but a shorter random backoff value. In order to solve this problem, authors of [27] have proposed two different formulas for generating the random backoff values so that the higher priority nodes are assigned shorter backoff time. Authors have demonstrated that this scheme has better performance than 802.11 DCF, in terms of throughput, access delay and frame loss probability for higher priority (real-time) traffic. It can support more than two traffic priorities. However, this scheme lacks the ability to provide deterministic delay bounds for real-time traffic. Moreover, normal data traffic suffers higher delay due to a longer backoff time even when no higher priority node is transmitting. Channel bandwidth is also wasted in such cases.

2.4.3 Enhanced Distributed Channel Access (EDCA)

IEEE 802.11 DCF is designed to provide a channel access with equal probabilities to all the contending nodes in a distributed manner. EDCA enhances the DCF protocol to provide differentiated channel access according to the frame priorities. It has been developed as a part of the hybrid coordination function (HCF) of IEEE 802.11e [22, 69]. Two new features introduced in the HCF are the concepts of access category (AC) and transmission opportunity (TXOP). Each station using the EDCA, has four ACs and for each AC there is one transmit queue with an independent EDCA function that contends for access to the medium. The four ACs have different priorities and are used for different kind of traffic: background (AC BK), best effort (AC BE), video (AC VI) and voice (AC VO). A TXOP is a bounded time interval, defined by a starting time and a maximum duration, during which a station may transmit multiple frames. In EDCA, each data frame is assigned a Access Category (AC) in the MAC header, based on its priority as determined in the higher layers. During the contention process, EDCA uses AIFS[AC], CWmin[AC], CWmax[AC] and TXOPLimit (Transmission Opportunity limit) for a frame belonging to a particular AC. Hence, each AC is an enhanced variant of the DCF. Flows that fall under the same AC are effectively given identical priority to access the channel. A station accesses the channel based on the AC of the frame to be transmitted. The contention among different ACs with the same node is handled with the help of virtual contention resolution. After successfully reserving the channel for data transmission, a station may transmit one or more frames based on the corresponding value of TXOPLimit. In EDCA, AIFS (Arbitration Inter Frame Space) duration is at least DIFS, and can be enlarged individually for each AC. The CWmin of the backoff mechanism is set differently for different priority classes. EDCA thus combines two measures to provide service differentiation. This mechanism assumes existence of an access point to control the QoS functionality of the system. Further, there is no hints about mapping user requirements into the corresponding TC at MAC level. Using EDCA, normal traffic suffers higher delays even when in absence of real-time traffic in the network.

2.4.4 Black Burst (BB) Contention

The Black Burst (BB) contention scheme proposed in [100] avoids packet collision in a very novel manner, and solves the packet starvation problem as well. Packets from two or more flows of the same service class are scheduled in a distributed manner with fairness guarantee. Nodes content for the medium after it has been idle for a period longer than the inter-frame space. Nodes with best-effort traffic and nodes with real-time traffic use different inter-frame space values in such a way that real-time traffic have higher priority than best-effort traffic. BB contention scheme is added to any CSMA/CA type protocol in the following manner. Before sending their packets when the medium remains idle long enough, real-time nodes first contend for transmission right by jamming the media with pulses of energy called BB's (Black Bursts). The main principle in this scheme is that each contending node is using a BB with different length which is directly proportional to the contention delay experienced by the node. Following each BB transmission, a node senses the channel for an observation interval. Since distinct nodes contend with BB's of different length, each node can determine without ambiguity whether its BB is of longest duration. Thus only one winner is produced after this contention, which will transmit its real-time packets successfully. BB contention ensures that real-time packets are transmitted without collisions and with priority over best-effort packets. It has the main drawbacks of requirement for an extra channel and precise generation of black-burst pulses.

2.4.5 Distributed Fair Scheduling (DFS)

Vaidya et al. [107] proposed the DFS scheme to ensure that different flows sharing a common wireless channel are assigned appropriate bandwidth corresponding to their weights or priorities. DFS is derived from the IEEE 802.11 DCF and requires no central coordinator to regulate access to the medium. The fundamental idea of DFS is that each packet is associated with start and finish timestamps. A higher priority packet is assigned a smaller 'finish-tag' and shorter backoff periods. This approach ensures that any flow that has packets of higher priority will consistently have shorter backoff times, thereby achieving a higher throughput. In DFS, the start and finish times for packets are calculated on the basis of the Self-Clocked Fair Queuing (SCFQ) algorithm proposed by Golestani [38]. Following the idea of SCFQ, every node also maintains a local virtual clock. DFS does not, however, remove short-term unfairness in certain cases The authors observe that the use of collision resolution schemes such as those proposed in [33] can resolve this anomaly. In order to calculate backoff intervals, the authors have proposed two alternate approaches: linear mapping and exponential mapping. A disadvantage of the linear mapping scheme is that if many packet flows have low priorities, all of them are assigned large backoff intervals. As a result, the system remains idle for long periods of time. The exponential mapping approach is proposed as one solution to this problem. Using simulations, the authors have shown that DFS obtains a higher throughput than IEEE 802.11. Also, they have verified that use of exponential mapping technique for calculating backoff intervals leads to higher throughput than linear mapping. However, the DFS has the problems of not solving the hidden terminal problem and not considering delay bound of real-time packets [96]. Further, selection of a proper mapping function is a difficult problem.

2.4.6 Multiple Access Collision Avoidance with Piggyback Reservation (MACA/PR)

Multiple Access Collision Avoidance with Piggyback Reservation (MACA/PR) [65] provides rapid and reliable transmission of best-effort datagram as well as guaranteed bandwidth support to real-time traffic. For the transmission of best-effort datagram in MACA/PR, a host with a packet to send must first wait for free "window" in the reservation table, which records all the reserved send and receive windows of any station within the transmission range. It then waits for an additional random amount of time on the order of a single hop round trip delay. If it senses the channel as free, it proceeds with RTS-CTS-PACKET-ACK dialogue for a successful packet transmission. If the channel is busy it waits until the channel becomes idle and repeats the above procedure. For the transmission of real-time packets, the behaviour of MACA/PR is different. In order to transmit the first data packet of a real-time stream, the sender initiates an RTS-CTS dialogue and then proceeds with PACKET-ACK dialogue if the CTS dialogue is received. For subsequent data packets (not the first one) of a real time stream, Reservation Tables, which are propagated among neighbors, are used to avoid the hidden-terminal problem as opposed to the RTS-CTS dialogue. The first data packet of the real-time stream then makes the reservation along the selected path. The source node piggybacks the real-time scheduling information for the next packet transmission in the header of its current data packet. The destination confirms the reservation with an ACK, which contains the scheduling information for the next packet reception. Thus, any node hearing a real-time packet updates its reservation table and defers transmission for the specified time. Note that if the sender fails to receive several ACK's, it restarts the connection with the RTS-CTS dialogue again. MACA/PR does not retransmit the real-time packets after collision. MACA/PR relies on two assumptions. Firstly, real-time packets arrive at constant time intervals and secondly, the communication links are symmetric. These, assumptions are not necessarily valid in a MANET environment [85].

2.4.7 Elimination by Sieving (ES-DCF) and Deadline Bursting (DB-DCF)

Pal et al [81, 82] proposed two variants of the IEEE 802.11 DCF that offer guaranteed time bound delivery for real-time traffic, by using deterministic collision resolution algorithms. Interestingly, they also employ black burst features. The ES-DCF has three phases of operation - elimination, channel acquisition and collision resolution. In elimination phase, every node is assigned a grade based on the deadlines and priority of its packets as in [27]. A closer deadline results in a lower numerical grade, which translates to lower than DIFS channel-free wait time. Therefore, the grade of the packet improves if it remains in the queue for a longer time. In the channel acquisition phase, the node transmits RTS packet to initiate the channel acquisition, when the channel has been free for the requisite amount of time, as decided by the grade of its data packet. If it receives a CTS packet in return, the channel is considered acquired successfully. Otherwise, the third phase of collision resolution is initiated by sending out a BB (as in [100]). The length of the BB corresponds to the node identification (Id) number. Higher Id numbers are given to the nodes that generate a lot of real-time data. The node that sends out the longest burst accesses the channel at the subsequent attempt.

In the DB-DCF, the first phase is for BB contention wherein the lengths of the BB packets are proportional to the urgency (i.e., relative deadlines) of the real-time packet. This is followed by phases for channel acquisition and collision resolution, which are similar to the corresponding phases in ES-DCF. Both schemes assign channel-free wait time longer than DIFS for best-effort nodes, such that these nodes are allowed to transmit only when the other real-time nodes have no data waiting to be sent. However, the results of the simulations carried out by the authors indicate that ES-DCF is more useful when hard real-time traffic is involved, and DB-DCF performs better in the case of nodes with soft real-time packets. However, due to its dependence on BB techniques, it inherits the similar problems in BB [100] as we discussed earlier.

2.5 QoS Routing Protocols

QoS routing is one of the most essential components of a QoS architecture. QoS routing protocols try to find end-to-end paths for flows with the knowledge of network resource availability and the QoS requirements of the corresponding applications such that there is a good chance of meeting the QoS requirements of the applications while transporting data packets over the discovered paths [15, 16, 23, 89, 99]. QoS routing is difficult in MANETs. Firstly, it has too high overhead due to maintenance and updation of link state information. Secondly, maintaining the precise link state information is very difficult. Thirdly, a feasible path may not guarantee a QoS path. Fourthly, finding a stable QoS route such that the discovered path will remain valid for a longer period of time. Finally, nodes can be selfish and unwilling to cooperate which causes difficulty in reserving resources [68]. Here, we discuss about some of the QoS routing protocols for MANETs.

2.5.1 Core Extraction Distributed Ad hoc Routing (CEDAR)

CEDAR [99], a Core Extraction Distributed Ad hoc Routing algorithm is proposed for small to medium size ad hoc networks. It dynamically establishes the core of the network, and then incrementally propagates the link states of stable high-bandwidth links to the core nodes. The route computation is on demand basis, and is performed by the core nodes using only local state information. CEDAR has three key components: Core Extraction - to select a set of nodes to form the core that maintains the local topology of the nodes in its domain, and also performs route computations. Link State Propagation- QoS routing in CEDAR is achieved by propagating the bandwidth availability information of stable links to all core nodes. Route Computation - Route computation first establishes a core path from the domain of the source to the domain of the destination. Using the directional information provided by the core path, CEDER iteratively tries to find a partial route from the source to the domain of the furthest possible node in the core path satisfying the requested bandwidth. The sheer amount of updates (in the form of flooding) it needs in order to maintain a global topology and state information makes CEDAR unsuitable, especially in random mobility environment of MANETs. QOLSR (QoS aware Optimal Link State Routing) [5, 6] has similar disadvantage due to heavy amount of link state updates required to maintain network topology information.

2.5.2 Ticket-based Probing

A distributed multipath QoS routing scheme for MANETs, called ticket-based probing is proposed in [16] to tolerate imprecise state information. This method uses tickets to limit the number of candidate paths searched during route discovery. When a source wishes to find QoS paths to a destination, it issues probe message with some tickets. One ticket corresponds to one path searching; and one probe message should carry at least one ticket. At an intermediate node, a probe with more than one ticket is allowed to split into multiple ones, each searching a different downstream sub-path. Hence, when an intermediate node receives a probe, it decides, on the basis of its available state information, whether the received probe should be split and to which neighbors the probe(s) should be forwarded. Some questions must be answered in the route search approach: First, how many tickets should be issued by the source? Second, when an intermediate host receives a probe message of n tickets, it will have to make the decisions regarding - ticket splitting and probe forwarding rules. Third, how are multiple-paths dynamically maintained?

2.5.3 Positional Attribute-based Next-hop Determination Approach (PANDA)

Positional Attribute-based Next-hop Determination Approach (PANDA) [57] discriminates the next hop nodes based on their location or capabilities. When a route-request is broadcasted, instead of using a random rebroadcast delay, the receivers opt for a delay proportional to their abilities in meeting the QoS requirements of the path. The decisions at the receiver side are made on the basis of a predefined set of rules. Thus the end-to-end path will be able to satisfy the QoS constrains as long as it is intact. A broken path will initiate a QoS route discovery process.

2.5.4 Ad hoc QoS On-demand Routing (AQOR)

AQOR[115] is a reservation-based, on-demand QoS-aware routing to provide QoS guarantees with the following features: (i) available bandwidth estimation and end-to-end delay measurement, (ii) bandwidth reservation, and (iii) adaptive route recovery. When a route is needed, the source host initiates a route request, in which the bandwidth and delay requirements are specified. The intermediate nodes check their available bandwidth and perform bandwidth admission hop-by-hop. If the bandwidth at the intermediate host is sufficient to support the request, an entry will be created in the routing table with an expiration time. If the reply packet does not arrive within the allotted time, the entry will be deleted. In this this approach, a reply packet whose delay exceeds the requirement will be deleted immediately in order to reduce overhead. To estimate available bandwidth for performing call admission control, each node puts its reserved bandwidth in periodic Hello messages that are sent to their neighbors. AQOR uses the sum of a node's neighbors traffic as the estimated total traffic affecting the node. The estimated traffic can be larger than the real overall traffic and imposes a stringent bandwidth admission control threshold. The available bandwidth is thus a lower bound on the real available bandwidth. End-to-end one way downstream delay is approximated by using half the round trip delay. With the knowledge of available bandwidth and end-to-end delay, the smallest delay path (corresponding to the first route reply packet come to the source) with sufficient bandwidth is chosen as the QoS route.

Temporary reservation is used to free the reserved resources efficiently at each node when the existing routes are broken. If a node does not receive data packets in a certain interval, the node immediately invalidates the reservation. This avoids using explicit resource release control packets upon route changes. The adaptive route recovery procedure includes detection of broken links and QoS violations and triggering of route recovery at the destination.

2.6 QoS-aware Multipath Routing Protocols

Standard on-demand routing protocols like AODV and DSR are used to discover a single route between a source and destination node. In such protocol, when an active route breaks, the nodes of the broken route simply drop the data packets because no alternate route to the destination is available. Multipath routing is useful for finding multiple paths between a source and destination in a single route discovery. These multiple routes between source and destination can be used to compensate for the dynamic topology changes in MANETs, which will reduce disruption of data transport. Recently, several different multipath routing mechanisms have been proposed where most of them are for enhancing best-effort traffic. QoS-aware multipath routing protocols proposed in [62, 66] are based on a CDMA/TDMA based MAC layer. Due to difficulty in realizing such centralized MAC scheme in MANETs, we consider only multipath routing based on a distributed MAC layer such as IEEE 802.11 DCF. Here, we discuss about some of the important multipath routing protocols which can be used (with some adaptations) for QoS support in MANETs.

2.6.1 Ad hoc On-demand Multipath Distance Vector (AOMDV)

Ad hoc On-demand Multipath Distance Vector (AOMDV) [71] is an extension of the AODV protocol to compute multiple loop-free and link-disjoint paths. The loop-freedomness is guaranteed by using a notion of "advertised hop count" and link-disjointness of multiple paths is achieved by using a particular property of flooding. To keep track of multiple paths, the routing table entries for each destination contain a list of next-hops along with the corresponding hop counts. All the next hops have the same sequence number. For each destination, a node maintains the advertised hop count, which is defined as the maximum hop count for all the paths to the destination. This is the hop count used for sending route advertisements of the destination. Each duplicate route advertisement received by a node defines an alternate path to the destination. To ensure loop-freedomness, a node only accepts a RREQ for the destination if it carries a lower hop count than the advertised hop count for that destination. Because the maximum hop count is used, the advertised hop count therefore does not change for the same sequence number. When a route advertisement is received for a destination with a greater sequence number, the next- hop list and advertised hop count are initialized. AOMDV can be used to find either node-disjoint or link-disjoint routes. To find node-disjoint routes, each node does not immediately reject duplicate RREQs. Each RREQ arriving via a different neighbor of the source defines a node-disjoint path. This is because nodes are not allowed to broadcast duplicate RREQs, so any two RREQs arriving at an intermediate node via a different neighbor of the source could not have traversed a common node. In an attempt to get multiple link-disjoint routes, the destination replies to duplicate RREQs regardless of their first hop. To ensure link-disjointness in the first hop of the RREP, the destination only replies to RREQs arriving via unique neighbors. After the first hop, the RREPs follow the reverse paths, which are node-disjoint and thus link-disjoint. High path diminution problem exists due to RREQ and RREP forwarding policies adopted in this routing protocol.

2.6.2 Ad-hoc On-demand Distance Vector Multipath (AODVM)

AODVM [118] is an extension to AODV for finding multiple node-disjoint paths. Intermediate nodes are not allowed to send a route reply directly to the source. Also, the duplicate RREQ packets are not discarded by intermediate nodes. Instead, information about all received RREQs are recorded in an RREQ table at the intermediate nodes. The destination sends a RREP for each of the received RREQ packets. An intermediate node forwards a received RREP packet toward the source through the neighbor found in the RREQ table that gives the shortest path to the source. To ensure that nodes do not participate in more than one route, whenever a node overhears one of its neighbors broadcasting an RREP packet, it deletes all the entries corresponding to that neighbor from its RREQ table. Because a node cannot participate in more than one route, the discovered routes must be node-disjoint. Due to the RREQ and RREP forwarding policies used in the protocol, the number of node-disjoint paths discovered by this protocol is restricted.

2.6.3 Mobility Prediction metric based AOMDV (MP-AOMDV)

This is an extension to AOMDV to support adaptive route maintenance based on mobility prediction metric of a path. MP-AOMDV [92] modifies AOMDV to give two types of multipath protocol- one is node-disjoint MP-AOMDV and other is Link-disjoint MP-AOMDV. In node-disjoint AOMDV, to enable discovery of multiple node-disjoint paths for a particular source-destination pair, the RREQ packet is modified to contain the address of the neighbor of the source through which the given RREQ packet has been forwarded. Each node forwards only one route request toward the destination. However, it stores the subsequent RREQ packets to remember the previous hops for each RREQ received through a unique neighbor of the source. The destination node replies to only those RREQs that come through distinct neighbors of the source. Since every intermediate node forwards only one RREQ toward the destination, each RREQ arriving at the destination has traveled along a unique path from source to destination. Once an intermediate node receives a RREP packet, it search the stored information to find out the best reverse route to the source. Two RREQs arriving at the destination by following the same neighbor of the source will have at least one node common to them. Two RREQs arriving at the destination by following a different neighbor of the source will have all node distinct (means they are node-disjoint). Here, if two or more RREQs come to the destination with same neighbor of the source, it will reduce the detection of number of node-disjoint path. This is mainly the race conditions created by RREQ packets due to forwarding of the first copy of RREQ packet by any intermediate node. Similar RREQ forwarding rule is followed for discovery of Link-disjoint paths. Each node forwards only one RREQ toward the destination during route discovery process; however, it maintains a queue of the previous hop nodes for each RREQ received from a unique neighbor of the source. The destination node does not use this source neighbor information, and reply to all RREQs. When an intermediate node receives an RREP, it forwards the RREP to the node at the head of its next hop queue, and then removes this node from the queue. This forwarding process is repeated for subsequent RREPs received at this intermediate node. And once the queue is empty, all received RREPs are dropped, because there is no more disjoint link available.

2.6.4 Multipath Dynamic Source Routing (MP-DSR)

MP-DSR [56] is a QoS aware multipath extension to DSR, proposed to provide increased reliability and stability of routes a QoS metric called end-to-end reliability is defined. Based on the reliability requirements, P_u, of an application, MP-DSR determines three parameters used for route discovery - (i) number of required disjoint paths to discover, m_0 (ii) the lowest path reliability requirement, Π_{lower}, for each path to satisfy applications' reliability requirements and (iii) the time window, t_w, within which the end-to-end guarantee provided by the selected routes hold. The source then sends m_0 RREQ messages intended for the destination to its neighbors containing information such as Π_{lower}, P_u, accumulated path reliability, traversed path etc. The intermediate nodes check to see in the RREQ message still meets the reliability requirements. If so, the node updates the reliability and path information in RREQ and then forwards the message to its neighbors. If a RREQ message no longer meets the requirements, the message is discarded. When the destination receives all the RREQ messages, it selectively chooses multiple disjoint paths from these messages and sends RREP messages back to the source through the recorded path in the selected RREQ messages. The number of copies of RREQ messages forwarded by intermediate nodes and received by the destination is determined by the parameter m_0 and the accumulated reliability in RREQ messages. When the source receives the RREP messages, it can begin using the multiple paths for transferring data. Route maintenance is performed either when all routes fail or the time window for the paths expire. For the case of all routes fail, a fresh route discovery process is initiated. In the expiry of time window, source check the validity of the routes with the help of communication to destination. If validation is unsuccessful, then new route discovery is triggered. Determination of the three parameters required for route discovery based on applications reliability requirements and local link availability is a problem in this method.

2.6.5 Adaptive Dispersity QoS Routing (ADQR)

ADQR [40] finds multiple disjoint paths with longer-lived connections based on the available network bandwidth information obtained during route discovery. Based on the user bandwidth requirements, it finds multiple disjoint paths which will together satisfy user bandwidth requirements, and then to reserve bandwidth and disperse data over these paths. During route discovery it prefers route which consists of links between physically closer nodes. To reduce the effect of rerouting due to broken paths, a fast rerouting is proposed with the aid of proactive monitoring of network topology where a rerouting is done before a path actually becomes unavailable. Based on signal strength observed on links connecting nodes, links, nodes and routes are classified accordingly as (from strongest to weakest) first, second and third. In the case the source has multiple routes, a route with stronger route class is given preference. A class three route, with nodes in the weakest link going apart from each other, is never selected. To discover a multipath, the source sends a route request packet which carries information regarding class of all intermediate links, path, QoS metric etc. Intermediate nodes append their address at the time of forwarding the route request packets and store information about all forwarded route request messages to enable them to forward only route request messages which come through some disjoint path as compared to already forwarded route request messages. The destination checks the node disjointness of the path while replying to a route request message along with the class of the route. Finally, the source selects the routes based on quality of the routes (route class and available bandwidth) and the user requirements. The source reserves bandwidth for data flow on the selected paths sending special reservation on the paths toward the destination. In this routing protocol, the state information maintained by each node regarding the topology of the network, for processing route request and supporting fast rerouting is very high. Further, proactive monitoring give rise to extra control overhead in the network.

2.6.6 QoS-aware Disjoint Multipath Routing Protocol (QDMR)

In [109], a Qos framework is proposed to support delay sensitive real-time applications in MANETs. It consists of two parts - (i) reliable and QoS-aware routing through disjoint paths discovery and maintenance and (ii) providing differentiated service to real-time and best-effort traffic through Diff-Serv. To compute QoS-aware disjoint multipath, source generates RREQ packet which carries ID of the one hop neighbor of the source, hop count and path information along with other information required for routing. Intermediate node does the admission control by maintaining and using admitted and admissible rates for QoS traffic like in the case of SWAN[3]. There is no admission control done for end-to-end delay during route discovery. To support delay guarantees to real-time applications, the paths with lower hop counts are preferred. Nodes forwarding a RREQ packet appends its ID in the forwarded RREQ packet. An intermediate node forwards the first copy of a RREQ packet after performing admission control. A duplicate RREQ will be re-broadcasted if it comes through a different first hop neighbor of the source with the previously forwarded ones and with hop count not larger than that of the previously forwarded ones. For every forwarded RREQ packet, bandwidth is temporarily reserved in the node. Destination after getting the first RREQ packet waits for a fixed time period to collect a number of RREQ packets. Then it computes two maximally disjoint paths and sends the RREP packet to the source separately for each of the selected paths. Source admits real-time packets into the primary path and secondary path is maintained through sending probe messages periodically. In the case, routing protocol detects only one path, traffic admission will be determined from the hop count of the path. During data transmission session using disjoint paths, if the primary path fails which is detected through RRER message, the real-time traffic will be switched to the alternate path and a route discovery is initiated to recover the failed path. During data communication session, route discovery for the alternate path is invoked if it fails.

2.6.7 Interference-aware Multipath Routing Protocol (IMRP)

IMRP [110] proposes an interfering-aware QoS multipath routing protocol for QoS-constraint multimedia and real-time applications in ad hoc wireless network. In general, when using multipath routing protocols for QoS, a sender node adopts multiple disjoint routing paths for data transmission based on pre-evaluated path bandwidth and applications QoS requirements. To minimize the path cost, paths with smaller hop counts are preferred. Selected paths using this policy will be very closed to each other and as a result expected total bandwidth of the paths will be less than the sum of their individual bandwidth due to path interference problems. Therefore, an interference-aware multipath routing protocol for QoS-constraint multimedia and real-time applications in ad hoc wireless networks. It is assumed that each mobile node can compute its available bandwidth and link stable time using some model. A source node starts route discovery by broadcasting a route discovery packet (RDP) in the network. The RDP carries QoS requirements, accumulated path stable time (which is minimum of already explored links), hop count and path information along with other information. Any intermediate node which have adequate bandwidth available, and the RDP is received through a link with link stable time greater than a threshold forwards a non duplicate RDP. While forwarding RDP all nodes reserve bandwidth for a pre-reservation time period which will be freed if no route reply packet is received within the time period. The pre-reserved bandwidth will be actually reserved in the node for path stable time while receiving route reply packet before the pre-reservation timer expires. After receiving a RDP, the destination reverses the path information carried by the RDP and sends back a route reply packet to the source through the original path discovered by the RDP. All nodes including the destination while forwarding route reply packet reserve the bandwidth specified in the packet. Once source receives a route reply packet, it will reserve the bandwidth recorded in the route reply packet and store the complete route information in the routing table. The path stable time of the selected paths are used for prediscovery of routing path before the actual path fails. Due to its RDP forwarding policy, it suffers from similar path diminution problem as in AOMDV, AODVM etc.

2.6.8 Mulipath QoS Routing Protocol of Supporting DiffServ (MQRD)

In this method [59], the Node-Disjoint Multipath Routing (NDMR) protocol in [58] is combined with DiffServ for supporting QoS in mobile ad hoc networks. NDMR is based on the modified version of AODV with path accumulation feature of DSR to discover multiple node-disjoint paths. To discover node-disjoint multiple paths between a source and a destination, the source generates a

RREQ packet and forward it to its neighbor through broadcasting. Each node forwarding the RREQ packet appends its address in the RREQ. When an intermediate node receives the first copy of a RREQ packet, it records the hop count of the RREQ packet as the shortest hop count for route discovery from the given source to the given destination. Any duplicate RREQ packet with higher hop count than the already recorded shortest hop count in reverse route table is discarded. Therefore, each node forwards RREQ multiple times based on hop count only. Once the first copy of RREQ packet reaches the destination, it selects the first node-disjoint path and saves the path information contained in the received RREQ packet in reverse route table. Destination generates a RREP packet which records the selected path in reverse order and forwards the RREP packet toward the source using the reverse route. For any duplicate copy of RREQ, destination checks the node-disjointness of the path with respect to already selected paths stored in reverse route table. Destination selects a maximum of three node-disjoint paths from the received RREQ packets. Because of RREQ forwarding policy and the path selection procedure adopted at the destination, a reasonable amount of path diminution problem exits in NDMR. In MQRD, both multipath routing using NDMR and DiffServ for supporting differentiated service work at the network layer as complementary techniques to provide end-to-end QoS for real-time applications. The source node will classify data packets and mark them with corresponding DSCP (DiffServ Code Point) in IP packet. All intermediate mobile nodes in the routing path from source to destination use DSCP in IP packet to determine scheduling and packet drop preference to provide differentiated service in the network. To further improve QoS performance of DiffServ, NDMR helps in reducing routing overhead and controlling congestion through load balancing over multiple node-disjoint paths.

2.7 Conclusion

In this chapter, we have briefly discussed about the mobile ad hoc networks and their characteristics. We outlined the requirements for quality of service support in MANETs. We found that research in QoS support in MANETs spans over all the layers of the protocol stack. Based on literature survey, we have presented some important approaches to provide QoS support in MANETs.

In the next chapter, we will discuss about the priority based QoS-aware MAC protocols designed to support quality of service at MAC layer in MANETs.

Chapter 3

QoS-aware MAC Protocols

3.1 Introduction

With the growing popularity of mobile ad hoc networks, it is reasonable to expect that users will demand some level of Quality of Service (QoS) from it, in terms of end-to-end delay, throughput, probability of packet loss, etc. However, the IEEE 802.11 Distributed Coordination Function (DCF) [25, 41], which is the most widely used Medium Access Control (MAC) protocol in MANETs, does not provide effective QoS support in MANETs due to its inherent problems [15, 49, 67, 75, 122]. Typically, random access MAC protocols in IEEE 802.11 focus primarily on solving the well-known hidden-terminal problem along with the basic contention-based channel access mechanism. While a solution to the hidden-terminal problem is paramount, a QoS-aware MAC protocol must also provide bounds on channel access delay for real-time flows. Fairness [44, 60] is also an important issue that affects the serviceability of the networks. Existing DCF based MAC protocols can not support delay sensitive real-time traffic due to the following drawbacks. (i) A station using DCF based MAC protocol might have to wait arbitrarily long time to send a frame, this results in delaying real-time traffic and (ii) Retransmitted frames are of little use in real-time traffic due to unexpected delay introduced by extra unbounded channel access delay. Here, for simplicity, we consider that traffic is classified into two priority classes - delay-sensitive real-time (RT) traffic and Best-effort (BE) traffic. Real-time traffic can be voice or video and Best-effort traffic is the normal data. Our goals in designing a QoS-aware MAC protocol are - (i) to reduce the channel access delay of real-time (higher priority) flows as compared to best-effort flows, (ii) reducing the end-to-end delay of different flows while forwarding traffic through the network (both real-time and best-effort) and to increase the overall throughput of the network and (iii) the proposed approach should be much simpler than IEEE 802.11e EDCA [42] and based on the legacy IEEE 802.11 DCF (which is the de facto standard for MAC in MANETs) and thus can be easily integrated into existing systems without much difficulty. These overall goals become difficult to achieve due to highly dynamic nature of MANETs.

In this chapter, we have proposed two different schemes for resolving MAC layer contention in the form of priority based variants of the DCF which aim at achieving an enhanced level of service differentiation among real-time and best-effort traffic. Further, we have included a priority-based queue management in both these schemes to resolve MAC layer contention within a node.

The first scheme proposed is a dynamic priority based MAC protocol, which aims at achieving an enhanced level of service differentiation among real-time and best-effort traffic in an end-to-end basis. In this scheme, nodes are assigned priorities based on their roles (relay or transmit) and type of traffic to handle. For each of the possible backoff stage, a non-overlapping range of contention window is assigned to each priority level where a higher priority level is assigned a contention window with lower range as compared to a lower priority level. This ensures that a node with higher priority has a better chance of accessing the channel earlier than a node with lower priority. Contention windows for two nodes with different priority may overlap only when they are in different backoff stage. Further, freezing of backoff counter during countdown periods of channel contention may lead to overlapping of backoff counter between lower priority and higher priority levels. We named this scheme as the Priority based QoS-aware MAC Protocol (PQAMP). This scheme is suitable for application areas where along with provisioning QoS for real-time traffic a minimum level of service should be assured to best-effort traffic.

The second scheme is rather a unique priority based version which aims at achieving a strict differentiation among real-time and best-effort traffic. In the scheme, to ensure that nodes of high priority always win the next transmission right when contending with nodes of low priority, a strict non-overlapping contention window is assigned to nodes with real-time and best-effort traffic irrespective of their backoff stage values. To further avoid overlapping of backoff timer during countdown period, freezing of backoff timer in low priority is replaced with a resetting of backoff timer mechanism. We named this scheme as the Strict Priority based QoS-aware MAC Protocol (SPQAMP) due to its strictness in maintaining non-overlapped backoff timer values between real-time and best-effort traffic. It is suitable for application areas where real-time traffic should be always assured better QoS without paying attention to service quality achievable for best-effort traffic.

The MAC level traffic differentiation provided by the proposed schemes can be exploited by higher layers as a powerful platform for QoS supports in MANETs. Here, we have not proposed any admission control mechanism along with the QoS-aware MAC protocols. It is obvious that delay and throughput experienced by real-time flows degrade when the number of active nodes in the interference range of a flow increases beyond a certain level. To ensure MAC level QoS, a priority-based admission control [32] should be incorporated so that the total load due to admitted traffic does not exceed the network capacity. Therefore, an admission control mechanism is indispensable for QoS assurance to avoid congested or overloaded network conditions. We have used Markov Chain model to analyze the performance of the proposed schemes in single hop scenarios, and the accuracy of the models are further validated using simulations. Our Markov chain models are based on adapting basic models proposed by Giuseppe Bianchi [8] and Yang Xiao [114]. Due to their simplicity in design, the proposed schemes can be implemented by modifying the standard DCF without much difficulty.

The organization of this chapter is as follows. In section 3.2, we discuss about the related work. A comparative study of some existing QoS-aware MAC protocols are presented in section. 3.3. The proposed priority-based QoS-aware MAC schemes namely PQAMP and SPQAMP are discussed in section 3.4 and section 3.5 respectively. In these sections, we evaluate the schemes by proposing analytical models and discuss the detailed simulation studies conducted for PQAMP and SPQAMP respectively. Finally, section 3.6 concludes the chapter.

3.2 Related Work

The main design goal of most of the MAC protocols for MANET is usually to solve medium contention, hidden/exposed terminal problems and to improve throughput. A QoS-aware MAC protocol must also provide a channel access mechanism that considers the QoS constrains of real-time flows [15, 84, 112]. Since it is not possible to provide upper bounds on channel access delay when using random access mechanisms, the goal is simply to reduce the channel access delay of real-time flows. To achieve any level of QoS at the medium access layer in MANETs, two levels of scheduling are required. These are packet-level scheduling and node-level scheduling. In packet-level scheduling, every node must define a scheduler and a queuing discipline that will determine which packet[1] should be sent during the next transmission time slot. This decision can be made by each node locally. On the other hand, node-level scheduling is a distributed problem in which all the nodes belonging to a neighborhood must cooperate to determine which nodes has channel access priority. The overall problem becomes more difficult due to dynamic topology and limited resources in the MANETs. The QoS-based MAC protocols can be broadly classified as either priority-based or reservation-based [75]. Some MAC protocols that have been proposed aiming at QoS support for real-time traffic rely on a centralized control, that are not viable in ad hoc networks [121]. IEEE 802.11e [20] which adds QoS features to IEEE 802.11 standard also rely on centralized control and hence belongs to the same category. Aad et al [1] proposed three different ways to modify the standard DCF to support service differentiation. These are based on differentiation of - Backoff increase function, Inter frame spacing (IFS) and maximum frame length. A variation of IEEE 802.11 protocol called RT-MAC (Real-Time Medium Access Control) [7] was proposed to support real-time traffic where backoff time depends on number of nodes in the system. Therefore, the number of nodes in the network should be known or at least estimated a priori. Using RT-MAC may result in wastage of bandwidth (due to larger contention window) in a lightly loaded network with large number of nodes. MACA/PR (Multiple Access Collision Avoidance with Piggyback Reservation) [65], which uses reservation information piggybacked on MAC frame has lower throughput due to periodic exchange of reservation tables among neighbors. The DFS (Distributed Fair Scheduling) [107] based MAC protocol does not consider delay bound traffic and has the problem in selection of mapping function. In the priority based protocols - DCF-PC (DCF with Priority Class) [27] and EDCA [22, 69], normal traffic suffers higher delays even when no real-time traffic is available. The Black Burst (BB) [100] contention scheme and its variants in ES-DCF (Elimination by Sieving DCF) [81, 82], DB-DCF (Deadline Bursting DCF) [81, 82] and proposal by Sheu et al [97] apply black-burst mechanism during priority classification period to separate the higher priority nodes from the lower priority ones. In these protocols, higher priority nodes content for free channel first and others have to wait until the transmission of higher priority nodes are over. All of them have the drawbacks of requirement for an extra channel and precise generation of BB pulses. In addition to wastage of channel capacity, lower priority traffic suffers due to longer IFS in [81, 82]. Veres et al proposed a virtual MAC and virtual source algorithms in [108] that operates parallel to the real MAC. A node can decide whether the channel can support a new flow or not by analyzing the delay experienced by the node with the help of running the DCF in virtual mode. Therefore, in their scheme, everything depends on estima-

[1]Although the terms 'frame' and 'packet' are used to mean protocol data units in MAC layer and network layer respectively, we will be using these two terms interchangeably throughout this paper to mean protocol data unit at MAC layer

tion model used for calculation. Ge et al [34] proposed a differentiated service based MAC protocol called Persistent Factor DCF (P-DCF), where each defined traffic class is associated with a persistent factor P (high-priority classes have smaller P). In P-DCF, backoff interval for a traffic class is a geometrically distributed random variable with parameter P, therefore, it gives a statistically proportional service and performs similar to the p-persistent based CSMA protocols. Kuang et al proposed a bidirectional multichannel MAC protocol [50] as a variant of IEEE 802.11 DCF which divides available bandwidth into one control channel and several data channels. Use of multiple channels require changes in NAV (Network Allocation Vector) and RTS/CTS frame formats to specify reservation for each channel. Bononi et al [10] proposed a MAC scheme called Differentiated Distributed Coordination Function (DDCF) to implement node differentiation based on distinct node roles (such as cluster head and leaf nodes), where nodes are assigned roles through a clustering scheme that runs on the upper layers. In DDCF, nodes belonging to a higher layer in the clustering structure are assigned higher channel access priority than a node in a lower layer. It is a best-effort based MAC mainly targeted at achieving higher channel efficiency. The modifications of DCF proposed by Ksentini et al [48], is for enhancing best-effort service in noisy channel. It aimed at adaptation of RTS/CTS mechanism to control updation of contention window depending on estimated channel condition.

Table 3.1: Comparison of QoS-aware MAC schemes

MAC scheme	Hidden Terminal problem	QoS technique used	Fairness	Strength	Weakness
RT-MAC [7]	not solved	dropping expired packets, piggybacked next backoff value	No	reduction in delay, missed deadlines and collisions as compared to IEEE 802.11	backoff time depends on no of nodes, so may result in wastage of b/w in light load
DCF-PC [27]	not solved	by shorter IFS (PIFS) and random backoff time for higher priority class	No	Better than 802.11 DCF in terms of throughput, delay and frame loss for higher priority	normal traffic suffers higher delays due to larger backoff time even when no real-time traffic is available
EDCA [22]	solved partially	By traffic differentiating with different access parameters for different ACs	No	reduction in delay and collisions, increase in throughput for higher priority classes as compared to IEEE 802.11	lower priority traffic suffers even in the absence of higher priority traffic
BB [100]	not solved	by shorter IFS and priority based BB period determined by the contention delay	within one hop	Provides bounded delay for real-time traffic	need of an extra channel and precise generation and detection of BB pulses
DFS [107]	not solved	by fair scheduling based on flow weight and packet size	short-term unfairness remains	Improved throughput as compared to IEEE 802.11	Delay bound of real-time traffic not considered, problem in proper selection of mapping function
MACA/PR [65]	solved partially	through piggybacked reservation and maintenance of reservation tables bandwidth is reserved for real-time traffic	No	lower end-to-end delay for real-time traffic as compared to IEEE 802.11	Lower throughput due to periodic exchange of reservation tables among neighbors
ES-DCF and DB-DCF [82]	solved partially	based on the deadlines and priority of packets, and a deterministic collision resolution algorithm	solved partially	They perform better for hard and soft real-time data packets respectively as compared to DCF-PC and IEEE 802.11	lower priority traffic suffers due to longer IFS and BB even in situations with very less real-time traffic

3.3 Comparison of Different Existing QoS-aware MAC protocols

Table 3.1 gives a brief comparison of the existing QoS-aware MAC protocols discussed above considering only the main parameters namely hidden terminal problems, QoS technique used, Fairness, Strength, and Weakness. From the comparative studies carried out, we can conclude that none of the MAC schemes are better suited for QoS provisioning in MANETs. This motivates us to design the two schemes that we will discuss in the following sections.

3.4 Priority based QoS-Aware MAC Protocol (PQAMP)

The proposed protocol called Priority based QoS-aware MAC Protocol (PQAMP) is based on classifying data frames into two QoS classes - Real-time and Best-effort. Accordingly, a node (in MAC layer) can be in any one of the corresponding state based on the type of data frame it is currently prepared to send. Further, in a multihop environment, based on whether a node is relaying/forwarding a data frame or transmitting a self generated data frame, it can have further derived states to improve QoS performance in an end-to-end basis. This leads to four possible states for a node in the MAC layer, where each of the state corresponds to a priority class. The detailed classification and their implications are as follows.

To provide QoS at MAC level, we require that a real-time data frame gets a higher channel access priority to that of a best-effort data frame in the network. We, therefore, provide higher channel access priority for frames belonging to real-time traffic with smaller contention window as compared to frames belonging to best-effort traffic. Further, we know that the dominating part of end-to-end delay in data transmission is the channel access delay over multiple hops that are incurred when transporting a data packet from source to the destination in a multihop ad hoc network. Every time a relay node in the path from source to destination forwards a frame, it has to compete for accessing the channel with its neighbor nodes, which leads to longer delay and hence degrades the throughput of the network [106]. Therefore, it is always better to provide higher priority to a node which has a frame to relay in comparison with a node with self generated frame for accessing the channel, because the relayed frame has already used up the resources of the network in terms of time and bandwidth. In our scheme, this is achieved by assigning a higher priority to relayed traffic, within a QoS class, to provide the relayed traffic higher channel access priority than that of the self-generated traffic within the same QoS class. The channel access priorities are realized by proper contention window selection method that will be discussed later. Besides the differentiated channel access mechanism at MAC layer, a proper queuing discipline or packet-level scheduling is implemented to improve overall QoS support through intra-node service differentiation.

In PQAMP, at any point in time, each node can be in one of the following five states: HIGHEST, HIGH, MEDIUM, LOW and NORMAL based on the type of the data frame it has to transmit or forward. Each of the state has a discrete value (with smaller value representing higher priority) which represents a *priority class* assigned to it. In our scheme, the values *0, 1, 2, 3 and 4* represent the priority classes assigned to the nodes in states HIGHEST, HIGH, MEDIUM, LOW and NORMAL respectively. The question is what *state* or *priority class* is assigned to which node? Based on the type of the frame (at the front of the interface queue) that a node is going to forward or send, it is is assigned an appropriate level of state among the four possible states - HIGHEST, HIGH, MEDIUM and LOW. The HIGHEST state is assigned to a node which has to relay a real-time data frame, since this type of data frame is delay sensitive and has already used up some of the network resources like time and bandwidth, it is required to expedite the forwarding of such frame until it reaches the destination. The HIGH state is assigned to a node which has a self generated real-time data frame to be transmitted. Because, the real-time data like voice and video are more important for timely delivery than that of non real-time data like ftp, http etc, a higher priority is assigned to this type of nodes. Similarly, MEDIUM state is assigned to a node which has a best-effort data frame to be forwarded; this is to expedite the forwarding of best-effort data frames as compared to freshly generated best-effort data frames at a node. This enhances end-to-end performance of best-effort traffic. The LOW state is assigned to a node which has to transmit a self generated best-effort data frame. Finally, NORMAL state is assigned to a node which has no activity and hence the default state of every node is set as NORMAL.

Nodes are assigned priorities dynamically based on their roles (sender, receiver, and forwarding node) and the type of traffic they have to forward or send. Initially, all nodes will start with the NORMAL state. When a node contents to access the channel, it chooses a random backoff timer value in the range $[W_{i,j} - W_j, W_{i,j} - 1]$, which is a function of two parameters namely backoff stage ($j = 0, \ldots, m$, where m is the retry limit) and priority class ($i = 0, 1, 2, 3$). $W_{i,j}$ and W_j are obtained as shown in Equations (3.1) and (3.2) respectively. In Equation (3.2), m' represents the backoff stage at which contention window for priority class i reaches the maximum value. It is worth mentioning that, this backoff timer selection procedure is different from the backoff timer selection procedure followed in IEEE 802.11 DCF. In our scheme, nodes with different states (or priority classes) are assigned distinct non-overlapping contention window ranges as depicted in Figure 3.1. It can be observed that the lowest range is assigned to the nodes which are in the HIGHEST state (or priority class 0) and the range increases as the priority goes down, until priority class 3 (or LOW state). This non-overlapping range of contention window ensures that a node in higher priority state always generate smaller backoff timer value as compared to a node in lower priority state, while attempting to access the medium with equal number of retrial attempts. Figure 3.1 shows changes of CW ranges with varying backoff stage (or retrial time) for four priority classes up to backoff stage 2, where we have assumed W_0 as 16. In PQAMP, we use smaller retry limit for transmission of frames belonging to real-time traffic, considering their tight deadline requirement. Therefore, we have chosen retry limit for priority classes 0 and 1 as m', whereas for priority classes 2 and 3, it is m (where $m \geq m'$).

$$W_{i,j} = (i+1) \times W_j \quad \text{for } j = 0, \ldots, m, i = 0, \ldots, 3 \tag{3.1}$$

$$W_j = \begin{cases} 2^j W_0 & \text{for } j = 0, \ldots, m' \\ 2^{m'} W_0 & \text{for } j = m'+1, \ldots, m \end{cases} \tag{3.2}$$

3.4.1 Rules for state transition for nodes at MAC layer

The following rules give a clear idea about how the nodes at MAC layer changes their states to achieve the priority based channel access.

- The default state is assumed to be NORMAL for each node in the beginning.

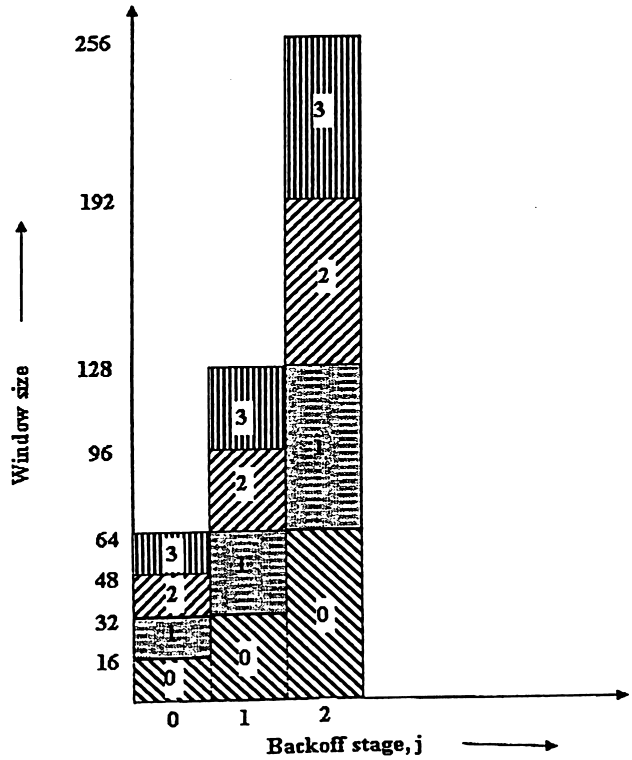


Figure 3.1: Range of backoff values for priority classes 0,1,2, and 3 (up to backoff stage 2) in PQAMP

- When a node in NORMAL state receives a real-time data frame to be forwarded to its neighbor node, the node's state is set to HIGHEST.
- When a node in NORMAL state receives a best-effort data frame to be forwarded to its neighbor node, the node's state is set to MEDIUM.
- A source node in NORMAL state, which has a real-time data frame to transmit changes its state from NORMAL to HIGH.
- A source node in NORMAL state, which has to transmit a best-effort data frame changes its state from NORMAL to LOW
- The state of a node after a successful transmission of data frame is set back to NORMAL.
- A node remains in the same state after an unsuccessful transmission of data frame until the retry limit for the particular state is reached.
- The state of a node is set back to NORMAL, once the retry limit for the particular state is reached and the transmission is unsuccessful.

3.4.2 Queue management and Packet scheduling in PQAMP

PQAMP uses priority based queue management and packet scheduling scheme for handling the incoming and the outgoing packets to and from the MAC layer of the nodes. A queue management scheme with features similar to RED (Random Early Detection) [30] provides intra-node service differentiation. Accordingly, when queue occupancy is more than a threshold but not full, only incoming high priority packets (real-time packets including control packets) are enqueued. If the queue occupancy is below the threshold, incoming packets are enqueued irrespective of their priorities. In the situation, when an incoming real-time packet finds the queue as full, the queue management scheme works to make room for the incoming real-time packet as follows. If there is at least one best-effort packet available in the queue, a best-effort packet is selected to drop; otherwise, the oldest real-time packet is selected to drop. Therefore, incorporating the queue management scheme improves the packet delivery ratio of real-time traffic, especially when there is some background traffic. The packet scheduler schedules the real-time packets before scheduling any best-effort packet for transmission over the wireless channel. Thus, average queuing delay of real-time packets is reduced. Further, fresh real-time packets are preferred for queuing, which makes sense in the networks supporting real-time communications.

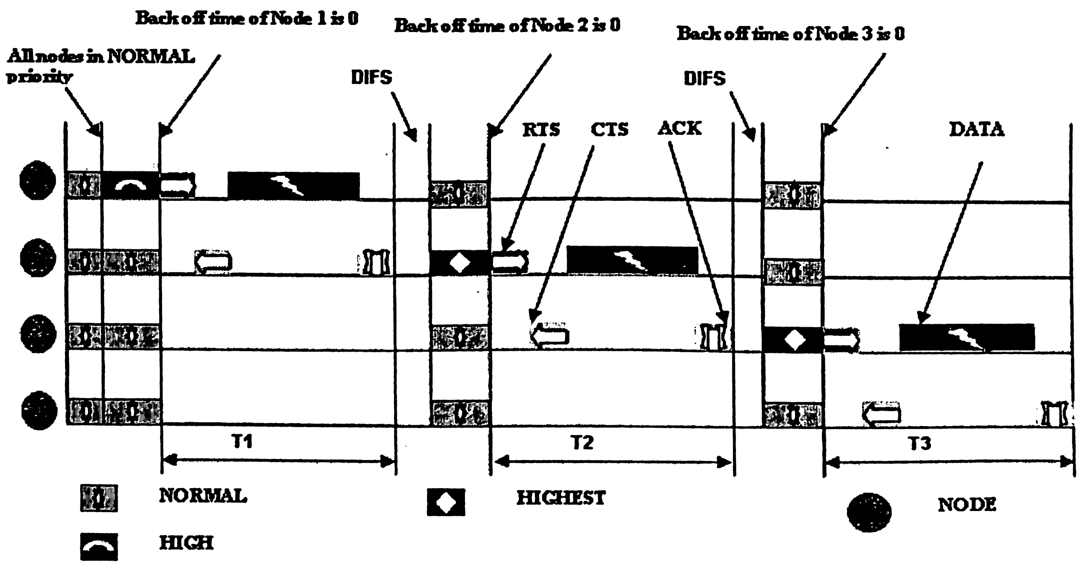


Figure 3.2: An Example showing state transition of nodes in PQAMP

3.4.3 An Example showing state transition of nodes during data transmission

Figure 3.2 shows the transitions of the node's state in transmitting real-time data packets in a MANET of linear topology consisting of four nodes. In this example, node 1 is the source node which wants to transmit real-time data packets to node 4, which is the destination node. We have assumed that (in the linear topology), nodes 2 and 3 are playing the role as relaying nodes for the route from node 1 to node 4. As shown in Figure 3.2, initially all nodes are in the default state i.e. NORMAL. Node 1 changes its state from NORMAL to HIGH before accessing the channel to gain higher channel access priority to transmit its real-time data frames. Accordingly, it performs the backoff procedure using the contention window related parameters for priority class 1. Once the backoff timer of node 1 counts down to 0, at the beginning of T1, it starts accessing the channel by initiating an RTS-CTS handshaking. After successful transmission of the data frame at the end of T1, node 1 changes its state back to NORMAL. Node 2, after receiving the data frame from node 1, prepares itself for forwarding the packet to node 3 and changes its MAC level state to HIGHEST before accessing the channel. Node 3 performs backoff process using the channel access parameters for priority class 0. At the beginning of T2, node 2 initiates an RTS-CTS handshaking for forwarding the data frame to node 3. At the end of T2, node 3 receives the data frame successfully and prepares itself to forward the data frame to node 4. Similar to node 2, node 3 changes the state (hence priority class) and forwards the data frame during T3 to node 4, which is the final destination of the data frame. Similar kind of state transition takes place in sending and forwarding best-effort data frames in the network. In a situation, where multiple nodes with similar state interfere with each other, the behaviour of the PQAMP will be like DCF, thus all of them will have equal chance. But, for a mixture of interfering real-time and best-effort traffic flows, PQAMP always gives higher preference to the real-time traffic flows due to its backoff timer selection procedure.

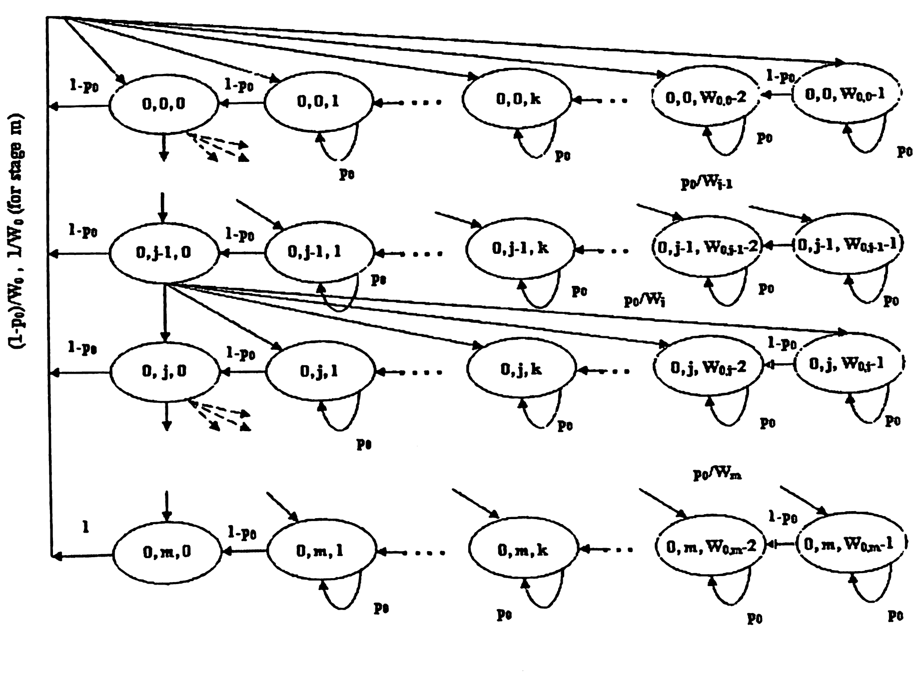


Figure 3.3: Markov Chain Model for the priority class 0 in PQAMP

3.4.4 Analytical Modeling of PQAMP

Based on Bianchi's basic model [8] and Xiao's model [114], an analytical model for multiple prioritized classes of traffic under high traffic condition is proposed (as shown in Figure 3.3 and Figure 3.4). Here, we have assumed that each station/node belongs to one and only one priority class (or

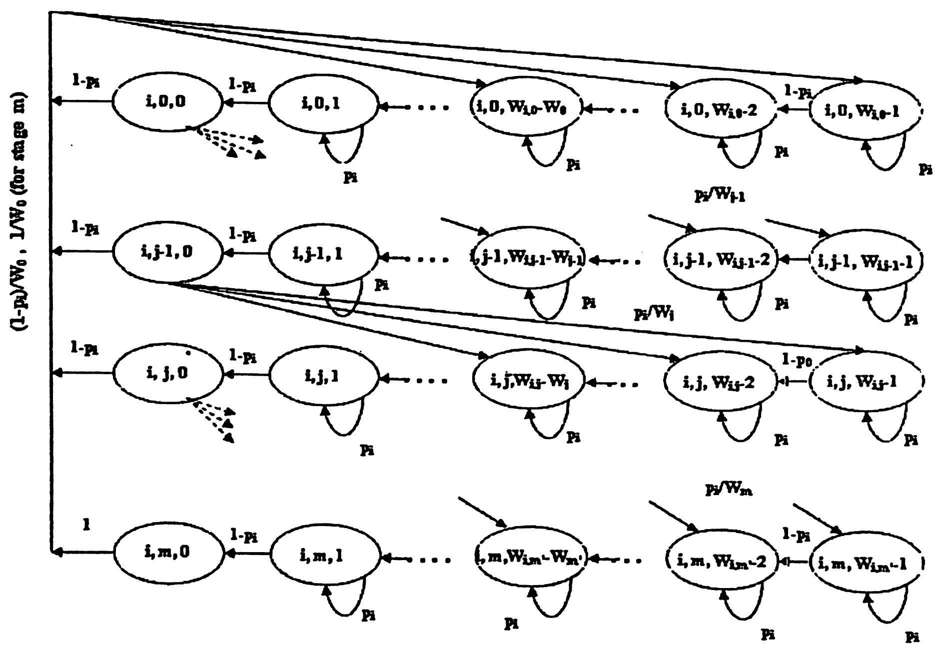


Figure 3.4: Markov Chain Model for the priority classes 1, 2, and 3 in PQAMP

state[2]) and always has frames to send. For a given station in the priority class i ($i = 0, \ldots, N-1$), $b(i,t)$ is defined as a random process representing the value of the backoff counter k at time t. Similarly, $s(i,t)$ is defined as the random process representing the backoff stage j ($j = 0, \ldots, m$) for the given station in the priority class i, at time t (where m is the retry limit). The value of the backoff counter $b(i,t)$ is uniformly chosen in the range $[W_{i,j} - W_j, W_{i,j} - 1]$ for i=0, 1, 2 and 3. The values $W_{i,j}$ and W_j are defined in Equation (3.1) and Equation (3.2) respectively. In Equation (3.2), W_0 represents the initial window size, i.e., similar to $CWmin$ of IEEE 802.11 DCF, m' is the backoff stage at which maximum contention window is reached and m is the retry limit. It can be seen that $W_{0,j}$ and W_j represents the same value.

Let p_i denotes the probability that a frame transmitted in a randomly chosen slot time by a station in priority class i experiences a collision. The bi-dimensional random process $\{s(i,t), b(i,t)\}$ is a discrete time Markov Chain as depicted in Figure 3.3 (for priority class 0) and Figure 3.4 (for priority classes 1 to 3) respectively. Therefore, the state of each node in the priority class i is described by $\{i, j, k\}$, where j and k represent backoff stage and backoff value respectively.

[2]we will be using the term *priority class* and *node state* interchangeably throughout this section

The non-null transition probabilities between states for priority class i are listed as follows.

(1) $P\{(i,j,k)|(i,j,k+1)\} = 1-p_i,$ for $0 \le k \le W_{i,j}-2$ and $0 \le j \le m$

(2) $P\{(i,0,k)|(i,j,0)\} = \dfrac{(1-p_i)}{W_0},$ for $W_{i,0}-W_0 \le k \le W_{i,0}-1$ and $0 \le j \le m$

(3) $P\{(i,j,k)|(i,j-1,0)\} = \dfrac{p_i}{W_j},$ for $W_{i,j}-W_j \le k \le W_{i,j}-1$ and $0 \le j \le m$

(4) $P\{(i,0,k)|(i,m,0)\} = \dfrac{1}{W_0},$ for $W_{i,0}-W_0 \le k \le W_{i,0}-1$

(5) $P\{(i,j,k)|(i,j,k)\} = p_i,$ for $1 \le k \le W_{i,j}-1$

The transition probabilities listed above account, respectively, for

1. the decrements of backoff timer at the beginning of each slot while detecting an idle channel;

2. the resetting of backoff stage, that is, after a successful transmission, the backoff timer of the new frame starts from the backoff stage 0;

3. the collision, that is, an unsuccessful transmission due to collision makes the backoff stage increase, provided it has not already reached the retry limit;

4. the last backoff stage, that is, at the maximum backoff stage (corresponding to retry limit), the contention window is reset irrespective of whether the transmission is successful or not;

5. the busy channel, that is, detecting busy channel during a backoff stage, the backoff timer is frozen.

Let $b_{i,j,k}$ be the stationary distribution of the given Markov chain.

$$b_{i,j,k} = \lim_{t\to\infty} (Pr[s(i,t)=j, b(i,t)=k])$$

In steady state, we can derive the following through chain regularities.

$$b_{i,j-1,0} * p_i = b_{i,j,0} \quad \text{where} \quad 1 \le j \le m \tag{3.3}$$

$$\text{We have} \quad b_{i,j,0} = p_i^j b_{i,0,0} \quad \text{where} \quad 0 \le j \le m \tag{3.4}$$

$$b_{i,j,k} = \begin{cases} \frac{1}{1-p_i} \cdot b_{i,j,0} & \text{for} \quad 0 \le k \le W_{i,j}-W_j-1 \\ \frac{W_{i,j}-k}{W_j} \cdot \frac{1}{1-p_i} \cdot b_{i,j,0} & \text{for} \quad W_{i,j}-W_j \le k \le W_{i,j}-1 \end{cases} \tag{3.5}$$

Now, using the normalization condition for stationary distribution, we get

$$\sum_{j=0}^{m} \sum_{k=0}^{W_{i,j}-1} b_{i,j,k} = 1 \tag{3.6}$$

From above equation,

$$
\begin{aligned}
1 &= \sum_{j=0}^{m}\sum_{k=0}^{W_{i,j}-1} b_{i,j,k} \\
&= \sum_{j=0}^{m}\left[\sum_{k=0}^{W_{i,j}-(W_j-1)} b_{i,j,k} + \sum_{k=W_{i,j}-W_j}^{W_{i,j}-1} b_{i,j,k}\right] \\
&= \sum_{j=0}^{m}\left[\frac{W_{i,j}-W_j}{1-p_i}\cdot b_{i,j,0} + \sum_{k=W_{i,j}-W_j}^{W_{i,j}-1}\frac{W_{i,j}-k}{W_j}\cdot\frac{1}{1-p_i}\cdot b_{i,j,0}\right] \\
&= \sum_{j=0}^{m}\left[\frac{W_{i,j}-W_j}{1-p_i}\cdot b_{i,j,0} + \frac{b_{i,j,0}}{1-p_i}\cdot\sum_{k=W_{i,j}-W_j}^{W_{i,j}-1}\frac{W_{i,j}-k}{W_j}\right] \\
&= \sum_{j=0}^{m}\left[\frac{iW_j}{(1-p_i)}\cdot b_{i,j,0} + \frac{b_{i,j,0}}{1-p_i}\cdot\frac{W_j+1}{2}\right] \\
&= \sum_{j=0}^{m}\left[\left(\frac{b_{i,j,0}}{1-p_i}\right)\left(\frac{(2i+1)W_j+1}{2}\right)\right] \\
&= \sum_{j=0}^{m'}\left[\left(\frac{b_{i,j,0}}{1-p_i}\right)\left(\frac{(2i+1)W_j+1}{2}\right)\right] + \sum_{j=m'+1}^{m}\left[\left(\frac{b_{i,j,0}}{1-p_i}\right)\left(\frac{(2i+1)W_j+1}{2}\right)\right] \\
&= \sum_{j=0}^{m'}\left[\left(\frac{p_i^j b_{i,0,0}}{1-p_i}\right)\left(\frac{(2i+1)2^jW_0+1}{2}\right)\right] + \sum_{j=m'+1}^{m}\left[\left(\frac{p_i^j b_{i,0,0}}{1-p_i}\right)\left(\frac{(2i+1)2^{m'}W_0+1}{2}\right)\right] \\
&= \frac{b_{i,0,0}}{2(1-p_i)}\left[(2i+1)W_0\frac{1-(2p_i)^{m'+1}}{1-2p_i} + \frac{(1-p_i^{m'+1})}{1-p_i}\right] \\
&+ \frac{b_{i,0,0}}{2(1-p_i)}\left[((2i+1)W_0 2^{m'}+1)\cdot\frac{(p_i^{m'+1}-p_i^{m+1})}{1-p_i}\right] \\
&= \frac{b_{i,0,0}}{2(1-p_i)}\left[\frac{(2i+1)W_0(1-(2p_i)^{m'+1})(1-p_i)+(1-p_i^{m'+1})(1-2p_i)}{(1-2p_i)(1-p_i)}\right] \\
&+ \frac{b_{i,0,0}}{2(1-p_i)}\left[\frac{(2i+1)W_0 2^{m'}(p_i^{m'+1}-p_i^{m+1})+(p_i^{m'+1}-p_i^{m+1})}{(1-p_i)}\right] \\
&= \frac{(2i+1)W_0 b_{i,0,0}}{2(1-p_i)^2(1-2p_i)}\left[(1-(2p_i)^{m'+1})(1-p_i)+2^{m'}(p_i^{m'+1}-p_i^{m+1})(1-2p_i)\right] \\
&+ \frac{b_{i,0,0}}{2(1-p_i)^2(1-2p_i)}\left[(1-p_i^{m+1})(1-2p_i)\right]
\end{aligned}
$$

$$
b_{i,0,0} = \frac{2(1-p_i)^2(1-2p_i)}{(2i+1)W_0\left[(1-(2p_i)^{m'+1})(1-p_i)+2^{m'}(p_i^{m'+1}-p_i^{m+1})(1-2p_i)\right]+(1-p_i^{m+1})(1-2p_i)} \qquad (m>m') \qquad (3.7)
$$

$$
= \frac{2(1-p_i)^2(1-2p_i)}{(2i+1)W_0(1-(2p_i)^{m+1})(1-p_i)+(1-p_i^{m+1})(1-2p_i)} \qquad (m=m') \qquad (3.8)
$$

Let τ_i be the probability that a station in a priority class i transmits during a generic slot time. Therefore, from the Markov chain we have τ_i as follows.

$$\tau_i = \sum_{j=0}^{m} b_{i,j,0} = \sum_{j=0}^{m} p_i^j b_{i,0,0} = b_{i,0,0}\frac{1-p_i^{m+1}}{1-p_i} \tag{3.9}$$

Therefore, from Equation (3.7), Equation (3.8) and Equation (3.9), we get τ_i as given in Equation (3.10) and Equation (3.11).

$$\tau_i = \frac{2(1-p_i)(1-2p_i)(1-p_i^{m+1})}{(2i+1)W_0\left[(1-(2p_i)^{m'+1})(1-p_i)+2^{m'}(p_i^{m'+1}-p_i^{m+1})(1-2p_i)\right]+(1-p_i^{m+1})(1-2p_i)} \quad (m > m') \tag{3.10}$$

$$= \frac{2(1-p_i)(1-2p_i)(1-p_i^{m+1})}{(2i+1)W_0(1-(2p_i)^{m+1})(1-p_i)+(1-p_i^{m+1})(1-2p_i)} \quad (m = m') \tag{3.11}$$

Let n_i denotes the number of stations in the priority class i ($0 \le i \le (N-1)$, N is the number of priority classes). A transmitted frame of priority class i collides with another frame, when one more station also transmits during the same slot time. The probability p_i that a frame transmitted in a generic slot time by a station in priority class i experiences a collision is given by Equation (3.12).

$$p_i = 1 - \left[\prod_{h=0}^{i-1}(1-\tau_h)^{n_h}\right](1-\tau_i)^{n_i-1}\left[\prod_{h=i+1}^{N-1}(1-\tau_h)^{n_h}\right] \tag{3.12}$$

Equation (3.10), Equation (3.11), and Equation (3.12) together represent a nonlinear system in eight unknowns τ_0, τ_1, τ_2, τ_3, p_0, p_1, p_2, and p_3 which have been solved by using numerical methods with the parameters set used for the proposed PQAMP scheme. This gave us the values of p_i and τ_i for each priority class i. Let P_b denotes the probability that the channel is busy, which happens when at least one station transmits in a given time slot(shown in Equation (3.13)).

$$P_b = 1 - \prod_{i=0}^{N-1}(1-\tau_i)^{n_i} \tag{3.13}$$

Saturation Throughput

To compute the saturation throughput S_i for a priority class i, we have defined the following parameters. $P_{s,i}$: probability that a successful transmission occurs in a given time slot for the priority class i, P_s: probability that a successful transmission occurs in a given slot, P_c: probability that the channel is busy with collision(means channel is neither idle nor successful). We have, $P_{s,i}$, P_s and P_c as defined in Equation (3.14), Equation (3.15) and Equation (3.16) respectively.

$$P_{s,i} = n_i\tau_i(1-\tau_i)^{n_i-1}\prod_{h=0,h\neq i}^{N-1}(1-\tau_h)^{n_h}. \tag{3.14}$$

$$P_s = \sum_{i=0}^{N-1} P_{s,i} = \sum_{i=0}^{N-1}\frac{n_i\tau_i}{1-\tau_i}(1-P_b) = (1-P_b)\sum_{i=0}^{N-1}\frac{n_i\tau_i}{1-\tau_i} \tag{3.15}$$

$$\begin{aligned} P_c &= 1-(1-P_b)-P_s \\ &= P_b - P_s \end{aligned} \tag{3.16}$$

Now, the throughput S_i can be calculated as,

$$\begin{aligned} S_i &= \frac{E\,[\text{Payload information transmitted in a slot time for class } i]}{E\,[\text{length of a slot time}]} \\ &= \frac{P_{s,i} \times T_{E[P]}}{T_{avg}} \\ &= \frac{P_{s,i} \times T_{E[P]}}{(1-P_b)\sigma + P_s T_{succ} + P_c T_{col}} \end{aligned} \tag{3.17}$$

where $T_{E[P]}$ is the time required to transmit average payload size $E[P]$ (determined by the data rate), and thus $P_{s,i} \times T_{E[P]}$ is the amount of time spent in a generic slot time in transmitting an average payload information successfully. T_{avg} is the average length of a slot time. σ, T_{succ} and T_{col} are the duration of a slot corresponding to an idle slot, a successful transmission and a collision, respectively. Note that idle time slot σ is specific to the physical layer of IEEE 802.11, e.g., $20\mu s$ for Direct Sequence Spread Spectrum (DSSS). The specific values of T_{succ} and T_{col} for RTS-CTS based channel access mechanism are as follows.

$$\begin{cases} T_{succ} = T_{RTS} + T_{CTS} + T_{DATA} + T_{ACK} + 3 * SIFS + DIFS \\ T_{col} = T_{RTS} + T_{CTS} + SIFS + DIFS \end{cases} \tag{3.18}$$

Where T_{RTS}, T_{CTS}, T_{DATA} and T_{ACK} are the duration in time corresponding to transmission of RTS, CTS, DATA and ACK frames in the MAC standard including the overhead due to PHY and MAC headers. SIFS and DIFS represent the time durations used for the corresponding Inter-frame Spaces.

Saturation Delay

Saturation delay, D_i for priority class i is the average delay under the saturation condition which includes medium access delay (due to backoff, collision), frame transmission time, and inter-frame spaces (such as SIFS, DIFS etc.). The average backoff delay depends on the value of a station's backoff counter and the duration when the counter freezes due to other stations' transmissions. Let X_i denotes the random variable representing the total number of backoff slots, which a frame in class i encounters without considering the case when the counter freezes. The probability that the frame is successfully transmitted after the jth retry is given by $p_i^j(1-p_i)/P_{i,succ}$, which uses conditional probability on a successful transmission with probability $P_{i,succ}$. Here, $P_{i,succ}$ is the probability that a frame in class i is transmitted successfully within the retry limit m and is equal to $1-p_i^{m+1}$. It is important to note that $P_{i,succ}$ is different from $P_{s,i}$, which is defined in Equation (3.14). Let $BO_{i,j}$ denotes the average number of backoff slots after the jth retry for a frame in class i, which is defined in Equation (3.19).

$$BO_{i,j} = \sum_{h=0}^{j} W_h(i + \frac{1}{2}) - 1 \tag{3.19}$$

Therefore, the (expected) total number of idle slots, $E(X_i)$ that the frame encounters during backoff stages is given as

$$E(X_i) = \sum_{j=0}^{m} \frac{p_i^j(1-p_i)}{1-p_i^{m+1}} BO_{i,j} = \sum_{j=0}^{m} \frac{p_i^j(1-p_i)}{1-p_i^{m+1}} \sum_{h=0}^{j} W_h(i+\frac{1}{2}) - 1 \tag{3.20}$$

Let B_i denotes the random variable representing the total number of slots when the counter freezes for class i, which a frame encounters during its backoff periods. The portion of idle slots which is used to decrease $E(X_i)$ is $(1-p_i)$. Therefore, the (expected) total number of busy slots, $E(B_i)$ that the frame encounters during backoff stages is given in Equation (3.21). Similarly, let $E(N_{i,retry})$ denotes the average number of retries for a frame in the priority class i.

$$E(B_i) = \frac{E(X_i)}{(1-p_i)} p_i \tag{3.21}$$

$$E(N_{i,retry}) = \sum_{j=0}^{m} \frac{jp_i^j(1-p_i)}{1-p_i^{m+1}} = \frac{(1-p_i^{m+1}) - mp_i^{m+1}(1-p_i)}{(1-p_i)(1-p_i^{m+1})} \tag{3.22}$$

Therefore, the expected saturation delay $E(D_i)$ for a priority class i is given as

$$E(D_i) = E(X_i)\sigma + E(B_i)\left[\frac{P_s}{P_b}T_{succ} + \frac{P_b - P_s}{P_b}T_{col}\right] + E(N_{i,retry})(T_{col} + T_{oh}) + T_{succ} \tag{3.23}$$

Where σ, $[(P_s/P_b)T_{succ} + \{(P_b - P_s)/P_b\}T_{col}]$, $(T_{col} + T_{oh})$ and T_{succ} are the average slot lengths for an idle slot, a busy slot, a failed transmission slot and a successful transmission respectively during backoff periods. Here, T_{oh} denotes the time that a station has to wait before sensing the channel again, when its frame transmission collides. T_{oh} for RTS-CTS based channel access method is defined in Equation (3.24).

$$T_{oh} = SIFS + T_{ACK} \tag{3.24}$$

Table 3.2: IEEE 802.11 parameters used in Calculation

Payload (bytes)	512
DATA	Payload + MAC header + PHY header
RTS	20 bytes + PHY header
CTS	20 bytes + PHY header
ACK	20 bytes + PHY header
MAC header	28 bytes
PHY header	192 μs
Basic Rate	1 Mbps
Data Rate	2 Mbps
Slot time, σ	20 μs
SIFS	10 μs
DIFS	50 μs

3.4.5 Analytical Results

In this section, we will present the analytical results in single-hop networks to show that the real-time traffic get better service in terms of higher throughput and reduced delay in comparison with the best-effort traffic. For performance analysis, we have used RTS-CTS based channel access mechanism in IEEE 802.11 DCF under the Direct Sequence Spread Spectrum (DSSS) physical layer with the values of the parameters listed in Table 3.2. We have adopted the following parameters in PQAMP for analytical results. These are: m=5 and m=7 for real-time and best-effort traffic respectively; m'=5; W_0=16.

From Table 3.3, it can be observed that the saturation throughput for the priority class 0 is the highest among all the priority classes and the saturation throughput decreases for the priority classes 1, 2 and 3 in that order. Moreover, it can also be observed that, in PQAMP, the total throughput is slightly higher than that of the DCF for total number of nodes less than or equal to 16 (4 nodes per priority class). This means that in lower to moderate traffic loads, besides supporting traffic differentiation, the proposed scheme achieves higher overall throughput as compared to DCF. In higher loads due to excessive collisions, the total throughput decreases as compared to DCF. This is mainly due to lower values of retry limit and $CWmax$ used for higher priority classes in PQAMP as compared to DCF. In general, overloaded situations are avoided through admission control to support real-time traffic with strict delay requirements. Therefore, very high load conditions are generally not encountered in network with QoS support. Table 3.4 shows that PQAMP ensures a very good level of service differentiation in terms of saturation delay. Saturation delay for all the priority classes increase with the number of nodes. Saturation delays for priority class 0 and priority class 1 are less as compared to standard DCF up to the number of nodes equal to 20 (5 per class). Because, in multihop ad hoc networks the intermediate relay nodes that forward real-time traffic use priority class 0 and the sender nodes use priority class 1 for transmission of real-time data frames, this will ensure lower delay for real-time traffic in multihop environment as compared with DCF. Further, it can be seen that the packet dropping probabilities of PQAMP is less for higher priority classes (0 and 1) for number of nodes less than or equal to 16 (as seen in Table 3.5). This is attributed to the lower values of retry limit and $CWmax$ used for higher priority classes as discussed earlier. Because, real-time traffic like voice and video can tolerate packet lost as compared to best-effort traffic, this higher value of packet dropping probabilities at higher load scenarios are acceptable. Thus, from the analytical results, we can judge the suitability of PQAMP to provide QoS in terms of throughput and delay as compared to IEEE 802.11 DCF.

3.4.6 Model Validation

In this section, we have conducted simulations on a single hop network of 60 nodes using NS-2 [80] to validate the proposed analytical model. We have adopted the following parameters in PQAMP for both simulation and analytical results. These are: m=5 and m=7 for real-time and best-effort traffic respectively; m'=5; and W_0=16. Further, for comparison purpose, we have also shown the simulation and analytical results of IEEE 802.11 DCF. The analytical model used for IEEE 802.11 DCF is similar to the model used for priority class 0 of PQAMP (as shown in Figure 3.3), but with parameters from IEEE 802.11 DCF standard. Frame sizes for both real-time and best-effort traffic are fixed at 1024 bytes. The simulation time is set as 5000 sec. The other simulation parameters are

Table 3.3: Analytical Result for Different Priority Classes (Saturation Throughput): PQAMP vs DCF

Nodes	PQAMP					IEEE 802.11 DCF	
(per PC)	PC 0	PC 1	PC 2	PC 3	Total	Per Class	Total
1	0.4783	0.1284	0.0742	0.0522	0.7331	0.1829	0.7315
2	0.4711	0.1317	0.0760	0.0536	0.7324	0.1809	0.7234
3	0.4661	0.1339	0.0763	0.0539	0.7302	0.1794	0.7175
4	0.4627	0.1356	0.0760	0.0537	0.7280	0.1782	0.7128
5	0.4343	0.1358	0.0803	0.0571	0.7075	0.1772	0.7088
6	0.4200	0.1330	0.0789	0.0562	0.6881	0.1763	0.7052
7	0.4068	0.1294	0.0769	0.0547	0.6678	0.1755	0.7019
8	0.3932	0.1254	0.0745	0.0530	0.6461	0.1747	0.6988
9	0.3786	0.1208	0.0718	0.0511	0.6223	0.1740	0.6959
10	0.3629	0.1159	0.0689	0.0491	0.5968	0.1733	0.6931

PC: Priority Class

as shown in Table 3.8. The simulation results are averages over 10 runs.

Table 3.6 and Table 3.7 compare the analytical and simulation results for both PQAMP and IEEE 802.11 DCF. From these tables, we can observe that the analytical models of both PQAMP and IEEE 802.11 DCF are accurate in capturing the trends and service differentiation provided by PQAMP in single hop networks as compared to IEEE 802.11 DCF. From Table 3.6, we can observe a maximum of 2.34% and 1.46% deviation between analytical and simulation results in saturation throughput for PQAMP and IEEE 802.11 DCF respectively. Similarly, we can see a maximum of 4.21 msec and 2.79 msec deviations between analytical and simulation results in saturation delay for PQAMP and IEEE 802.11 DCF respectively. For best-effort traffic the deviations between analytical and simulation results are comparatively less as compared to real-time traffic. From both simulation and analytical results, we can observe that PQAMP improves saturation throughput for real-time traffic by 40-42% as compared to DCF. Similarly, PQAMP reduces saturation delay for real-time traffic by 30-150%. It can be seen that both PQAMP and DCF have almost same average saturation throughput.

3.4.7 Simulation Results and Analysis

After evaluation of PQAMP with analytical results (in section 3.4.5) and then validating them with simulations (in section 3.4.6) in single hop environments under saturation condition, we have conducted simulation studies to further validate analytical results for different scenarios (including both single hop and multihop scenarios) under non-saturated conditions. As noted earlier, in networks with QoS support, overloaded or saturated situations are avoided through admission control. Therefore, in most of the cases, networks with QoS support run under non-saturated condition. This necessitates performance evaluation of PQAMP (in comparison with IEEE 802.11 DCF) using simulations in single hop as well as multihop networks under non-saturated conditions. We used the similar parameters for PQAMP as used in analytical results in section 3.4.5 and we added the required changes in IEEE 802.11 DCF of NS-2 [80] to implement the enhanced backoff procedure in

Table 3.4: Analytical Result for Different Priority Classes (Saturation Delay in msec): PQAMP vs DCF

Nodes (per PC)	PQAMP				IEEE 802.11 DCF
	PC 0	PC 1	PC 2	PC 3	
1	9.69	27.56	55.93	79.12	27.88
2	18.30	62.54	107.83	152.65	67.06
3	26.89	91.08	159.33	225.42	112.96
4	35.45	118.78	211.74	299.33	163.06
5	143.83	209.64	1454.45	2083.35	216.36
6	255.66	490.72	3773.47	5391.03	272.08
7	531.40	1930.01	7078.48	10081.41	330.04
8	785.11	2779.92	11260.14	15998.02	389.89
9	1062.85	3733.19	16213.61	22994.48	450.95
10	1203.80	4791.86	21883.81	30995.67	514.23

PC: Priority Class

PQAMP. Further, we have modified the priqueue queuing discipline in queuing model to give higher preference to real-time traffic as compared to best-effort traffic. We compared the performance of PQAMP with IEEE 802.11 DCF (with priority queue which gives preference to routing packets). Simulations are performed on four different topologies as follows:

- *Topology 1*: A single hop topology
- *Topology 2*: A chain topology with varying hop counts
- *Topology 3*: A Grid topology of size 6×6 (36 nodes)
- *Topology 4*: A Random topology of 50 nodes in a flat rectangular area of size 1000m×500m with random waypoint mobility model with maximum node speed of 10 m/s.

Performance parameters

For performance evaluation we have considered the following parameters -

- Throughput (TP): Throughput means the average number of successfully delivered bytes at the destinations per second. It is an important metric to provide minimum level of service in a network.
- Delay (D): This gives the average time delay that a data packet has encountered from the time it was sent by the source to the time it was delivered at the destination. Delay is a very important parameter for delay sensitive real-time traffic.
- Packet Dropped (PD): This is the count on number of real-time packets dropped during the communication sessions. In the situations where other network parameters are kept fixed for carrying out the experiments, this parameter provides the basis for implication of the number of packet dropped due to the selection of the underlying MAC layer protocol. This measures

Table 3.5: Analytical Result for Different Priority Classes (Packet Dropping Probability): PQAMP vs DCF

Nodes (per PC)	PQAMP				IEEE 802.11 DCF
	PC 0	PC 1	PC 2	PC 3	
1	0.00000004	0.00000504	0.00000017	0.00000021	0.00000844
2	0.00001942	0.00009121	0.00000550	0.00000617	0.00024367
3	0.00013303	0.00032640	0.00002718	0.00002924	0.00092359
4	0.00037954	0.00070734	0.00007268	0.00007660	0.00203126
5	0.01378110	0.01992913	0.00590167	0.00612111	0.00352143
6	0.04959659	0.06412192	0.02733374	0.02806212	0.00534990
7	0.10668140	0.12879780	0.06815676	0.06950581	0.00749254
8	0.18063611	0.20820413	0.12785627	0.12977754	0.00992246
9	0.26508303	0.29559954	0.20236568	0.20470852	0.01259841
10	0.35358423	0.38479677	0.28589878	0.28847561	0.01555556

PC: Priority Class

Table 3.6: Analytical Results Vs. Simulation Results for Saturation Throughput (PQAMP and DCF)

Traffic Class	Protocol	S/A	$N_{RT} = N_{BE} = 1$	$N_{RT} = N_{BE} = 5$	$N_{RT} = N_{BE} = 10$	$N_{RT} = N_{BE} = 15$
Real-Time	PQAMP	A	0.5037	0.5191	0.5181	0.5047
		S	0.4803	0.5014	0.4983	0.4887
	DCF	A	0.3640	0.3637	0.3593	0.3558
		S	0.3496	0.3558	0.3511	0.3502
Best-Effort	PQAMP	A	0.2031	0.2116	0.2101	0.2110
		S	0.2117	0.2156	0.2083	0.1997
	DCF	A	0.3640	0.3637	0.3593	0.3558
		S	0.3496	0.3558	0.3511	0.3502

S: Simulation; A: Analytical; RT: Real-Time; BE: Best-Effort; N_{RT}: Number of RT traffic;N_{BE}: Number of BE traffic

the number of data frames dropped due to successive collisions at the MAC layer after retransmission for the retry limit number of times.

- Packet Delivery Ratio (PDR): This is the ratio of number of data packets received at the destination to the actual number of data packets sent by the source. This measures how effectively the source generated data packets are delivered at the destination. Lower value of PDR means higher amount of lost of data packets during their journey from source to destination. It is related to the packet dropping probabilities used in analytical results.

For simulation studies, Real-time data traffic are generated using UDP based constant bit rate (CBR) traffic sources with 40/80 Kbps data rate. On the other hand, best-effort traffic are represented by TCP based FTP sessions with infinite data.

Table 3.7: Analytical Results Vs. Simulation Results for Saturation Delay (in msec) (PQAMP and DCF)

Traffic Class	Protocol	S/A	$N_{RT} = N_{BE}$ $= 1$	$N_{RT} = N_{BE}$ $= 5$	$N_{RT} = N_{BE}$ $= 10$	$N_{RT} = N_{BE}$ $= 15$
Real-Time	PQAMP	A	9.20	40.35	78.97	305.61
		S	10.31	44.12	78.83	307.28
	DCF	A	12.51	79.19	200.73	351.84
		S	13.27	80.73	203.42	351.07
Best-Effort	PQAMP	A	21.18	97.59	194.15	936.72
		S	23.06	100.34	195.76	939.04
	DCF	A	12.51	79.19	200.73	351.84
		S	13.27	80.73	203.42	351.07

S: Simulation; A: Analytical; RT: Real-Time; BE: Best-Effort; N_{RT}: Number of RT traffic;N_{BE}: Number of BE traffic

Table 3.8: Simulation Parameters in NS-2

Parameters	Values
Propagation Model	Two Ray Rayleigh Fading
Data Rate	2 Mbps
Basic Rate	1 Mbps
Transmission range	250m
Carrier sensing range	250m
TCP Packet size	512 Bytes
UDP Packet size	512 Bytes
MAC protocols	PQAMP, SPQAMP, 802.11 DCF

A Single Hop Topology

We selected a single hop topology of 60 nodes randomly placed within a rectangular area of size 120m×120m. To avoid extra control overhead due to routing protocol, we used NOAH (No Ad Hoc routing) from [79] that supports direct communication between wireless nodes without generating any routing related packets. We generated a number of real-time and best-effort connections between any randomly selected pair of nodes. The other parameters are set as shown in Table 3.8.

For performance analysis, we have varied the number of real-time and best-effort connections from 1 to 10 and measured the three parameters namely *Delay*, *Throughput* and *Packet Dropped* for real-time traffic. Similarly, we have also measured the *Throughput* and *Delay* for the best-effort traffic.

It can be seen from Table 3.9 and Table 3.10 that PQAMP provides a very good level of service differentiation and provide better service for real-time traffic in terms of *Delay*, *Throughput* and *Packet Dropped* as compared with DCF. With the increase in both real-time and best-effort traffic, PQAMP performs better with respect to service provided to real-time traffic without substantial degradation of best-effort service quality. Initially, at number of connections less than or equal to 2, delay for real-time traffic in PQAMP is slightly more than that in DCF. With the increase in number of traffic, PQAMP is able to consistently maintain a smaller delay (less than 1 sec) for real-time traffic as compared to DCF. We can observe an exponential trend in increase of delay for real-time traffic in DCF with increase of traffic load and reaches more than 3.5 secs at 10 connections each (as shown in Table 3.9). At the load of 7 connections each, PQAMP performs 15.48 times better than DCF in terms of delay for real-time traffic. Similarly, we can observe a consistently better performance of PQAMP in terms of *throughput* and *packet dropped* for real-time traffic. Although at traffic loads ranging from 1 to 4, *throughput* for real-time traffic in PQAMP and DCF are almost similar, from traffic load 5 onward PQAMP performs significantly better than DCF and at the highest load of 10 connections, PQAMP's performs almost 45% better than that of DCF. It is interesting to observe that, PQAMP even maintains a very low *packet dropped* level from light to high load conditions (as shown in Table 3.9) as compared to DCF. PQAMP performs best at 7 connections with 71 times better result as compared to DCF. Table 3.10 shows that PQAMP can maintain a good level of QoS for Real-time traffic without degrading the performance of best-effort service in most of the load conditions. It can be observed that at load conditions up to 5 connections each, PQAMP has almost $25 - 50\%$ less delay for best-effort traffic as compared to DCF, whereas at higher loads (from 7 to 10 connections) PQAMP has 20-30% higher delay with respect to DCF. Similarly, for load conditions of up to 6 connections, PQAMP has only 0.2-4.4% less throughput as compared to DCF, whereas at higher loads it goes up to 33% less throughput as compared to DCF. Thus, the above simulation results show the effectiveness of PQAMP in providing service differentiation in single hop environments.

Table 3.9: Simulation Results for Real-Time Traffic in a Single Hop Topology (PQAMP vs DCF)

#RT	#BE	TP (Kbps)		D (msec)		PD	
		DCF	PQAMP	DCF	PQAMP	DCF	PQAMP
1	1	80.97	82.33	10.06	18.59	0	1
2	2	162.86	162.32	23.27	23.96	0	2
3	3	244.85	245.45	59.13	32.34	12	7
4	4	327.39	328.98	159.42	43.64	2	9
5	5	399.44	407.68	512.25	70.93	191	12
6	6	453.38	486.61	1293.94	97.89	907	21
7	7	451.00	569.50	2323.13	150.12	2848	40
8	8	486.79	646.24	2655.39	437.29	4049	190
9	9	496.85	709.26	3019.84	751.98	5661	625
10	10	496.75	720.23	3538.32	769.27	7676	2262

#RT: Number of Real-time connections; #BE: Number of best-effort connections; TP: Throughput; D: Delay; PD: Packet dropped

A chain topology with varying hop counts

We have considered a chain topology of n nodes numbered as 0 through $n-1$, where the distance between any two successive nodes is taken as 200 meters. We generated a number of real-time and best-effort connections from node 0 to node $n-1$. We have varied the number of hops from 1 to 7. we used NOAH (No Ad Hoc routing) from [79] that support fixed routes in chain topology. Simulation

Table 3.10: Simulation Results for Best-Effort Traffic in a Single Hop Topology (PQAMP vs DCF)

#RT	#BE	TP (Kbps)		D (msec)	
		DCF	PQAMP	DCF	PQAMP
1	1	677.65	647.74	94.92	75.80
2	2	622.09	607.76	200.90	144.33
3	3	557.13	557.02	372.94	251.06
4	4	490.27	503.28	502.27	341.99
5	5	435.66	449.57	813.72	529.40
6	6	396.10	395.97	757.27	620.76
7	7	386.66	338.44	871.89	825.63
8	8	363.50	285.47	899.12	1151.29
9	9	351.16	241.87	944.66	1381.11
10	10	349.58	231.31	1019.07	1540.70

#RT: Number of Real-time connections; #BE: Number of best-effort connections; TP: Throughput; D: Delay

time is set to 200 seconds. The other parameters are set as shown in Table 3.8. We have carried out three different type of experiments to examine the effectiveness of the proposed MAC mechanism in chain topology and compared to IEEE 802.11 DCF. In the first set of experiments (called *Experiment 1*), we have taken only real-time traffic of moderate load (with number of connections = 8) and study the performance of PQAMP and IEEE 802.11 DCF for hop length starting from 1 up to 7. This will show the effectiveness of PQAMP to handle higher hop length i.e. in multihop environment as compared to DCF. Our second set of experiments (called *Experiment* 2) study the effect of best-effort traffic[3] on ongoing real-time traffic. For this purpose, we initially fixed the number of real-time traffic as 9 and 6 respectively for the hop lengths - 3 and 6, and then vary the number of best-effort traffic from 1 to 9. The number of real-time connections for a particular hop length is chosen by estimating the number of real-time connections at which the maximum throughput for a particular hop length is obtained. Through some initial simulation runs, we have obtained the number of real-time traffic at which the maximum throughput are achieved for hop lengths - 3 and 6 as 9 and 6 respectively. Our third set of experiments (called *Experiment 3*) study the performance of real-time traffic with varying loads of real-time traffic while keeping number of ongoing best-effort traffic fixed. In this case, we have taken a chain of length 5 (hop length 4) and fixed the number of best-effort traffic at 3 and then we go on varying the number of real-time traffic from 1 to 9.

Figure 3.5 and Figure 3.6 show the performance comparison of PQAMP and IEEE 802.11 DCF for *Experiment 1* of chain topology. It can be seen from Figure 3.5 and Figure3.6 that, PQAMP has higher throughput and lower delay as compared to DCF in all situations except for hop count 1. From hop length 2 onward, the throughput of PQAMP starts exceeding that of DCF. The difference in throughput is increasing with the increase in hop length and reaches the maximum at hop length 5 which is 14.55% more than that of DCF, after that the difference is gradually decreasing till hop length 7 (where it is almost 12% more than that of DCF). With the increase in hop length, the number of relay nodes increases, hence the relay nodes with highest priority (priority class 0) access the channel with smaller backoff values as compared to nodes sending self-generated frames. This

[3] which is assumed as the background traffic

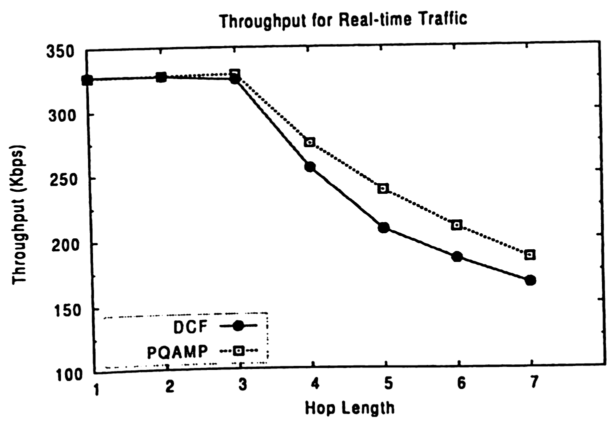


Figure 3.5: Throughput for Real-time traffic for a chain topology (Experiment 1): PQAMP vs DCF

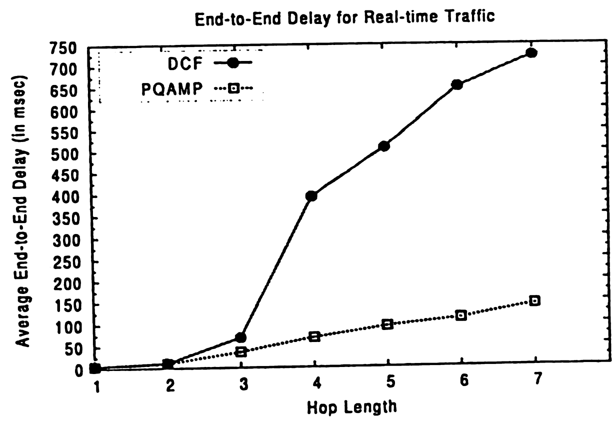


Figure 3.6: Delay for Real-time traffic for a chain topology (Experiment 1): PQAMP vs DCF

leads to the higher throughput (and lower delay) for chain with higher number of relay nodes. But, after a particular limit of hop length, the number of contending relay nodes increases beyond a level which leads to increase in amount of collision among relay nodes. This leads to gradual decrease of throughput and gradual increase in delay for PQAMP in higher hop length. The increasing level of collision also give rise to dropping of more real-time packets as compared to lower hop length, because the retry limit used in higher priority classes (priority classes 0 and 1) of PQAMP are lower as compared to DCF. With a similar argument, we can explain the causes of lower delay in PQAMP as compared to DCF at all hop lengths except for single hop. The difference in delay for DCF and PQAMP increases with increasing hop length. At hop length ≥ 4, delay in PQAMP is more than 5 times less as compared to DCF (as shown in Figure 3.6).

This ensures the suitability of the PQAMP for supporting QoS to real-time traffic in chain topology in the network environment in presence of only real-time traffic as compared to DCF.

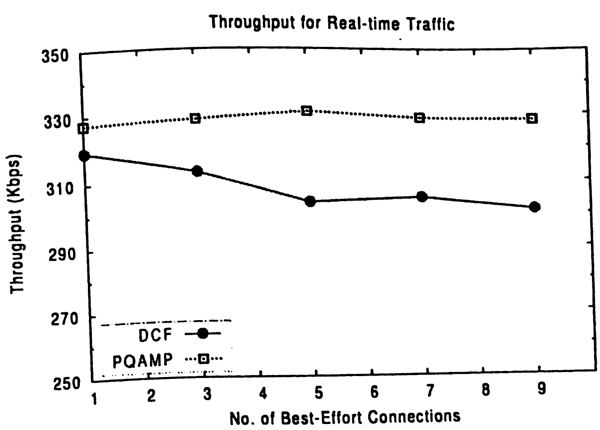


Figure 3.7: Throughput for Real-time traffic for a chain topology with hop length 3 and with 9 real-time traffic (Experiment 2): PQAMP vs DCF

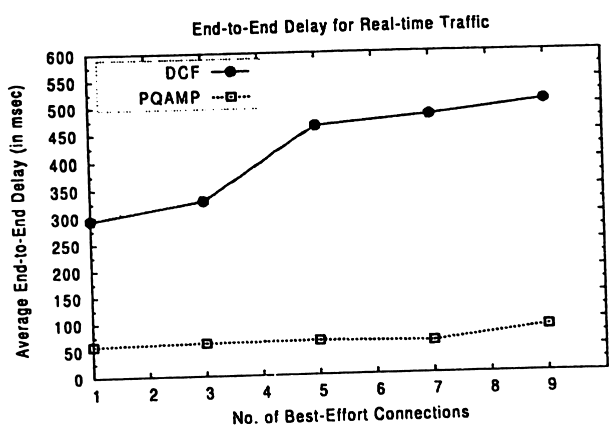


Figure 3.8: Delay for Real-time traffic for a chain topology with hop.length 3 and with 9 real-time traffic (Experiment 2): PQAMP vs DCF

Figure 3.7 to Figure 3.10 show throughput and delay of real-time traffic with varying number of best-effort flows for *Experiment* 2 of chain topology. It can be observed that, with increasing

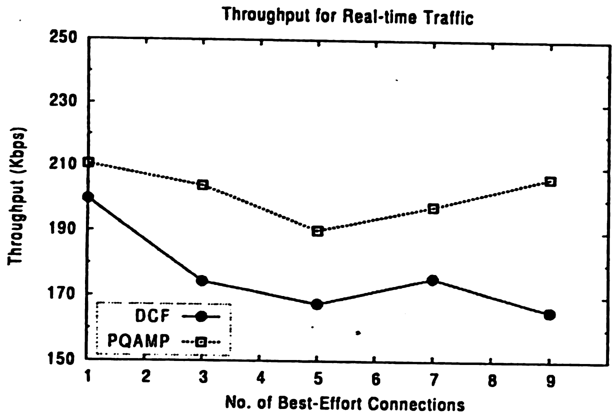


Figure 3.9: Throughput for Real-time traffic for a chain topology with hop length 6 and with 6 real-time traffic (Experiment 2): PQAMP vs DCF

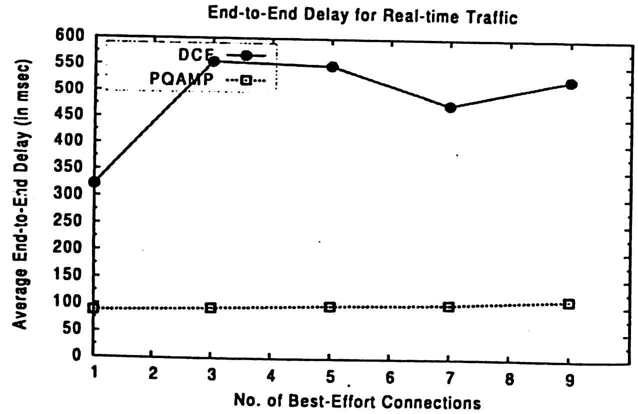


Figure 3.10: Delay for Real-time traffic for a chain topology with hop length 6 and with 6 real-time traffic (Experiment 2): PQAMP vs DCF

load of best-effort traffic, throughput and delay for real-time traffic are not affected much as compared to DCF. Due to the effective traffic differentiation provided by PQAMP, real-time traffic are not affected much in presence of best-effort traffic. Figure 3.8 shows that PQAMP has 5-8 times less delay for chain topology with hop length 3 as compared to DCF. Similarly, a delay improvement of 3.66-5.58 times can be observed in PQAMP for chain topology with hop length 6 (as shown in Figure 3.10). This difference in delay performance increases with increasing load of both real-time and best-effort traffic (as can be seen from Figure 3.8 and Figure 3.10 respectively). Similarly, throughput for real-time traffic improves 2.5-9.00% and 5.45-25.00% for hop lengths 3 and 6 respectively, with larger difference in higher hop length and higher loads (as can be seen from Figure 3.7 and Figure 3.9 respectively). This ensures the suitability of PQAMP to provide traffic differentiation in chain topology with varying best-effort traffic load conditions.

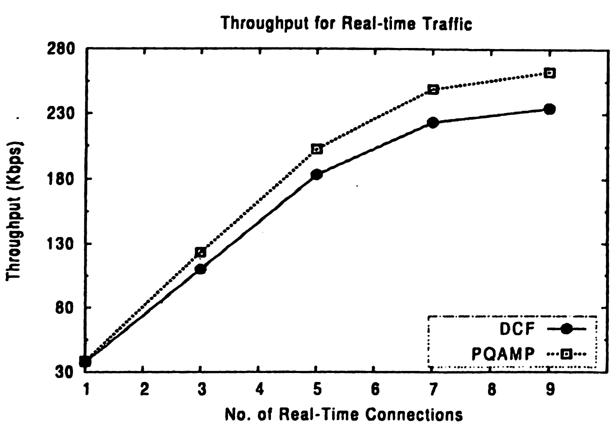


Figure 3.11: Throughput for Real-time traffic for a chain topology with hop length 4 and with 3 best-effort traffic (Experiment 3): PQAMP vs DCF

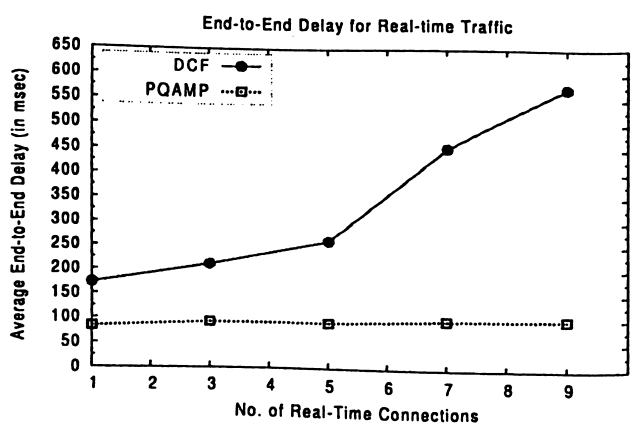


Figure 3.12: Delay for Real-time traffic for a chain topology with hop length 4 and with 3 best-effort traffic (Experiment 3): PQAMP vs DCF

Figure 3.11 and Figure 3.12 show the throughput and delay of real-time traffic with varying number of real-time connections for *Experiment 3* of chain topology. It can be seen that with the increase in number of real-time traffic, delay in PQAMP increases from 82.76 msec to 101.95 msec, whereas we can notice a rapid increase in delay from 172.51 msec to 570.65 msec in DCF. This gives PQAMP to provide 2.08 to 5.61 times delay improvement for real-time traffic as compared to DCF. As can be observed from Figure 3.11, throughput for real-time traffic in PQAMP are 10.0 - 12.5% better as compared to DCF for all real-time traffic load conditions. This ensures the suitability of PQAMP to support QoS for real-time traffic in chain topology for varying real-time traffic load conditions in presence of constant background/best-effort traffic loads.

A Grid topology of size 6×6 (36 nodes)

In this topology, 36 nodes, numbered as 0 through 35, are arranged in a 6×6 grid with a grid spacing of 200 meters. We have generated a number of real-time connections from the nodes in the leftmost positions to the nodes in the rightmost position in the same row. Similarly, we have generated a number of best-effort flows from the nodes in the top row to the nodes in the bottom row of the same column. We have used NOAH (No Ad Hoc routing) from [79] that supports fixed routes in grid topology. Two different sets of experiments are conducted to study the effectiveness of the MAC scheme in supporting QoS for real-time traffic in terms of throughput and delay in the grid topology. In the first set of experiments, we have kept the number of real-time traffic fixed at 3 and vary the number of best-effort connections from 1 to 7. In our second set of experiments, we have kept the number of best-effort connections fixed at 3 and vary the number of real-time connections from 1 to 9. The other simulation parameters are set as shown in Table 3.8.

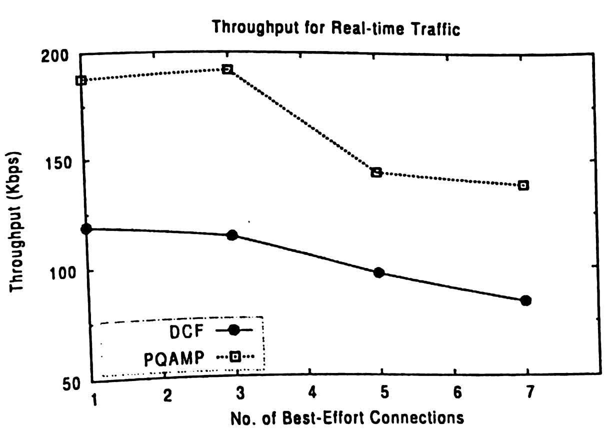


Figure 3.13: Throughput for Real-time traffic for a Grid topology with 3 real-time traffic (PQAMP vs DCF)

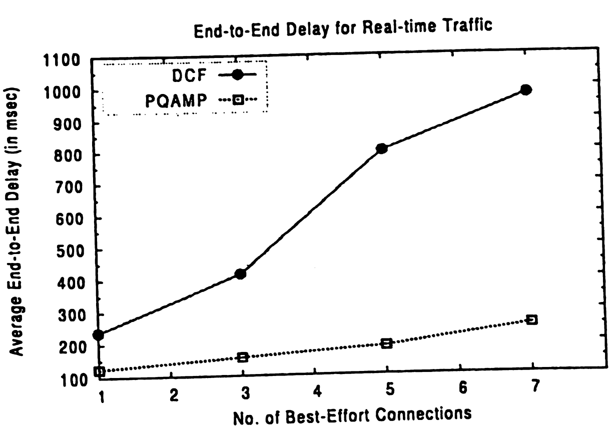


Figure 3.14: Delay for Real-time traffic for a Grid topology with 3 real-time traffic (PQAMP vs DCF)

The Grid topology with the traffic pattern mentioned above represents a very high interfering scenario, where real-time as well as best-effort flows are encountering interference from both type of traffic flows.

Due to the non-overlapping contention windows set for the different priority classes and the higher preference given to nodes with forwarding traffic, PQAMP is able to support adequate level of QoS in terms of throughput and delay.

From Figure 3.13 and Figure 3.14, it can be seen that, although with increasing number of best-

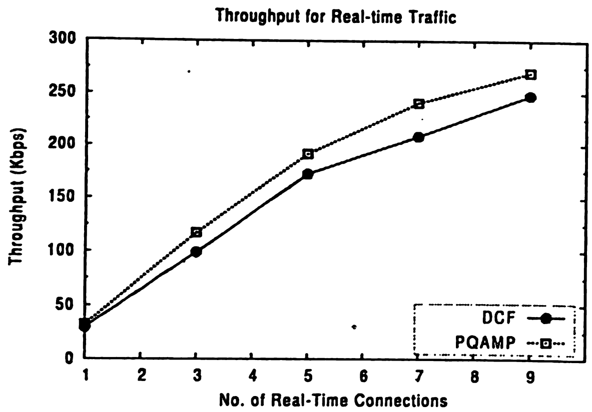


Figure 3.15: Throughput for Real-time traffic for a Grid topology with 3 best-effort traffic (PQAMP vs DCF)

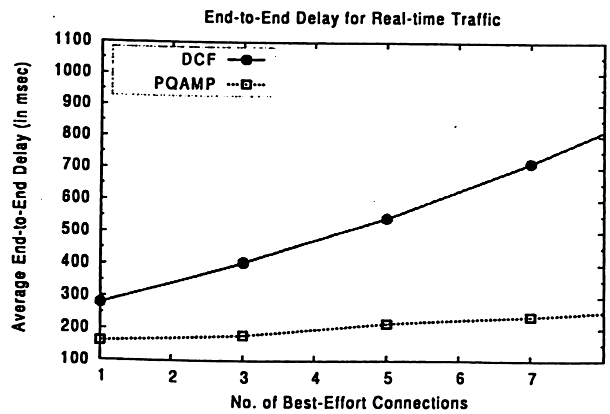


Figure 3.16: Delay for Real-time traffic for a Grid topology with 3 best-effort traffic (PQAMP vs DCF)

effort traffic, throughput and delay for real-time traffic in both the protocols get affected, the affect is much less in PQAMP as compared to DCF. In this situation, PQAMP can still provide an adequate level of QoS for real-time flows as compared to DCF in terms of both throughput and delay, which is mainly due to effective service differentiation and use of higher priority for relay nodes in PQAMP. We can observe a delay improvement of 2.0 to 4.27 times for real-time traffic as compared to DCF with the difference increases with increasing load of best-effort traffic. Similarly, we can observe a throughput improvement of 4.26 to 64.47% for real-time traffic in PQAMP as compared to DCF (as shown in Figure 3.13). We can observe similar poor performance in DCF to support QoS for real-time traffic as compared to PQAMP, when varying real-time traffic load and keeping best-effort traffic load fixed at 3 (as shown in Figure 3.15 and Figure 3.16). We can observe a delay improvement of 1.73 to 3.43 times for real-time traffic in PQAMP as compared to DCF, with larger difference in higher loads (as shown in Figure 3.16). Similarly, we can see 7.43 to 18.66% improvement in throughput for real-time traffic in PQAMP as compared to DCF (as shown in Figure 3.15). There is a marginal improvement of throughput (2.28%) for PQAMP as compared to DCF for real-time traffic load of 9, which is due to heavy load in the network. These simulation results show the suitability of PQAMP's traffic differentiation scheme in grid topology which represents a very high level of traffic interference.

A Random topology of 50 nodes

We have considered a random topology of 50 nodes numbered as 0 through 49 which are arranged in a flat rectangular area of size 1000 m $\times$ 500 m. The nodes are allowed to move using random waypoint mobility model with a maximum speed of 10m/s and pause time of 0 second. To cope up with dynamic behaviour, we have taken AODV [85] as the routing protocol. The total simulation time for the experiments are set as 500s. We have carried out three different sets of experiments to study the effectiveness of PQAMP to support QoS for real-time traffic (in terms of throughput and delay) in a random topology. Different set of experiments are conducted under different real-time traffic load conditions both in the absence and presence of best-effort traffic. In the first set of experiments, we have allowed only the real-time traffic and changes the number of real-time traffic from 1 to 11. The source and destination of a real-time traffic is chosen randomly and a selected source starts generating real-time traffic randomly within the time period between 1.0s to 100.0s. In our second set of experiments, we have kept the number of real-time connections fixed at 7 and vary the number of best-effort connections from 1 to 7. In this case, all real-time connections are started before starting any best-effort connection. The third set of experiments are carried out by keeping the number of best-effort traffic fixed at 5, and varying the real-time connection from 1 to 11. In this experiment, all best-effort connections are started before any real-time connection. The other parameters are set as shown in Table 3.8. For each set of experiments, a total of 10 simulations are performed with 10 different scenarios and averages are taken as the results.

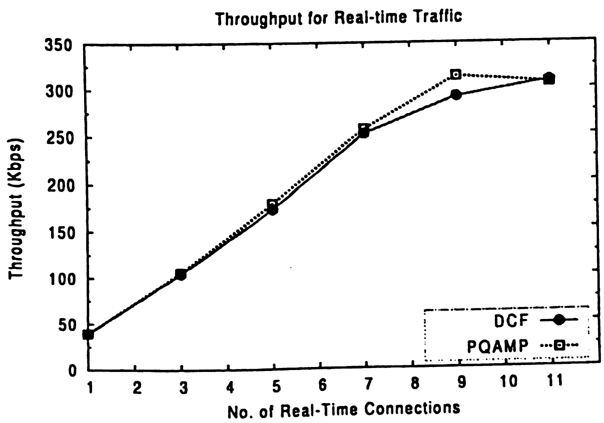


Figure 3.17: Throughput for Real-time traffic for a random topology with only real-time traffic (PQAMP vs DCF)

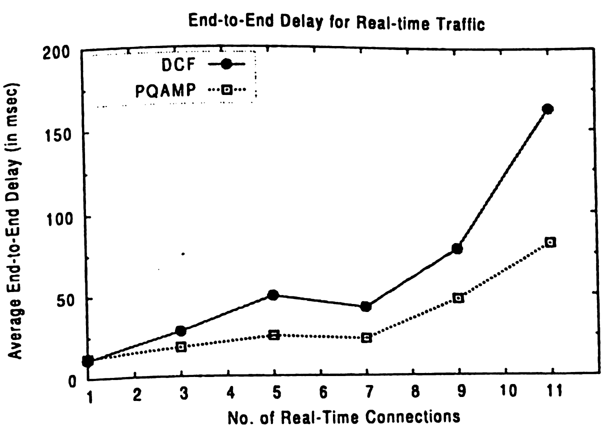


Figure 3.18: Delay for Real-time traffic for a random topology with only real-time traffic (PQAMP vs DCF)

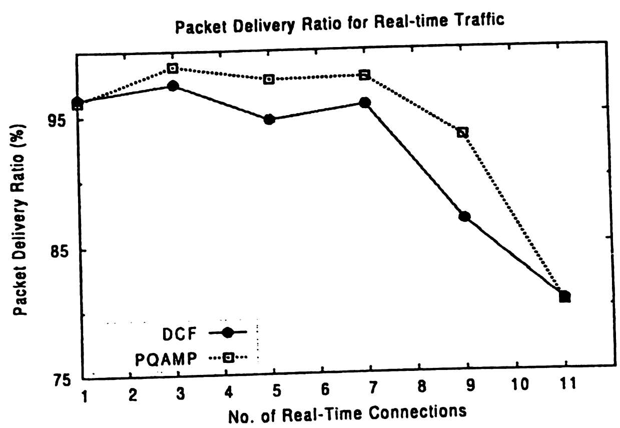


Figure 3.19: Packet Delivery Ratio for Real-time traffic for a random topology with only real-time traffic (PQAMP vs DCF)

From Figure 3.17 to Figure 3.19, we can observe that PQAMP is able to provide better QoS for real-time traffic in a random topology without any best-effort traffic load. Figure 3.18 shows that, PQAMP is able to maintain 55 to 100% lower delay for real-time traffic with growing network load

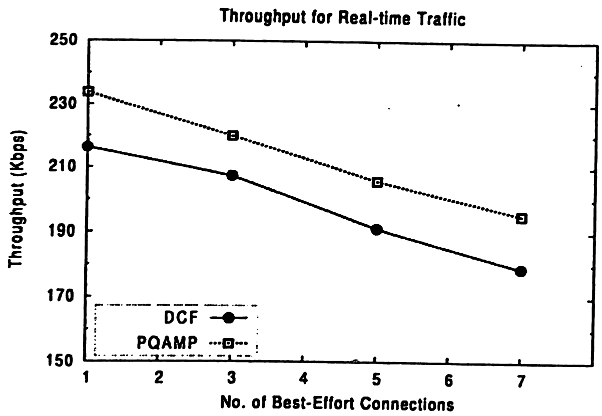


Figure 3.20: Throughput for Real-time traffic for a random topology with 7 real-time traffic (PQAMP vs DCF)

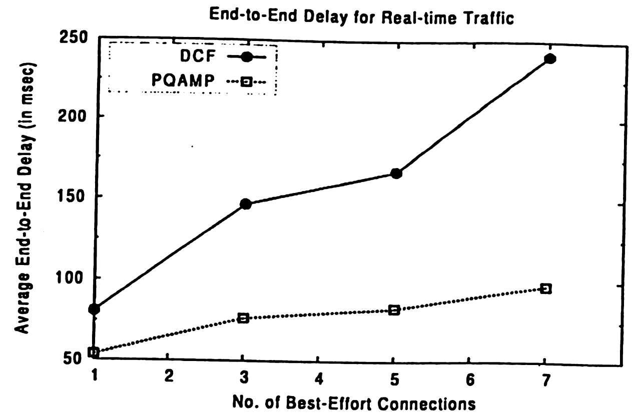


Figure 3.21: Delay for Real-time traffic for a random topology with 7 real-time traffic (PQAMP vs DCF)

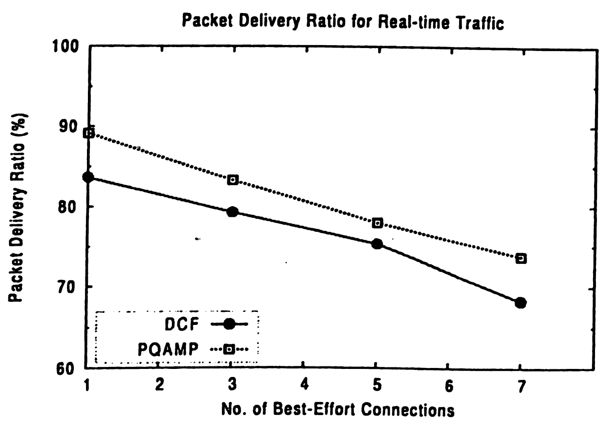


Figure 3.22: Packet Delivery Ratio for Real-time traffic for a random topology with 7 real-time traffic (PQAMP vs DCF)

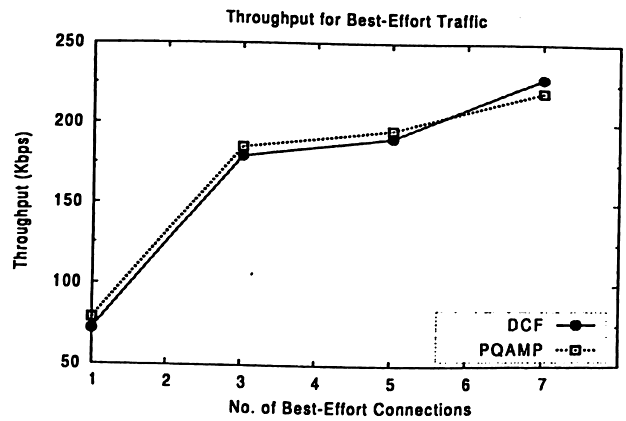


Figure 3.23: Throughput for Best-Effort traffic for a random topology with 7 real-time traffic (PQAMP vs DCF)

as compared to DCF, while the difference increases with increasing load. Similarly, a throughput improvement of 2 to 7% can be seen for real-time traffic in PQAMP as compared to DCF from Figure 3.17. From Figure 3.19, we can observe that, at traffic load condition of only 1 real-time traffic, both DCF and PQAMP are able to maintain a higher Packet Delivery Ratio (PDR) of 96.36 and 96.14 respectively, and the PDR decreases with increasing traffic load conditions. It is seen that PDR of PQAMP remains higher than DCF in all loads except for the traffic load of 1 connection. At the maximum load of 11 connections, PDR of DCF and PQAMP are dropped to 77.32% and 80.39% respectively. These observed performance improvements in PQAMP (as compared to DCF) in terms of delay, throughput and PDR are mainly attributed to assignment of higher priorities to relayed frames.

Figure 3.20 to Figure 3.22 show that PQAMP performs better to support QoS for real-time traffic as compared to DCF in random topology with growing best-effort traffic. PQAMP maintains a delay improvement for real-time traffic, which is more than 2.2 times less as compared to DCF (as

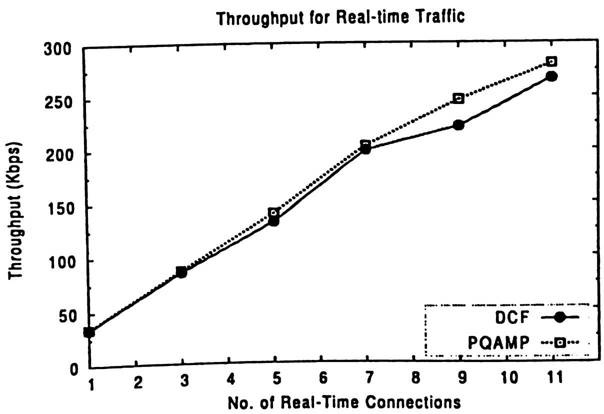


Figure 3.24: Throughput for Real-time traffic for a random topology with 5 best-effort traffic (PQAMP vs DCF)

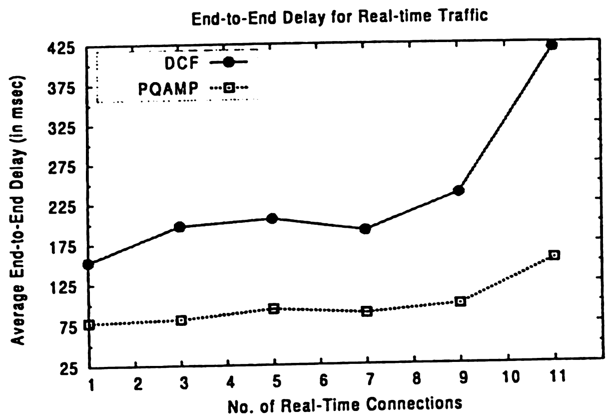


Figure 3.25: Delay for Real-time traffic for a random topology with 5 best-effort traffic (PQAMP vs DCF)

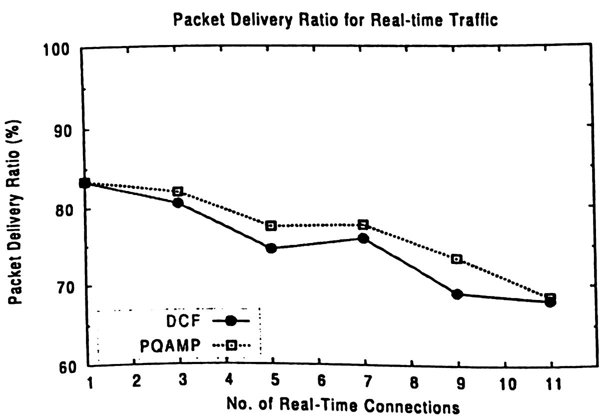


Figure 3.26: Packet Delivery Ratio for Real-time traffic for a random topology with 5 best-effort traffic (PQAMP vs DCF)

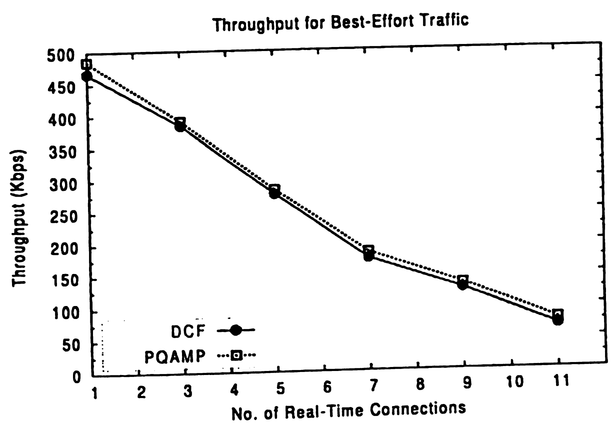


Figure 3.27: Throughput for Best-Effort traffic for a random topology with 5 best-effort traffic (PQAMP vs DCF)

shown in Figure 3.21). Similarly, we can observe a throughput improvement of 8.00 to 9.15% for real-time traffic in PQAMP as compared to DCF (as shown in Figure 3.20). Moreover, for both delay and throughput, the difference in performance between PQAMP and DCF increases with increasing loads. Further, PQAMP maintains a 2.00 to 11.41% higher throughput and a delay improvement of 1.96 to 2.72 times for real-time traffic as compared to DCF for the situations of varying real-time traffic loads while keeping best-effort traffic load fixed (shown in Figure 3.24 and Figure 3.25 respectively). PQAMP is able to maintain a higher PDR for both the situations of varying best-effort traffic and varying real-time traffic, where the improvement is more for the first case (as shown in Figure 3.22 and Figure 3.26). It is interesting to observe that, PQAMP maintains a similar throughput improvement for best-effort traffic as shown in Figure 3.23 and Figure 3.27 respectively. This indicates that along with supporting QoS for real-time traffic in terms of delay, throughput and PDR, PQAMP also achieves an improvement of throughput utilization in MANETs. Therefore, PQAMP's traffic differentiation is able to improve QoS support for real-time traffic in a random topology and at the same time improves network's bandwidth utilization by special treatment of relayed frames in their respective QoS classes.

From all the above simulation results and analysis, we can conclude that PQAMP outperforms IEEE 802.11 DCF in terms of delay and throughput for real-time traffic in all types of topology for MANETs including - single hop, chain, grid and random topology. The performance gain for real-time traffic is more in the presence of best-effort traffic with multihop environments. Further, PQAMP also improves overall network utilization and support minimum service for best-effort traffic. Therefore, PQAMP can be taken as a MAC protocol for supporting QoS in resource constrained MANETs.

3.4.8 Summary on PQAMP

We have proposed PQAMP, a priority based MAC scheme designed for supporting QoS in MANETs. It is based on utilizing non-overlapping contention windows for different priority classes in the same backoff stage, with lower contention window ranges assigned for higher priority classes. With PQAMP, nodes are assigned priorities based on their roles and type of data frame traffic to handle. This priority scheme at MAC level improves the QoS assurance of real-time traffic in terms of reduced end-to-end delay and higher throughput while maximizing overall network utilization. Analytical model shows the superiority of PQAMP's in supporting QoS for real-time traffic in single hop ad hoc networks under saturation condition which is validated through a detailed simulation study. Further, simulation studies in single hop as well as multihop environments reveal the suitability of PQAMP to support QoS at MAC layer in MANETs. By allowing priority inversion in certain cases, PQAMP avoids starvation of best-effort traffic in MANETs. Priority inversion in PQAMP may take place due to (i) freezing and resuming of backoff counter (ii) difference in backoff stages of nodes with real-time and best-effort traffic and (iii) use of post-backoff[4]. The possibility of priority inversion is less in lower load conditions, whereas in presence of heavy best-effort traffic load, it leads to increased end-to-end delay and lower throughput for real-time traffic. Therefore, admission control of best-effort traffic is must to ensure higher level of QoS assurance to real-time traffic. Our next protocol called Strict Priority based QoS-aware MAC Protocol (SPQAMP) implements a strict priority to avoid the situations of priority inversion for further enhancing the level of service differentiation.

3.5 Strict Priority based QoS-Aware MAC Protocol (SPQAMP)

The proposed protocol is based on classifying data frames into two QoS classes - *Real-time* and *Best-effort*, where *Real-time* and *Best-effort* represent High priority and Low priority traffic respectively. Accordingly, a node (in MAC layer) can be in any one of the corresponding state based on the type of data frame it is currently prepared to send. SPQAMP differentiates traffic in such a manner that high priority traffic are completely unhindered by low priority traffic. Here, low priority traffic can be relegated to the background traffic and allowed to transmit only when no high priority frame is currently contending for the channel. That is, the protocol resolves MAC layer contention in such a way that nodes having high priority traffic always win the next transmission right when contending with nodes having low priority traffic. To provide QoS at MAC level, we require that real-time

[4]used to eliminate channel capture by immediate channel access in IEEE 802.11 DCF

data frames get higher channel access priority to that of non real-time or best-effort data frames in the network. This is achieved by assigning smaller contention window to real-time data traffic as compared to ordinary or best-effort traffic. Further, by assigning non-overlapping backoff timers for high and low priority traffic irrespective of their backoff stage, the level of service differentiation is improved. Still due to countdown and freezing of backoff counter (followed in DCF), nodes with low priority traffic can reasonably win channel access before nodes with high priority traffic access the channel. Therefore, countdown and freezing of backoff timer is substituted with resetting of backoff counter for low priority traffic in the situation when a node with low priority traffic detects a busy channel and the value of its backoff timer had entered into the possible backoff timer range of high priority traffic. For the nodes with high priority traffic, they behave like nodes in pure DCF. Thus, using the proper contention window selection method with additional mechanism for resetting backoff counter for nodes with low priority traffic ensures SPQAMP to maintain an enhanced level of service differentiation, where nodes with high priority traffic always win the next transmission right when contending with nodes with low priority traffic. This means the high priority traffic can not be affected by the presence of low priority traffic that exists in the background. Despite SPQAMP gives high priority traffic the exclusive right to access the channel, low priority traffic can still achieve a throughput close to DCF in the absence of high priority traffic.

In SPQAMP, at any point in time, each node of the network can be in one of the following three states: HIGH, LOW and NORMAL based on the type of data frame the node is going to transmit or forward. The states HIGH, LOW and NORMAL are assigned a discrete value which represents a *priority class* assigned to them. The values 0, 1 and 2 represent the priority classes assigned to the nodes in states HIGH, LOW and NORMAL respectively. Based on the type of the frame (at the front of the interface queue) that a node is going to send, it is assigned an appropriate level of state between HIGH and LOW. The HIGH state is assigned to a node which has a delay sensitive real-time data frame to transmit. Similarly, the LOW state is assigned to a node which has a best-effort data frame to be transmitted. The NORMAL state is assigned to a node which has no activity (i.e. no data frame to be transmitted) and hence the default state of every node is set as NORMAL.

SPQAMP adopts the well-known Binary Exponential Backoff (BEB) algorithm as its contention resolution technique, when a node in priority class 0 contents to access the channel. After sensing the channel as idle for DIFS, the station waits for a random backoff interval selected from $[0, CW-1]$, where CW is the current value of the contention window. As long as the channel is sensed idle, the backoff timer is decremented in terms of slot time until it reaches 0. Backoff timer is frozen on detection of any transmission over the channel, and reactivated when the channel is sensed idle for DIFS again. At the time when backoff counter reaches 0, the node transmits its frame. After every successful transmission, a node resets the CW to its initial value $CWmin$ and initiates a backoff[5] before starting a new transmission. At every retransmission attempt (also called backoff stage), the contention window is doubled until it reaches the maximum limit, $CWmax$. After that CW remains at the maximum value until the retry limit (m) is reached. A frame is dropped, if the retry count exceeds the retry limit m and in this situation CW is reset to $CWmin$.

For nodes in priority class 1, the effective backoff timer value at any stage is larger than the maximum possible CW value for priority class 0. At each transmission attempt, a node in priority class 1 selects a random backoff timer in the range from $[W', W' + CW - 1]$, where W' is the

[5] called post backoff in IEEE 802.11 DCF

maximum contention window size for priority class 0 and CW is the contention window size in current backoff stage for priority class 0. For strict priority implementation, we must ensure that backoff timer for priority class 1 is reset while detecting a busy channel, if the value of the backoff timer had entered into the possible contention window range for priority class 0. This means, in such situations, the backoff counter for priority class 1 must be reset to a value uniformly chosen from within the current contention window range (it means backoff stage value remains same).

We have followed the convention that W and $W_{i,j}$ to represent $CWmin$ of IEEE 802.11 DCF and CW size at backoff stage j for priority class i respectively. Equation (3.25) gives $W_{i,j}$ for priority class i in backoff stage j. In Equation (3.26) and Equation (3.27, m' represents the backoff stage at which contention window for priority class i reaches the maximum value. The maximum CW value for priority class 0 is also W'_1 (as can be inferred from Equation (3.26)) which is also written as W' (in subsequent discussion). In this way, a node in priority class i and at backoff stage j, selects a random backoff timer in the range $[W'_i, W_{i,j} - 1]$. Considering the life time of RT frames, we have taken retry limit for priority class 0 as m', whereas for priority class 1, it is m (where $m \geq m'$).

$$W_{i,j} = W'_i + W_j \quad \text{for } i = 0, 1 \text{ and } j = 0, \ldots, m \tag{3.25}$$

$$W'_i = \begin{cases} 0 & \text{for } i = 0 \\ 2^{m'}W & \text{for } i = 1 \end{cases} \tag{3.26}$$

$$W_j = \begin{cases} 2^j W & \text{for } j = 0, \ldots, m' \\ 2^{m'}W & \text{for } j = m' + 1, \ldots, m \end{cases} \tag{3.27}$$

From the above equations, we can write that - at backoff stage j, the backoff counter for priority class 0 and 1 are uniformly chosen from $[0, W_j - 1]$ and $[W', W' + W_j - 1]$ respectively. Figure 3.28 shows the contention window ranges for priority classes 0 and 1 at varying backoff stage with $W = 32$, $W' = 512$, $m' = 4$ and $m = 5$.

Like in PQAMP, this protocol also provides priority-based scheduling at two different levels - at the packet level within a station using a queuing discipline, and at MAC level while contending for the channel. We have implemented support at each of these two levels such that high priority packets are transmitted before low priority packets regardless of whether the low priority packets arrive before or after the high priority packets. The queuing discipline used in SPQAMP is the same as used in PQAMP which is already discussed in section 3.4.2 (in page 34).

3.5.1 Rules for state transition for nodes at MAC layer

The following rules give a clear idea about how nodes at MAC layer changes their states to achieve the strict priority based channel access mechanism in SPQAMP.

- The default state is assumed to be NORMAL for each node in the beginning.
- When a node in NORMAL state receives or generates a real-time data frame to be forwarded to its neighbor node, the node's state is set to HIGH.
- When a node in NORMAL state receives or generates a best-effort data frame to be forwarded to its neighbor node, the node's state is set to LOW.

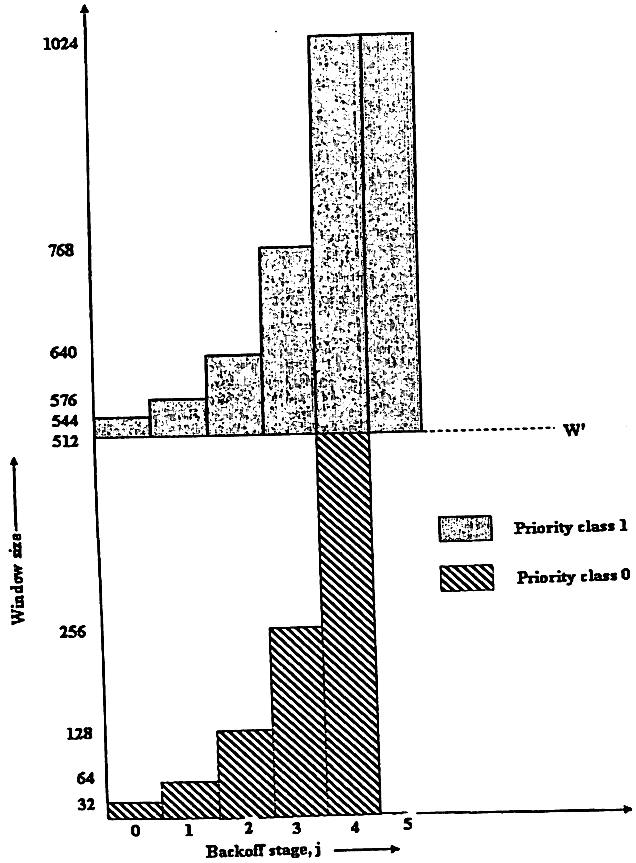


Figure 3.28: Contention Windows for priority class 0 and priority class 1 in SPQAMP

- The state of a node after a successful transmission of data frame is reset to NORMAL state.
- A node remains in the same state after an unsuccessful transmission of data frame until the retry limit for the particular state is reached.
- The state of a node is set back to NORMAL, once the retry limit for the particular state is reached and the transmission is unsuccessful.

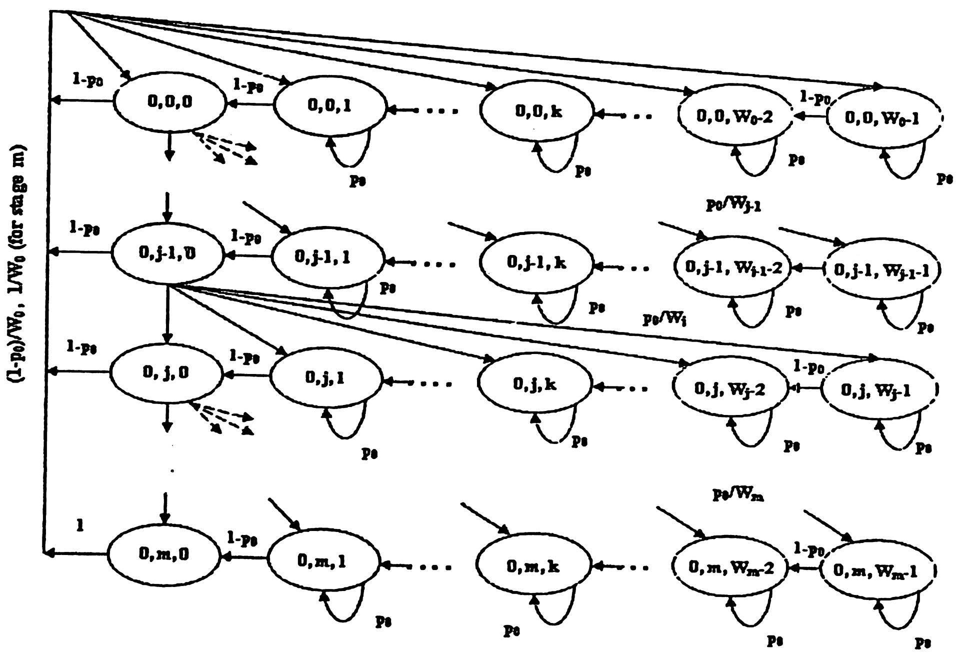


Figure 3.29: Markov Chain Model for priority class 0 in SPQAMP

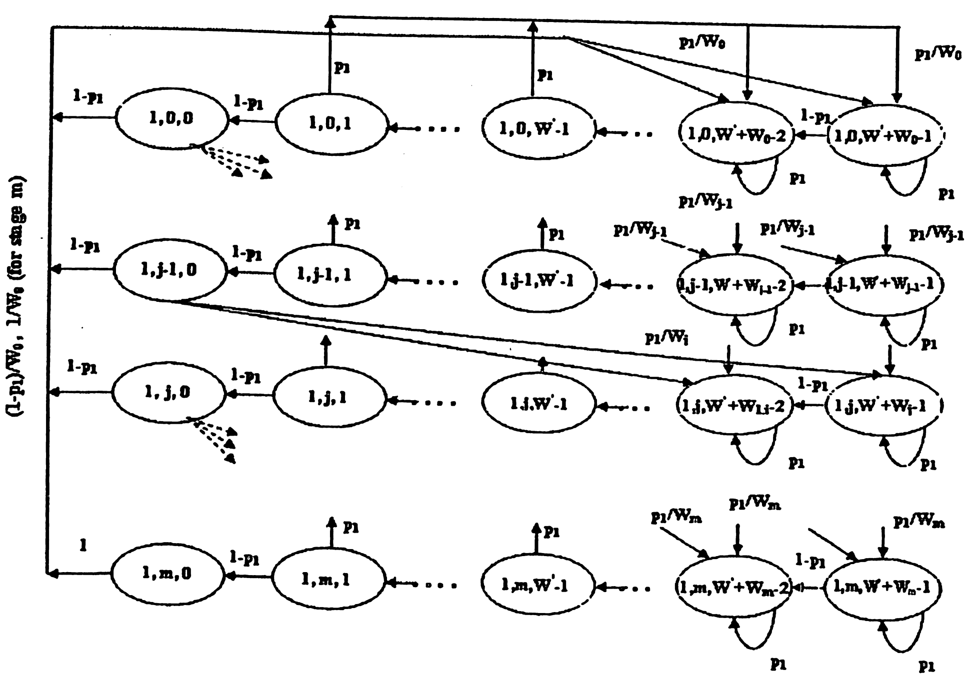


Figure 3.30: Markov Chain Model for priority class 1 in SPQAMP

3.5.2 Analytical Modeling of SPQAMP

An analytical model for two prioritized classes of traffic under high traffic condition is proposed. Here, we have assumed that each station belongs to one and only one priority class and always has frames to send. For a given station in the priority class i ($i = 0, 1$), $b(i,t)$ is defined as a random process representing the value of the backoff counter k at time t, and $s(i,t)$ is defined as the random process representing the backoff stage j ($j = 0, \ldots, m$), where m is the retry limit. The value of the backoff counter $b(i,t)$ is uniformly chosen in the range $[0, W_j - 1]$ and $[W', W' + W_j - 1]$ for priority class 0 and priority class 1 respectively (as can be derived from Equation (3.25), Equation (3.26) and Equation (3.27)).

Let p_i denotes the probability that a frame transmitted in a generic slot time by a station in priority class i experiences a collision. The bi-dimensional random process $\{s(i,t), b(i,t)\}$ is a discrete time Markov Chain as depicted in Figure 3.29 (for priority class 0) and Figure 3.30 for (priority class 1) respectively. Therefore, the state of each node in the priority class i is described by $\{i, j, k\}$, where j and k represent backoff stage and backoff value respectively.

The non-null transition probabilities between states for class 0 (i.e. the High Priority) and class 1 (i.e. the Low Priority) are listed as follows. The transition probabilities for priority class 0, that are shown in (1) to (5) are quite similar to the transition probabilities in the basic/pure DCF model. Transition probabilities for priority class 1 are listed in (6) to (11)

$$
\begin{array}{lll}
(1) & P\{(0,j,k)|(0,j,k+1)\} = 1 - p_0, & \text{for} \quad 0 \le k \le W_j - 2 \text{ and } 0 \le j \le m \\
(2) & P\{(0,0,k)|(0,j,0)\} = \dfrac{(1-p_0)}{W_0}, & \text{for} \quad 0 \le k \le W_0 - 1 \text{ and } 0 \le j \le m \\
(3) & P\{(0,j,k)|(0,j-1,0)\} = \dfrac{p_0}{W_j}, & \text{for} \quad 0 \le k \le W_j - 1 \text{ and } 0 \le j \le m \\
(4) & P\{(0,0,k)|(0,m,0)\} = \dfrac{1}{W_0}, & \text{for} \quad 0 \le k \le W_0 - 1 \\
(5) & P\{(0,j,k)|(0,j,k)\} = p_0, & \text{for} \quad 1 \le k \le W_j - 1 \text{ and } 0 \le j \le m \\
(6) & P\{(1,j,k)|(1,j,k+1)\} = 1 - p_1, & \text{for} \quad 0 \le k \le W' + W_j - 2 \text{ and } 0 \le j \le m \\
(7) & P\{(1,0,k)|(1,j,0)\} = \dfrac{(1-p_1)}{W_0}, & \text{for} \quad W' \le k \le W' + W_0 - 1 \text{ and } 0 \le j \le m \\
(8) & P\{(1,j,k)|(1,j-1,0)\} = \dfrac{p_1}{W_j}, & \text{for} \quad W' \le k \le W' + W_j - 1 \text{ and } 0 \le j \le m \\
(9) & P\{(1,0,k)|(1,m,0)\} = \dfrac{1}{W_0}, & \text{for} \quad W' \le k \le W' + W_0 - 1 \\
(10) & P\{(1,j,k)|(1,j,k)\} = p_1, & \text{for} \quad W' \le k \le W' + W_j - 1 \text{ and } 0 \le j \le m \\
(11) & P\{(1,j,k)|(1,j,k')\} = \dfrac{p_1}{W_j}, & \text{for} \quad 0 \le k' \le W' - 1 \text{ and } W' \le k \le W' + W_j - 1
\end{array}
$$

The transition probabilities that account for different functionality of the underlying MAC protocols are as follows.

- (1) and (6) are for the decrements of backoff timer;
- (2) and (7) are for resetting of contention window size after a successful transmission;
- (3) and (8) are for an unsuccessful transmission due to collision;
- (4) and (9) are for the resetting of contention window at the maximum backoff stage (corresponding to retry limit);

- (5) and (10) are for detection of busy channel during a backoff stage and hence freezing of the backoff timer;

- finally (11) ensures that the backoff timer for priority class 1 (i.e. low priority) is reset when the backoff counter enters into possible contention window range for priority class 0 (i.e. high priority) and the medium is sensed as busy.

Let $b_{i,j,k}$ be the stationary distribution of a given Markov chain for a station with priority class i. Therefore, $b_{i,j,k}$ can be defined as follows.

$$b_{i,j,k} = \lim_{t \to \infty} (Pr[s(i,t) = j, b(i,t) = k]) \qquad for \quad i \quad = \quad 0,1$$

In steady state, from the Markov chain for priority class 0 (as shown in Figure 3.29), we can derive the following through chain regularities.

$$b_{0,j,0} \quad = \quad p_0^j \times b_{0,0,0} \qquad \text{where} \quad 1 \leq j \leq m \tag{3.28}$$

Further, we have

$$b_{0,j,k} \quad = \quad b_{0,j,k-1} \times (1 - p_0) + \left(\frac{1}{W_j} \times b_{0,j,0} \right) + (p_0 \times b_{0,j,k})$$

This gives

$$\begin{aligned} b_{0,j,k} &= \frac{1}{(1-p_0)} \times \left[(1-p_0)(b_{0,j,k-1}) + \left(\frac{b_{0,j,0}}{W_j} \right) \right] \text{for } 1 \leq k \leq W_j - 2 \\ b_{0,j,W_j-1} &= \frac{1}{1-p_0} \times \left[\frac{b_{0,j,0}}{W_j} \right] \end{aligned}$$

Therefore, we get

$$b_{0,j,k} \quad = \quad \frac{b_{0,j,0}}{1-p_0} \times \frac{W_j - k}{W_j} \text{for } 0 \leq k \leq W_j - 1 \tag{3.29}$$

Now, using the normalization condition for stationary distribution, we get

$$\sum_{j=0}^{m} \sum_{k=0}^{W_j-1} b_{0,j,k} = 1 \tag{3.30}$$

From Equation (3.30), we have -

$$
\begin{aligned}
1 &= \sum_{j=0}^{m}\sum_{k=0}^{W_j-1} b_{0,j,k} \\
&= \sum_{j=0}^{m}\sum_{k=0}^{W_j-1}\left[\frac{b_{0,j,0}}{1-p_0}\times\frac{W_j-k}{W_j}\right] \\
&= \sum_{j=0}^{m}\left[\frac{b_{0,j,0}}{1-p_0}\times\sum_{k=0}^{W_j-1}\left(\frac{W_j-k}{W_j}\right)\right] \\
&= \sum_{j=0}^{m}\left[\frac{b_{0,j,0}}{W_j\times(1-p_0)}\times\frac{(W_j)(W_j+1)}{2}\right] \\
&= \sum_{j=0}^{m}\frac{(p_0)^j\times(W_j+1)\times b_{0,0,0}}{2(1-p_0)} \\
&= \frac{b_{0,0,0}}{2(1-p_0)}\times\left[\sum_{j=0}^{m'-1}(p_0)^j 2^j W_0+\sum_{j=m'}^{m}(p_0)^j 2^m W_0+\sum_{j=0}^{m}(p_0)^j\right] \\
&= \frac{b_{0,0,0}}{2(1-p_0)}\times\left[W_0\times\frac{1-(2p_0)^{m'}}{1-(2p_0)}+W_0\times 2^m\times\frac{(p_0)^{m'-1}-(p_0)^{m+1}}{1-p_0}+\frac{1-(p_0)^{m+1}}{1-p_0}\right] \\
\Rightarrow b_{0,0,0} &= \frac{2(1-p_0)}{W_0\times\left[\frac{1-(2p_0)^{m'}}{1-(2p_0)}+2^m\times\frac{(p_0)^{m'+1}-(p_0)^{m+1}}{1-p_0}\right]+\frac{1-(p_0)^{m+1}}{1-p_0}}
\end{aligned}
\tag{3.31}
$$

Similarly, in steady state for priority class 1, we can derive the following through chain regularities.

$$b_{1,j,0} = p_1^j \times b_{1,0,0} \qquad \text{where} \quad 1 \le j \le m \tag{3.32}$$

Let v represents the stationary probability of an imaginary state, that we have added to model the resetting of countdown timer during a backoff slot t within a stage j, when a node with priority class 1 detects the channel as busy. When the backoff counter of a node with priority class 1 is reset, it undergoes a state transition from a state, where its backoff counter value is k ($0 \le k \le W'-1$), to a state within the same backoff stage j with a new backoff counter value k' in the range between W' and $W'+W_j-1$. Further, we have assumed u to represent the state $b_{1,j,0}$. With the introduction of these two new symbols (u and v), we get the following

$$
\begin{aligned}
b_{1,j,W'+W_j-1} &= \frac{u}{W_j}+\frac{v}{W_j}+(p_1\times b_{1,j,W'+W_j-1}) \\
\Rightarrow b_{1,j,W'+W_j-1} &= \left[\frac{u+v}{W_j}\times\frac{1}{1-p_1}\right]
\end{aligned}
\tag{3.33}
$$

Now, for $W' \le k \le W'+W_j-2$, we get

$$
\begin{aligned}
b_{1,j,k} &= (1-p_1)(b_{1,j,k+1})+\frac{u}{W_j}+\frac{v}{W_j}+p_1\times(b_{1,j,k}) \\
\Rightarrow b_{1,j,k} &= b_{i,j,k+1}+\frac{u+v}{W(1-p_1)}
\end{aligned}
\tag{3.34}
$$

From Equation (3.33) and Equation (3.34), we have

$$b_{1,j,k} = \frac{1}{1-p_1} \times \left[\frac{u+v}{W_j} \times (W' + W_j - k)\right] \quad \text{for } W' \leq k \leq W' + W_j - 1 \tag{3.35}$$

As a special case, we see

$$b_{1,j,W'} = \frac{u+v}{1-p_1} \tag{3.36}$$

Further, we have noticed

$$\begin{aligned} b_{1,j,W'-1} &= (1-p_1) \times b_{1,j,W'} \\ b_{1,j,W'-2} &= (1-p_1)^2 \times b_{1,j,W'} \end{aligned}$$

To generalize, we have

$$b_{1,j,k} = (1-p_1)^{W'-k} \times b_{1,j,W'} \quad \text{for } 0 \leq k \leq W' - 1 \tag{3.37}$$

Now, from Equation (3.36) and Equation (3.37), we have

$$b_{1,j,k} = (u+v) \times (1-p_1)^{W'-k-1} \quad \text{for } 0 \leq k \leq W' - 1 \tag{3.38}$$

Therefore,

$$\begin{aligned} \frac{b_{1,j,k}}{b_{1,j,0}} &= \frac{(1-p_1)^{W'-k-1}}{(1-p_1)^{W'-1}} \quad \text{for } 0 \leq k \leq W' - 1 \\ \Rightarrow b_{1,j,k} &= \frac{b_{1,j,0}}{(1-p_1)^k} \quad \text{for } 0 \leq k \leq W' - 1 \end{aligned} \tag{3.39}$$

Similarly, from Equation (3.35), we get

$$\begin{aligned} \frac{b_{1,j,k}}{b_{1,j,0}} &= \frac{(\frac{1}{1-p_1}) \times \frac{u+v}{W_j} \times (W' + W_j - k)}{(u+v) \times (1-p_1)^{W'-1}} \quad \text{for } W' \leq k \leq W' + W_j - 1 \\ \Rightarrow b_{1,j,k} &= \left[\frac{W' + W_j - k}{W \times (1-p_1)^{W'}}\right] \times b_{1,j,0} \quad \text{for } W' \leq k \leq W' + W_j - 1 \end{aligned} \tag{3.40}$$

We shall use the Equation (3.39) and Equation (3.40) to get the value of $b_{1,0,0}$. Using the normalized condition for stationary distribution, we get

$$\begin{aligned}
1 &= \sum_{j=0}^{m} \sum_{k=0}^{W'+W_j-1} b_{1,j,k} \\
&= \sum_{j=0}^{m} \left[\sum_{k=0}^{W'-1} b_{1,j,k} + \sum_{k=W'}^{W'+W_j-1} b_{1,j,k} \right] \\
&= \sum_{j=0}^{m} \left[\sum_{k=0}^{W'-1} \left(\frac{1}{(1-p_1)^k} \times b_{1,j,0} \right) + \sum_{k=W'}^{W'+W_j-1} \left(\frac{W'+W_j-k}{W_j(1-p_1)^{W'}} \times b_{1,j,0} \right) \right] \\
&= \sum_{j=0}^{m} \left[b_{1,j,0} \times \left(\frac{1-(1-p_1)^{W'}}{p_1(1-p_1)^{W'-1}} \right) + \frac{b_{1,j,0}}{W_j(1-p_1)^{W'}} \times \frac{W_j(W_j+1)}{2} \right] \\
&= \frac{1-(1-p_1)^{W'}}{p_1(1-p_1)^{W'-1}} \times \left[\sum_{j=0}^{m} b_{1,j,0} \right] + \frac{1}{2(1-p_1)^{W'}} \times \left[\sum_{j=0}^{m} (b_{1,j,0} \times (W_j+1)) \right] \\
&= \frac{1-(1-p_1)^{W'}}{p_1(1-p_1)^{W'-1}} \times b_{1,0,0} \times \left[\sum_{j=0}^{m} p_1^j \right] + \frac{b_{1,0,0}}{2(1-p_1)^{W'}} \times \\
&\quad \left[\left[\sum_{j=0}^{m'-1} (p_1)^j \times W_j \right] + \left[\sum_{j=m'}^{m} (p_1)^j \times W_j \right] + \left[\sum_{j=0}^{m} (p_1)^j \right] \right] \\
&= \frac{1-(1-p_1)^{W'}}{p_1(1-p_1)^{W'-1}} \times b_{1,0,0} \times \frac{1-(p_1)^{m+1}}{1-p_1} + \frac{b_{1,0,0}}{2(1-p_1)^{W'}} \times \\
&\quad \left[\frac{W_0(1-(2p_1)^{m'})}{1-2p_1} + \frac{2^{m'} W_0 (p_1^{m'+1} - p_1^{m+1})}{1-p_1} + \frac{1-p_1^{m+1}}{1-p_1} \right] \\
&= \frac{b_{1,0,0}}{(1-p_1)^{W'}} \times \left[\frac{(1-(1-p_1)^{W'})(1-p_1^{m+1})}{p_1} + \frac{W_0}{2} \times \right. \\
&\quad \left. \left(\frac{1-(2p_1)^{m'}}{1-2p_1} + \frac{2^{m'}(p_1^{m'+1} - p_1^{m+1})}{1-p_1} \right) + \frac{1-p_1^{m+1}}{1-p_1} \right]
\end{aligned}$$

$$\Rightarrow b_{1,0,0} = \frac{(1-p_1)^{W'}}{\frac{(1-(1-p_1)^{W'})(1-p_1^{m+1})}{p_1} + \frac{W_0}{2} \times \left(\frac{1-(2p_1)^{m'}}{1-2p_1} + \frac{2^{m'}(p_1^{m'+1} - p_1^{m+1})}{1-p_1} \right) + \frac{1-p_1^{m+1}}{1-p_1}} \quad (3.41)$$

Let τ_i be the probability that a station in a priority class i transmits during a generic slot time. Therefore, from the Markov chain we have τ_i as follows.

$$\tau_i = \sum_{j=0}^{m} b_{i,j,0} = \sum_{j=0}^{m} p_i^j b_{i,0,0} = b_{i,0,0} \frac{1-p_i^{m+1}}{1-p_i} \quad (3.42)$$

Therefore, from Equation (3.31), Equation (3.41) and Equation (3.42), we have obtained τ_0 and τ_1

as given in Equation (3.43) and Equation (3.44) respectively.

$$\tau_0 = \frac{2(1-p_0^{m+1})}{W_0 \times \left[\frac{1-(2p_0)^{m'}}{1-(2p_0)} + 2^m \times \frac{(p_0)^{m'+1}-(p_0)^{m+1}}{1-p_0}\right] + \frac{1-(p_0)^{m+1}}{1-p_0}} \quad (3.43)$$

$$\tau_1 = \frac{(1-p_1)^{W'-1}(1-p_1^{m+1})}{\frac{(1-(1-p_1)^{W'})(1-p_1^{m+1})}{p_1} + \frac{W_0}{2} \times \left(\frac{1-(2p_1)^{m'}}{1-2p_1} + \frac{2^{m'}(p_1^{m'+1}-p_1^{m+1})}{1-p_1}\right) + \frac{1-p_1^{m+1}}{1-p_1}} \quad (3.44)$$

Let n_0 and n_1 denote the number of stations in the priority class 0 and 1 respectively. A transmitted frame of priority class i collides with another frame, when one more station also transmits during the same slot time. The probability p_i that a frame transmitted in a generic slot time by a station in priority class 0 and priority class 1 experiences a collision are given by Equation (3.45) and Equation (3.46) respectively.

$$p_0 = 1-(1-\tau_0)^{n_0-1} \times (1-\tau_1)^{n_1} \quad (3.45)$$

$$p_1 = 1-(1-\tau_1)^{n_1-1} \times (1-\tau_i)^{n_0} \quad (3.46)$$

Equation (3.43), Equation (3.44), Equation (3.45) and Equation (3.46) together represent a nonlinear system in four unknowns τ_0, τ_1, p_0 and p_1, which have been solved by using numerical methods with the parameters set used for the SPQAMP scheme. This gave us the values of p_i and τ_i for each priority class i. Let P_b denotes the probability that the channel is busy, which happens when at least one station transmits in a given time slot(shown in Equation (3.48)).

$$P_b = 1-(1-\tau_0)^{n_0} \times (1-\tau_1)^{n_1} \quad (3.47)$$

Saturation Throughput of SPQAMP

To compute the saturation throughput S_i for a priority class i, we have defined and used the parameters $P_{s,i}$, P_s and P_c as already defined in Equation (3.14), Equation (3.15) and Equation (3.16) respectively. Saturation throughput, S_i of a priority class i is defined as below (as shown in Equation (3.17)).

$$S_i = \frac{P_{s,i} \times T_{E[P]}}{(1-P_b)\sigma + P_s T_{succ} + P_c T_{col}}$$

where $T_{E[P]}$ is the time required to transmit average payload size $E[P]$ (determined by the data rate), and thus $P_{s,i} \times T_{E[P]}$ is the amount of time spent in a generic slot time in transmitting an average payload information successfully. T_{avg} is the average length of a slot time. σ, T_{succ} and T_{col} are the duration of a slot corresponding to an idle slot, a successful transmission and a collision, respectively. Note that idle time slot σ is specific to the physical layer of IEEE 802.11, e.g., $20\mu s$ for Direct Sequence Spread Spectrum (DSSS). The specific values of T_{succ} and T_{col} for RTS-CTS based channel access mechanism are as shown in Equation (3.18) (on page 40). Parameters used for calculation of saturation throughput are as used in PQAMP and are specific to Direct Sequence Spread Spectrum (DSSS) physical layer of IEEE 802.11 standard as shown in Table 3.2 (on page 41).

Saturation Delay

Saturation delay, D_i, for a priority class i is the average delay under the saturation condition which includes medium access delay (due to backoff, collision), frame transmission time, and inter-frame spaces (such as SIFS, DIFS etc.). The average backoff delay depends on the value of a station's backoff counter and the duration when the counter freezes due to other stations' transmissions. Let X_i denotes the random variable representing the total number of backoff slots, which a frame in priority class i encounters without considering the case when the counter freezes. The probability that the frame is successfully transmitted after the jth retry is given by $p_i^j(1-p_i)/P_{i,succ}$, which uses conditional probability on a successful transmission with probability $P_{i,succ}$. Here, $P_{i,succ}$ is the probability that a frame in class i is transmitted successfully within the retry limit m and is equal to $1-p_i^{m+1}$. It is important to note that $P_{i,succ}$ is different from $P_{s,i}$, which is defined in Equation (3.14) (on page 39). Let $BO_{i,j}$ denotes the average number of backoff slots after the jth retry for a frame in priority class i. In SPQAMP, low priority stations spend considerable amount of time in resetting the backoff counter values whenever busy channel is detected. This leads to larger values for $BO_{1,j}$ as compared to high priority station (which is similar to standard DCF). Therefore, the (expected) total number of idle slots, $E(X_i)$ that the frame encounters during backoff stages are different for priorities 0 and 1 which are given below. For high priority, we have

$$BO_{0,j} = \sum_{h=0}^{j} \frac{W_h - 1}{2} \tag{3.48}$$

$$E(X_0) = \sum_{j=0}^{m} \frac{p_0^j(1-p_0)}{1-p_0^{m+1}} BO_{0,j} = \sum_{j=0}^{m} \frac{p_0^j(1-p_0)}{1-p_0^{m+1}} \sum_{h=0}^{j} \frac{W_h - 1}{2} \tag{3.49}$$

For low priority, estimation of $BO_{1,j}$ and hence $E(X_1)$ require elaborate calculations due to complicated structure of the Markov chain resulting from resetting of backoff counter value once it becomes $\leq W'$.

From the Markov chain (shown in Figure 3.30), the probability that there is no cycle (due to resetting of backoff timer) for a station in low priority is $(1-p_1)^{W'}$. Therefore, the average number of cycles per frame transmission is

$$\left[\frac{1}{(1-p_1)^{W'}} - 1\right]$$

Similarly, we can find out the average length of cycles i.e. the expected value of cycle length if number of cycles is ≥ 1 as

$$\frac{W_j}{2} + \left[\frac{1}{p_1} - \frac{W'(1-p_1)^{W'}}{1-(1-p_1)^{W'}}\right]$$

The average number of slots a station needs to wait before transmission, in the case where no resetting is performed is given by

$$\left(W' + \frac{W_j}{2}\right)$$

This gives $BO_{1,j}$ and $E(X_1)$ as follows

$$BO_{1,j} = \left[\frac{1}{(1-p_1)^{W'}} - 1\right] \times \left[\frac{W_j}{2} + \frac{1}{p_1} - \frac{W'(1-p_1)^{W'}}{1-(1-p_1)^{W'}}\right] + \left(W' + \frac{W_j}{2}\right) \quad (3.50)$$

$$\begin{aligned} E(X_1) &= \sum_{j=0}^{m} \left[\frac{p_1^j(1-p_1)}{1-p_1^{m+1}} \times BO_{1,j}\right] \\ &= \frac{1-p_1}{1-p_1^{m+1}} \times \sum_{j=0}^{m} \left[p_1^j \left(\frac{1}{(1-p_1)^{W'}} - 1\right)\left(\frac{W_j}{2} + \frac{1}{p_1} - \frac{W'(1-p_1)^{W'}}{1-(1-p_1)^{W'}}\right) \right. \\ &\quad \left. + p_1^j\left(W' + \frac{W_j}{2}\right)\right] \\ &= \left(\frac{1-p_1}{1-p_1^{m+1}}\right) \times \left[\left(\frac{1}{(1-p_1)^{W'}} - 1\right)\left(\frac{1}{2}\sum_{j=0}^{m} p_1^j W_j\right)\right. \\ &\quad \left. + \left(\sum_{j=0}^{m} p_1^j\right)\left[\left(\frac{1}{(1-p_1)^{W'}} - 1\right)\left(\frac{1}{p_1} - \frac{W'(1-p_1)^{W'}}{1-(1-p_1)^{W'}}\right) + W'\right]\right] \quad (3.51) \end{aligned}$$

where

$$\sum_{j=0}^{m} p_1^j = \frac{1-p_1^{m+1}}{1-p_1} \quad (3.52)$$

and

$$\sum_{j=0}^{m} p_1^j W_j = W_0 \times \left[\frac{1-(2p_1)^{m'}}{1-(2p_1)} + \frac{2^{m'}(p_1^{m'+1} - p_1^{m+1})}{1-p_1}\right] \quad (3.53)$$

Let B_i denotes the random variable representing the total number of slots when the counter freezes for class i, which a frame encounters during its backoff periods. The portion of idle slots which is used to decrease $E(X_i)$ is $(1-p_i)$. Therefore, the (expected) total number of busy slots, $E(B_i)$ that the frame encounters during backoff stages is given as (like in Equation (3.21)) -

$$E(B_i) = \frac{E(X_i)}{(1-p_i)} p_i$$

Similarly, $E(N_{i,retry})$ denotes the average number of retries for a frame in the priority class i, which is given as below (like in Equation (3.22)).

$$E(N_{i,retry}) = \sum_{j=0}^{m} \frac{j p_i^j (1-p_i)}{1-p_i^{m+1}} = \frac{(1-p_i^{m+1}) - m p_i^{m+1}(1-p_i)}{(1-p_i)(1-p_i^{m+1})}$$

Therefore, the expected saturation delay $E(D_i)$ for a priority class i is given as (like in Equation (3.23))

$$E(D_i) = E(X_i)\sigma + E(B_i)\left[\frac{P_s}{P_b}T_{succ} + \frac{P_b - P_s}{P_b}T_{col}\right] + E(N_{i,retry})(T_{col} + T_{oh}) + T_{succ}$$

Where σ, $[(P_s/P_b)T_{succ} + \{(P_b - P_s)/P_b\}T_{col}]$, $(T_{col} + T_{oh})$ and T_{succ} are the average slot lengths for an idle slot, a busy slot, a failed transmission slot and a successful transmission respectively during backoff periods. Here, T_{oh} denotes the time that a station has to wait before sensing the channel again, when its frame transmission collides. T_{oh} for RTS-CTS based channel access method is already defined in Equation (3.24) of section 3.4.

3.5.3 Analytical Results

In this section, we have presented the analytical results in single-hop networks to show that SPQAMP is very effective in terms of service differentiation as compared to DCF. In SPQAMP, the real-time traffic get better service in terms of higher throughput, reduced delay and lower packet dropping probability in comparison with the best-effort traffic. For performance analysis, we used RTS-CTS based channel access mechanism in IEEE 802.11 DCF under the Direct Sequence Spread Spectrum (DSSS) physical layer with the values of the parameters listed in Table 3.2. We adopted the following parameters in SPQAMP for analytical results. These are: m=5 and m=7 for real-time and best-effort traffic respectively; m'=5; and W=32.

Table 3.11 and Table 3.12 show the analytical results in presence of both high and low priority traffic. It can be observed from these tables that SPQAMP provide a much better *throughput*, *delay* and *packet dropping probability* for real-time traffic as compared to DCF in saturation condition. In Table 3.12 and Table 3.13, by delay value ∞ we mean delay which is $\geq 1000s$. It can be observed that throughput and delay for *priority class 1* (or best-effort traffic) are 0 and ∞ respectively in saturation condition, when there exists nodes with priority class 0 (or real-time traffic) in the network. It is mainly due to the resetting of backoff counter in *priority class 1* while detecting busy channel during backoff period. Throughput for *priority class 0* is more than 100% higher as compared to DCF for number of nodes per priority class $\leq$ 10. With increase in number of nodes per priority class up to 20, throughput of priority class 0 in SPQAMP becomes 63% higher as compared to DCF. Similarly, the total throughput of SPQAMP is slightly higher than DCF up to the number of nodes per priority class $\leq$ 10, after that it reduces to 21.6% less as compared to DCF. This is due to smaller values of $CWmax$ and retry limit used in *priority class 0* of SPQAMP. From Table 3.12, we can observe 2 to 5 times less delay in *priority class 0* of SPQAMP as compared to DCF for number of nodes (per priority class) from 1 to 9. After that, delay difference between SPQAMP and DCF decreases gradually, and at number of nodes per priority class $\geq$ 19, delay in priority class 0 of SPQAMP becomes slightly larger than DCF. It is interesting to observe that SPQAMP has lower packet dropping probability as compared to DCF for both high and low priority traffic up to number of node per priority class are 10 and 9 respectively. At higher number of nodes, due to lower values of both $CWmax$ and retry limit for priority class 0 and resetting of backoff counter for priority class 1, packet dropping probability for both priority class 0 and priority class 1 increases as compared to DCF. From the analytical results shown in Table 3.11 and Table 3.12, we can clearly visualize the strict traffic differentiation provided by SPQAMP.

Table 3.13 shows the analytical results in presence of only one of high or low priority traffic. It is observed that SPQAMP with only *priority class 0* (or real-time traffic), outperforms DCF in terms of delay and throughput, where the delay improvement of almost 5 times can be noticed at higher load (40 nodes). Throughput improvement is marginal in almost all the loads, whereas packet dropped probability is more as compared to DCF. It can be noticed that due to backoff counter

Table 3.11: Analytical Results (Saturation Throughput): SPQAMP vs DCF

Nodes	SPQAMP			IEEE 802.11 DCF	
(per PC)	PC 0	PC 1	Total	Per PC	Total
1	0.7327	0.0000	0.7327	0.3679	0.7357
2	0.7351	0.0000	0.7351	0.3658	0.7315
3	0.7340	0.0000	0.7340	0.3636	0.7271
4	0.7324	0.0000	0.7324	0.3617	0.7234
5	0.7309	0.0000	0.7309	0.3601	0.7202
6	0.7294	0.0000	0.7294	0.3588	0.7175
7	0.7280	0.0000	0.7280	0.3575	0.7150
8	0.7267	0.0000	0.7267	0.3564	0.7128
9	0.7255	0.0000	0.7255	0.3554	0.7107
10	0.7115	0.0000	0.7115	0.3544	0.7088
11	0.6996	0.0000	0.6996	0.3535	0.7069
12	0.6879	0.0000	0.6879	0.3526	0.7052
13	0.6759	0.0000	0.6759	0.3518	0.7035
14	0.6632	0.0000	0.6632	0.3510	0.7019
15	0.6498	0.0000	0.6498	0.3502	0.7003
16	0.6356	0.0000	0.6356	0.3494	0.6988
17	0.6205	0.0000	0.6205	0.3487	0.6973
18	0.6045	0.0000	0.6045	0.3480	0.6959
19	0.5877	0.0000	0.5877	0.3473	0.6945
20	0.5699	0.0000	0.5699	0.3466	0.6931

PC: Priority Class

resetting technique adopted for low priority traffic in SPQAMP, DCF outperforms SPQAMP in terms of throughput and delay for *priority class 1* (best-effort traffic). With growing loads, throughput for *priority class 1* improves whereas delay deteriorates. This shows the efficiency of the individual priority class in SPQAMP against DCF.

Thus, from the above analytical results, we can judge the suitability of SPQAMP to provide QoS for real-time traffic in terms of throughput and delay as compared to DCF.

Table 3.12: Analytical Results (Saturation Delay and Packet Dropping Probability): SPQAMP vs DCF

Nodes (per PC)	SPQAMP				IEEE 802.11 DCF	
	SD(msec)		PDP		SD(msec)	PDP
	PC 0	PC 1	PC 0	PC 1		
1	6.62	∞	0.00000000	0.00000004	12.57	0.00000001
2	11.40	∞	0.00000077	0.00000122	27.88	0.00000844
3	15.58	∞	0.00001494	0.00000679	46.39	0.00007499
4	19.41	∞	0.00006596	0.00002000	67.06	0.00024367
5	23.02	∞	0.00016911	0.00004321	89.39	0.00052740
6	26.49	∞	0.00033254	0.00007865	112.96	0.00092359
7	29.79	∞	0.00055457	0.00012657	137.48	0.00142497
8	32.90	∞	0.00082643	0.00018565	163.06	0.00203126
9	35.96	∞	0.00115667	0.00025963	189.25	0.00272803
10	75.40	∞	0.01101198	0.00442363	216.36	0.00352143
11	113.24	∞	0.02934609	0.01468267	243.80	0.00439071
12	152.18	∞	0.05567774	0.03184098	272.08	0.00534990
13	193.13	∞	0.08978444	0.05653607	300.90	0.00638738
14	236.25	∞	0.13088564	0.08879417	330.04	0.00749254
15	281.65	∞	0.17780623	0.12807525	359.53	0.00866443
16	329.48	∞	0.22912632	0.17337619	389.89	0.00992246
17	379.96	∞	0.28332692	0.22337432	420.46	0.01123903
18	433.39	∞	0.33891871	0.27658508	450.95	0.01259841
19	490.17	∞	0.39454288	0.33150935	482.31	0.01404198
20	550.74	∞	0.44903867	0.38675251	514.23	0.01555556

PC: Priority Class; SD: Saturation Delay; PDP: Packet Dropping Probability

Table 3.13: Analytical Results in presence of only one priority class in SPQAMP

No. of Nodes	PC 0			PC 1		
	STP	SD(msec)	PDP	STP	SD(msec)	PDP
2	0.7351	11.41	0.00000078	0.3496	29.32	0.00000000
4	0.7324	19.51	0.00006792	0.4524	80.02	0.00000000
6	0.7293	26.69	0.00034426	0.5075	225.35	0.00000000
8	0.7265	33.34	0.00086984	0.5436	586.49	0.00000000
10	0.7240	39.61	0.00163172	0.5714	1605.85	0.00000000
12	0.7217	45.60	0.00260789	0.5952	4187.01	0.00000000
14	0.7196	51.37	0.00377571	0.6100	12904.23	0.00000000
16	0.7177	56.90	0.00510380	0.6247	31008.10	0.00000000
18	0.7159	62.27	0.00658317	0.6336	93577.18	0.00000000
20	0.7141	67.50	0.00820192	0.6437	283601.73	0.00000000
22	0.7124	72.55	0.00992554	0.6521	604224.07	0.00000000
24	0.7109	77.51	0.01176720	0.6569	∞	0.00000000
26	0.7093	82.33	0.01369874	0.6632	∞	0.00000000
28	0.7079	87.05	0.01572253	0.6687	∞	0.00000000
30	0.7064	91.64	0.01780867	0.6736	∞	0.00000000
32	0.7051	96.16	0.01997742	0.6779	∞	0.00000000
34	0.7037	100.61	0.02221909	0.6817	∞	0.00000000
36	0.7024	104.96	0.02451621	0.6852	∞	0.00000000
38	0.7011	109.28	0.02688704	0.6865	∞	0.00000000
40	0.6999	113.51	0.02930577	0.6894	∞	0.00000000

PC: Priority Class; STP: Saturation Throughput; SD: Saturation Delay; PDP: Packet Dropping Probability

3.5.4 Model Validation

In this section, we have conducted simulations on a single hop network of 60 nodes using NS-2 [80] to validate the proposed analytical model. We adopted the following parameters in SPQAMP for both simulation and analytical results. These are: m=5 and m=7 for real-time and best-effort traffic respectively; m'=5; and W=32. Further, for comparison purpose, we have also shown the simulation and analytical results of IEEE 802.11 DCF. The analytical model used for IEEE 802.11 DCF is similar to the model used for priority class 0 of SPQAMP (as shown in Figure 3.29), but with parameters from IEEE 802.11 DCF standard. Frame sizes for both real-time and best-effort traffic were fixed at 1024 bytes. The simulation time is set as 5000 sec. The other simulation parameters are as shown in Table 3.8. The simulation results are averages over 10 runs.

Table 3.14 and Table 3.15 compare the analytical and simulation results for both SPQAMP and IEEE 802.11 DCF. From these tables, we can observe that the analytical models of both SPQAMP and IEEE 802.11 DCF are accurate in capturing the trends and service differentiation provided by SPQAMP in single hop networks as compared to IEEE 802.11 DCF. From Table 3.14, we can observe a maximum of 3.92% and 1.46% difference between analytical and simulation results in saturation throughput (STP) for SPQAMP and IEEE 802.11 DCF respectively. Except the first case, SPQAMP shows a lower saturation delay (SD) for real-time traffic in simulation results as compared to the analytical results. We observed a maximum of 2.79 msec deviations between analytical and simulation results in saturation delay (SD) for IEEE 802.11 DCF (as shown in Table 3.15). It is worth mentioning that, although in our analytical result for SPQAMP, we got zero saturation throughput and saturation delay of very large quantity (which we have denoted as ∞) for best-effort traffic, in simulations we got some non-zero amount of throughput and some smaller delays. This is due to few successful transmissions by nodes with low priority, which is possible because of adoption of post-backoff mechanism in SPQAMP from standard DCF. For best-effort traffic in IEEE 802.11 DCF, the difference between analytical and simulation results are comparatively less as compared to Real-time traffic. From both simulation and analytical results, we can observe that SPQAMP improves saturation throughput for real-time traffic by 95 - 99% as compared to DCF. Similarly, SPQAMP reduces saturation delay of real-time traffic by 1.4 - 22 times and 1.8 - 6.0 times, respectively for simulation and analytical results, as compared to DCF. It can be seen that both SPQAMP and DCF have almost similar aggregate saturation throughput.

3.5.5 Simulation Results and Analysis

After evaluation of SPQAMP with analytical results (in section 3.5.3) and then validated the analytical results with simulation (in section 3.5.4) in single hop environments under saturation condition, we have carried out simulation studies to further validate analytical results for different scenarios (including both single hop and multihop) under non-saturated conditions. As noted earlier, in networks with QoS support, overloaded or saturated situations are avoided through admission control. This necessitates performance evaluation of SPQAMP in comparison with IEEE 802.11 DCF using simulation studies in single hop as well as multihop ad hoc networks under non-saturated conditions. We adopted the following parameters in SPQAMP for simulation results. These are: m=5 and m=7 for real-time and best-effort traffic respectively; m'=5; and W=32. We added the required changes in IEEE 802.11 DCF of NS-2 [80] to implement the enhanced backoff procedure in SPQAMP. Fur-

Table 3.14: Analytical Results Vs. Simulation Results for Saturation Throughput (SPQAMP and DCF)

Traffic Class	Protocol	S/A	$N_{RT} = N_{BE}$ $= 1$	$N_{RT} = N_{BE}$ $= 5$	$N_{RT} = N_{BE}$ $= 10$	$N_{RT} = N_{BE}$ $= 15$
Real-Time	SPQAMP	A	0.7123	0.7318	0.7294	0.7263
		S	0.6816	0.6945	0.6997	0.6871
	DCF	A	0.3640	0.3637	0.3593	0.3558
		S	0.3496	0.3558	0.3511	0.3502
Best-Effort	SPQAMP	A	0.0000	0.0000	0.0000	0.0000
		S	0.0418	0.0246	0.0209	0.0181
	DCF	A	0.3640	0.3637	0.3593	0.3558
		S	0.3496	0.3558	0.3511	0.3502

S: Simulation; A: Analytical; RT: Real-Time; BE: Best-Effort; N_{RT}: Number of RT traffic;N_{BE}: Number of BE traffic

Table 3.15: Analytical Results Vs. Simulation Results for Saturation Delay (in msec) (SPQAMP and DCF)

Traffic Class	Protocol	S/A	$N_{RT} = N_{BE}$ $= 1$	$N_{RT} = N_{BE}$ $= 5$	$N_{RT} = N_{BE}$ $= 10$	$N_{RT} = N_{BE}$ $= 15$
Real-Time	SPQAMP	A	6.78	24.96	42.74	57.86
		S	9.45	10.26	11.36	15.38
	DCF	A	12.51	79.19	200.73	351.84
		S	13.27	80.73	203.42	351.07
Best-Effort	SPQAMP	A	∞	∞	∞	∞
		S	112.38	441.64	507.80	935.74
	DCF	A	12.51	79.19	200.73	351.84
		S	13.27	80.73	203.42	351.07

S: Simulation; A: Analytical; RT: Real-Time; BE: Best-Effort; N_{RT}: Number of RT traffic;N_{BE}: Number of BE traffic

ther, we have modified the priqueue queuing discipline in queuing model to give higher preference to real-time traffic as compared to best-effort traffic. We compared the performance of SPQAMP with IEEE 802.11 DCF (with priority queue which gives preference to routing packets). Simulations are performed on four different topologies (similar to PQAMP) as follows:

- *Topology 1*: A single hop topology
- *Topology 2*: A chain topology with varying hop counts
- *Topology 3*: A Grid topology of size 6×6 (36 nodes)
- *Topology 4*: A Random topology of 50 nodes in a flat rectangular area of size 1000m×500m with random waypoint mobility model with maximum node speed of 10m/s.

The performance parameters we have considered here are the same as used in simulation studies of PQAMP. We used similar traffic specification for real-time and best-effort traffic as used in PQAMP.

A Single Hop Topology

We have selected a single hop topology of 60 nodes randomly placed within a rectangular area of size $120\text{m} \times 120\text{m}$. To avoid extra control overhead due to routing protocol, we used NOAH (No Ad Hoc routing) from [79] that supports direct communication between wireless nodes without generating any routing related packets. We generated a number of real-time and best-effort connections between any randomly selected pair of nodes. The simulation time is set to 200 seconds. The other parameters are set as shown in Table 3.8.

For performance analysis, we have varied the number of real-time and best-effort connections from 1 to 10 and measure the three parameters namely *Delay*, *Throughput* and *Packet Dropped* for real-time traffic. Similarly, we have also measured the *Throughput* and *Delay* for the best-effort traffic. It can be seen from Table 3.16 and Table 3.17 that SPQAMP is very effective in providing strict traffic differentiation. In SPQAMP, real-time traffic has much better performance in terms of *Delay*, *Throughput* and *Packet Dropped* as compared with DCF. As the number of traffic increases, SPQAMP performs better with respect to service provided to real-time traffic as compared to DCF. With the increase in number of traffic, SPQAMP is able to consistently maintain a smaller delay in the range from 6.81 msec to 8.54 msec whereas delay in DCF grows exponentially with increasing load (from 10.06 msec to 3538.32 msec). This results in 1.48 to 414.40 times higher delay for real-time traffic in DCF than that of SPQAMP (as shown in Table 3.16). Similarly, we can observe a consistently better performance of SPQAMP in terms of *throughput* and *packet dropped* for real-time traffic as compared to DCF. Although at traffic loads ranging from 2 to 5, *throughput* for real-time traffic in SPQAMP are marginally higher (0.04 to 1.81%) than DCF, from traffic load 6 onwards SPQAMP performs consistently better than DCF with 7.8 to 63.00% higher *throughput* than that of DCF. It is interesting to observe that, SPQAMP even maintains a very low *packet dropped* value from light to high load conditions (as shown in Table 3.16) as compared to DCF. Although for number of traffic up to 2, SPQAMP performs similar to DCF in terms of *packet dropped*, from number of traffic 3 to 10 SPQAMP performs 2 to 118 times better than that of DCF. As illustrated in Table 3.17, DCF maintains a good level of service quality in terms of *throughput* and *delay* for best-effort traffic as compared to SPQAMP. Due to strict service differentiation maintained by SPQAMP, throughput for best-effort traffic sharply decreases from 464.92 Kbps to 29.38 Kbps with increase in traffic load, which are 31 to 91% less as compared to DCF. Similarly, we can observe that when the load changes from 1 to 10, delay for best-effort traffic in SPQAMP changes from 201.46 msec to 5548.28 msec that are 2.12 to 5.4 times higher with respect to DCF. Thus, we can observe suitability of SPQAMP in providing QoS to real-time traffic in single hop environment. Further, we can observe SPQAMP outperforming PQAMP in this case.

Table 3.16: Simulation Results for Real-Time Traffic in a Single Hop Topology (SPQAMP vs DCF)

#RT	#BE	TP (Kbps)		D (msec)		PD	
		DCF	SPQAMP	DCF	SPQAMP	DCF	SPQAMP
1	1	80.97	81.53	10.06	6.81	0	0
2	2	162.86	163.37	23.27	6.19	0	5
3	3	244.85	244.94	59.13	6.02	12	6
4	4	327.39	325.94	159.42	6.09	2	7
5	5	399.44	406.67	512.25	5.82	191	22
6	6	453.38	488.71	1293.94	5.88	907	27
7	7	451.00	568.42	2323.13	6.79	2848	45
8	8	486.79	650.20	2655.39	6.38	4049	37
9	9	496.85	726.69	3019.84	7.98	5661	81
10	10	496.75	809.61	3538.32	8.54	7676	65

#RT: Number of Real-time connections; #BE: Number of best-effort connections; TP: Throughput; D: Delay; PD: Packet Dropped

Table 3.17: Simulation Results for Best-Effort Traffic in a Single Hop Topology (SPQAMP vs DCF)

#RT	#BE	TP (Kbps)		D (msec)	
		DCF	SPQAMP	DCF	SPQAMP
1	1	677.65	464.92	94.92	201.46
2	2	622.09	379.53	200.90	483.44
3	3	557.13	302.59	372.94	817.21
4	4	490.27	244.56	502.27	1350.41
5	5	435.66	188.85	813.72	2068.99
6	6	396.10	141.44	757.27	2735.01
7	7	386.66	105.81	871.89	3672.37
8	8	363.50	72.44	899.12	3781.12
9	9	351.16	50.59	944.66	4960.51
10	10	349.58	29.38	1019.07	5548.28

#RT: Number of Real-time connections; #BE: Number of best-effort connections; TP: Throughput; D: Delay

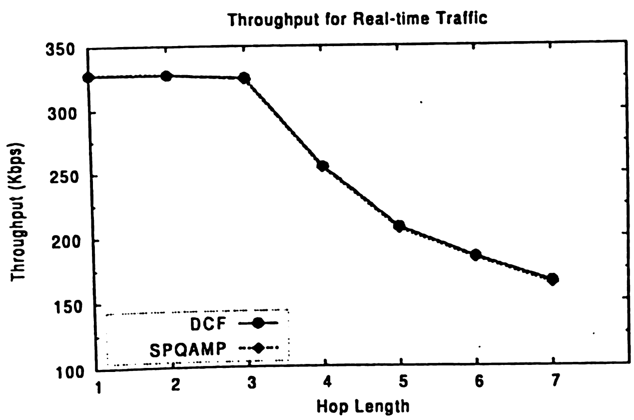


Figure 3.31: Throughput for Real-time traffic for a chain topology (Experiment 1): SPQAMP vs DCF

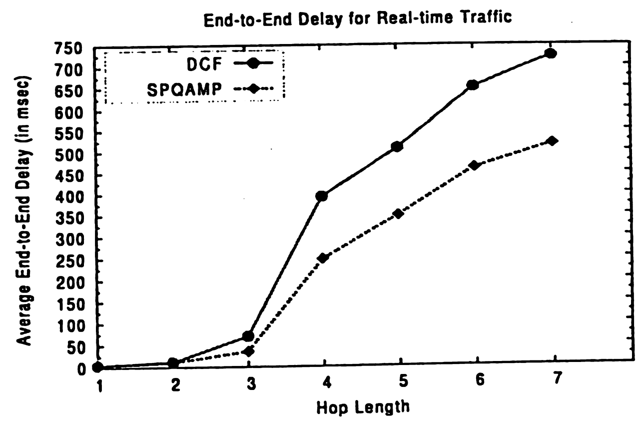


Figure 3.32: Delay for Real-time traffic for a chain topology (Experiment 1): SPQAMP vs DCF

A Chain Topology

We have considered a chain topology of n nodes numbered as 0 through $n-1$, where the distance between any two successive nodes is taken as 200 meters. We generated a number of real-time and best-effort connections from node 0 to node $n-1$. We varied the number of hops from 1 to 7. we used NOAH (No Ad Hoc routing) from [79] that support fixed routes in chain topology. Simulation time is set to 200 seconds. The other parameters are set as shown in Table 3.8. We carried out three different type of experiments to examine the effectiveness of SPQAMP in chain topology and compared to IEEE 802.11 DCF. In the first set of experiments (called *Experiment 1*), we have taken only real-time traffic of moderate load (with number of connections = 8) and studied the performance of SPQAMP and IEEE 802.11 DCF for hop length starting from 1 up to 7. This has shown the effectiveness of SPQAMP to handle higher hop length i.e. in multihop environment as compared to DCF. Our second set of experiments (called *Experiment* 2) study the effect of best-effort traffic[6] on ongoing real-time traffic. For this purpose, we initially fixed the number of real-time traffic as 9 and 6 respectively for the hop lengths - 3 and 6, and then vary the number of best-effort traffic from 1 to 9. Our third set of experiments (called *Experiment 3*) study the performance of real-time traffic with varying loads of real-time traffic while keeping number of ongoing best-effort traffic fixed. In this case, we have taken a chain of length 5 (hop length 4) and fix the number of best-effort traffic at 3 and then we go on varying the number of real-time traffic from 1 to 9.

Figure 3.31 and Figure 3.32 show the performance comparison of SPQAMP and IEEE 802.11 DCF for *Experiment 1* of chain topology. It can be seen from Figure 3.31 that SPQAMP has almost similar throughput as compared to DCF in all hop lengths. Due to absence of best-effort traffic and hence with no service differentiation, throughput performance of DCF and SPQAMP are almost similar (as shown in analytical results). With the increase in hop length, the delay of real-time traffic changes from 3.92 msec to 515 msec which are 0.93 to 4.17 times lower than that of DCF (as shown in Figure 3.32). This is mainly due to lower values of backoff counter and retry limit used for real-time traffic in SPQAMP, hence results in slightly higher amount of packet dropped as compared to

[6]which is assumed as the background traffic

DCF. The marginal increase in packet dropped in SPQAMP can easily be tolerated in delay sensitive real-time applications.

This ensures the suitability of the SPQAMP for real-time traffic as compared to DCF in chain topology without any background (or best-effort) traffic. Further, we can observe PQAMP to outperform SPQAMP in this situation, specially due to suitability of PQAMP in multihop environment.

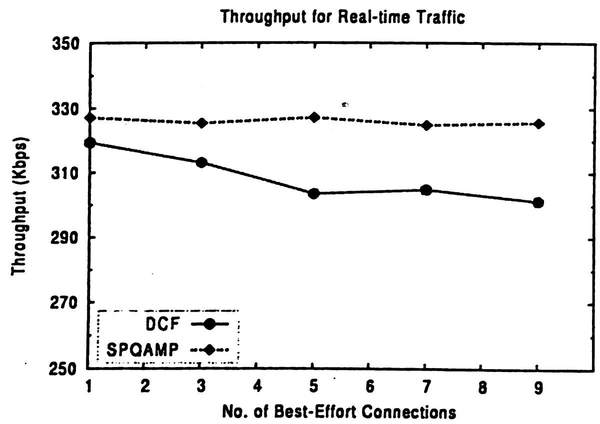


Figure 3.33: Throughput for Real-time traffic for a chain topology with hop length 3 and with 9 real-time traffic (Experiment 2): SPQAMP vs DCF

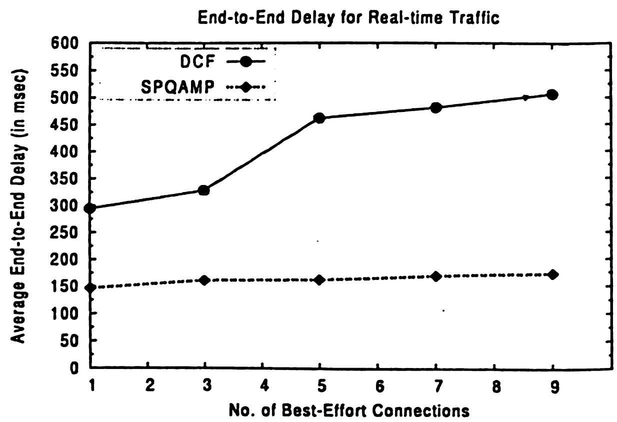


Figure 3.34: Delay for Real-time traffic for a chain topology with hop length 3 and with 9 real-time traffic (Experiment 2): SPQAMP vs DCF

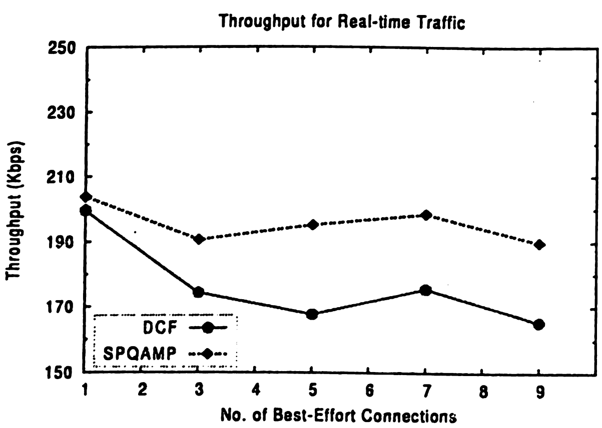


Figure 3.35: Throughput for Real-time traffic for a chain topology with hop length 6 and with 6 real-time traffic (Experiment 2): SPQAMP vs DCF

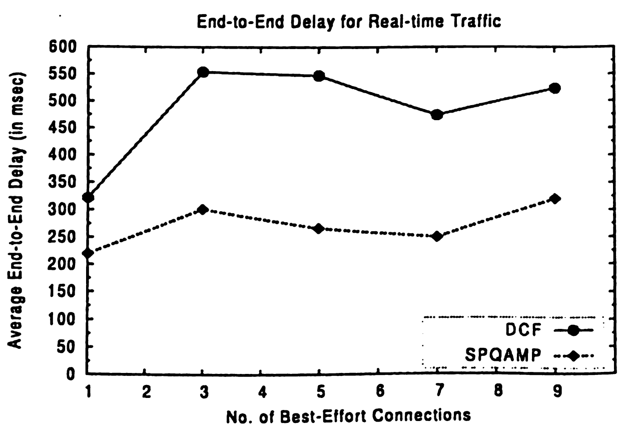


Figure 3.36: Delay for Real-time traffic for a chain topology with hop length 6 and with 6 real-time traffic (Experiment 2): SPQAMP vs DCF

Figure 3.33 to Figure 3.36 show the throughput and delay of real-time traffic over varying number of best-effort traffic for *Experiment 2* of chain topology. It can be seen that with increasing number of best-effort traffic, the throughput and delay of SPQAMP does not change much as compared to DCF. With the increase in number of best-effort traffic, the throughput for real-time traffic in DCF starts decreasing. Similarly, with increasing best-effort traffic, delay for real-time traffic get reasonably affected in DCF. Due to the effective strict service differentiation provided by SPQAMP, real-time traffic are not affected much in presence of growing best-effort traffic. It can be seen from Figure 3.34 that at hop length 3, with increase in best-effort traffic, delay for real-time traffic in SPQAMP changes from 145.87 msec to 172.93 msec which are 2.02 to 2.93 times lower with respect to DCF. Similarly, for hop length 6, with change in best of traffic from 1 to 9, we can observe 1.47 to 2.04 times lower delay for real-time traffic in SPQAMP with respect to DCF (as seen from Figure 3.36). It can be noticed that delay difference between SPQAMP and DCF is more with higher best-effort traffic and lower hop length. From Figure 3.33, with hop length 3, we can observe throughput improvements for real-time traffic in SPQAMP from 2.43 to 8.09% as compared to DCF. A similar improvements of throughput for real-time traffic from 2.10 to 16.46% can be seen from Figure 3.35 for hop length 6. We can notice that the throughput difference between SPQAMP and DCF is more with higher best-effort traffic load and at higher hop length. This shows suitability of SPQAMP as QoS-aware MAC scheme to provide strict service differentiation in chain topology with varying best-effort traffic loads.

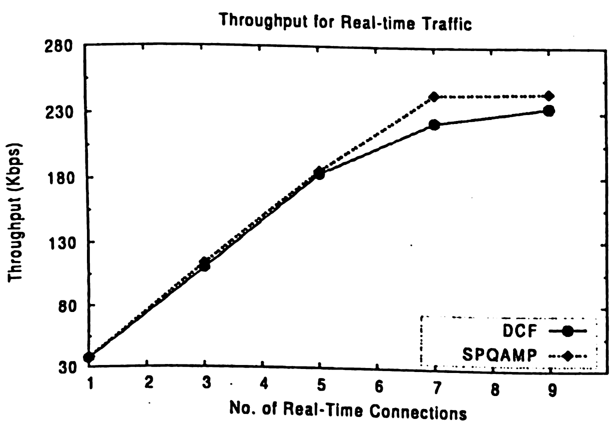


Figure 3.37: Throughput for Real-time traffic for a chain topology with hop length 4 and with 3 best-effort traffic (Experiment 3): SPQAMP vs DCF

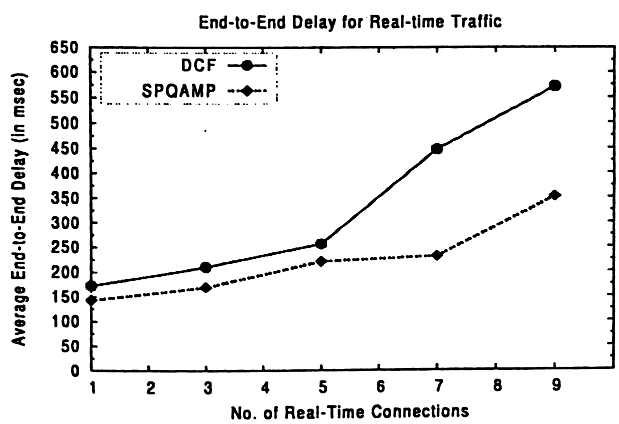


Figure 3.38: Delay for Real-time traffic for a chain topology with hop length 4 and with 3 best-effort traffic (Experiment 3): SPQAMP vs DCF

Figure 3.37 and Figure 3.38 show the throughput and delay of real-time traffic over varying number of real-time connections for *Experiment 3* of chain topology. It can be seen that with the increase in number of real-time traffic, delay in SPQAMP increases from 142.74 msec to 351.67 msec, whereas we can observe a rapid increase of delay from 172.51 msec to 570.65 msec in DCF. Therefore, SPQAMP can provide 21 to 94% improvement in delay as compared to DCF. As can be observed from Figure 3.37, with increase in number of real-time traffic, throughput for SPQAMP improves from 0.19 to 9.5% as compared to DCF. Therefore, SPQAMP is able to provide QoS through effective service differentiation in chain topology with varying load of real-time traffic in presence of fixed best-effort traffic load.

A Grid topology of size 6×6 (36 nodes)

In this topology, 36 nodes, numbered as 0 through 35, are arranged in a 6×6 grid with a grid spacing of 200 meters. We have generated a number of real-time connections from the nodes in the leftmost positions to the nodes in the rightmost position in the same row. Similarly, we have generated a number of best-effort flows from the nodes in the top row to the nodes in the bottom row of the same column. We used NOAH (No Ad Hoc routing) from [79] that supports fixed routes in grid topology. We carried out two different set of experiments to study the effectiveness of SPQAMP in providing QoS in terms of *throughput* and *delay* in the grid topology. In the first set of experiment, we kept the number of real-time traffic fixed at 3 and varied the number of best-effort connections from 1 to 7. In our second set of experiments, we kept the number of best-effort connections fixed at 3 and varied the number of real-time connections from 1 to 9. The simulation parameters are set as shown in Table 3.8.

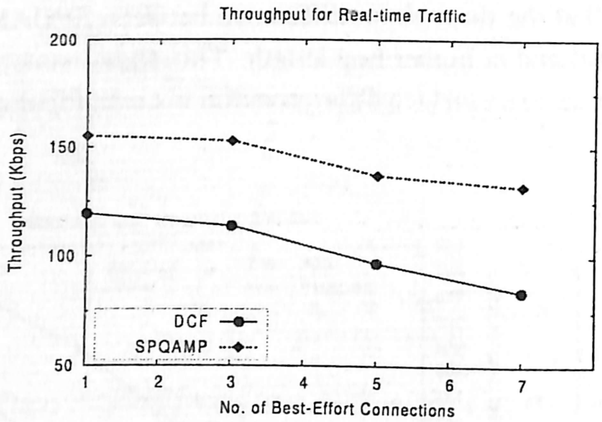


Figure 3.39: Throughput for Real-time traffic for a Grid topology with 3 real-time traffic (SPQAMP vs DCF)

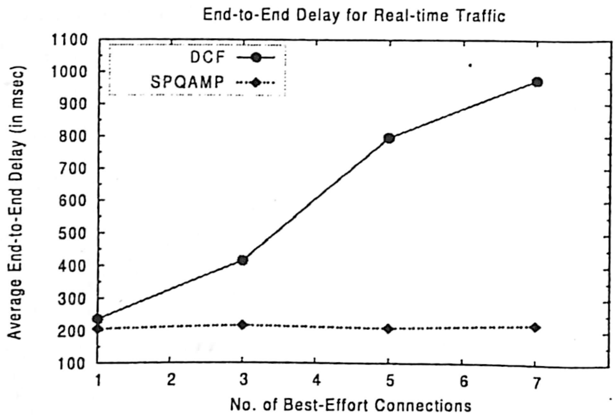


Figure 3.40: Delay for Real-time traffic for a Grid topology with 3 real-time traffic (SPQAMP vs DCF)

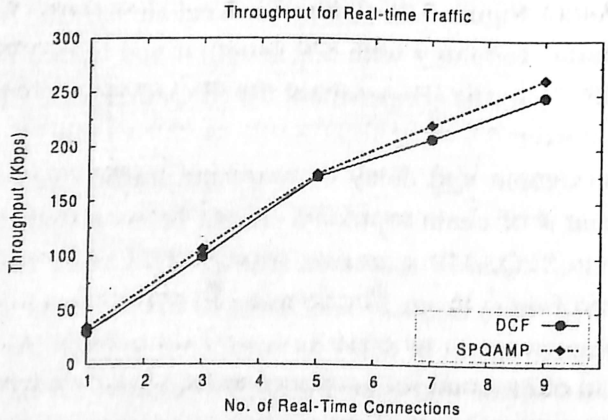


Figure 3.41: Throughput for Real-time traffic for a Grid topology with 3 best-effort traffic (SPQAMP vs DCF)

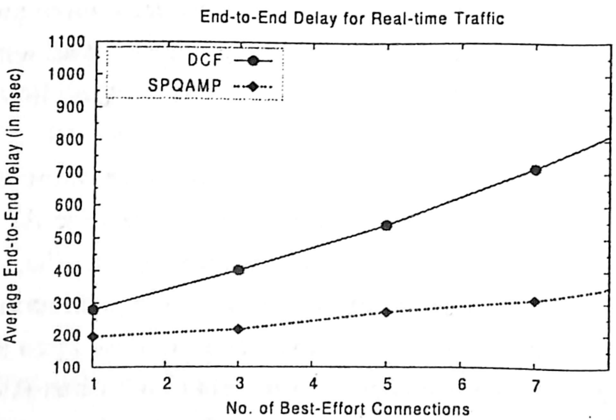


Figure 3.42: Delay for Real-time traffic for a Grid topology with 3 best-effort traffic (SPQAMP vs DCF)

The Grid topology with the traffic pattern mentioned above represents a very high interfering scenario, where real-time as well as best-effort flows encounter interference from both type of traffic flows. Due to strict non-overlapping contention windows set for nodes with real-time and best-effort data frames to transmit, and the strict priority based service differentiation, SPQAMP is able to support adequate level of QoS in terms of throughput and delay.

From Figure 3.39 and Figure 3.40, it can be seen that, with the increase in number of best-effort traffic, throughput and delay for DCF get affected much as compared to SPQAMP. It is mainly due to the high interference created in the network. SPQAMP provides a very good level of QoS to real-time flows due to its effective service differentiation through strict priority based MAC scheme as compared to DCF. With the increase of interference, due to increasing load of best-effort traffic, we can observe a marginal change of delay in SPQAMP from 205.68 msec to 223.06 msec, whereas a rapid increase in delay from 236.26 msec to 974.81 can be observed in DCF. This leads to a delay improvement of 1.148 - 4.37 times in SPQAMP as compared to DCF with the difference being increases with growing load of best-effort traffic (as shown in Figure 3.40). Similarly, we can also observe a marginal change in throughput for SPQAMP with increase in best-effort traffic load, which leads to a throughput improvement of 30.2 to 56.8% in SPQAMP as compared to DCF (as shown in Figure 3.39). we can observe similar poor performance by DCF in comparison with SPQAMP when the number of best-effort traffic is fixed at 3 (as shown in Figure 3.41 and Figure 3.42). In this case, we can observe 1.43 to 2.37 times less delay in SPQAMP as compared to DCF, with larger difference in higher traffic loads (as shown in Figure 3.42). Similarly, we can see 15.25 to 6.09% improvement in throughput for SPQAMP as compared to DCF (as shown in Figure 3.41). There is a marginal improvement of throughput (1.58%) for SPQAMP as compared to DCF for real-time traffic load of 5. These simulation results show the suitability of SPQAMP's strict service differentiation scheme for grid topology, which represents a scenario with very high level of traffic interference, to support QoS for real-time traffic.

A Random topology of 50 nodes

We have considered a random topology which consists of 50 nodes numbered as 0 through 49 which are arranged in a flat rectangular area of size 1000 m × 500 m. The nodes are allowed to move using random waypoint mobility model with a maximum speed of 10m/s and pause time of 0 second. We have taken AODV [85] as the routing protocol for such a dynamic environment. The total simulation time for the experiments are set as 500s. We carried out three different sets of experiments to study the effectiveness of SPQAMP in providing QoS in terms of *throughput* and *delay* in random topology under different load conditions both in the absence of background traffic and with background traffic. In the first set of experiments, we allowed only the real-time traffic to be present in the network where the number of real-time traffic are changed from 1 to 11. The source and destination of a real-time traffic are chosen randomly where a selected source starts generating real-time traffic randomly within the time period between 1.0s to 100.0s. In our second set of experiments, we kept the number of real-time connections fixed at 7 and vary the number of best-effort connections from 1 to 7. In this case, all the real-time connections are started before starting any best-effort connection. The third set of experiments are carried out by keeping the number of best-effort traffic fixed at 5, and varying the real-time connections from 1 to 11. In this experiment, all best-effort connections are started before starting any real-time connection. The other parameters are set as shown in Table 3.8. For each of the experiment, a total of 10 simulations are performed with 10 different scenarios and averages are taken as the results.

From Figure 3.43 and Figure 3.44, we can observe that SPQAMP is able to maintain similar throughput and delay as compared to DCF in absence of best-effort (or low priority) traffic. As mentioned earlier, SPQAMP with only high priority traffic works similar to basic DCF (due to lack

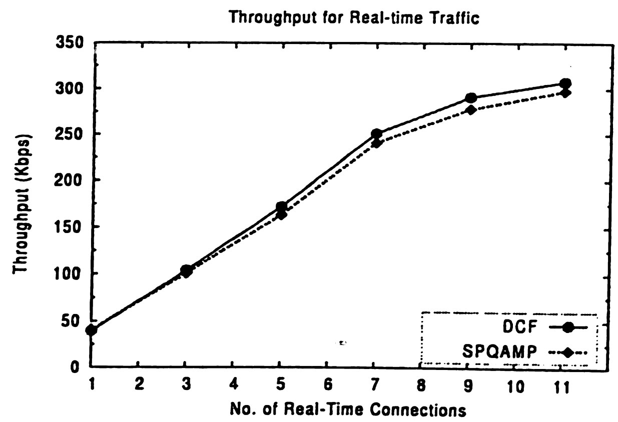


Figure 3.43: Throughput for Real-time traffic for a random topology with only real-time traffic (SPQAMP vs DCF)

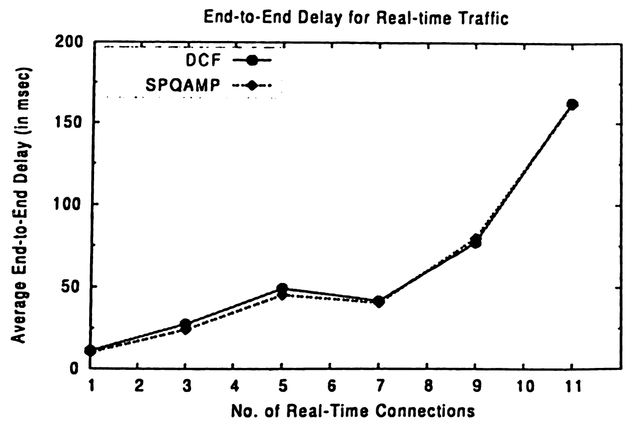


Figure 3.44: Delay for Real-time traffic for a random topology with only real-time traffic (SPQAMP vs DCF)

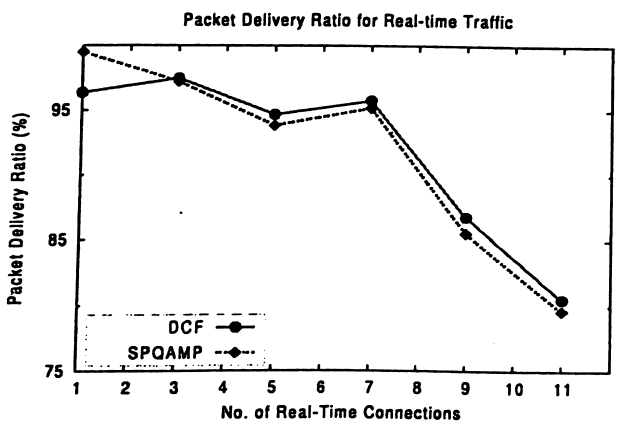


Figure 3.45: Packet Delivery Ratio for Real-time traffic for a random topology with only real-time traffic (SPQAMP vs DCF)

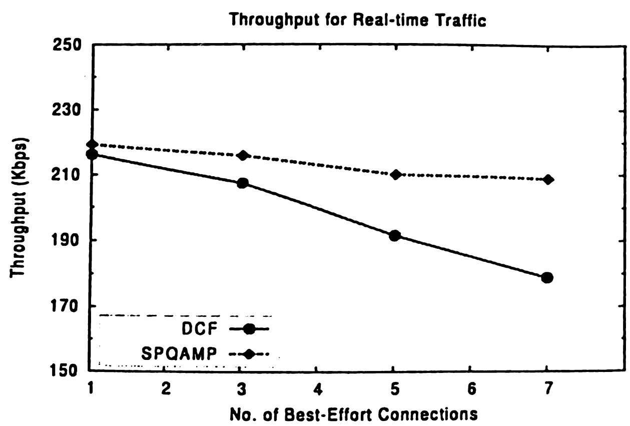


Figure 3.46: Throughput for Real-time traffic for a random topology with 7 real-time traffic (SPQAMP vs DCF)

of strict service differentiation). As can be observed from Figure 3.44, for number of real-time traffic up to 7, SPQAMP has slightly lower delay as compared to DCF. But, with further increase in load, delay in SPQAMP exceeds that of DCF by a small margin. This is mainly due to smaller values of $CWmax$ and retry limit used for real-time traffic in SPQAMP as compared to DCF, which also leads to slightly smaller values of PDR for SPQAMP in higher loads as compared to DCF (as shown in Figure 3.45). Similarly, We can observe a marginally smaller throughput as compared to DCF for number of real-time traffic from 3 to 11 (as seen from Figure 3.43). This shows the performance compatibility of SPQAMP and DCF in the presence of only real-time traffic.

From Figure 3.46 and Figure 3.47, we can observe that with growing load of best-effort traffic, throughput and delay for real-time traffic are not affected much in SPQAMP. Whereas we can observe severe degradation of throughput and delay performance for real-time traffic in DCF with growing load of best-effort traffic. With increase in best-effort traffic load, delay in SPQAMP changes from 61.44 msec to 75.87 msec, whereas we can observe a rapid change of delay from 80.57 msec to

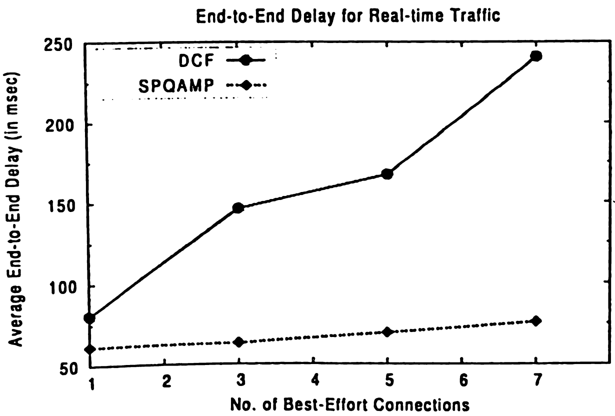


Figure 3.47: Delay for Real-time traffic for a random topology with 7 real-time traffic (SPQAMP vs DCF)

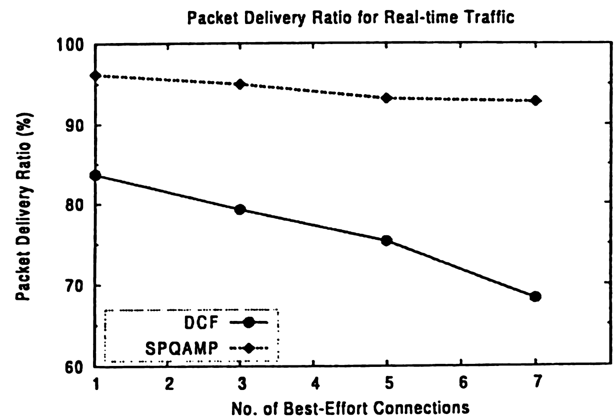


Figure 3.48: Packet Delivery Ratio for Real-time traffic for a random topology with 7 real-time traffic (SPQAMP vs DCF)

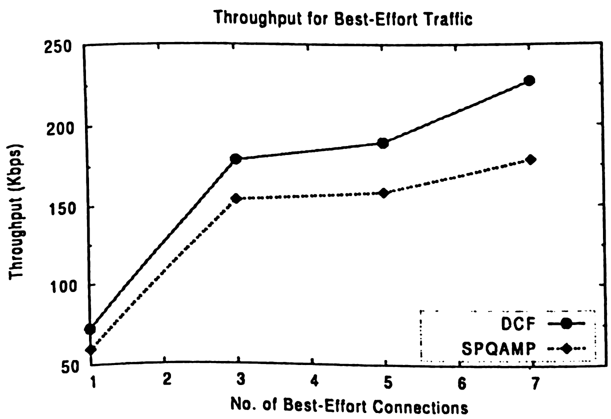


Figure 3.49: Throughput for Best-Effort traffic for a random topology with 7 real-time traffic (SPQAMP vs DCF)

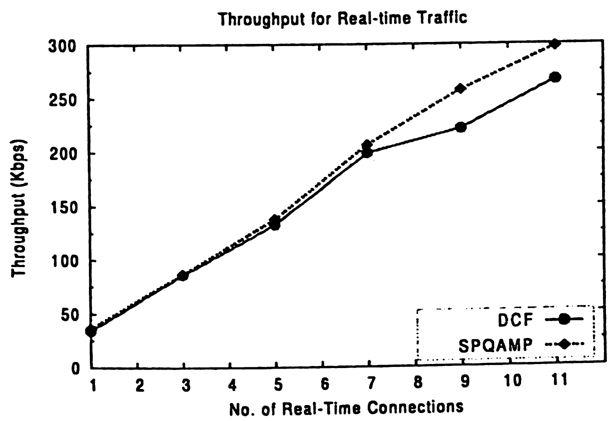


Figure 3.50: Throughput for Real-time traffic for a random topology with 5 best-effort traffic (SPQAMP vs DCF)

240.69 msec in DCF, this leads to 1.31 to 3.17 times lower delay in SPQAMP as compared to DCF (as shown in Figure 3.47). Similarly, with growing best-effort traffic, we can see a small changes of throughput from 219.25 Kbps to 208.24 Kbps in SPQAMP, whereas the corresponding throughput changes in DCF are from 216.28 Kbps to 178.92 Kbps, which gives 1.37 to 16.39% lower throughput in DCF as compared to SPQAMP (as shown in Figure 3.46. This performance improvement in SPQAMP is mainly due to the effective service differentiation provided by strict priority scheme in SPQAMP. SPQAMP also maintains lower delays for varying real-time traffic (and fixed best-effort traffic) loads, where delay increases from 36.05 msec to 174.01 msec with increase in real-time traffic loads from 1 to 11, which are 4.24 to 2.41 times lower as compared to DCF (as shown in Figure 3.51). From Figure 3.50, we can also observe throughput improvements for real-time traffic in SPQAMP from 6.18 to 11.56% as compared to DCF. Similarly, SPQAMP maintains a higher PDR for both the situations of varying best-effort and real-time traffic loads, where the improvement is more for the first case (as shown in Figure 3.48 and Figure 3.52). Due to effective service differ-

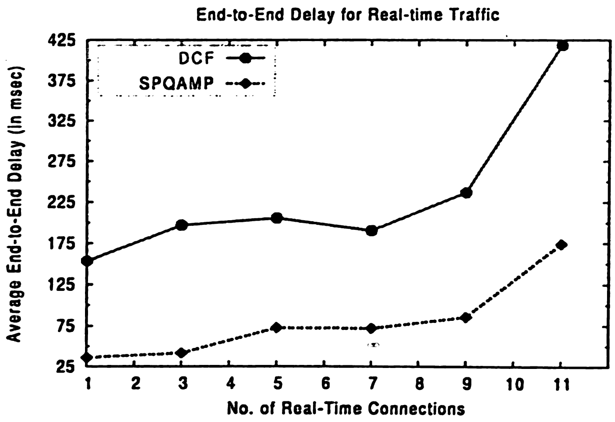


Figure 3.51: Delay for Real-time traffic for a random topology with 5 best-effort traffic (SPQAMP vs DCF)

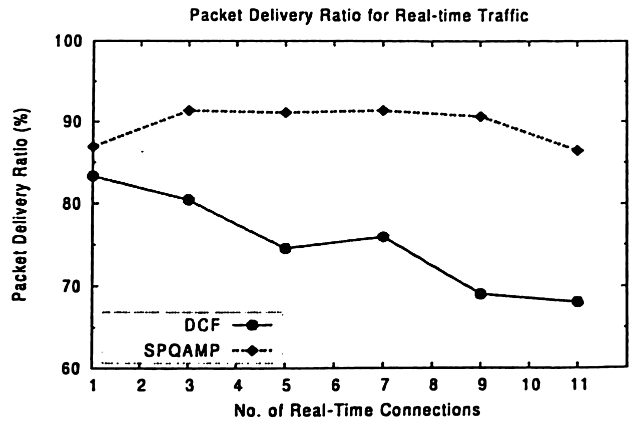


Figure 3.52: Packet Delivery Ratio for Real-time traffic for a random topology with 5 best-effort traffic (SPQAMP vs DCF)

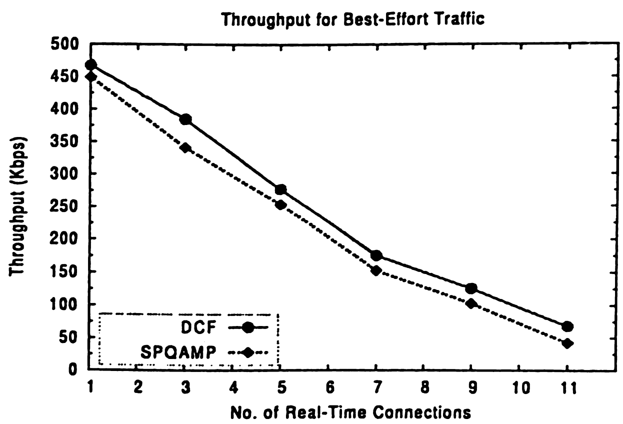


Figure 3.53: Throughput for Best-Effort traffic for a random topology with 5 best-effort traffic (SPQAMP vs DCF)

entiation through strict priority scheme in SPQAMP, we can observe degradation of throughput for best-effort traffic in the range of $13-21\%$ and $4-38\%$ in scenarios of varying best-effort traffic and varying real-time traffic respectively as compared to DCF (shown in Figure 3.49 and Figure 3.53). It is to be noted that despite the use of strict service differentiation in SPQAMP, we can observe some level of service for best-effort traffic. This is mainly due to (i) the possibility of channel access by best-effort traffic while contending with real-time traffic in non-saturated condition, (ii) post backoff scheme in SPQAMP adopted from DCF to eliminate the possibility of channel capture by immediate access and (iii) channel access possibility by best-effort traffic without any contention from real-time traffic in a random topology with random traffic distribution.

From these analysis, we can conclude that SPQAMP outperforms IEEE 802.11 DCF in terms of delay and throughput for real-time traffic in all types of topology for MANETs including - single hop, chain, grid and random topology. The performance gain in SPQAMP for real-time traffic is very high in the presence of best-effort traffic as compared to DCF. Although in non saturated condition, there is no starvation for best-effort, the performance of best-effort traffic suffers in terms of longer delays. Therefore, SPQAMP can be taken as a MAC protocol for supporting QoS in MANETs where real-time traffic needs strict service differentiation.

3.5.6 Summary on SPQAMP

We have proposed SPQAMP as a QoS-aware MAC scheme for achieving a strict service differentiation between real-time and best-effort traffic, where real-time and best-effort traffic are represented by high priority and low priority traffic respectively. SPQAMP ensures that nodes with real-time data frames always win the next transmission right when contending with nodes with best-effort traffic, by adopting the following mechanisms into standard IEEE 802.11 DCF. These are :-

- In SPQAMP, a non-overlapping range of contention windows are assigned to nodes with real-time and best-effort data frames to transmit respectively. This non-overlapping property is maintained irrespective of the backoff stage.

- To avoid overlapping of backoff counter during countdown, freezing of backoff timer for best-effort traffic is replaced with a resetting of backoff timer mechanism.

The analytical results and their validations through simulation studies confirm that, in single hop scenarios under saturation condition, real-time traffic are completely unaffected by the presence of best-effort traffic. To eliminate channel capture by a node through immediate access, we have adopted post-backoff procedure from standard DCF in the implementation of SPQAMP. This leads to some amount of channel access by low priority traffic in the network. Simulation results in multihop environments show the potential of SPQAMP as a QoS-aware MAC scheme to be used in mobile ad hoc networks.

3.6 Conclusion

In this chapter, we have proposed two dynamic priority based schemes for resolving MAC layer contention to provide enhanced level of service differentiation in mobile ad hoc networks. PQAMP is based on assigning priorities to nodes based on their roles and type of traffic to handle, whereas SPQAMP classify nodes based on the type of data frame they have to transmit. In both the schemes, channel access priorities are realized through assigning contention windows of different ranges for nodes with different priority classes. In PQAMP, to ensure different channel access priorities for different priority classes, at each backoff stage, a non-overlapping range of contention window is assigned to nodes with different priority classes, with lower range assigned to higher priority. To reduce the possibility of priority inversions that are possible in PQAMP, SPQAMP assigns non-overlapping contention windows to nodes with different priority classes irrespective of their backoff stage. Further, in SPQAMP, freezing of backoff counter mechanism in DCF is replaced with a resetting of backoff counter mechanism for nodes with best-effort traffic. From both analytical and simulation results, we draw the following conclusions

- Both PQAMP and SPQAMP outperform DCF in terms of service provided to real-time traffic in MANETs

- Level of service differentiation is more in SPQAMP as compared to PQAMP
- PQAMP is best suited for multihop environment, particularly in low to medium loads
- SPQAMP always performs better in single hop environment due to its strict service differentiation as compared to PQAMP
- With growing load of best-effort traffic, performance degradation of real-time traffic is less in SPQAMP as compared to PQAMP in almost all situations.
- PQAMP is more effective in providing some minimum level of service for best-effort traffic while maximizing the network utilization
- SPQAMP is more effective in applications, where the service provided to real-time traffic should not be affected by the presence of best-effort traffic

After working on priority based MAC protocols to improve QoS support at MAC layer, our next investigation is for a QoS-aware routing protocol which takes route stability as an additional constraint to improve durability/reliability of the discovered QoS routes. In the next chapter, we discuss about the design of our route stability based QoS routing (RSQR) protocol for MANETs.

Chapter 4

A Route Stability based QoS Routing (RSQR) for Mobile Ad Hoc Networks

4.1 Introduction

Ensuring a data path to be valid for sufficiently longer period of time is a very difficult problem in MANET due to its highly dynamic nature. In this chapter, we have proposed a Route Stability based QoS Routing (RSQR) protocol for Mobile Ad Hoc Networks, which have used route stability as an additional QoS metric along with throughput and delay, to improve the durability of the discovered QoS path.

It is found that, though there exist a number of on-demand QoS routing algorithms [16, 19, 47, 86, 115] in MANETs to compute routes with throughput and delay constraints, none of them address the issue of reducing the path breakage during data transmission. Most of the QoS routing algorithms proposed for Ad Hoc networks are based on on-demand routing [15, 73]. In these routing algorithms, when a route is broken, route recovery and maintenance procedures are initiated. These procedures consumes substantial amount of resources (in terms of wireless bandwidth, power, processing capacity at nodes) and introduce extra delay in QoS flows. To reduce the possibilities of route breaks during data transmission, it is important to find routes that endure longer time. Shortest path route has short lifetime, especially in highly dense ad hoc networks, due to selection of longer links (also called edge effects [64]). The longer is the physical distance of a link, the larger is the packet transmission error rate, as it approaches the transmission range of mobile nodes. In this case, a small movement of any of the nodes involved may cause packet loss due to link failure. Furthermore, packets are more likely to be lost due to external environmental factors like white noise and wireless interference if the signal strength is not very strong. Link stability indicates how stable the link is and how long it can support communications between two nodes. Stability of links can be estimated using many parameters like - signal strength [12, 29, 54, 64, 104], pilot signals [105], relative speed between two nodes forming the link [70], link duration distributions [35, 90] and remaining battery power of the nodes etc. Stability or lifetime of a route is determined by the number of links that compose the route and the stability of each link in the route. It can be shown that the probability of route failure can be reduced by reducing either the link failure rate or the number of links that compose the route. But in ad hoc networks, trying to reduce one of these parameters will result in increase in the other parameter. This means, most of the route searching process will either ends up with shortest path with longer (hence unstable) links or stable route (shorter links) with longer route. Therefore, there should be some compromise in selecting shortest routes and stable routes. It is important to note that delay constrained routes avoid selecting very long routes.

Here, we have developed a route stability based QoS-aware routing protocol with the following key components. *(i)* We have proposed a model for computing link stability based on measurements of received signal strengths of two successively received packets from a neighbor. Because, link stability will be in the interval $[0, 1]$, we can use it as the probability of survival of the link in the near future. Therefore, the route stability is calculated as a product of link stability of all the links which constitute the route under consideration. *(ii)* A QoS-aware route discovery with throughput and delay constraints. The routing protocol incorporates the stability model (as mentioned in *(i)*) in route discovery process to find QoS routes with highest stability. *(iii)* A hop by hop admission control during route discovery. *(iv)* A QoS violation detection and recovery mechanism.

Our method is based on an enhancement of AODV which is one of the most widely used on-demand routing protocol in MANETs. Results of our experiments show performance improvements in terms of packet delivery ratio, control overhead and average end-to-end delay in comparison with a QoS routing protocol proposed by Q. Xue and A. Ganz [115].

The rest of the chapter is organized as follows. Section 4.2 discusses the related works. Section 4.3 defines the problem and section 4.4 gives a brief overview of our route stability model (RSM). The proposed Route stability based QoS Routing protocol in MANETs is presented in section 4.5, and its performance is evaluated and compared with AQOR[115] in section 4.6. Finally, section 4.7 concludes this chapter.

4.2 Related Works

AODV[85], due to its minimum control overhead, is mostly suitable for throughput constrained routing in MANETs. A QoS extension of AODV with throughput and delay constraints, called QAODV (QoS-aware AODV), is defined in the *Internet draft* [86] for QoS routing in MANETs. The draft for QAODV neither specify the methods for calculating available bandwidth and forwarding delay nor the route recovery process due to throughput violation. Hence, no practical implementation of QAODV is still available. AQOR[115] is a reservation-based, on-demand QoS-aware routing protocol based on AODV. It has incorporated the following features for providing QoS - (i) available bandwidth estimation and end-to-end delay measurement, (ii) bandwidth reservation, and (iii) adaptive route recovery. Temporary or soft state reservation is used to free the reserved resources efficiently at each node when the existing routes are broken. The adaptive route recovery procedure includes detection of broken links and QoS violations and triggering of route recovery at the destination. Similar AODV-based QoS routing protocols are proposed in [19, 47]. With the knowledge of available bandwidth and end-to-end delay, the smallest delay path with sufficient bandwidth is chosen as the QoS route in [19, 86, 115]. Where as [47] selects the lowest cost path among the feasible QoS paths. Sivakumar et al [99] proposed a QoS routing which dynamically establishes a core network and then incrementally propagates the link state information of stable high-bandwidth links to all the nodes in the core. Route computation is limited only to core nodes using their local states. A proposal called courtesy piggybacking [63] have tried to alleviate the conflict between throughput and fairness for different prioritized traffic in ad hoc network. By exploiting the channel dynamics and stochastic traffic behaviour in the network, higher priority traffic helps the lower priority traffic by sharing unused residual bandwidth with courtesy. It mainly targeted at elimination of bandwidth starvation problem in lower priority traffic. The Proposal in [119] takes into account the effect of self-traffic to accurately estimate the transmission time along a candidate QoS path. To predict the quality of a path after the admission of a real-time flow, they use a new traffic-aware routing metric called Path Predicted Transmission Time (PPTT), which basically estimates the packet service time and the queuing delay for the path. None of the QoS routing protocol mentioned above considers selecting QoS-aware route with highest stability which is essential to reduce the possibility of path breakage during data transmission.

From the best of our knowledge, no stability based QoS routing with throughput and delay constraints is available in the literature. Though an attempt is made by Zhu et al in [123], their method considers only delay as the QoS parameter and it was developed as an enhancement of Ticket Based Probing [16]. Ticket based probing is not a widely accepted method for QoS-aware routing in MANETs due to its inherent problems. Among the link stability based routing without QoS, Signal Stability based Adaptive Routing (SSA)[29] distinguishes strongly connected link from weakly connected link by estimating link stability based on signal strength. To obtain route with higher stability, it prefers a route with shorter links than a shortest route with longer links. Their method is not valid for nodes with long pause time. Associativity Based Routing (ABR)[105] uses periodically sent pilot signals to determine the link stability. It assumes that after a threshold period, all nodes move with similar speeds and directions and tend to stay together. This assumption is not accurate or acceptable because not all nodes follow the mobility pattern assumed by ABR. A proposal in [102] reduces the size of control packet in ABR which results in performance improvements in terms of average throughput and delay. Link Life Based Routing Protocol (LBR)[70] estimates link lifetime based on estimated distance and maximum speed of nodes. Proactive maintenance is proposed for route recovery due to link failure. However, estimated route lifetime is not valid when a route is broken and therefore, backup route may be unstable. Advanced Signal Strength based link stability estimation model (ASBM)[64] is an enhancement of SSA [29], where direction of node movement is taken into account. That is, links with increasing signal strengths are considered as stable. ASBM has the drawback that it assumes the link between two nodes as unstable that are initially very close to each other and then immediately start to move slowly away from each other. Actually, it might take a long time before the two nodes come out of communication range of each other. In [90], a statistical model for Link stability is proposed based on the assumption that link stability in different network environments follows certain distributions. However, statistical model for link stability based on node movement pattern is not adequate for general applications, because the mobility patterns of nodes cannot be known a priori. In [12], authors propose an Ad Hoc Routing based on the stability of Routes to mitigate the possibility of route recovery by the selection of more stable paths (paths with expected longer lifetime). Stability of a route is defined as the minimum signal strength (MSS) of all the links of the route. If multiple route replies are allowed, the source can select the path based on the maximum value of MSS. Route Fragility based Route Selection in MANETs [104] is based on the idea that a less dynamic route lasts longer. The method uses a metric called Route Fragility Coefficient (RFC) to describe the dynamic nature of a route. RFC estimates the rate at which a given route expands or contracts. Lower value of RFC gives more static (or less fragile) route. This metric is used as a basis for route selection such that route discovery yields routes that last longer and hence increases throughput while reducing routing overhead. In [35], adaptive metrics are developed to identify stable links based on the link duration distributions. They compute the residual lifetime of a link with age a from the average lifetime of all links with lifetime durations above the age a. Lifetimes (durations) of the links can be measured and recorded in a histogram by periodic hello message transfer like in the case of ABR. So, link stability estimation is based on past information and the estimation will be valid only when the nodes follow similar stable mobility patterns. Stability and Hop-count based Algorithm for Route Computation (SHARC)[101] considers residual lifetime as stability metric of links and hop count for computing path in MANETs. It computes path stability through a simple histogram based estimator like in [35] and tries to find the most stable route among all shortest hop routes. Entropy-Based Long-Life Distributed QoS Routing (EBLLD)[17] algorithm was designed to select the long-life path satisfying the delay requirement with the help of entropy metrics. In their method, a different representation called entropy metric is used to define the stability metric of a link. The entropy for a link is defined as a function of the relative positions, relative velocities, and the transmission ranges of the nodes connected by the link under consideration. Though their approach has the potential to be more accurate than the methods using statistical model, however, this comes at the price of assuming that each node has GPS like devices which is not feasible and unrealistic in most of the ad hoc network environment. The PANDA [57] routing protocol is based on positional attributes such as location and velocity information in determining the rebroadcast delay while finding a longer-lived route with smaller hop counts. Although PANDA can be used to achieve routing goals like power conservation and load balancing, it is not designed for satisfying throughput and end-to-end delay of real-time applications.

Our proposal uses a resource reservation based QoS routing with throughput and delay requirements, and apply the route stability metric to improve the durability of established route such that the probability of the reservation to remain valid for duration of data transmission is more. Route stability model we have developed - is based on simple calculations on measured received signal strengths obtained from link layer, has less control overhead, require no extra hardware, and has the potential to give good approximation on link stability in most of the real life mobility scenarios.

4.3 Problem Formulation

We can model the topology of a MANET as an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges (links) connecting the nodes. Each node has a unique identifier, by which routing protocol can identify it. Note that the links of a MANET can be broken and formed at any time owing to the mobility of the nodes. Any link $e = (u, v) \in E$ has an associated residual bandwidth $B(e)$, delay $D(e)$ and link stability $LS(e)$. A path between two arbitrary nodes u and v is given as $P(u, v) = (u, e(u, x), x, e(x, y), y, ..., e(z, v), v)$. It can be emphasized that a path between any two nodes is a set consisting of all possible paths between them. Formally, $P(u, v) = \{P_0, P_1, ..., P_n\}$ where each P_i is a candidate path between u and v. Delay, bottleneck bandwidth and stability for a path P_i are defined as follows.

$$Delay(P_i) = \sum_{e \in P_i} D(e) \tag{4.1}$$

$$BotBand(P_i) = min\{B(e_0), B(e_1), ..., B(e_n)\} \tag{4.2}$$

$$Stability(P_i) = \prod_{e \in P_i} LS(e) \tag{4.3}$$

where $e_0, e_1, ..., e_n$ are the edges making up the path.

Now, we can state the problem as - **"To find the most stable path from source s to destination d which satisfies both minimum bandwidth constraint(Δb) and maximum delay constraint (Δd)."**

We call the path as QoS Aware Stable Path (QASP). Suppose, $P(s,d) = \{P_0, P_1, ...P_n\}$ for a given source s and destination d. Now, we can define the problem mathematically as follows:

$$QASP(s,d) = P_i \tag{4.4}$$

$$\begin{aligned} with\ Stability(P_i) &= max\{Stability(P_0), Stability(P_1)...Stability(P_n)\} \\ \forall i\ BotBand(P_i) &\geq \Delta b \\ \forall i\ Delay(P_i) &\leq \Delta d \end{aligned}$$

Therefore, the objective of RSQR is to find a path with highest stability among all the candidate paths from source to destination that have the bottleneck bandwidth of at least the requested minimum bandwidth and the total end-to-end delay experienced by a data packet is at most the requested maximum delay.

4.4 Route Stability Model (RSM)

We have proposed a route stability model with the following characteristics. Firstly, it considers node mobility and signal strengths for computing the probability of link failure rather than using probability distribution of link lifetimes. Hence, it has the potential to be more accurate. Secondly, it does not require geographic location information for nodes which needs GPS like devices to determine relative positions and mobility speed of nodes. Thirdly, it uses signal strength values from MAC layer to compute link stability and uses less control overhead without enforcing periodic pilot packet exchange to compute link stability. Fourthly, it considers stability values in the range [0,1], unlike binary level stability values considered in most of the existing methods. Lastly, it does not depend on any particular mobility pattern. In fact, it is applicable to any real life node mobility pattern.

Though it is very difficult to predict the exact probability that a link will be broken in the near future, it is possible to estimate the relative stability of the link based on recent and current received signal strengths over the link. The received signal strength is a MAC layer information used by routing layer through cross-layer interaction. When a node i receives a route request from the previous hop, say j, MAC layer of node i can measure the signal strength of the route request packet ($SS^1_{i,j}$). The recent received signal strength ($SS^2_{i,j}$) value over the link from node j is obtained from a table called *Neighbor Information Table* (NIT). Each node stores the recent received signal strength ($SS^2_{i,j}$) value for each of the neighbor and updates it periodically. Depending on whether $SS^2_{i,j}$ value for node j is available in NIT or not, node i will compute the link stability according to our model. After the estimation of stability of the link between node i and node j, node i updates $SS^2_{i,j}$ value for node j with $SS^1_{i,j}$. Our model for link stability estimation is given by the following rules (Rules 1 to 3). Here, rules are tried one after another in the sequence from Rule 1 to Rule 3 until a matching rule is found. That is, if the condition specified in a rule is evaluated to *true*, then the link stability estimation is done using the given rule and the remaining rules are skipped. Otherwise, the following rules are tried in the same way. In our routing protocol, routing packet with signal strength $< Thr_2$ are dropped through admission control. Therefore, link stability estimation for links with signal strength $< Thr_2$ are not shown below.

- $L_{i,j}$: The link between nodes *i* and and *j*.
- $RxThr$: Reception Threshold for received signal strength, assumed to be same for all nodes.
- $SS^1_{i,j}$: Current received Signal strength for the link between nodes *i* and *j*.
- $SS^2_{i,j}$: The Received Signal Strength for the link between nodes *i* and *j* at recent time; It is stored in NIT and periodically updated.
- Thr_1: Threshold for received signal strength with the property that - when a node receives a packet from a node with signal strength $\geq Thr_1$, the link between the nodes can be assumed to be very strong (stable).
- Thr_2: Threshold for received signal strength, below which the signal strength will not be strong for reception. We have $Thr_1 > Thr_2$.
- $DSS_{i,j}$: The differentiated signal strength between nodes *i* and *j* to indicate whether the signal strength is going stronger or weaker.
- u_1, u_2: u_1 is a threshold for DSS to handle small variations in signal strength due to temporary environmental factors like fading and interference. On the other hand, $u_2(> u_1)$ is used for detection of situations in which two nodes are going away from each other very fast.
- $LS_{i,j}$: Link stability between nodes *i* and *j* and its value $\in [0, 1]$.
- $LU_{i,j}$: Link Uncertainty between nodes *i* and *j*, means the link's stability can not be determined due to lack of its recent signal strength value $SS^2_{i,j}$ in NIT. It is a binary flag.
- ***Rule 1:*** If $SS^1_{i,j} \geq Thr_1$ then nodes *i* and *j* are very close and $L_{i,j}$ is considered to be sufficiently stable so that we set $LS_{i,j} = 1$ and $LU_{i,j} = 0$.
- ***Rule 2:*** If $SS^1_{i,j} \geq Thr_2$ and node *j* is a new neighbor of node *i*, that is, node *j* was not the neighbor of node *i* recently, then we set $LU_{i,j} = 1$.
- ***Rule 3:*** If $SS^1_{i,j} \geq Thr_2$ and $SS^2_{i,j} \geq RxThr$ then node *i* and node *j* may be - approaching each other, leaving each other, or stay stationary based on the value of differentiated signal strength, $DSS_{i,j} = SS^2_{i,j} - SS^1_{i,j}$. In all the rules from $3(a)$ to $3(c)$, we set $LU_{i,j} = 0$.
- ***Rule 3(a):*** If $DSS_{i,j} \leq u_1$, we set $LS_{i,j} = 1$.
- ***Rule 3(b):*** If $DSS_{i,j} > u_1$ and $DSS_{i,j} \leq u_2$, we set $LS_{i,j} = \frac{(u_2 - DSS_{i,j})}{(u_2 - u_1)}$.
- ***Rule 3(c):*** if $DSS_{i,j} > u_2$, we set $LS_{i,j} = 0$.

Hence, the path stability of P, PS_P, is defined as follows: $PS_P = \prod_{e \in P} LS_e$, Where e is a link in the path P, and P is composed of the links connecting the source to the destination. Similarly, Path Uncertainty of P, PU_P, is defined as the number of uncertain links in the path P from source to destination. That is, $PU_P = \sum_{e \in P} LU_e$. So higher is the value of path stability, the higher is the possibilities of selecting the path as the most stable path provided that the path has the smallest or admissible value for path uncertainty.

4.4.1 An example showing Link stability estimation

Figure 4.1 shows the estimation of link stability using our model for a link formed by connecting two nodes *Node 1* and *Node 2*. Here we have assumed that *Node 1* remains stationary (which is shown at the centre of the concentric circles) and *Node 2* starts moving from initial position '*a*' and follows a path denoted by the positions in the sequence '*a*'→ '*b*'→ '*c*'→'*d*'→'*e*'→'*f*'→'*g*'→'*h*' →'*i*' respectively. The positions on the path represents the time instance when we have calculated the link stability of the link connecting *Node 1* and *Node 2*. We denote the area around *Node 1* where the signal strength is greater than or equal to Thr_1 as *Area A*. Any position of *Node 2* within this area will form a stable link to *Node 1* with Link Stability value 1.0 (according to *Rule 1*). In Figure 4.1, position of *Node 2* at '*f*' corresponds to a link with stability value 1.0. The area around *Node 1* where signal strength is greater than or equal to $RxThr$ but smaller than Thr_2 is represented by the *Area C*. Any node position in this area will form a link with *Node 1* with stability value of 0.0. Links of this kind are excluded from consideration in forwarding route request/reply packet in our QoS aware routing protocol. Positions of *Node 2* at '*a*', '*h*' and '*i*' correspond to link stability value of 0.0. Finally, the area around *Node 1* where signal strength is greater than or equal to Thr_2 but less than Thr_1 is represented by *Area B*. Any node position within this area will have a link to *Node 1* with stability value determined from both previously received and currently received signal strengths. *Rule 3(a)*, *3(b)* or *3(c)* of our stability model will be applicable based on the difference in signal strength in the two successive positions. Positions of *Node 2* at '*b*', '*c*', and '*e*' correspond to applications of *Rule 3(a)* and hence Link stability value of 1.0. In positions '*d*' and '*g*' based on the chosen values for u_1 and u_2, we may get application of *Rule 3(a)*, *3(b)* or *3(c)*. It can be seen from the figure that, we should select u_1 and u_2 in such a way that position '*d*' correspond to application of *Rule 3(b)* and position *g* should correspond to application of *Rule 3(c)*. Therefore, proper selection of the values for u_1 and u_2 are very important to get more accurate link stability prediction. In our routing algorithm, values for u_1 and u_2 have been heuristically selected through a broad set of simulations run with varying parameter set under NS-2 [80].

4.5 Route Stability based QoS Routing (RSQR)

In the proposed routing protocol, we have used a new QoS parameter called *Route Stability* along with *throughput* and *delay*. *Route Stability Model (RSM)* explained in the previous section (Section 4.4) combines the individual link's stability behaviour to present a unified picture of the route stability. Our routing algorithm forwards route request through all the feasible paths (means satisfying both delay and throughput requirements of the application) from source to destination avoiding very weak links during this forwarding process. After receiving the first route request, the destination node waits for a fixed small time interval, called Route Reply Latency (RRL), for more route request

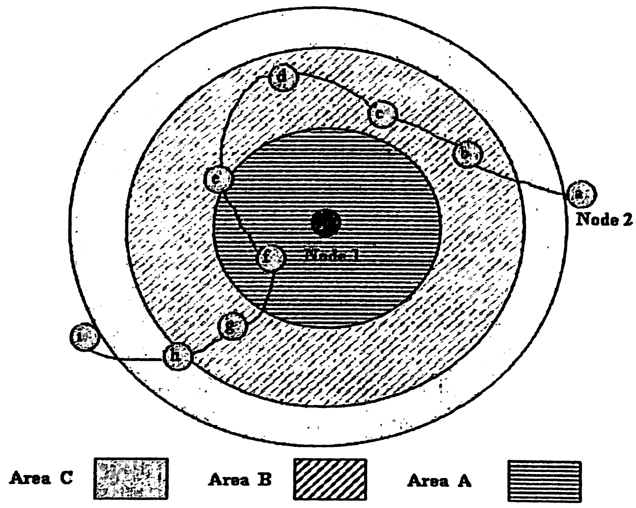


Figure 4.1: An example showing link stability estimation between Node 1 and Node 2

packets to arrive. Destination selects, among all feasible paths the one with the highest route stability value to reply to the source. Therefore, use of route stability during route discovery yields route that last longer and hence increases throughput and reduces delay while reducing routing overhead. The detail routing algorithm is discussed in the following sub sections.

4.5.1 Assumptions

We have made no assumptions about the global state of the network at the nodes. Instead, we have proposed an on-demand distributed routing algorithm where all the nodes contain only information about their one hop neighbors. The following network model assumptions are made.

- Neighborhood is commutative property i.e. if node A can hear node B this implies node B also hear node A. This means wireless links are symmetric
- CSMA/CA like MAC protocol is used for unicast communication and it solves the hidden terminial problem with the help of RTS-CTS control packets
- There is a close interaction between MAC layer and Network layer
- Combinatorial stability of network is assumed, it means topology changes occur sufficiently slowly to allow successful propagation of all topology updates as necessary
- Hello intervals to update neighbor information is reasonable to capture the dynamics of the network
- Transmission range and carrier sensing range are assumed to be same for available bandwidth calculation at a node
- Initial route discovery latency is tolerable in applications

4.5.2 QoS parameters

The Quality of Service parameters we have used in RSQR are the minimum throughput requirements, B_{min}, and maximum delay, D_{max} of the applications. RSQR uses a QRREQ (QoS Route Request) and QRREP (QoS Route Reply) with the QoS and route stability related fields along with the fields used in RREQ and RREP of AODV [85]. RSQR allows mobile nodes in a MANET to specify as part of the route request, QoS requirements that a route to the destination must satisfy. In particular, QRREQ includes minimum throughput (B_{min}) and maximum delay (D_{max}) requirements of the applications. To facilitate measurements to be accumulated for end-to-end delay, RSQR also provides an accumulated value for delay called $ADELAY$. In addition, to support route stability measurements to be accumulated for a complete path, RSQR includes two values - APS (Accumulated Path Stability) and APU (Accumulated Path Uncertainty).

4.5.3 Control packets used in Routing

Before we discuss the route discovery and maintenance process, we first briefly describe about control packets used in the proposed routing protocol.

- ***QRREQ***: This packet is basically AODV Route Request (RREQ) packet with QoS extension, which is used for sending route request in the proposed protocol. It contains QoS and route stability related fields along with the fields used in RREQ packet in AODV.

- ***QRREP***: This packet is basically AODV Route Reply (RREP) packet with QoS extension, which is used for sending route reply in the proposed protocol. It contains QoS and route stability related fields along with the fields used in RREP packet in AODV.

- ***RUPD***: This is Route Update packet for QoS route recovery from destination to source, performs similar task like QRREQ, but is treated as QRREP both at the intermediate node and at source node. It contains similar fields like QRREQ and QRREP packets. Time to Live (TTL) field of RUPD is set to hop count of already broken route plus a small positive constant to limit the scope of broadcasting of RUPD packet.

- ***RERR***: This is the Route Error packet generated after detecting link lost by an intermediate node. It is similar to the Route Error (RERR) packet used in AODV.

- ***HELLO***: This Hello packet is required for Neighborhood maintenance, one hop delay measurement, and providing bandwidth reservation information. It contains a special field indicating whether immediate reply is required or not. *HELLO* packet is always sent with TTL set as 1.

4.5.4 Tables used in Routing

In order to implement the routing algorithm, we have used the following tables in each node of the network.

- ***Neighbor Information Table (NIT)***:This table is used to keep information about neighbors, where one entry per neighbor is maintained. The fields contained in an entry are - *Node_ID*:

neighbor's address, *Delay*: to store average delay to the neighbor, *B_Consumed*: Bandwidth consumed by active flows at neighbor, *SS*: Signal strength value of last packet received from the neighbor, and *Timestamp*: for checking the validity of signal strength value.

- ***Flow Table (FT)***: For reserving the required bandwidth for each of the active flow passing through the node for reservation period.
- ***Route Request Forwarded Table (RFT)***: This table is used to store information about QRREQ packets received and forwarded by an intermediate node during route discovery to assist selection of highest stable route during QRREP processing.
- ***Routing Table (RT)***: To setup and remember forward path and reserve path entry after processing QRREP and QRREQ packets respectively as used with AODV.

4.5.5 Available Bandwidth Estimation

The difficulty of calculating bandwidth of a wireless channel arises from the fact that it is shared by multiple nodes unlike in the wired scenario. A simple method of estimating the total available bandwidth of the wireless channel of a node is by calculating the total bandwidth consumed by all the nodes in the interference region and deducting it from the raw bandwidth of the node. This follows from the fact that a node can not use the channel if other interfering nodes are transmitting or receiving at the same time. The bandwidth estimation technique proposed in [115] is used for this purpose and it is briefly explained here for clarity.

The following symbols are used in estimating the bandwidth of a wireless channel around a node:

- B: Raw data rate of a node (same for all nodes)
- $N(I)$: Set of neighbor of node I
- B_{avail}: Bandwidth available at the node
- B_{int}: Total interfering traffic at the node
- B_{resv}: Bandwidth reserved at the node by different flows
- B_{cons}: Bandwidth consumed by the requesting flow

To estimate the available bandwidth at any node, the total traffic in its interference region needs to be calculated. It can be observed that there are three kinds of traffic that contribute to total interfering traffic, B_{int}. These are -

- Self Traffic $B_{self}(I)$, bandwidth consumed by the traffic transmitted or received by node I.
- Neighborhood traffic $B_{neigh}(I)$, total traffic between I's neighbors, $N(I)$.
- Boundary traffic $B_{boundary}(I)$, total traffic between I's neighbors and nodes that are outside I's range, whose connection crosses the boundary of node I's accessible range.

Therefore, the total interfering traffic at node I can be calculated as:

$$B_{int}(I) = B_{self}(I) + B_{neigh}(I) + B_{boundary}(I) \tag{4.5}$$

This can be better understood from the Figure 4.2, where node I has four neighbors A, B, C and D and node E is outside of I's range. There is self-traffic between node I and node A that equals T_{IA}, neighborhood traffic between nodes B and C which equals T_{BC} and the boundary traffic between nodes D and E which is T_{DE}. Therefore, the total interfering traffic, B_{int}, is the sum of all three kinds of traffic. However, in some cases neighborhood traffic and boundary traffic may not interfere with each other and hence can be transmitted simultaneously. In this sense, the above estimation of $B_{int}(I)$ is a conservative estimate. It has been shown in [115] that, $B_{int}(I)$ can be estimated as follows.

$$B_{int}(I) = \sum_{J \in N(I)} B_{self}(J) \tag{4.6}$$

The available bandwidth of the node I can be estimated as

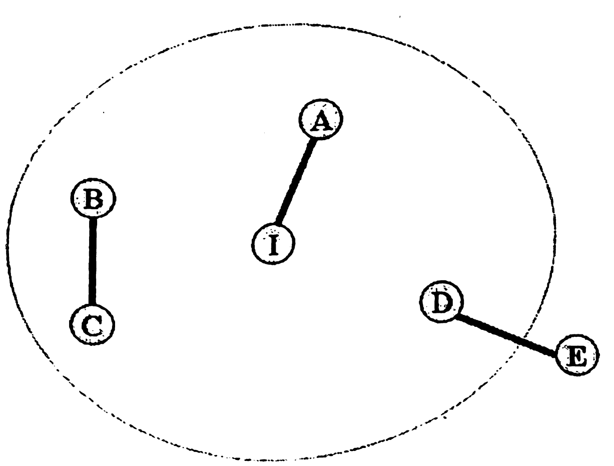


Figure 4.2: Interfering Traffic at node I

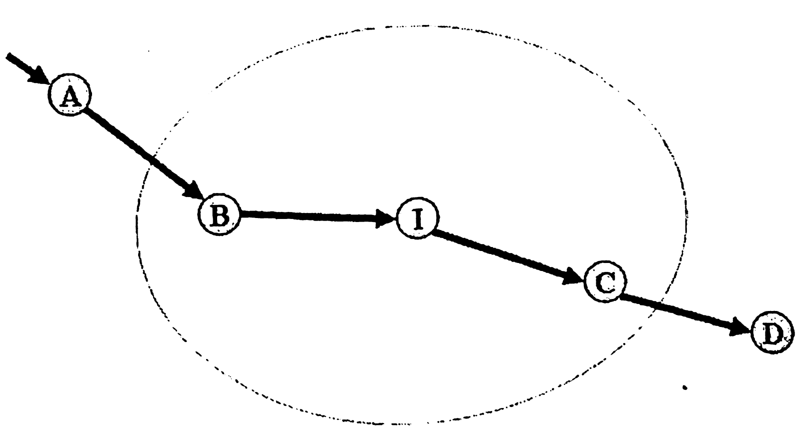


Figure 4.3: Additional Traffic by Requesting Flow

$$B_{avail}(I) = B - B_{int}(I) \tag{4.7}$$

The bandwidth consumed at a node is different from the requested minimum bandwidth, B_{min}, of a flow. This is because to satisfy B_{min} for a flow going through node I, node I needs to reserve a bandwidth considering the interference that will be generated in the neighborhood of node I due to transmission of packets belonging to the flow. To understand the situation, see the Figure 4.3. Let $B_I(j)$ be the bandwidth to be reserved at node I for a flow j with requested minimum bandwidth B_{min}. Then, $B_I(j)$ is defined as follows.

$$B_I(j) = \begin{cases} B_{min} & \text{if A or D} \\ 2 \times B_{min} & else \end{cases} \tag{4.8}$$

Since, the intermediate nodes need to receive and forward the flow j, the reservation required at the node is double the requested bandwidth. However, $B_I(j)$ doesn't include the additional boundary traffic, brought by the flow, to node I ; e.g., T_{AB} and T_{CD} as shown in the Figure 4.3. Due to traffic aggregation, both the new self-traffic and new boundary traffic introduced by the requesing flow is included in computing the total bandwidth consumed by flow j at node I, $B_{cons}(I, j)$, as follows.

$$\begin{aligned} B_{cons}(I,j) &= T_{AB} + T_{BI} + T_{IC} + T_{CD} \\ &= B_B(j) + B_C(j) \end{aligned} \tag{4.9}$$

With respect to node I, we call B and C as uplink and downlink neighbors respectively. Therefore, $B_{cons}(I, j)$ can be expressed as

$$B_{cons}(I,j) = B_{uplink(I)}(j) + B_{downlink(I)}(j)$$

Now, the total bandwidth reserved at node I for all flows can be calculated as -

$$B_{cons} = \sum_{\forall j} B_{cons}(I,j) \tag{4.10}$$

4.5.6 Admission Control

In RSQR, admission control decision at each node should ensure that the requested minimum throughput (B_{min}) and maximum end-to-end delay (D_{max}) constraints are satisfied in each hop of the path. Each node estimates the available bandwidth at the node by subtracting the consumed aggregated bandwidth in the interference range of the node from the raw bandwidth (as discussed in section 4.5.5). To enable this estimation of available bandwidth, each node maintains the information about bandwidth reservation at the node and periodically exchanges this information with neighbors through *Hello* message. Therefore, if a node receives a QRREQ with certain throughput requirement, it can decide whether the requested throughput can be supported by the node or not. Besides the available bandwidth, a node has to consider the intra-flow contention due to the requested flow to perform throughput admission control. *Algorithm* 4.1 shows the details to estimate the available residual bandwidth ($Bavail$) and the required bandwidth ($Breq$) by a route request at an intermediate node. A mechanism to calculate forwarding delay at each node is required to perform end-to-end delay admission control at each hop. Per hop delay is calculated by exchanging a pair of time stamped *Hello* packets between the end points of a link and taking weighted average of this value. The calculated delays are maintained by the nodes in *Neighbor Information Table* (NIT) and are periodically refreshed. To perform end-to-end delay admission control, a node will compare the combined value of accumulated delay ($ADELAY$) and per hop delay for the previous hop with D_{max}. If the combined value is less than or equal to D_{max}, the route request passes the delay admission control. The route request packets for which either end-to-end delay or throughput can not be satisfied are dropped and others are allowed to proceed further. Further, to avoid the possibility of selecting unstable paths, route request packets with signal strength $< Thr_2$ are dropped without performing any processing.

Algorithm 4.1 Algorithm to check bandwidth availability at node Q

Input: S, D, Previous Hop ID P, $Bmin$
Input: NIT of node Q, set of Neighbor of Q, NQ
Output: $Bavail$ (Residual Bandwidth at Q), $Breq$ (Bandwidth required at Q to satisfy $Bmin$)

```
Breq := Bmin
Bint := 0 {* Bint is the total interfering traffic at node Q *}
for all ( i in NQ) do
    Bint := Bint + NIT[i].Bcons {* Bcons is the total bandwidth consumed at a neighbor i *}
end for
Bavail := B - Bint {* B is Raw bandwidth, which is assumed to be constant *}
if (Q ≠ S) AND (Q ≠ D) then
    if (P ≠ S) AND (D not in NQ) then
        Breq := 4 * Bmin {*When neither its uplink neighbor is S nor its downlink neighbor is
        D*}
    else if (P ≠ S) OR (D not in NQ) then
        Breq := 3 * Bmin {*When either its uplink neighbor is S or its downlink neighbor is D *}
    else {When its uplink neighbor is S and downlink neighbor is D}
        Breq := 2 * Bmin
    end if
else {when the node is either source or destination}
    Breq := 2 * Bmin
end if
if (Breq > Bavail) then
    print "Throughput Admission Fails"
end if
```

4.5.7 Route Discovery

In this section, we outline the algorithms that need to be executed at each node to compute a Route Stability based QoS route between a given pair of source and destination. RSQR's route discovery is done on-demand using selective flooding. To compute a QoS route to a destination D , the source S generates a QoS Route Request (QRREQ) packet with values for B_{min} and D_{max} from the application's requirements. The other fields are specified as: 0 for $ADELAY$, 1 for APS, 0 for APU, and *seq_no*+1 for *seq_no*. Rest of the fields are used as in AODV[85]. After processing and sending a QRREQ packet, source node starts a timer of duration $3 \times D_{max}$ before waiting for a QRREP packet from the destination. The timer is set as $3 \times D_{max}$, because it represents the maximum time that will be required to receive a QRREP from the destination through a route whose maximum one way end-to-end delay will be $\leq D_{max}$. In our routing protocol, destination is given an upper bound of time period which is RRL (where RRL is always $< D_{max}$) to process the QRREQ packets to compute a QoS route with higher stability. If source node does not receive any QRREP within the timeout period, it assumes failure of route discovery and the source node will initiate rediscovery after a backoff period of $3 \times D_{max}$. Every successive route discovery failure will increase the backoff period exponentially till some maximum retry limit after which the user will be notified of the route discovery failure. The operations performed by different nodes for the processing of QRREQ and QRREP are explained in the following subsections.

Algorithm 4.2 Algorithm to process and update QRREQ packets at intermediate node Q in RSQR

Input: A QRREQ packet from node P with S, D, B_{min}, D_{max}, $ADELAY$, APS and APU
Input: Signal strength values SS^1 (current) and SS^2 (recent), list of neighbors of Q, N_Q

```
if (SS^1 < Thr_2) then
    Drop QRREQ packet {*Signal Strength is very poor*}
    return
end if
if (delay or throughput admission control fails) then
    Drop QRREQ packet
    return
else
    ADELAY := ADELAY + NIT[P].Delay
end if
Estimate LS and LU using Algorithm 4.3
if (LU ≠ 1) then
    APS := APS * LS
else
    APU := APU + LU
end if
Update QRREQ with ADELAY, APS and APU
if (QRREQ is not already forwarded) or (QRREQ has better value for APU or same value of
    APU but better value for APS) then
    Make an RFT entry with (S,D,seq_no,ADELAY,APU,APS,P)
end if
if (QRREQ is not already forwarded) then
    Make a temporary reverse RT entry, a FT entry with timeout = 3 * D_max
    Broadcast the QRREQ packet to next hop
end if
```

Algorithm 4.3 Algorithm to compute Link Stability metric

Input: SS^2, SS^1 : recent and current signal strengths values
Output: LS (Link Stability), LU (Link Uncertainty)

LS := - 1.0
if ($SS^1 \geq Thr_1$) **then**
    LS :=1.0 ; LU := 0
    return
end if
if (($SS^1 \geq Thr_2$) AND ((SS^2 not defined) or ($SS^2 < RxTh$)) **then**
    LU := 1 ; return
end if
if ($SS^1 \geq Thr_2$) AND ($SS^2 \geq RxTh$) **then**
    $DSS := SS^2 - SS^1$
    $LU := 0$
    if ($DSS < u_1$) **then**
        $LS := 1.0$
    else if ($DSS < u_2$) **then**
        $LS := \frac{(u_2 - DSS)}{(u_2 - u_1)}$
    else
        $LS := 0.0$
    end if
else
    $LS := 0.0$
end if

Operations performed by Intermediate node

An intermediate node i, after receiving a QRREQ packet, checks the signal strength of QRREQ and simply drops the packet if its strength is very poor (less than a threshold Thr_2). Otherwise, node i performs the delay and throughput admission control. If the QRREQ passes both delay and throughput admission control, node i makes a temporary reverse route entry in *RT* (*Routing Table*) with $timeout = 3 \times D_{max}$ and also remembers the *seq_no* of the QRREQ packet. The reason for setting $timeout = 3 \times D_{max}$ is same as explained in section 4.5.7. Node i computes the *Link Stability* (*LS*) and/or *Link Uncertainty* (*LU*) (using *Algorithm* 4.3) of the link through which it has received the QRREQ. After these processing, some fields in QRREQ are modified such that the modified values contain the route stability and end-to-end delay of the explored route up to the current node. Before rebroadcasting of QRREQ, node i stores relevant information from QRREQ packet in a table called *Route Request Forwarded Table*(*RFT*). *seq_no* field in QRREQ packet is used to detect the duplicate QRREQ packet. Before discarding duplicate QRREQ packets, an intermediate node performs similar processing as is done for a new QRREQ packet. For a duplicate QRREQ packet, which passes all admission control criteria and contains better values for route stability or delay or both, an entry in RFT is made before discarding the packet. If a node receives no QRREP before the expiry of a reverse route entry, that reverse route entry in *RT* is purged and any late QRREP packet for the entry is simply dropped. *Algorithm* 4.2 gives the detail processing of an QRREQ packet by an intermediate node.

On Receiving a QRREP packet, node i performs similar processing as done when receiving a QRREQ packet as shown in *Algorithm* 4.4. If the received QRREP packet passes admission control, node i looks up *RFT* for the corresponding QRREQ to select the entry with highest stability which satisfies both delay and throughput constraint. Node i updates *APS* and *APU* fields in QRREP so that they will represent stability of the route to be followed by the QRREP packet. Before forwarding the QRREP packet toward the source, node i makes an *RT* entry (with $timeout = 2 \times D_{max}$) and reset the timeout of the *Flow Table* (*FT*) entry for the flow to $2 \times D_{max}$. The reason for setting a timeout period for *RT* and *FT* entries at the time of forwarding a QRREP packet is to enable removal of unused entries automatically from the corresponding tables. Here, we have assumed that source starts using a route without delay once it receives the QRREP packet for the route. In this case, after forwarding the QRREP packet for a selected route, an intermediate node should not take more than $2 \times D_{max}$ to receive the first packet using the given route. This is why the timeout is set as $2 \times D_{max}$. In the case, none of the entries in *RFT* satisfies both delay and throughput constraints, the QRREP packet is discarded and related table entries for the route request (created during the forwarding of corresponding QRREQ) are removed. If no data packet for a selected route is received within a time period of $2 \times D_{max}$ after forwarding of a QRREP packet, the corresponding entries in *RT* and *FT* are purged automatically. Only after receiving the first data packet, before timeout of an *RT* entry, the throughput reservation at the node is activated in *FT*. *Algorithm* 4.4 shows the processing of QRREP packets by an intermediate node.

Operations performed by Destination node

Destination node may receive multiple QRREQ packets that reach the destination by following different candidate routes. Destination node, after receiving a QRREQ, performs admission control

Algorithm 4.4 Algorithm to process and update QRREP packets at intermediate node Q in RSQR

Input: A QRREP packet from node P with S, D, B_{min}, D_{max}, $ADELAY$, APS and APU
Input: Signal strength values SS^1 (current) and SS^2 (recent), list of neighbors of Q, N_Q

if ($SS^1 < Thr_2$) **then**
  Drop QRREP packet {Signal Strength very poor}
  return
end if
if (delay or throughput admission control fails) **then**
  Drop QRREP packet
  return
else
  $ADELAY := ADELAY + NIT[P].Delay$
end if
K:= no. of entry in RFT for corresponding QRREQ
L:= index of the first entry in RFT for corresponding QRREQ
$R := Previous\_Hop\_ID$ in reverse RT entry for QRREQ
$APU^* := APU$ value in reverse RT entry for QRREQ
$APS^* := APS$ value in reverse RT entry for QRREQ
$i := L$
$Count := K$
while ($Count > 1$) **do**
  j:= Next entry for QRREQ in RFT
  if ($ADELAY + RFT[i].ADELAY \leq D_{max}$) **then**
    if ($RFT[j].APU < RFT[i].APU$) or (($RFT[i].APU = RFT[j].APU$) and ($RFT[i].APS < RFT[j].APS$)) **then**
      $i := j$
    end if
  end if
  $Count := Count - 1$
end while
if ($i \neq L$) **then**
  $APU := APU - (APU^* - RFT[i].APU)$
  $APS := APS * RFT[i].APS/APS^*$
  $R := RFT[i].Prev\_Hop\_ID$
end if
Construct a QRREP with $ADELAY$, APS and APU
Make a RT entry with $timeout = 2 * D_{max}$
Reset FT entry with $timeout = 2 * D_{max}$
Remove RFT entries for QRREQ
Forward QRREP to R

and computes route stability as performed by an intermediate node. After receiving the first QRREQ packet which satisfies the QoS constraints, it waits for a small period of time called *Route Reply Latency (RRL)* to receive more QRREQ packets belonging to the same route request. We have set RRL in such a way that it is always $< D_{max}$, but larger enough to collect sufficient QRREQ packets for computing highest stable QoS routes. After the expiry of the waiting time, destination generates a QRREP packet for the QRREQ packet which corresponds to QoS route with highest route stability value. It is worth pointing out here that in DSR and AODV, route selection algorithm prefers the route corresponding to the route request that reaches the destination node first. Similarly, in AQOR, route reply that reaches the source first is selected. The basic reason for selecting the first route request/reply is that, this route is less congested than the others. Of course, this route may contain highly unstable links, which makes route failure more likely during data communication. Therefore, it is more reasonable to look at the route stability value of the candidate routes to make final selection of the route. Further, as compared to AQOR where multiple route reply packets are sent from destination to source, in our proposal, destination sends only one route reply which corresponds to the highest stable QoS route from source to destination. This reduces considerable amount of control overhead in our proposed protocol. In RSQR, destination does the following before generating a route reply for the selected route. It copies QoS and route stability related fields of QRREQ to QRREP and initializes $ADELAY$ with zero. Destination creates both an RT entry and a FT entry for the requested flow with ($timeout = 2 \times D_{max}$). Only after receiving the first data packet belonging to a flow on or before this time period, the throughput reservation for the flow is made active for a reservation period, T, in FT. T is determined from the application's throughput requirement and is defined as $(k * N * 8/B_{min})$ [115], where k is the allowed packet loss and N is the maximum packet length (in bytes) used at PHY level. The reservation of throughput in FT is required to guarantee the availability of the requested throughput during the communication session of the flow. However, absence of data packet within a reservation period will lead to the deletion of the entry from FT, which leads to QoS violation and subsequent route recovery process in the network. Similar throughput reservation is done at every node in the route from source to destination. *Algorithm* 4.5 explains the operations by the destination.

4.5.8 An example showing Route Discovery using RSQR

The route discovery process of RSQR is shown in Figure 4.4 - Figure 4.8 for an arbitrary network of 10 nodes. Here, for the sake of simplicity, we have not shown processing related to admission control. Further, we have assumed that all QRREQ/QRREP packets pass admission control due to bandwidth and delay. In the figures, the label shown below a link represents the current link stability value of that link. As shown in Figure 4.4, node S initiates a route discovery to find a QoS route to node D by broadcasting a QRREQ packet. Node S initializes the fields in QRREQ packet in accordance with QoS requirement of the flow as discussed in Sec. 4.5.7. Every intermediate node, after receiving a QRREQ packet, processes the QRREQ packet as explained in *Algorithm* 4.2. Figure 4.5 shows the processing of the QRREQ packet by the intermediate node H after it receives the first copy of QRREQ from E. Along with creating a reverse route entry in RT (which is not shown in the figure), node H creates an RFT entry to remember the stability (APS=0.64, APU=0) and accumulated delay (ADELAY=0.024) values for the explored path from S to H. It can happen that the first copy of QRREQ packet received at a node may not correspond to the best possible and computes route stability as performed by an intermediate node. After receiving the first QRREQ packet which satisfies the QoS constraints, it waits for a small period of time called *Route Reply Latency (RRL)* to receive more QRREQ packets belonging to the same route request. We have set RRL in such a way that it is always $< D_{max}$, but larger enough to collect sufficient QRREQ packets for computing highest stable QoS routes. After the expiry of the waiting time, destination generates a QRREP packet for the QRREQ packet which corresponds to QoS route with highest route stability value. It is worth pointing out here that in DSR and AODV, route selection algorithm prefers the route corresponding to the route request that reaches the destination node first. Similarly, in AQOR, route reply that reaches the source first is selected. The basic reason for selecting the first route request/reply is that, this route is less congested than the others. Of course, this route may contain highly unstable links, which makes route failure more likely during data communication. Therefore, it is more reasonable to look at the route stability value of the candidate routes to make final selection of the route. Further, as compared to AQOR where multiple route reply packets are sent from destination to source, in our proposal, destination sends only one route reply which corresponds to the highest stable QoS route from source to destination. This reduces considerable amount of control overhead in our proposed protocol. In RSQR, destination does the following before generating a route reply for the selected route. It copies QoS and route stability related fields of QRREQ to QRREP and initializes $ADELAY$ with zero. Destination creates both an RT entry and a FT entry for the requested flow with ($timeout = 2 \times D_{max}$). Only after receiving the first data packet belonging to a flow on or before this time period, the throughput reservation for the flow is made active for a reservation period, T, in FT. T is determined from the application's throughput requirement and is defined as $(k * N * 8/B_{min})$ [115], where k is the allowed packet loss and N is the maximum packet length (in bytes) used at PHY level. The reservation of throughput in FT is required to guarantee the availability of the requested throughput during the communication session of the flow. However, absence of data packet within a reservation period will lead to the deletion of the entry from FT, which leads to QoS violation and subsequent route recovery process in the network. Similar throughput reservation is done at every node in the route from source to destination. *Algorithm* 4.5 explains the operations by the destination.

4.5.8 An example showing Route Discovery using RSQR

The route discovery process of RSQR is shown in Figure 4.4 - Figure 4.8 for an arbitrary network of 10 nodes. Here, for the sake of simplicity, we have not shown processing related to admission control. Further, we have assumed that all QRREQ/QRREP packets pass admission control due to bandwidth and delay. In the figures, the label shown below a link represents the current link stability value of that link. As shown in Figure 4.4, node S initiates a route discovery to find a QoS route to node D by broadcasting a QRREQ packet. Node S initializes the fields in QRREQ packet in accordance with QoS requirement of the flow as discussed in Sec. 4.5.7. Every intermediate node, after receiving a QRREQ packet, processes the QRREQ packet as explained in *Algorithm* 4.2. Figure 4.5 shows the processing of the QRREQ packet by the intermediate node H after it receives the first copy of QRREQ from E. Along with creating a reverse route entry in RT (which is not shown in the figure), node H creates an RFT entry to remember the stability (APS=0.64, APU=0) and accumulated delay (ADELAY=0.024) values for the explored path from S to H. It can happen that the first copy of QRREQ packet received at a node may not correspond to the best possible

(APS=0.80, APU=0) and modify the QRREP packet accordingly (as explained in *Algorithm* 4.4) so that the stability value contained in the final QRREP packet will correspond to the stability value of the route to be followed by QRREP. Node H forwards the QRREP packet to node F(obtained from *Prev Hop* value in the selected RFT entry). All the intermediate nodes B, F and H set up their forward RT entry during the time of forwarding QRREP packet toward the source. Finally, when the source S receives the QRREP packet and a QoS route from S to D is established (as shown in Figure 4.8).

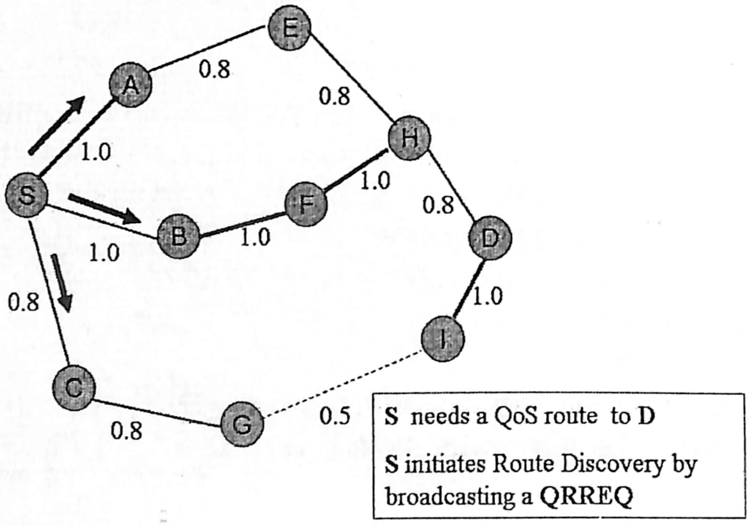


Figure 4.4: An example showing route discovery (S initiates route discovery)

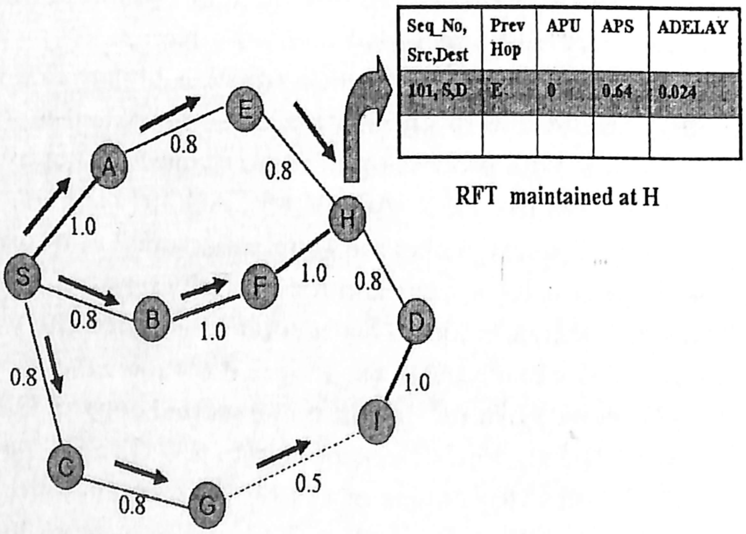


Figure 4.5: An example showing route discovery (intermediate nodes are processing and forwarding QRREQ)

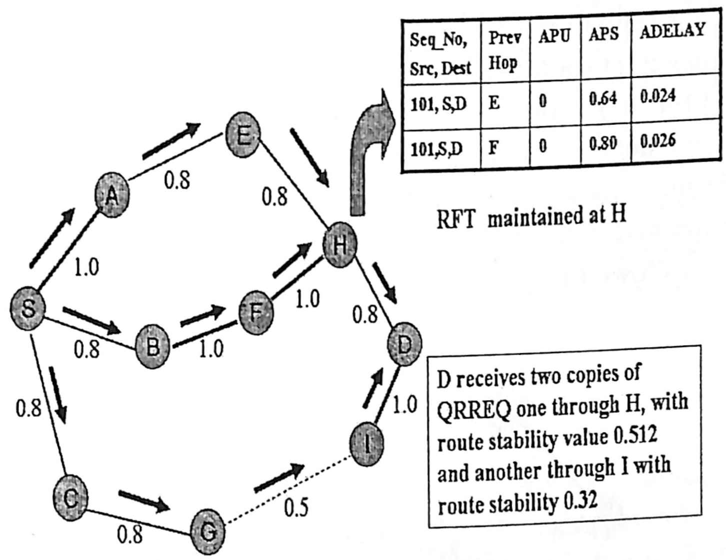


Figure 4.6: An example showing route discovery (Destination D is receiving two copies of QRREQ)

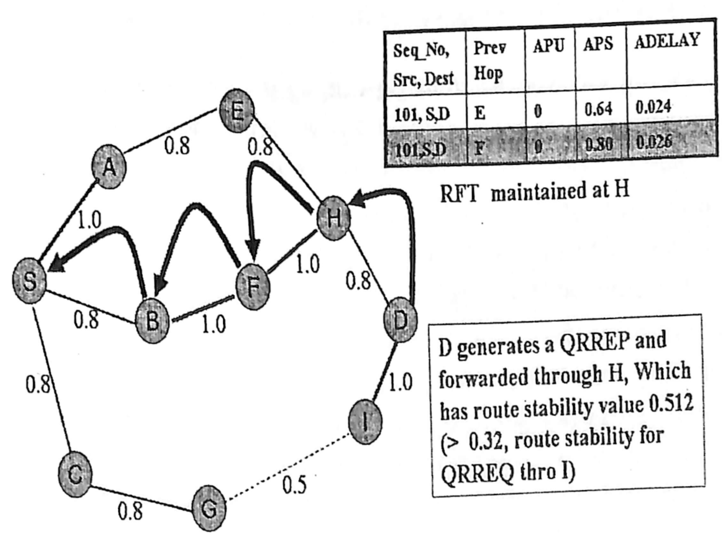


Figure 4.7: An example showing route discovery (Destination computes the highest stable QoS route and generates QRREP)

4.5.9 Adaptive Route Recovery

The QoS-aware routes in MANETs experience frequent route failure and/or QoS violation mainly due to node mobility and unreliable shared wireless media. Incorporating route stability into routing reduces the frequency of route failures by favoring highly stable route and thereby making resource reservation valid for longer period of time. QoS violation is encountered due to prevailing network conditions like channel degradation, network congestion etc. Similarly, route failures take place as a result of node mobility and node failures. Therefore, to reduce QoS disruption an instant QoS violation detection and recovery mechanism is a must. Below we will discuss the adaptive route recovery process followed in RSQR.

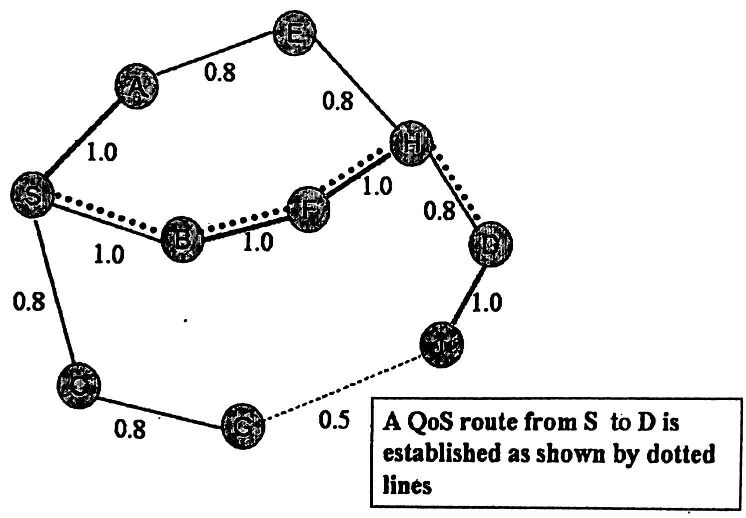


Figure 4.8: An example showing route discovery (A QoS route between S and D is established)

End-to-end Delay violation detection during data transmission session is done at the destination through computation of one way delay (downstream delay) experienced by the data packets. The clock offset between source and destination is computed by the source from the round trip delay between source and destination and the destination's time stamp on QRREP packet. Source sends this clock offset value to the destination at the beginning of data flow. From the clock offset, and the source's time-stamp on data packet, destination can compute one way delay, T_d of a data packet. Therefore, if destination finds $T_d > D_{max}$ for a received data packet, it detects delay violation for the data packet. This violation may be temporary, so destination observes for n (a constant, to be determined for the network) number of continuous delay violations to trigger a delay violation alarm. There is an indirect way to detect possible throughput violation by detecting absence of data packets of a flow in the destination for a particular time interval called throughput reservation timer (T) (as defined in section 4.5.7 under *Operation performed by Destination node*). We have maintained a throughput reservation timer T at destination and QoS violation is indicated if no data packet for a flow is received within this interval. Bandwidth reservation timer is also maintained in all intermediate nodes of a QoS route for canceling out (tear down) bandwidth reservation in broken or obsolete route. Recovery from QoS violation is done by sending a Route Update, RUPD (unsolicited QRREP) from destination to the source for discovering a new QoS route to the source. Here, we have assumed that the route from the destination to the source is similar to the route from the source to the destination due to bidirectional wireless link assumption. The RUPD is sent with limited flooding by using TTL equal to hop count of the already broken route plus a small constant. The RUPD packets are treated similar to QRREP packets by the source and QRREQ packets by the intermediate nodes. If at least one RUPD packet arrives at the source before bandwidth reservation timer expires, the QoS route recovery is successful and the source switches its flow from old route to new route. In the case, recovery by destination fails, the route recovery by source is initiated as in [85] with the help of RERR packet generated by the upstream node of a broken link. Any intermediate node which receives a RERR packet marks the path using the link as invalid and forwards the RERR packet toward the source. Finally, after receiving the RERR packet source initiates a new route discovery for the same destination if it requires to perform further communications with the destination.

4.6 Simulations and Performance Evaluation

In this section, we have investigated the performance of our proposed protocol and compare it with AQOR[115] through an extensive set of simulations. We have taken AQOR [115] for performance comparison with RSQR, because AQOR [115] is the closest protocol to RSQR as compared with other QoS-aware protocols found in [16, 17, 86, 123]. Both RSQR and AQOR take delay and throughput as the QoS parameters. The protocol proposed in [19] is almost similar to [115] without performance improvement over AQOR. A performance improvement over AQOR [115] was obtained in [47] by defining another QoS metric called path cost. Because, our protocol does not consider path cost, so we have not considered [47] for performance comparison. Rest of the routing protocols in the reference are either purely best-effort based with stability metric or they consider route length as a QoS parameter. Therefore, we found AQOR [115] as the best available QoS routing protocol to compare its performance with RSQR.

4.6.1 Performance Metrics and Simulation parameters

Following metrics have been used for performance analysis.

- *Traffic Admission Ratio (TAR):* This is the ratio of number of data packets actually sent by the sources and the number of data packets generated at the sources. This measures the effectiveness of admission control policy.

- *Packet Delivery Ratio (PDR):* This is the ratio of number of data packets received at the destinations and the number of data packets actually sent to the network. This measures the quality of the discovered path.

- *Normalized Control Overhead (NCO):* This is the ratio of number of control packets sent to the network hop-wise and the number of data packets delivered at the destinations. This measures the overhead induced by the protocol.

- *End-to-End Delay (E2ED):* This gives the average time delay that a data packet has encountered from the time it was sent by a source to the time it was delivered at the destination. This measures the delay performance of the QoS flows.

To evaluate the performance of RSQR protocol and perform the comparative study with AQOR [115] under different mobility and network load conditions, an extensive simulation is performed using NS-2[80]. For each type of scenario, we have run the simulation for 10 times (with random scenarios with different seeds) to take average values in the measured performance metrics. We have used two types of network loads to represent moderate load and high load situations. The connection pattern consists of 10 flows (for high load 15 flows) between randomly chosen source-destination pairs among the 50 nodes. The QoS flow taken is CBR with packet size 512 bytes at rate of 10 packets per second. All the generated flows have same QoS requirements. The detailed simulation parameters are summarized in Table 4.1. The parameters for stability model are heuristically selected through a set of simulation runs.

Table 4.1: Simulation Parameters

Parameter Name	Value
Simulation area	$1000m \times 500m$ flat-grid area
Number of nodes	50
Mobility Model	Random Waypoint
MAC layer	IEEE 802.11 DCF
Channel Capacity (Mb/s)	2
Radio Propagation Model	Two Ray Rayleigh Fading
Antenna Type	Omni directional
Transmission Range (m)	200
Carrier Sensing Range (m)	200
Maximum Node Speed (m/s)	0, 2, 5, 10, 15, 20
Pause time (s)	0
Simulation Time (s)	500
Number of flows	10, 15
Traffic Type	CBR
Packet Size (B)	512
Traffic Rate (packets/s)	10
Minimum Throughput (kbps)	40
Maximum Delay (s)	0.1
Thr_1, Thr_2	$1.5 \times RxThr, 1.2 \times RxThr$
u_1, u_2	$0.05 \times RxThr, 0.3 \times RxThr$
T (s), RRL (msec), n	1.0, 75, 3

4.6.2 Results Analysis

From Figure 4.9 and Figure 4.10, we can observe that the Traffic Admission Ratio (TAR) of both AQOR and RSQR decreases with increased mobility due to admission control policy. TAR is significantly reduced in high load condition as shown in Figure 4.10. In RSQR, TAR is further reduced as compared to AQOR due to the RSQR's restrictions in using weak or transient links for route selections in high mobility scenarios. For moderate load condition, though at stationary scenario, TARs of AQOR and RSQR are 99.3% and 98.2% respectively, at higher mobility (i.e. at 20 m/s) TAR reduces to 79.7% and 74.3% respectively. We found reduced TAR for RSQR, but better PDR (97.2% for RSQR vs 90% for AQOR in 20 m/s) as seen from Figure 4.9 and Figure 4.11. Similarly, for high load condition, TAR of AQOR and RSQR are 78% and 75% respectively at stationary scenario, where as it reduces to 57.2% and 52.3% at at higher mobility of 20 m/s. The PDRs of AQOR and RSQR at high load are 85% and 95.3% respectively as shown in Figure 4.12. It can be seen that irrespective of mobility and load conditions RSQR maintains a very high degree of PDR, which is essential for improving the level of QoS assurance. The reasons for the observed higher PDR in RSQR are *(i)*reduction in TAR for RSQR at higher mobility reduces packet collisions as compared to AQOR, *(ii)* use of highly stable routes which reduces the possibility of route breaks as compared to AQOR and *(iii)* reduction of waiting time in queue and subsequent reduction of packet lost due to long waiting time in the queue. Therefore, RSQR can easily provide higher assurance to reliable delivery as compared to AQOR. Figure 4.13 and Figure 4.14 depict the NCO of both the protocols, which increases (for both AQOR and RSQR) with increase in mobility. It is due to frequent route failures and QoS violations at higher mobility. Further, increase in load also increases the amount of control traffic in the network. Due to selection of highly stable routes, number of route recoveries due to route failure in RSQR will be reasonably less as compared to AQOR. In RSQR, destination sends back only one route reply that corresponds to the best stable QoS aware route computed from source to destination. Whereas AQOR sends back as many route replies as the number of route request packets received by the destination. Therefore, control overhead in AQOR increases due to increase in number of route reply packets. Further, NCO is determined by the number of admitted flows, which is lower in RSQR. The trends in increase of NCO with mobility is very marginal in RSQR compared to AQOR. In moderate load, AQOR has 40.3% NCO at the mobility of 20 m/s, whereas it is only 28.6% in case of RSQR. This difference in NCO is more in high load condition, where AQOR has NCO of almost 50% and RSQR has 36% (as can be seen Figure 4.14). From Figure 4.15 and Figure 4.16, it can be seen that average delay is below 0.04s for both the protocols in all scenarios, which is much lower than the requested maximum delay of 0.1 s. But RSQR has lower delay compared to AQOR, because of lower TAR and less number of route recoveries results in reduction in waiting time in queues and hence reduction in E2ED.

From the above analysis, we can conclude that RSQR shows significant performance improvement in dynamic network environments and in moderate to high network load conditions and hence is a promising solution for QoS routing in MANETs. Although AQOR and RSQR protocols are similar in the sense both of them are resource reservation based on-demand routing protocols. But in AQOR, the interruptions in real-time flows due to node mobility or channel variations is more frequent and consequently the service quality provided is poor as compared to RSQR. RSQR avoids frequent route failure and hence improve the service quality by finding durable QoS routes without extra overhead for measuring stability of the computed routes.

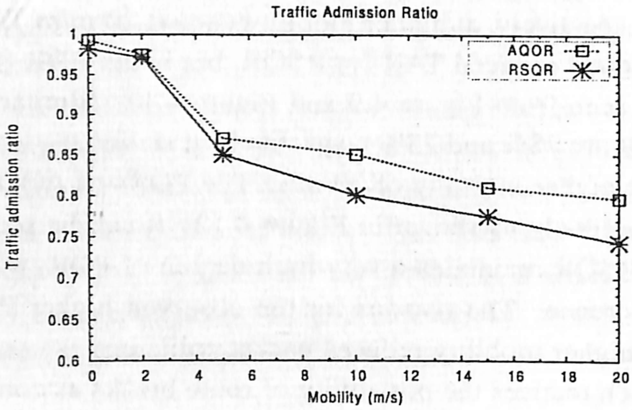


Figure 4.9: Traffic admission ratio vs node mobility

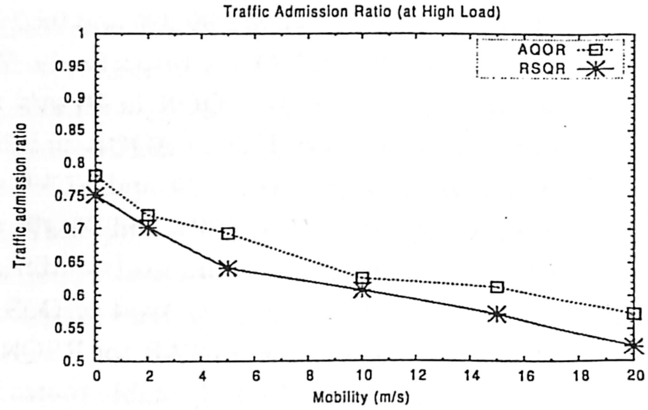


Figure 4.10: Traffic admission ratio vs node mobility (High Load)

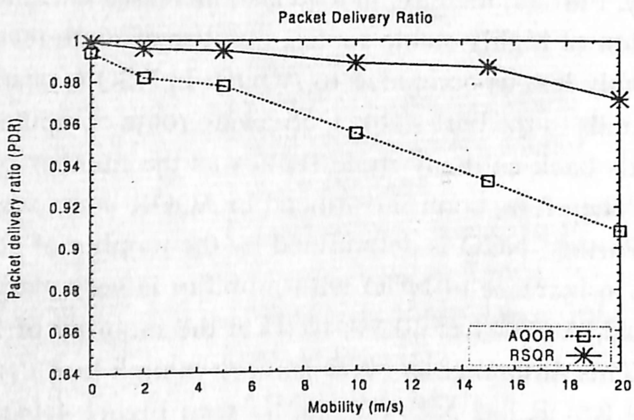


Figure 4.11: Packet delivery ratio vs node mobility

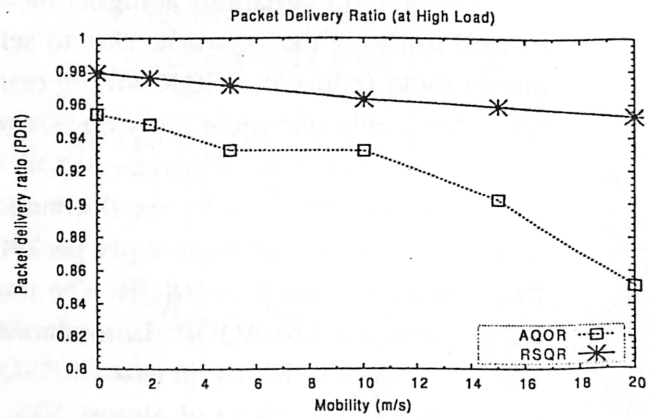


Figure 4.12: Packet delivery ratio vs node mobility (High Load)

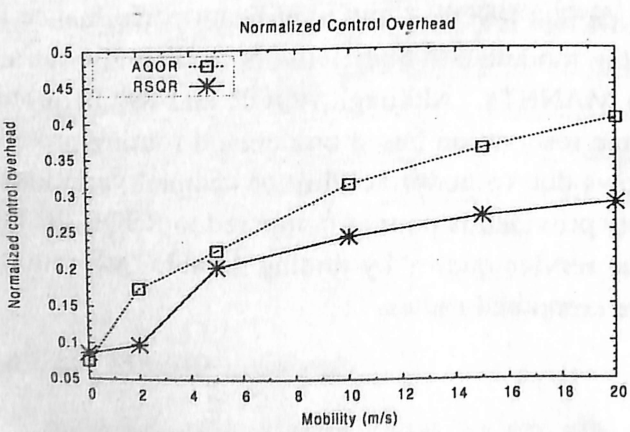


Figure 4.13: Normalized control overhead vs node mobility

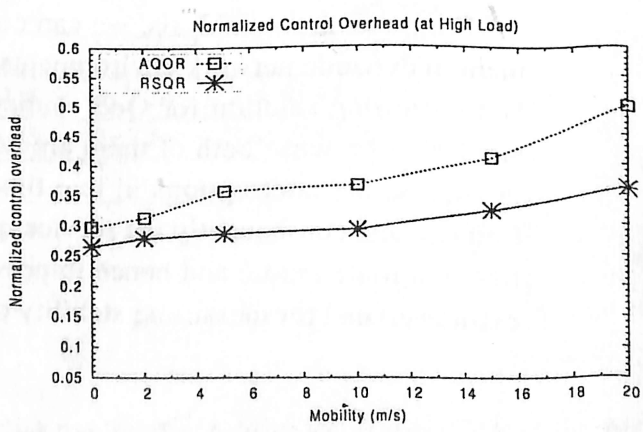


Figure 4.14: Normalized control overhead vs node mobility (High Load)

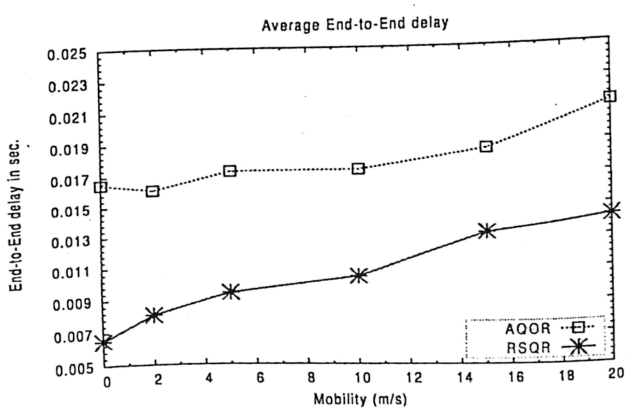


Figure 4.15: Average end-to-end delay vs node mobility

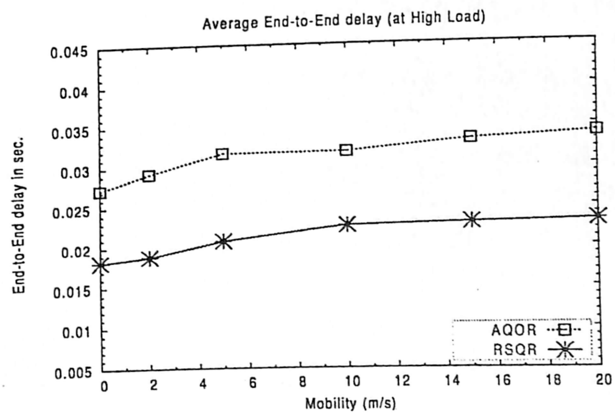


Figure 4.16: Average end-to-end delay vs node mobility (High Load)

4.7 Conclusion

In this chapter, we have proposed a route stability based QoS aware routing for Mobile Ad Hoc Networks to provide QoS assurance to applications in terms of maximum delay and minimum throughput. We found that similar QoS aware routing available for MANETs lacks inclusion of a proper stability model which is very important to ensure that discovered path remain valid for a longer period of time. We have shown that use of a simple route stability model in the proposed routing significantly reduces the number of route recoveries required during QoS data transmission. Simulation results show significant performance improvements in terms of control overhead, average end-to-end delay and packet delivery ratio especially in highly mobile scenarios with marginal decrease in traffic admission ratio in comparison with AQOR.

Detection of potential link breaks or QoS violations before actual link breaks or QoS violation takes place and use of proactive route maintenance will further save route recovery time and reduce possibility of packet loss during active communication. In this direction, we have developed a multipath version of RSQR, called route stability based multipath QoS routing or SMQR in short, which will be discussed in the next chapter.

Chapter 5

A Route Stability based Multipath QoS Routing (SMQR) for MANETs

5.1 Introduction

Routing protocols used for QoS support in mobile ad hoc networks are mostly based on On-demand unipath routing protocols due to their relative advantages as compared to the proactive routing protocols. But, there is a basic limitation of using on-demand unipath QoS routing protocols as follows. When an active route fails, due to either link failures or QoS violations, these on-demand protocols have to invoke a route discovery to recover from route failure. In dynamic environment, the frequent route failures cause the nodes on the failed routes to drop packets due to unavailability of alternate paths. Further, the route recoveries are very expensive in MANETs because it results in extra control overhead due to flooding of routing packets. This results in high end-to-end delay and lower throughput for the affected applications.

Using a QoS-aware multipath routing, a number of QoS-aware alternate paths can be made available at the source (based on some selection criteria) with a single route discovery. Then, out of the number of alternate paths discovered, one can be designated as the primary path and the others as secondary paths (based on some stability criteria on these discovered paths). In such multipath protocols, if a link failure or a QoS violation is detected on the primary path (through which actual data transmission is taking place), the source can switch to an alternate path instead of initiating a route discovery/recovery process. A new discovery takes place only when all pre-computed paths break. This results in reduction in end-to-end delay and packet loss for the affected flows, since packets do not need to be buffered for long time at the source when an alternate path is available. Further, due to reduction of control overheads, overall performance of the network will improve substantially.

In this chapter, we have proposed a Route Stability based Multipath Quality of Service Routing (SMQR) protocol for Mobile Ad Hoc Networks (MANETs) which is an extension of RSQR (already discussed in Chapter 4) to establish multiple node-disjoint QoS paths for a given source-destination pair. We have observed that, although in RSQR the source actually discovers multiple QoS routes during route discovery, the destination selects only the best QoS route based on route stability and discards the rest. In RSQR, to recover from QoS violation due to link failure, delay violation, or possible throughput violation, route recovery by destination has been used whenever possible. Otherwise, route recovery by source node is initiated. Route recovery by the destination or source results in considerable performance degradation due to extra delay and increased packet loss, though performance degradation is comparatively less in the case of route recovery by the destination. Therefore, using an alternate route to handle the potential route failures or QoS violation is one of the best options to reduce possible number of route recoveries during active communication in the network.

In the proposed multipath routing, reliability of the QoS-aware multipath is improved through the node-disjointness and stability properties of the discovered routes. Further, to increase the possibility of obtaining a valid alternate path after the failure of the primary path, a periodic validation of alternate paths is incorporated in SMQR. This is very much essential, because due to high mobility of nodes, alternate paths might become stale or invalid when the primary path fails. Using such stale paths may result in increasing number of dropped packets while trying each of the alternate paths one after another. Most of the existing multipath routing protocols do not perform any kind of maintenance of the alternate (secondary) paths. Further, due to RREQ packet forwarding policy combined with arbitrary arrival order of RREQ packets in intermediate nodes (also called *path diminution* problem), only a few disjoint multipath routes are possible in a network of moderate density. Though it is very difficult to remove the path diminution problem without excessive control and computation overheads, we have reduced the path diminution by adopting some special route request forwarding policies. Further, a proactive path switching by the source is used such that a source can always select the highest stable route among the multiple paths for transporting data to the destination. This is possible in SMQR through periodic maintenance of the multipath. Nasipuri et. al show in their paper [76] that performance gain in multipath routing is marginal beyond a few number of paths. Therefore, SMQR is designed to select a maximum of three QoS-aware disjoint paths.

The rest of the chapter is organized as follows. Section 5.2 gives an overview of the related works. A comparative study is included in section 5.3. The proposed Route stability based Multipath QoS Routing (SMQR) and its route maintenance are presented in sections 5.4 and 5.5 respectively. Performance of SMQR is evaluated and compared with AOMDV[71] in section 5.6. Finally, section 5.7 concludes the chapter.

5.2 Related Works

In standard on-demand routing protocols (like AODV and DSR), when an active route breaks, the nodes of the broken route simply drop the data packets because no alternate route to the destination is available. Some local route maintenance schemes were proposed in [13, 52, 111] to reduce the route recovery cost in on-demand protocols. AODV-BR [52] alleviates the temporary packet drop/delay problem while rediscovering a new routing path using intermediate backup nodes. The proposal in [111] proposed an optimization of the AODV-BR. In this approach [111], instead of using only the overheard RREP packets to learn backup paths, it uses both RREP and data packets to increase the possibility of learning more backup routes. Backup routes may solve the route failure problem temporarily and is not effective in QoS-aware data transmission. In [13], to recover from a broken path, the source uses a limited route request flooding within a few hops around the broken path. The route recovery using the localized flooding is not a promising solution specially in a highly mobile environment because applications require QoS guarantee on an end-to-end basis. A proposal by Goff et al [36] uses preemptive routing to predict the route failure and preemptively recover the route before the actual route fails. To be effective, their proposal requires an accurate route failure prediction model, which is difficult to design in MANETs.

Multiple routes provided by multipath routing can be used to compensate for the dynamic topology changes in MANETs. Earlier works on multipath routing [53, 59, 71, 92, 118] show that providing more than one path can better support high mobility network with low end-to-end delay, high packet delivery ratio and lower control overhead. All these protocols are designed to work for enhancing best-effort traffic. QoS-aware multipath routing protocols proposed in [62, 66] are based on a CDMA/TDMA based MAC layer. Due to difficulty in realizing such centralized MAC scheme in MANETs, we have considered only multipath routing based on a distributed MAC layer such as IEEE 802.11 DCF. In MP-DSR [56], a new QoS metric called end-to-end reliability is included with multipath discovery, but it considers traffic with neither throughput nor delay constraint. ADQR [40] finds multiple disjoint paths which can together satisfy user throughput requirements with longer-lived connections based on the available network bandwidth information obtained during route discovery. Distributing traffic among different paths has the inherent problem of packet reordering, which is not addressed in the protocol. The state information maintained by each node regarding the topology of the network, for processing route request and supporting fast rerouting, is very high. Further, proactive monitoring give rise to extra control overhead in the network. In QDMR [109], a QoS framework is proposed to support delay sensitive real-time applications in MANETs. It consists of two parts - i)a reliable and QoS-aware routing through disjoint paths discovery and maintenance and ii) providing differentiated service to real-time and best-effort traffic through DiffServ. Although it has included a mechanism to periodically maintain the alternate paths, it has the drawbacks that it does not use any estimation algorithm to compute available bandwidth in a node and uses hop count to represent path delay. IMRP [110] proposes an interfering-aware QoS multipath routing protocol for multimedia applications in ad hoc wireless network. Although it proposed a protocol for computing disjoint paths based on available bandwidth and the link stability computed by the nodes, the methods used for estimation of bandwidth and link stability are not specified. The protocol in [59] combines NDMR [58] with DiffServ for supporting QoS in mobile ad hoc networks. To improve QoS performance of DiffServ, NDMR helps in reducing routing overhead and controlling congestion through load balancing over multiple node-disjoint paths. Finally, in all the multipath routing protocols found in literature, a reasonable amount of path diminution problem [2] exist. The main reasons are the RREQ forwarding policy and the path selection procedure adopted at the intermediate nodes and the destination respectively.

5.3 Comparative study of different multipath routing protocols

In this section, we have compared the main properties of the multipath routing protocol discussed in the previous section. Table 5.1 compares these protocols based on the properties as discussed below.

- *RREQ Forwarding Policy*: This property says what policy is chosen by an intermediate node to handle duplicate RREQ packets such that the protocol is able to discover multiple path with the specified characteristics. The two extremes are dropping all the duplicate RREQ packets and rebroadcasting all the RREQ without considering the number of copies received.

- *Type and No. of multipaths*: This property says - what will be the disjointness property of the discovered multiple paths which will be selected either by the destination or by the source and how many such paths will be selected at the maximum.

- *Route Stability*: This property says what type of measures are taken, if any, to compute stable routes.

- *Admission Control*: This property mention whether there are some restriction, either link-wise or path-wise, besides due to the *RREQ Forwarding Policy* to process and forward received RREQ packet.

- *QoS Support*: It indicate what type of QoS metrics are incorporated in the proposed multipath routing.

- *Path Diminution*: Path diminution means a badly selected path at an early stage of route discovery in a network can block other possible disjoint paths. This is due to the RREQ forwarding policy (RREQ is forwarded only once) generally adopted in a routing protocol by an intermediate node. Though it is very difficult to remove the path diminution problem without excessive control and computation overheads, it can be reduced by adopting some policies.

- *Route Computation Overheads*: This gives the amount of processing required for computing disjoint paths at a node (generally at the destination) after collecting and processing the RREQ packets.

- *Path Rerouting*: It indicates what triggers a new route recovery in a multipath routing protocol.

- *Amount of Flooding*: It gives an estimation of amount of flooding based on the RREQ forwarding policy adopted in an intermediate node.

- *Alternate Route Maintenance*: It says whether a particular multipath routing algorithm after discovering multiple paths maintains the alternate paths for making them up to date.

From the comparative studies shown in Table 5.1, we can conclude that none of the multipath routing protocols are adequate to provide delay and throughput assurance to applications requiring QoS in MANETs. Most of them suffers from severe path diminution problems and do not maintain the alternate paths to keep them upto date. Further, except one [92], none of them pay attention to proactive route maintenance through detection of potential link breaks to save route recovery times. Designing of a multipath QoS routing for MANETs that can alleviate or reduce these drawbacks is the main objective of SMQR.

5.4 Route Stability based Multipath QoS Routing (SMQR) Protocol

In the proposed route stability based multipath QoS routing, SMQR, uses the basic route discovery procedure of QoS-aware unipath protocol RSQR with some modifications. SMQR like its unipath counterpart uses a new QoS parameter called *Route Stability* along with *throughput* and *delay* and uses the route stability model (RSM) discussed in section 4.4 of chapter 4 for computing route stability. Thus, inclusion of some extra fields in route request/reply packets of the routing protocol help in

Table 5.1: Comparison of Main Characteristics of existing multipath routing protocols

Protocol ⇒ Characteristics ⇓	AOMDV [71]	AODVM [118]	MP-AOMDV [92]	MP-DSR [56]	ADQR [40]	MQRD [59]	IMRP [110]	QDMR [109]
RREQ forwarding policy	Only first copy and saves other copies with distinct FID	Only first copy and saves all other copies	Same as AOMDV	Determined by required number of disjoint paths	Only disjoint copies	Based on hop count	Only first copy,(no copy saved)	Based on hop count and with distinct FID
Type and No. of multipaths	Link and Node disjoints, N	Node-disjoint, N	Link and Node disjoints, N	Node-disjoint, N (determined by end-to-end reliability reqts)	Node-disjoint (w.r.t the first copy), N (user b/w reqts)	Node-disjoint, 3,(w.r.t the first copy)	Node-disjoint, N (w.r.t the first copy)	Maximally disjoint, 2
Route stability	No	No	Signal strength on links	Path reliability	Signal Strength on links	No	Path stable time	No
Admission Control	No	No	No	Path reliability and link availability	No	No	Link b/w, link stable time	admissible rate of real-time traffic
QoS Support	No	No	No	End-to-End reliability	Bandwidth, route class	Through DiffServ	Bandwidth	DiffServ
Path Diminution	High	High	High	High	Less	High	High	Low
Route Computation Overheads	Less	Less	Less	High	Moderate	Moderate	Moderate	Moderate
Path Rerouting	if any route fails	if any route fails	if all routes fail	if all routes fail or time window for a route expires	When active route fails or active route class degrades	if all except one fail	if path stable time expires or any route fails	if any route fails
Amount of Flooding	Less	Less	Less	Less	High	High	Moderate	Moderate
Alternate Route Maintenance	No	No	Yes	No	All paths	No	No	Yes

FID: First Hop ID

selecting routes with higher stability among all the feasible paths between a given source-destination pair. SMQR computes a maximum of three node-disjoint QoS routes from source to destination and uses the route with maximum route stability as the primary path and the others as the secondary paths. Figure5.1 shows two types of disjoint paths between node S and D, where Figure 5.1(a) is for node-disjoint paths and Figure 5.1(b) for link-disjoint paths. It is easy to show that the number of node-disjoint or link-disjoint paths between a given pair of source and destination is bounded by the number of one-hop neighbors of the source and the destination. Our protocol computes node-disjoint paths to increase the likelihood that when the primary path breaks due to node movement or QoS violation, source can still find one of the alternate paths which is valid (validity is ensured through route maintenance). Further, preferring routes with higher stability improve the reliability of the discovered paths. Previous studies [71, 92, 118] show that such node-disjoint paths typically fail independently. The possible number of completely node-disjoint paths with QoS constraints at a moderate node densities is very limited [118] and performance improvement due to increase in number of disjoint paths beyond two or three paths are very marginal [76]. In SMQR, a periodic maintenance and validation of the alternate paths is performed which results in switching of primary route to an alternate route, if the stability value of the alternate route is higher than that of the primary route. A new route discovery is initiated only when all the paths in multipath fail. This means the policy for selecting multipath should be based on selecting paths with minimum number of common nodes or maximally disjoint paths. For the route rediscovery policy, where route discovery is performed if one of the paths in multipath fails, multipath selection should be based on number of hops in the paths. This can be shown by the following simple analysis on path failure probability.

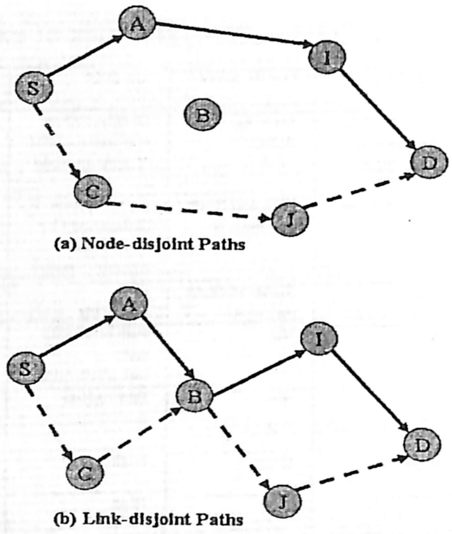


Figure 5.1: An example showing Disjoint Paths between S and D

Let us assume for simplicity that each multipath computed by a routing consists of two paths. We define two multipaths MP_1 and MP_2 as follows. $MP_1 = \{P_{11}, P_{12}\}$ and $MP_2 = \{P_{21}, P_{22}\}$. Where P_{11} and P_{12} are two *node-disjoint* paths with m and n numbers of nodes (except source and destination) respectively. Similarly, P_{21} and P_{22} are two paths with m and n number of nodes (except source and destination) respectively with one intermediate node in common. Let us further assume that each node in the network has the constant and independent probability to break a link due to mobility and is equal to ρ. Therefore, each node in the network remains stable with probability $\tau = 1 - \rho$. We assume that the nodes move independently in the network. Therefore, the probability of path break, P_b, and path stability, P_s, for any unipath, P, with k number of intermediate nodes can be defined as : $P_b(P) = 1 - (1 - \rho)^k$, and $P_s(P) = (1 - \rho)^k$. Now, for a multipath MP, the path break probability, $P_b(MP)$, can be defined based on the policy used for rediscovery during route failure. Now, we apply the two rediscovery policies to our example multipaths MP_1 and MP_2 to see which one is a better choice in which multipath rediscovery policy.

First, we apply the rediscovery policy that failure of any one of the paths in multipath should result in route discovery. In this case, the path break probabilities for MP_1 and MP_2 are given in Equation (5.1) and Equation (5.2) respectively.

$$\begin{aligned}
P_b(MP_1) &= P_b(P_{11}) \bigcup P_b(P_{12}) \\
&= 1 - P_s(P_{11}) \bigcap P_s(P_{12}) \\
&= 1 - (1-\rho)^m \times (1-\rho)^n \\
&= 1 - (1-\rho)^{m+n} \qquad (5.1)
\end{aligned}$$

Similarly,

$$\begin{aligned}
P_b(MP_2) &= P_b(P_{21}) \bigcup P_b(P_{22}) \\
&= 1 - P_s(P_{11}) \bigcap P_s(P_{12}) \\
&= 1 - (1-\rho)^m \times (1-\rho)^{n-1} \\
&= 1 - (1-\rho)^{m+n-1}
\end{aligned} \tag{5.2}$$

From Equation (5.1) and Equation (5.2), we have found that $P_b(MP_1) \geq P_b(MP_2)$. This means that lesser the number of distinct nodes or more the number of common nodes in the multipath, smaller is the probability of break for the multipath and better is the stability for multipath.

Now, consider the route rediscovery policy wherein failure of both of the paths will result in the failure of the multipath and hence rediscovery process will be initiated. The path break probabilities for MP_1 and MP_2 are as given in Equation (5.3) and Equation (5.4) respectively.

$$\begin{aligned}
P_b(MP_1) &= P_b(P_{11}) \bigcap P_b(P_{12}) \\
&= (1-(1-\rho)^m) \times (1-(1-\rho)^n)
\end{aligned} \tag{5.3}$$

$$\begin{aligned}
P_b(MP_2) &= P_b(P_{21}) \bigcap P_b(P_{22}) \\
&= \left(1-(1-\rho)^{m-1}\right) \times \left(1-(1-\rho)^{n-1}\right) + \rho
\end{aligned} \tag{5.4}$$

To find out the relationship between $P_b(MP_1)$ and $P_b(MP_2)$, we calculate $P_b(MP_2) - P_b(MP_1)$ using Equation (5.3) and Equation (5.4) as follows.

$$\begin{aligned}
P_b(MP_2) - P_b(MP_1) &= \left(1-(1-\rho)^{m-1}\right) \times \left(1-(1-\rho)^{n-1}\right) + \rho \\
&- (1-(1-\rho)^m) \times (1-(1-\rho)^n) \\
&= (1-\tau)\left[1-\tau^{n-1}-\tau^{m-1}+\tau^{n-1}\tau^{m-1}(1+\tau)\right] \\
&\geq (1-\tau)(1-\tau^{n-1}-\tau^{m-1}+\tau^{n-1}\tau^{m-1}) \\
&= (1-\tau) \times (1-\tau^{n-1}) \times (1-\tau^{m-1}) \\
&\geq 0
\end{aligned} \tag{5.5}$$

From Equation (5.5), we can conclude that $P_b(MP_1) \leq P_b(MP_2)$. This means, for rediscovery policy where the route discovery process is initiated when all the paths in multipath fail, multipath with no common nodes or less number of common nodes among the paths are better than the multipath with common nodes or larger number of common nodes. The above analysis shows why we prefer node-disjoint paths in our QoS-aware multipath routing protocol.

Route discovery procedure in SMQR is based on the following policies - *(i)* Selective forwarding of duplicate route request message is adopted to maximize the possibility of tracing all the feasible disjoint paths from source to destination and *(ii)* Route request message received by the destination will carry complete path information from source to destination (like in DSR) that helps the destination to compute a set of node-disjoint paths by processing those path information. These two policies adopted are particularly important in a QoS aware routing, due to the limitation of number of feasible paths in the network.

5.4.1 Assumptions

We have made no assumptions about the global state of the network at the nodes. Instead, we proposed an on-demand distributed routing algorithm where all the nodes contain only information about their one hop neighbors. The following network model assumptions are made.

- Neighborhood is commutative property i.e. if node A can hear node B this implies node B also hears node A
- CSMA/CA like MAC protocol is used for unicast communication and it solves the hidden terminal problem with the help of RTS-CTS control packets.
- There is a close interaction between MAC layer and Network layer
- Combinatorial stability of network is assumed, it means topology changes occur sufficiently slowly to allow successful propagation of all topology updates as necessary
- *Hello* intervals to update neighbor information is reasonable to capture the dynamics of the network
- Transmission range and carrier sensing range are assumed to be same for available bandwidth calculation at a node
- Initial route discovery latency is tolerable in the supported applications

5.4.2 Delay and Bandwidth Estimation

We have measured per hop delay by using MAC delay at a node which is a cross layer information. A node computes the MAC delay (d) by subtracting the time (t_s) that a packet is passed to the MAC layer from the time (t_r) an ACK packet is received from the receiver for the same packet. In order to minimize the biasness against transient changes in delay, an Exponentially Weighted Moving Average (EWMA) estimator is used to measure the average of MAC delay (d^i_{avg}) as shown in Equation (5.6). Here, d^i and d^{i-1}_{avg} are measured MAC delay and previously stored average MAC delay in a node. η is a positive constant which determines how much effect the current measure delay has on the average value, and in our experiment we set it as 0.4.

$$d^i_{avg} = \eta \times d^{i-1}_{avg} + (1-\eta) \times d^i \tag{5.6}$$

Initially, MAC delay in a node is set as the sum of propagation delay, the backoff delay due to half of the initial window size, DIFS and SIFS (which are based on the IEEE 802.11 PHY layer standard). Each node estimates the available bandwidth at the node by subtracting the consumed aggregated bandwidth in the interference range of the node (getting from Neighbor Information Table) from the raw bandwidth as used in RSQR (see section 4.5.5 in chapter 4). Therefore, to facilitate estimation of available bandwidth at a node, each node stores the bandwidth consumed by the active flows passing through the node, and then making these information available to their neighbors through periodic exchange of *Hello* packet.

5.4.3 Control packets used in Routing

Before we discuss the route discovery and maintenance process of SMQR, we briefly describe about the control packets used in our proposed routing protocol.

- ***QRREQ***: This packet is basically the AODV Route Request (RREQ) packet with QoS extension, which is used for sending route request for finding node-disjoint multipath in the proposed protocol. It carries Path information, QoS and route stability related fields along with the fields used in RREQ packet in AODV.

- ***QRREP***: This packet is basically AODV Route Reply (RREP) packet with QoS extension, which is used for sending route reply in the proposed protocol. It contains complete path information from source to destination with QoS and route stability related fields along with the fields used in RREP packet in basic AODV protocol.

- ***RouteM***: This is the Route Maintenance packet sent from source to destination and vice-versa, to keep the secondary paths active (during its journey from source to destination) and to measure the route stability of the maintained routes (during its journey from destination back to the source).

- ***RERR***: This is the Route Error packet generated after detecting link lost or QoS violation by an intermediate node/destination. It is similar to the Route Error (RERR) packet used in AODV except that it carries the type of error for which the RERR packet is generated. An RERR can be generated due to - link failure, delay violation and throughput violation.

- ***HELLO***: This Hello packet is required for Neighborhood maintenance, refreshing signal strength value from neighbors, providing bandwidth reservation information. It is sent always with TTL set as 1.

5.4.4 Tables used in Routing

Implementation of SMQR requires the use of following tables in the nodes of the network.

- ***Neighbor Information Table (NIT)***:This table is used to keep information about neighbors, where one entry per neighbor is maintained. Per neighbor, information about bandwidth consumed by active flows at the neighbor and signal strength value of last packet received from the neighbor with the validity period are saved.

- ***Flow Table (FT)***: For reserving the required bandwidth for each of the active flow passing through the node for reservation period.

- ***Route Request Forwarded Table (RFT)***: This table is used to store route stability information about QRREQ packets received and forwarded by an intermediate node, per unique first hop neighbor of the source. It is used during route discovery to assist selection of highest stable node-disjoint routes at the destination.

- ***Routing Table (RT)***: To setup and remember forward path and reserve path entry after processing QRREP and QRREQ packets respectively as used with AODV.

- ***Route List Table (RLT)***: This table is used only by the destination to store the sorted (according to route stability value) list of received QRREQ packets to be processed for computing node-disjoint paths at the destination.

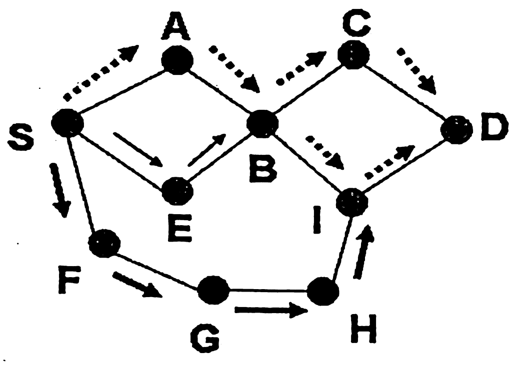


Figure 5.2: An example showing the need of special forwarding rule at node I

5.4.5 Route Discovery

In SMQR, we have modified the basic AODV protocol with additional mechanism to enable discovery of multiple node-disjoint QoS paths for a source and destination pair. *Algorithm* 5.1 gives the outlines of the route discovery process initiated by the source node. The route request processing done at an intermediate node and at the destination are given in *Algorithm* 5.3 and *Algorithm* 5.4 respectively. The admission control performed during route request processing is shown in *Algorithm* 5.2. *Algorithm* 5.5 gives the overview of the processing done at an intermediate node during forwarding of route reply packets from destination toward the source.

To compute a QoS-aware multipath to a destination D, source S generates a QoS Route Request (QRREQ) packet with values for B_{min}(minimum throughput) and D_{max} (maximum delay) as the application's QoS requirements. Among the other fields carried by the QRREQ are - ADELAY (Accumulated delay) to store end-to-end delay of the explored path, APS/APU (Accumulated Path stability/Accumulated Path Uncertainty) for storing route stability of the explored path, and *seq_no* to uniquely identify the route. Besides these fields, QRREQ packets contain a field to store the address of the one hop neighbor of the source through which a particular copy of QRREQ packet has been forwarded by the source node. It also contains a field to store the hop count of the path from source to the current node as used in AODV. QRREQ packet further records the IDs of the nodes in a field called APATH (Accumulated Path) through which a QRREQ.packet starting from the source reaches the destination. QRREQ packets which are received with signal strength less than a threshold, Thr_2, are rejected to avoid possibility of selecting path with very weak links. Admission control for throughput and delay is performed hop by hop basis during forwarding of QRREQ/QRREP packets for QoS assurance to applications.

Algorithm 5.1 Algorithm for route discovery by S for a QoS multipath to D in SMQR

Input: B_{min} (Minimum bandwidth), D_{max} (Maximum end to end delay)

$Seq\_no := Seq\_no + 1$ {* Seq_no is used to identify QRREQ *}
$ADELAY := 0$; $APATH := S$; $HopCount := 0$; $Bmin := B_{min}$; $Dmax := D_{max}$
$APU := 0$; $APS := 1$; $First\_Hop\_ID := null$; $Last\_Hop\_ID := null$
Check throughput admission control for $Bmin$
if (Admission control fails) **then**
  print "Not possible to find a Path!"
  exit(1)
end if
Construct a QRREQ with the required fields
start-timer ($3 \times Dmax$)
Broadcast QRREQ
$N := 0$ {* To Count the number of corresponding QRREP *}
$APU^* := 10$; $APS^* := 0$; $Next\_Hop := S$; $APATH^* := null$
while (not (timeout) and ($N < 3$)) **do**
  if (A QRREP (S, D, Seq_no, $First\_Hop\_ID$, $Last\_Hop\_ID$, $HopCount$, $APATH$, $Dmax$, $Bmin$, $ADELAY$, APS, APU) is received from j) **then**
    Check admission control for delay and throughput
    if (admission control fails) **then**
      discard the QRREP; continue to receive more QRREP
    end if
    $N := N + 1$
    Make a Routing Table entry with (S, D, Seq_no, S, j, $Bmin$, $Flag =$ "A") {* $Flag =$ "A" is used for setting Routing table entry as Alternate Path *}
    if (($APU < APU^*$) or (($APU = APU^*$) and ($APS > APS^*$))) **then**
      $APU^* := APU$; $APS^* := APS$; $Next\_Hop := j$; $APATH^* := APATH$
    end if
  end if
end while
if ($N = 0$) **then**
  print "No Route to D found!"
  Wait for a backoff time and start route discovery
else
  Mark the RT entry for D with $Next\_Hop$ as the next hop as Primary ($Flag =$ "P")
  Set Timeout for RT entries for the flow as T
  Make the FT entry with timeout as T
end if

To control the amount of broadcast of QRREQ packets in SMQR, an approach to selectively broadcast duplicate copies of QRREQ packet is used. We have used the advertised hop count to limit the number of duplicate QRREQ packets that can be forwarded by an intermediate node. This advertised hop count is stored in a reverse routing table entry when the first copy of the QRREQ packet for a given source destination pair is forwarded by a node. This results in elimination of forming routing loops and prevent QRREQ packets from traversing in the reverse direction. Each reverse routing table entry is stored only for a timeout period of $3 \times D_{max}$. After this timeout period, if no QRREP packet comes, the reverse routing table entry along with any temporary entries created in RFT for the route request are removed. The reason for setting timeout period as $3 \times D_{max}$ is to allow the possible maximum time to receive the corresponding QRREP packet. This maximum time includes round trip time from the node to the destination, which is $2 \times D_{max}$ and the extra processing/waiting time used in the destination, which we set in the order of D_{max}. Any duplicate QRREQ packets with hop count less than or equal to the advertised hop count and coming through a distinct first hop neighbor of the source or through the same first hop neighbors of the source with better route stability are forwarded by the intermediate node. RFT maintained in each node remembers the first hop neighbor and corresponding route stability values for the already forwarded QRREQ packets for each route request. Therefore, any QRREQ packet that arrived at an intermediate node will be dropped, if it does not pass the admission control check or can not be forwarded due to the forwarding policy adopted as discussed above.

To reduce possible path diminution, we have adopted a special QRREQ forwarding rule as follows. Suppose, there is a disjoint path (with distinct first hop neighbor of the source) traced by a QRREQ packet from source to an intermediate node with hop count greater than the stored advertised hop count for the same route request packet. In this case, the general QRREQ forwarding policy discussed above will not allow the node to forward this QRREQ packet. If the node under consideration is happened to be a neighbor of the destination, forwarding this QRREQ will enable destination to detect another node-disjoint path without much overhead. We have adopted this extra forwarding rule in our routing protocol. As shown in Figure 5.2, suppose, S starts a multipath discovery for D and the QRREQ packet from S following the path S-A-B-I reaches node I before the QRREQ packet which follows the path S-F-G-H-I. Without this special QRREQ forwarding rule, node I will drop the copy of QRREQ packet receives from H, which will eliminate the possibility of detecting two disjoint paths. By adopting the special forwarding rule, node I will forward the second copy of QRREQ, which will result in detection of a multipath with two node-disjoint paths (S-A-B-C-D and S-F-G-H-I-D) or (S-E-B-C-D and S-F-G-H-I-D) at the destination D.

Algorithm 5.2 Algorithm to perform Admission Control at node i

Input: A QRREQ packet from j with S, D, Seq_no, $First\_Hop\_ID$, $Last\_Hop\_ID$, $HopCount$, $APATH$, $Dmax$, $Bmin$, $ADELAY$, APS and APU
Input: SS^2, SS^1 : recent and current signal strengths values

if ($SS^1 < Thr_2$) **then**
  Drop the QRREQ packet; return (-1)
end if
Check admission control for $Dmax$ and $Bmin$
if (Admission control fails) **then**
  Drop QRREQ packet; return (-1)
end if
if ($First\_Hop\_ID$ in QRREQ is NULL) **then**
  Insert i as the $First\_Hop\_ID$ in QRREQ
end if
$HopCount := HopCount + 1$
Compute LS and LU using $Algorithm$ 4.3 {* shown in Chapter 4*}
Modify $ADELAY$, APS, APU, $HopCount$ in QRREQ
Append i in $APATH$ of QRREQ
return (0)

Algorithm 5.3 Algorithm to process and update QRREQ packets at intermediate node i in SMQR

Input: A QRREQ packet from j with S, D, Seq_no, $First\_Hop\_ID$, $Last\_Hop\_ID$, $HopCount$, $APATH$, $Dmax$, $Bmin$, $ADELAY$, APS, and APU

Input: SS^2, SS^1 : recent and current signal strengths values

$Flag :=$ Algorithm 5.2(QRREQ) {*Performs admission control and compute link stability*}
if ($Flag < 0$) **then**
    return {*QRREQ does not pass admission control*}
end if
if (QRREQ is a fresh request) **then**
    Remember the Seq_no
    Make a reverse route table entry and save $HopCount$ as the $advertised\_hop\_count^D$ for QRREQ
    Store (S, D, Seq_no, $First\_Hop\_ID$, APS, APU) in RFT; Rebroadcast QRREQ packet
else if (QRREQ is a duplicate request) **then**
    if ($HopCount \leq advertised\_hop\_count^D$) **then**
        if ($First\_Hop\_ID$ for QRREQ is not present in RFT) or (New QRREQ has better Stability value) **then**
            Store (S, D, Seq_no, $First\_Hop\_ID$, APS, APU) in RFT; Rebroadcast QRREQ packet
        end if
    else if (D is a neighbor of i) and ($First\_Hop\_ID$ for QRREQ is not present in RFT) **then**
        Store (S, D, Seq_no, $First\_Hop\_ID$, APS, APU) in RFT; Rebroadcast QRREQ packet {*A Special case to reduce path diminution*}
    else
        Drop QRREQ packet {* QRREQ is having a hop count greater than $advertised\_hop\_count^D$*}
    end if
else
    Drop QRREQ packet {* Outdated QRREQ *}
end if

Algorithm 5.4 Algorithm to process QRREQ packets at destination node D in SMQR

Input: A QRREQ packet from j with S, D, Seq_no, $First\_Hop\_ID$, $Last\_Hop\_ID$, $HopCount$, $APATH$, $Dmax$, $Bmin$, $ADELAY$, APS, and APU

Input: SS^2, SS^1 : recent and current signal strengths values

$Flag$:= Algorithm 5.2(QRREQ)
if ($Flag < 0$) **then**
  return {*Admission control fails*}
end if
if (QRREQ is a fresh request) **then**
  Remember the Seq_no; Insert j in $Last\_Hop\_ID$; Make a reverse route table entry for QRREQ
  Store the relevant information of QRREQ (i.e., S,D,Seq_no, $First\_Hop\_ID$, $Last\_Hop\_ID$, $APATH$, APS and APU) in Route List Table(RLT)
  start_timer(RRL) {*RRL is Route Reply Latency *} {*For the 1st copy of QRREQ only*}
  while (not time-out) **do**
    if (QRREQ with (S, D, Seq_no, $First\_Hop\_ID$, $Last\_Hop\_ID$, $HopCount$, $APATH$, $Dmax$, $Bmin$, $ADELAY$, APS, APU) received from j) **then**
      Flag:= Algorithm 5.2(QRREQ) {*check for admission control*}
      if (Flag < 0) **then**
        Discard the QRREQ and wait for the next QRREQ arrival
      end if
      Insert j in $Last\_Hop\_ID$
      if (QRREQ with $First\_Hop\_ID$ and $Last\_Hop\_ID$ exists in RLT with better Route stability) **then**
        Discard the QRREQ and wait for the next QRREQ arrival
      else if (QRREQ with $First\_Hop\_ID$ and $Last\_Hop\_ID$ does not exists in RLT) **then**
        Store the relevant information of QRREQ in RLT
      else {New QRREQ carries better route}
        Replace the APS, APU and $APATH$ fields in the stored RLT entry with corresponding values in QRREQ
      end if
    end if
  end while
  Sort the RLT entries in descending order of route stability to get $FeasiblePaths := \{P_0, P_1, ..., P_{m-1}\}$ {*m is the number of paths*}
  $FeasibleSet := \{0, 1, ..., m-1\}$; $SelectedPath := \phi$
  call $PathSelection(FeasibleSet, SelectedPath)$ {*call Algorithm 5.6*}
  PRI_PATH := Path information in lst element of $SelectedPath$
  if ($|SelectedPath| > 1$) **then**
    $ALTER\_PATH_1$:= Path information in 2nd element of $SelectedPath$
  end if
  if ($|SelectedPath| > 2$) **then**
    $ALTER\_PATH_2$:= Path information in 3rd element of $SelectedPath$
  end if
  Sends a QRREP for each of the path computed; Make a Routing table entry for S
end if

Algorithm 5.5 Algorithm to process QRREP at intermediate node i in SMQR

Input: A QRREP packet from j with S, D, Seq_no, $APATH$, $Dmax$, $Bmin$, $ADELAY$, APS, and APU
Input: SS^2, SS^1 : recent and current signal strengths values
1: Perform admission control for QRREP {* similar to Algorithm 5.2*}
2: **if** (admission control fails) **then**
3: Drop the QRREP; remove reverse route entry for QRREQ; remove RFT entries for QRREQ
4: return
5: **end if**
6: $P :=$ previous node of i in $APATH$
7: Make a Routing Table entry with (S, D, Seq_no, P, j, $Bmin$) with timeout $2 \times D_{max}$
8: Make a Routing Table entry with (D, S, Seq_no, j, P) with timeout $2 \times D_{max}$
9: Make a Flow Table entry with (S, D, Seq_no, $Bmin$) with timeout $2 \times D_{max}$
10: Unicast QRREP packet to P

5.4.6 Route Selection and Establishment

The destination after receiving the first copy of QRREQ packet with sufficient signal strength ($\geq Thr_2$) that passes the admission control, waits for a small time period called *Route Reply Latency* (RRL) to allow receiving all possible feasible QRREQ packets (satisfying QoS requirements). For each incoming link of the destination (called $Last\_Hop\_ID$), the destination is allowed to store at most one QRREQ (corresponding to the highest stable QoS path) per $First\_Hop\_ID$ of the source. This restricts the maximum number of QRREQ packets that should be handled by the destination as $d_1 \times d_2$, where d_1 and d_2 are the degrees of source and destination respectively. After waiting for the RRL, destination sorts the QRREQ packets in descending order of their stability values and runs a path selection algorithm to compute a maximum of three node-disjoint paths from the sorted QRREQ packets.

In $PathSelection$ algorithm (shown as Algorithm 5.6), two arguments are passed. The first one, $PathSet$, is the set of integers representing the given set of feasible paths in sorted order (of route stability) and the second one, $DisjointSet$, is the set of node-disjoint paths the procedure will return once it terminates. It can be seen that $PathSelection$ returns immediately once it detects three node-disjoint paths. Due to the order maintained for checking disjointness, these returned paths are always the highest stable node-disjoint paths. The number of paths to be examined for checking node-disjointness reduces in each of the subsequent steps. $PathSelection$ at the destination returns three, two or one node-disjoint paths depending on the availability of node-disjoint paths. Destination unicasts QRREP packets to the source through the selected paths. While forwarding the QRREP packets all intermediate nodes perform admission control, creates routing table entries (with certain timeout period) one for destination and one for source, and reserves the required bandwidth in the Flow Table. The timeout period for both forward and reverse entries in RT are set as $2 \times D_{max}$. This is to allow the maximum time period within which either a data packet or a $RouteM$ packet will be received for the forward entry. The reverse RT entry which are used only to return $RouteM$ packets back to the source are made active as long as the corresponding forward RT entry is active. The timeout periods for RT entries are reset to T, after forwarding the first data packet or $RouteM$ packet using the corresponding RT entry. Similarly, timeout period for an FT entry is set as T after forwarding first data/$RouteM$ packet through the corresponding RT entry. Source, after receiving the QRREP packets, sets the routing table entry with the associated flag indicating whether the corresponding path is a primary or a secondary path. As shown in Algorithm 5.1, source selects the route with highest stability value as the primary and the others (if any) as the secondary.

Algorithm 5.6 PathSelection ($PathSet$, $DisjointSet$)

Input: $FeasiblePaths$ (the sorted list of QoS-aware path computed at destination)

Output: $DisjointSet$ (A set of node-disjoint QoS-aware paths)

```
BestSet := {0} {* 0 represents the 1st element in PathSet *}
m := |PathSet|
WorkingSet := PathSet
for (i = 0 to m − 1) do
    WorkingSet := WorkingSet \ {i}
    DisjointSet := {i}
    TempSet := WorkingSet
    for (k = 1 to 2) do
        for all (j ∈ TempSet) do
            TempSet := TempSet \ {j}
            if (j is disjoint to DisjointSet) then
                DisjointSet := DisjointSet ⋃{j}
                if (|BestSet| < |DisjointSet|) then
                    BestSet := DisjointSet
                end if
                break
            end if
        end for
    end for
    if (|DisjointSet| = 3) then
        return
    end if
end for
DisjointSet := BestSet
```

5.5 Route maintenance

After successful discovery of QoS routing paths, the source admits real-time traffic into the primary path. To maintain QoS assurance, the maintenance of the primary path as well as the secondary paths are very important. Accordingly, we have two types of route maintenance - one is due to the QoS violation on the active primary path and the other is continuous maintenance of alternate paths to ensure that only valid paths are maintained. Both of these methods reduce QoS disruptions.

5.5.1 Maintenance due to QoS Violation

QoS violation due to end-to-end delay violation is detected by the destination by computing one way delay (T_d) experienced by the data packets from the sender's timestamp on the received data packets and already stored clock offset of the destination with respect to the source. Therefore, destination detects delay violation for a data packet, if $T_d > D_{max}$. Similar to RSQR (as discussed in section 4.5.9) a constant, n, is used to handle the case of a temporary delay violation. Bandwidth violation is detected by each of the node including the destination, by observing the absence of data packet in an active route for a particular time interval called bandwidth reservation timer (T)[115], which is defined as ($k * N * 8/B_{min}$), where k is the allowed packet loss and N is the maximum packet length (in bytes) used at PHY level. Link failure detection follows the method using periodic *Hello* packets. Any node which detects either a QoS violation or a link failure, informs the source by sending a route error (RERR) packet. All intermediate nodes which forward the RERR packets release the resources reserved by them for the flow.

5.5.2 Maintenance of Alternate Paths

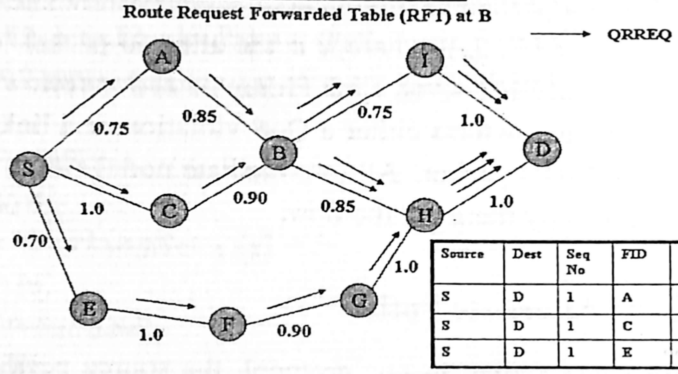
To make the secondary route active, in our protocol, the source periodically sends some dummy packets forwarded through these routes. We have used a special dummy packet called *Route Maintenance* (*RouteM*), which is sent periodically to keep the secondary route(s) active. The period for sending route maintenance packet should be adjusted according to the active routing table entry timer for a route. We have used *RouteM* packet for two purposes - one is to keep the secondary route active and the other is to measure route stability of the maintained routes. To enable continuous measurement of route stability of the primary path, we have used the same *RouteM* packet on primary path also. Once a *RouteM* packet is received by a destination through a route it sends a *RouteM* packet back to the source immediately. During its journey from destination to the source, the *RouteM* packet will measure the route stability of the path. To enable a forwarding node to know whether or not to update route stability values (in APS and APU fields) a special flag is included in the *RouteM* message. Finally, based on the received values of the route stabilities for different routes (secondary routes and primary route) the source will decide whether to switch to a new route or not. To safeguard unnecessary route oscillations (switching back and forth), the difference in stability value between current primary path and a secondary path with higher route stability should exceed a threshold (defined as Δs) to trigger the route switching. If source does not receive a *RouteM* packet in response to its previously sent *RouteM* packet through a secondary path, the source assumes the failure of the corresponding path and a route recovery is initiated (as discussed in section 5.5.3). To handle temporary packet transmission errors, source declares a route failure if it does not receive two consecutive *RouteM* packets for the same path. Any intermediate node which receives a *RouteM* but can not forward it due to failure of the uplink, will generate a RERR message for the route.

5.5.3 Route Recovery by Source

Finally, when the source gets a RERR message for a route, it will take action as follows. If the RERR message is received through a secondary route, the source will remove the corresponding secondary route entry from its routing table. Otherwise, if the RERR message is received through the primary route, it will immediately switch the primary route to the highest stable secondary route available in the routing table. In the case, where there is no valid secondary route available and a RERR message is received through the primary route, the source will initiate a new route discovery process, if there remain more data packets to be sent for the flow.

Source	Dest	Seq No	FID	APS, APU
S	D	1	A	0.638, 0
S	D	1	C	0.9, 0

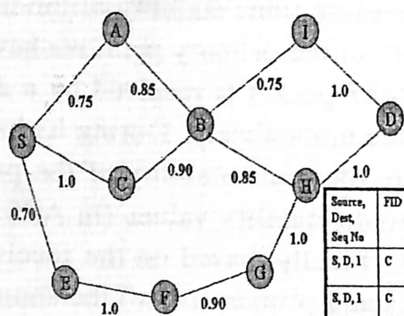
Route Request Forwarded Table (RFT) at B



Source	Dest	Seq No	FID	APS, APU
S	D	1	A	0.542, 0
S	D	1	C	0.765, 0
S	D	1	E	0.63, 0

Route Request Forwarded Table (RFT) at H

Figure 5.3: An example of Node-disjoint multipath route discovery (S initiates route discovery)



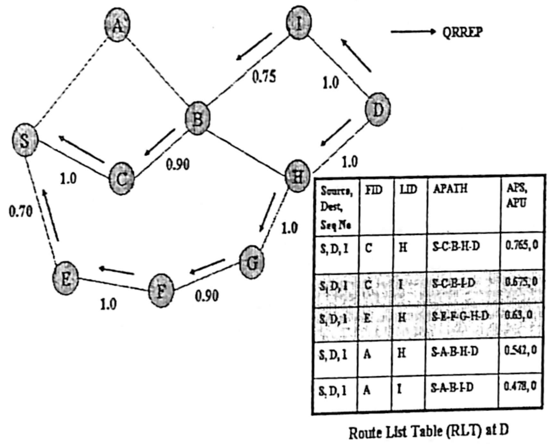
Source, Dest, Seq No	FID	LID	APATH	APS, APU
S, D, 1	C	H	S-C-B-H-D	0.765, 0
S, D, 1	C	I	S-C-B-I-D	0.675, 0
S, D, 1	E	H	S-E-F-G-H-D	0.63, 0
S, D, 1	A	H	S-A-B-H-D	0.542, 0
S, D, 1	A	I	S-A-B-I-D	0.478, 0

Route List Table (RLT) at D

Figure 5.4: An example of Node-disjoint multipath route discovery (D prepares a Route List Table)

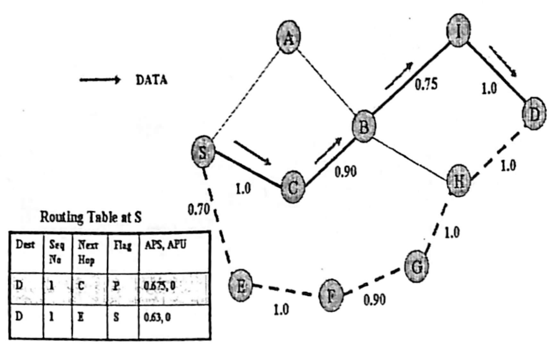
5.5.4 An example showing route discovery and maintenance

Figure 5.3 to Figure 5.6 show the node-disjoint multipath route discovery process of SMQR for an arbitrary network of 10 nodes. Here, for the sake of simplicity, we have not shown processing related to admission control. Further, we have assumed that all QRREQ/QRREP packets pass admission control due to bandwidth and delay. In the figures, the label shown below a link represents the current link stability value of that link. As shown in Figure 5.3, node *S* initiates a route discovery to find a multipath QoS route to node *D* by broadcasting a QRREQ packet. Node *S* initializes the fields in QRREQ packet in accordance with QoS requirement of the flow as discussed in section 5.4.5. Every intermediate node, after receiving a QRREQ packet, processes the QRREQ packet as explained in



Source, Dest, Seq No	FID	LID	APATH	APS, APU
S, D, 1	C	H	S-C-B-H-D	0.765, 0
S, D, 1	C	I	S-C-B-I-D	0.675, 0
S, D, 1	E	H	S-E-F-G-H-D	0.63, 0
S, D, 1	A	H	S-A-B-H-D	0.542, 0
S, D, 1	A	I	S-A-B-I-D	0.478, 0

Figure 5.5: An example of Node-disjoint multipath route discovery (D computes 2 node-disjoint paths and sends QRREP packets)



Dest	Seq No	Next Hop	Flag	APS, APU
D	1	C	P	0.675, 0
D	1	E	S	0.63, 0

Figure 5.6: An example of Node-disjoint multipath route discovery (S creates Routing Table entries and set these entries as primary or secondary)

Algorithm 5.3. Arrows shown along with the links (in Figure 5.3) indicate the forwarding of the QR-REQ packet according to the rules as discussed earlier. The number of arrows on a link represents the number of QRREQ packet forwarded over the link. The nodes use Route Request Forwarded Table (RFT) and advertised hop count (which is stored in reverse route entry) for realizing the QRREQ forwarding policy. The RFT entries created in nodes *B* and *H* during the route discovery process are shown in the Figure 5.3. We have assumed that destination *D* receives a total of five QRREQ packets within its RRL. Destination *D* stores the relevant fields from the received QRREQ packets in Route List Table (RLT) (in the descending order of route stability) as shown in Figure 5.4. Then it executes the route selection algorithm shown in Algorithm 5.6 to compute the multipath. The shaded entries in RLT at *D* (in Figure 5.5) correspond to the selected stable node-disjoint paths by the destination. For the selected node-disjoint paths S-C-B-I-D and S-E-F-G-H-D, *D* sends QRREP packets to *S* through the nodes *I* and *H* respectively (as shown in Figure 5.5. A QRREP packet generated by the destination contains full path information along with stability value of the selected path. Nodes *I* and *H* forward the QRREP packets according to the method discussed in *Algorithm* 5.5. After receiving the QRREP packets, node *S* determines S-C-B-I-D as primary (with higher route stability

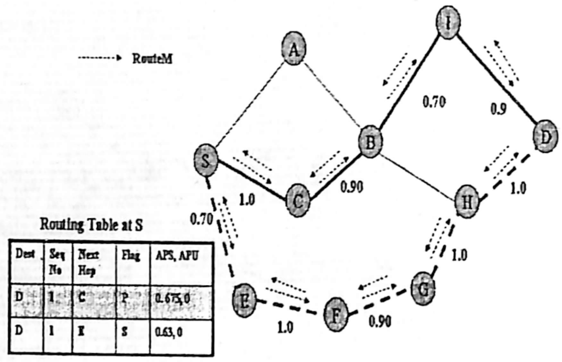


Figure 5.7: An example of Node-disjoint multipath route maintenance (S sends RouteM message through both the paths and receives RouteM with route stability values)

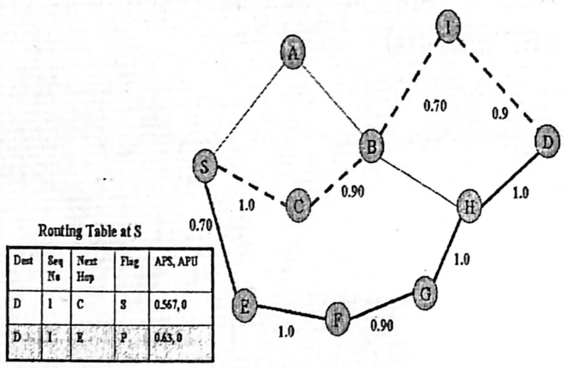


Figure 5.8: An example of Node-disjoint multipath route maintenance (Primary route get changed due to change in route stability values obtained from RouteM messages)

value, shown in solid lines) and S-E-F-G-H-D as the secondary route (shown in dashed lines) and creates routing table entries as shown in Figure 5.6. After setting up the routes, S starts transporting data packets through the selected primary path. During the communication sessions, to keep the secondary path active and to measure the route stability values of both the primary and the secondary paths, S periodically sends the *RouteM* message over both the primary and secondary paths. On receipt of a *RouteM* message through any one of the route, D sends a *RouteM* message back to the source which will measure the stability of the route in its journey to S. In Figure 5.7, due to mobility of the node I on the primary path, the stability values received by S through route maintenance are (APS=0.567, APU=0) and (APS=0.63, APU=0) respectively. This results in switching of the the primary path from S-C-B-I-D to S-E-F-G-H-D as shown in Figure 5.8. Figure 5.9 to Figure 5.13 show the route maintenance due to route failure. A similar procedure applies for route maintenance due to QoS violation, only difference is only destination generates the RERR messages due to QoS violations. In Figure 5.9, a link failure between nodes I and D is detected by node I and hence forwards a RERR message back to the source S. All nodes which forward the RERR packet release resources reserved in the path. Finally, S receives the RERR message and removes the corresponding route entry as shown in Figure 5.10. Because, there is one more active route for the destination D, S does not start a new route discovery. Figure 5.11 shows route failure due to the link break between F

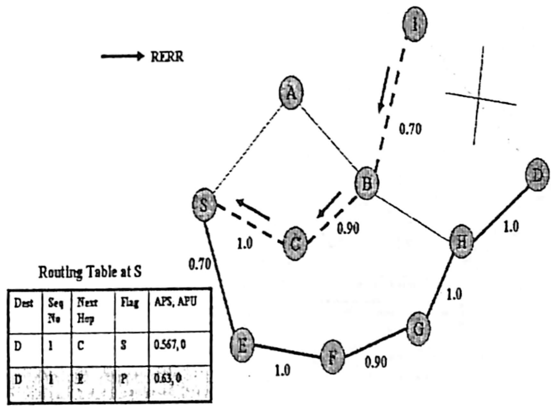


Figure 5.9: An example of Node-disjoint multipath route maintenance (Link failure between I and D in the secondary path is detected by I, so it forwards a RERR packet toward S)

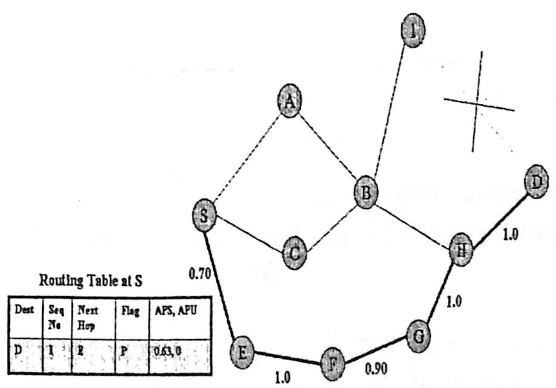


Figure 5.10: An example of Node-disjoint multipath route maintenance (After receiving a RERR packet S removes the corresponding Routing Table entry)

and G detected by F. After receiving the RERR message which is generated and forwarded by F, S removes the only routing table entry for node D (as shown in Figure 5.12) and initiates a new route discovery to compute a new node-disjoint multipath between S and D. Finally, S discover a multipath consisting of only one path and use it as the primary path for data transport as shown in Figure 5.13.

5.6 Simulation and Performance Evaluation

In this section, we have evaluated the QoS performance of SMQR protocol through extensive set of simulations, under various mobility conditions. For comparison purpose, we have considered the node-disjoint version of the AOMDV [71] which is a widely accepted standard for multipath routing in MANETs. We have taken AOMDV [71] for performance comparison with SMQR, because AOMDV [115] is the closest protocol to SMQR as compared with other multipath protocols (both QoS-aware and without QoS) found in the literature.

QoS-aware multipath protocols found in [62, 66] are based on CDMA/TDMA based MAC protocols, and therefore they are excluded for performance comparison with SMQR. Multipath protocol

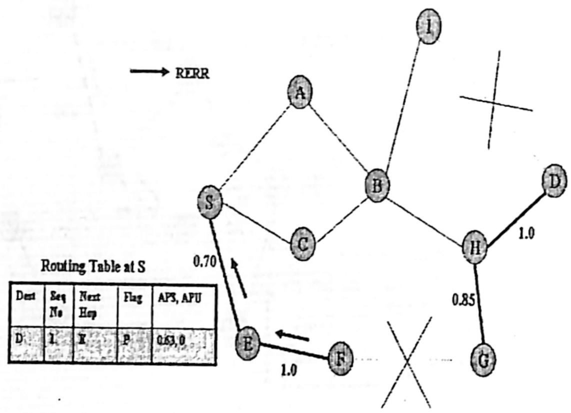


Figure 5.11: An example of Node-disjoint multipath route maintenance (Link failure between F and G in the primary path is detected by F, so it forwards a RERR packet toward S)

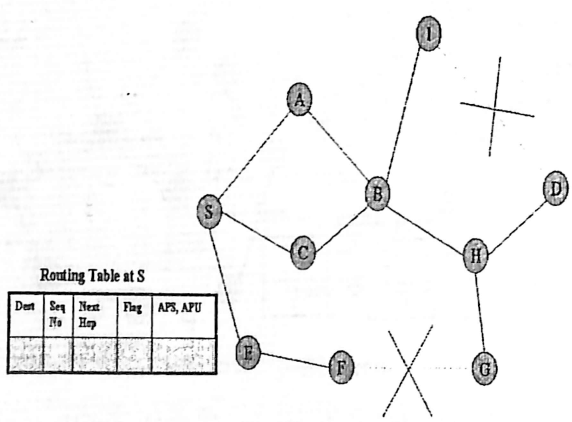


Figure 5.12: An example of Node-disjoint multipath route maintenance (After receiving a RERR packet S removes the corresponding Routing Table entry, so no path between S and D exits)

proposed in MP-DSR [56] considers only reliability of packet delivery as the application's QoS requirement without delay and throughput parameters. Therefore, it is not compatible with SMQR. ADQR [40] is based on simultaneous use of the multiple disjoint paths to support application's throughput requirement and hence is not compatible with SMQR. Similarly, proposals in [59, 109] use DiffServ model for service differentiation and are therefore not compatible with SMQR. Proposal in IMPR [110] considers throughput and link stability for computing multipath without mentioning about the methods used for estimation of bandwidth and path stability of the computed paths. Hence, IMPR specification is incomplete and we have not considered IMPR for performance comparison with SMQR. From the above analysis of existing QoS-aware multipath routing protocols, we have found that none of the existing QoS-aware multipath routing protocols for MANETs are compatible and complete to consider for performance comparison with SMQR. We, therefore, have chosen AOMDV [71] which is one of the most widely accepted, practically implemented best-effort on-demand multipath protocol used in MANETs for performance comparison.

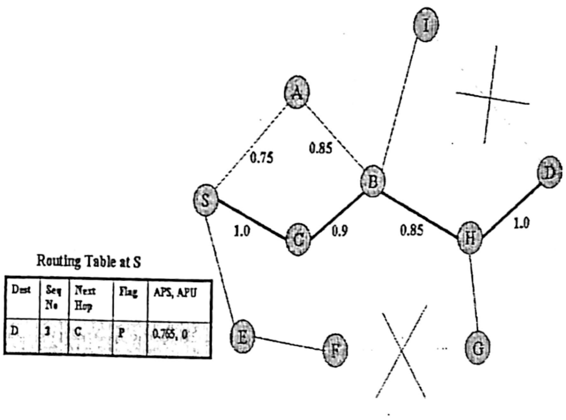


Figure 5.13: An example of Node-disjoint multipath route maintenance (A new route discovery is initiated and a new path formed between S and D)

5.6.1 Performance Metrics and Simulation parameters

Following metrics have been used for performance analysis.

- *Packet Delivery Ratio (PDR):* This is the ratio of number of data packets received at the destinations and the number of data packets actually sent to the network. This measures the quality of the discovered path.

- *Normalized Control Overhead (NCO):* This is the ratio of number of control packets sent to the network hop-wise and the number of data packets delivered at the destinations. This measures the overhead induced by the protocol.

- *End-to-End Delay (E2ED):* This gives the average time delay that a data packet has encountered from the time it was sent by a source to the time it was delivered at the destination. This measures the delay performance of the QoS flows.

- *Maximum Jitter(MJ):* This gives the maximum variations in delay among the delivered data packets at the destinations.

- *Dropped Packets (DP):* This gives the total number of data packets dropped during communication in the network.

To evaluate the performance of SMQR protocol and perform the comparative study with AOMDV [71] under different mobility and network load conditions, an extensive simulation is performed using NS-2[80]. For each type of scenario, we have run the simulation for 10 times (with random scenarios with different seeds) to take average values in the measured performance metrics. We have used two types of network loads to represent moderate load and high load situations. The connection pattern consists of 10 flows (for high load 15 flows) between randomly chosen source-destination pairs among the 50 nodes. The QoS flow taken is CBR with packet size 512 bytes at rate of 10 packets per second. All the generated flows have same QoS requirements. The detailed simulation parameters are summarized in Table 5.2. The parameters for stability model, the period for *RouteM* and the threshold for route switching are heuristically selected through a set of simulation runs.

Table 5.2: Simulation Parameters

Parameter Name	Value
Simulation area	1000 × 500 flat-grid area
Number of nodes	50
Mobility Model	Random Waypoint
MAC layer	IEEE 802.11 DCF
Channel Capacity (Mb/s)	2
Radio Propagation Model	Two Ray Rayleigh Fading
Antenna Type	Omni directional
Transmission Range (m)	200
Carrier Sensing Range (m)	200
Maximum Node Speed (m/s)	0, 2, 5, 10, 15, 20
Pause time (s)	0
Traffic Type	CBR
Packet Size (B)	512
Simulation Time (s)	500
Number of flows	10, 15
Traffic Rate (packets/s)	10
Minimum Throughput (kbps)	40
Maximum Delay (s)	0.1
Thr_1, Thr_2	$1.5 \times RxThr, 1.2 \times RxThr$
u_1, u_2	$0.05 \times RxThr, 0.3 \times RxThr$
T (s), Δt (or RRL) (msec), n	1.0, 100, 3
No. of disjoint paths	≤3
Threshold for route switching, Δs	10% higher in route stability
RouteM sending period (s)	1.0

5.6.2 Results Analysis

From Figure 5.14 and Figure 5.15, we can observe that the packet delivery ratio (PDR) of SMQR improves significantly in comparison with AOMDV protocol. For 10 flows, PDR of SMQR is more than 97% in all mobility cases whereas the PDR of AOMDV reduces up to 81%. Similarly, SMQR shows better performance for 15 flows with minimum PDR at 94.8% at 20 m/s mobility speed where as performance of AOMDV drastically changes to a minimum of 68% at the mobility speed of 20m/s. This is mainly due to (i) SMQR always selects the most stable QoS path out of the selected node-disjoint paths for data transport (ii) it performs admission control before admitting a flow in the network, thereby avoiding congested network paths and (iii) most of the cases primary route is switched before it breaks or becomes unstable due to node movements. These give rise to reduction of data packet loss during data communication session which are shown in Figure 5.16 and Figure5.17 respectively. As can be seen from these figures, the amount of data packet loss in AOMDV for 10 and 15 flows scenarios are more than 2 and 3 times respectively as compared to SMQR. For QoS assurance, maintaining end-to-end delay of the data packets within the specified maximum delay is very important. The average end-to-end delay of data packets include - (a)waiting time in buffer during route discovery due to a route failure or recovery from QoS violation, (b) waiting time in nodes' interface queue and (c) delays at MAC layer during transmission/retransmission. Selection of a QoS aware stable node-disjoint path together with the use of - stability based route switching, admission control mechanism, and delay violation detection and recovery techniques in SMQR significantly reduces and controls the end-to-end delay of the data packets. From Figure 5.20 and Figure 5.21, it can be observed that in all mobility scenarios of 10 and 15 flows, SMQR maintains average end-to-end delay below 22 msec which is less than the required maximum delay of 100 msec. In the case of AOMDV, delay is not bounded and it reaches up to 198 msec and 1 sec for 10 and 15 flows respectively. It is not possible to maintain the maximum delay bound for each and every data packet in a random access MAC protocol based MANET. Though our protocol detects delay violation, it is not instantaneous. So, there will be always some number of packets which violate the delay requirements. Figure 5.22 and Figure 5.23 show the maximum delay jitter for both SMQR and AOMDV in all simulation scenarios. For SMQR, maximum delay jitters are only between 1 to 1.5 seconds and 1 to 2 seconds in all the scenarios of 10 and 15 flows respectively. Significant delay jitters of the order of 7 to 8 seconds and 12 to 19 seconds in the scenarios of 10 and 15 flows respectively can be observed in the case of AOMDV. In our protocol, network control overhead include both control packets generated during route discovery/recovery and due to periodic maintenance of alternate paths through RouteM messages. The amount of control overhead due to route discovery is more compared to the amount of control overhead due to alternate route maintenance, the first one using network wide flooding and the second one using unicast communication. Though the overhead due to alternate route maintenance is not present in AOMDV, but due to its route selection procedure, most of the selected routes fail after a short period of their discovery which lead to frequent route recoveries to be initiated by the source. Due to selection of the highly stable paths and careful selection of the alternate route maintenance period in SMQR, its network control overhead remain lower in all mobility and traffic load scenarios which can be observed in Figure 5.18 and Figure 5.19. Further the increase on network control overhead with mobility is very marginal in SMQR, where there is a sharp increase of NCO in case of AOMDV due to rapid increase of required number of route recoveries. Finally, along with providing the QoS assurance to real-time applications, SMQR improves the utilization of the network resources. This is done by - reducing amount of collisions due to network overload, reducing number of dropped packets by efficient route selection and route switching, and reducing unnecessary control overhead due to route failures. The average network throughput obtained in SMQR are respectively 10% and 12% higher than AOMDV in the two load scenarios as shown in Figure 5.24 and Figure 5.25 respectively.

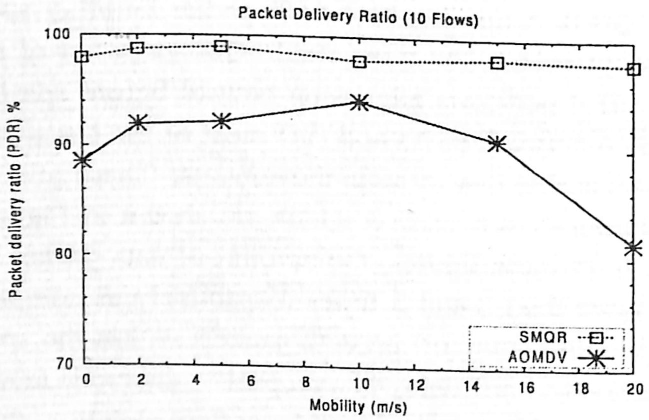


Figure 5.14: Packet Delivery Ratio vs Node Mobility (at Moderate Load)

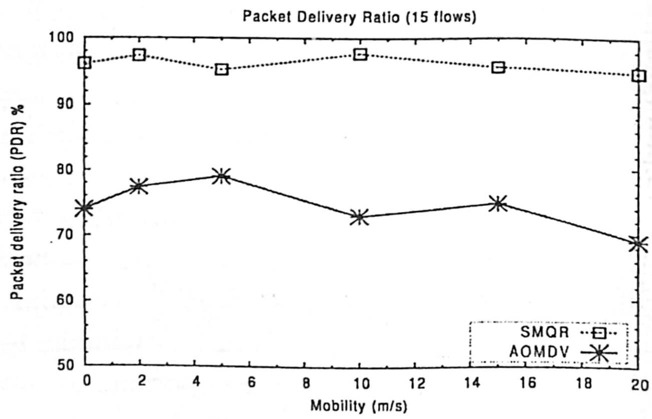


Figure 5.15: Packet Delivery Ratio vs Node Mobility (at High Load)

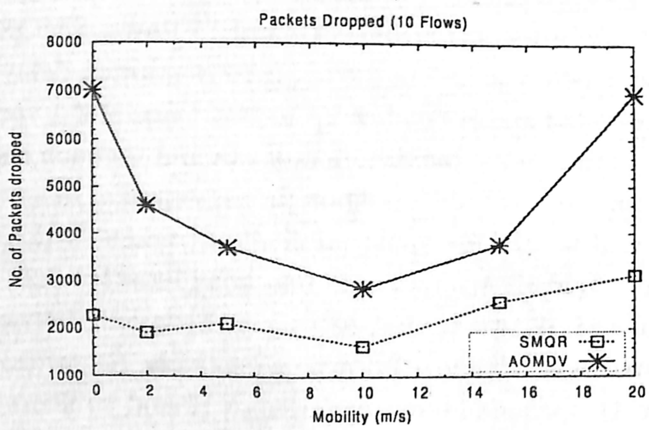


Figure 5.16: Packets drooped vs Node Mobility (at Moderate Load)

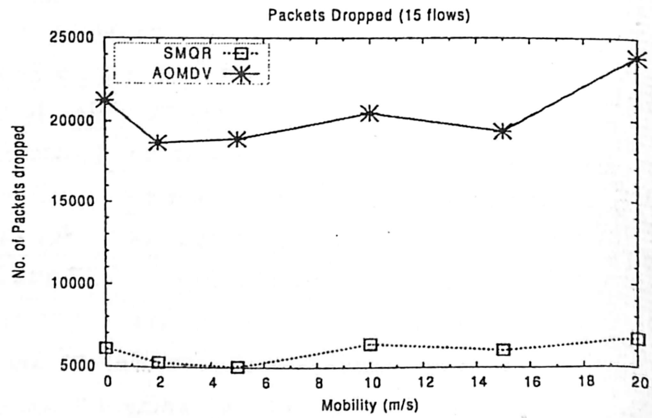


Figure 5.17: Packets dropped vs Node Mobility (at High Load)

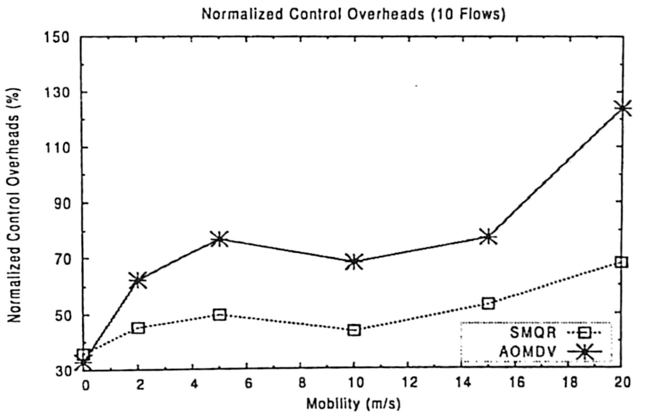


Figure 5.18: Network Control Overhead vs Node Mobility (at Moderate Load)

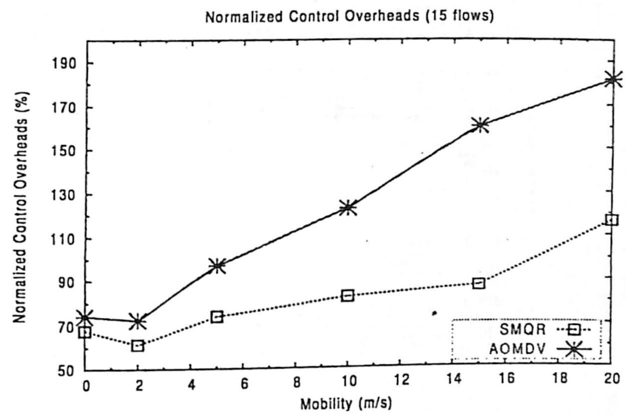


Figure 5.19: Network Control Overhead vs Node Mobility (at High Load)

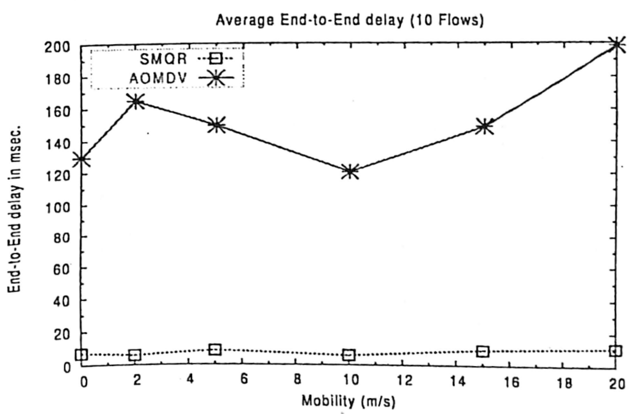


Figure 5.20: End-to-End Delay vs Node Mobility (at Moderate Load)

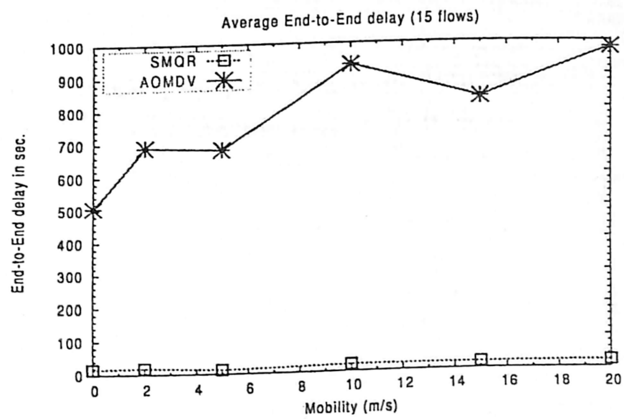


Figure 5.21: End-to-End Delay vs Node Mobility (at High Load)

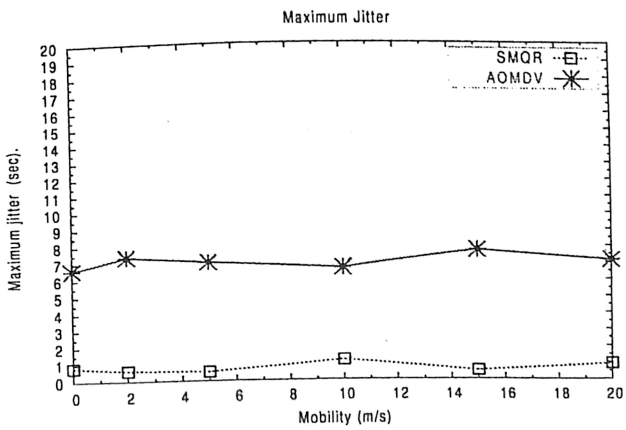


Figure 5.22: Maximum Jitter vs Node Mobility (at Moderate Load)

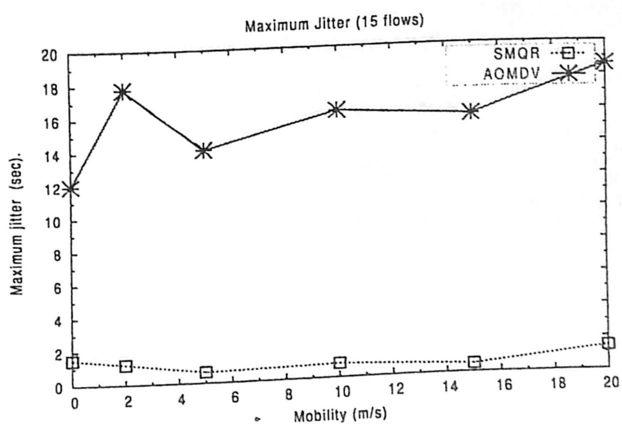


Figure 5.23: Maximum Jitter vs Node Mobility (at High Load)

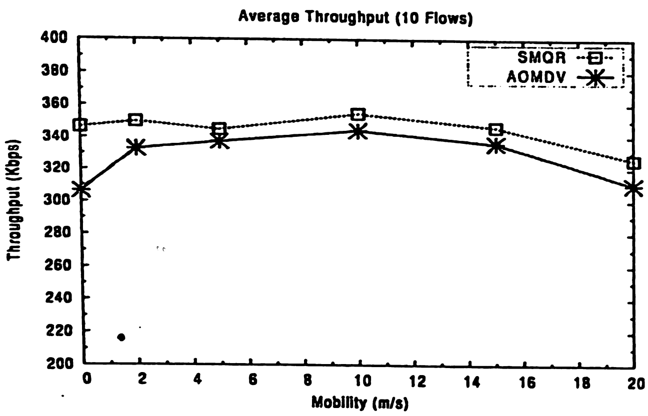


Figure 5.24: Average Throughput vs Node Mobility (at Moderate Load)

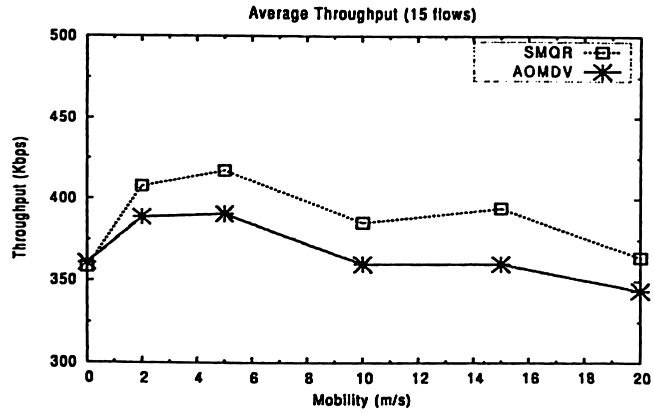


Figure 5.25: Average Throughput vs Node Mobility (at High Load)

From the above analysis, it is evident that SMQR performs significantly better to provide QoS assurance in both high and low mobility scenarios for moderate load and high load conditions. SMQR outperforms node-disjoint version of AOMDV in terms of end-to-end delay, packet delivery ratio, maximum jitter and network control overhead. Further, it is shown that the network resources are utilized efficiently while providing QoS support with improved average throughput.

5.7 Conclusion

In this chapter, we have proposed a route stability based multipath QoS routing protocol for Mobile Ad Hoc Networks which provides QoS assurance to applications in terms of maximum delay and minimum throughput. Design of SMQR is mainly based on use of the following mechanisms which ensures significant reduction in the number of route recoveries during QoS data transmission. These are :-

- a simple route stability model to compute the stability of the paths
- special route request forwarding policies to reduce path diminution problem
- node-disjointness and stability properties to improve the reliability of the discovered paths
- a route maintenance mechanism to maintain alternate paths as valid and
- a proactive path switching mechanism to select the highest stable path for transporting data.

Simulation results show better performance of SMQR in terms of average end-to-end delay, packet delivery ratio, maximum delay jitter, especially in highly mobile scenarios without degrading network throughput. In SMQAR, detection of potential route failures and switching to an available stable route before actual route break, saves route recovery time and reduces possibility of packet loss during active communication.

After designing QoS techniques at MAC layer and Network layer, we have investigated the QoS provisioning in MANETs from a multilayer perspective with mapping of user specified QoS parameters across different layers including application layer, network layer and MAC layer. In the next chapter, we will discuss about the proposed cross-layer QoS framework.

Chapter 6

A Cross-layer QoS framework for Mobile Ad Hoc Networks

6.1 Introduction

Future mobile ad hoc networks are expected to be based on all-IP architecture and should be capable of carrying different real-time multimedia applications such as voice and video flows along with normal data flows. It is necessary for MANETs to have an efficient quality of service (QoS) framework to support the diverse applications.

In this chapter, we have proposed a cross-layer QoS framework for mobile ad hoc networks, CLQM, to support a stronger notion of per class service guarantees in terms of delay and throughput in MANETs. Our framework is based on a cross-layer mapping of QoS parameters across three main protocol layers - Application, Network and MAC layers. MAC layer QoS metric indicates the quality of the link in the network. MAC layer is responsible for scheduling and allocation of the shared wireless channel, which eventually determine the link level QoS parameters namely the available link bandwidth and MAC delay. These link level QoS parameters affects the decision at the network layer for path selection to satisfy end-to-end QoS requirements. The network layer QoS metric indicates the quality of the end-to-end path. The routing protocol at the network layer selects proper links which can together satisfy QoS while relaying packets to the destination. Finally, the application layer QoS metric represents the QoS requirements of the user application. User QoS requirements are specified as - service class with pre-defined aggregated service guarantees in terms of delay and throughput. The user QoS requirements are transformed to application layer QoS metric in terms of - delay, throughput and best-effort. These are aggregated QoS metrics at the application layer. Throughput and delay values for different classes of service are based on their sensitivity to these two parameters. The proposed framework try to maintain an aggregated level of service to different QoS classes by mapping QoS classes to corresponding network layer and MAC layer scheduling. Further, to maintain the per class service guarantees in dynamic environment, both MAC layer and network layer are made adaptive.

The proposed cross-layer QoS framework aims at (i) mapping of QoS requirements across different layers starting from application layer to network layer and finally to link layer, (ii) developing an admission control module in network layer based on application layer QoS metrics and the prevailing network conditions, (iii) developing a strategy to dynamically adapt the behaviour of the link layer, and (iv) detection of network overload situation and reduction of traffic load to ensure better service quality for admitted flows. Overall aim of the framework is to select paths with sufficient resource to provide QoS guarantee for high priority traffic, and dynamically adapt the behaviour of MAC layer and network layer to maintain the QoS assurance to admitted flows based on prevailing network load and service quality achieved in the network for different QoS class. The proposed QoS framework provides per class service differentiation in an end-to-end basis.

The rest of the chapter is organized as follows. In section 6.2, we briefly discuss about the related work. Section 6.3 describes the QoS mapping and computation of QoS parameters across different layers. Our cross-layer QoS framework is discussed in section 6.4. In section 6.5, we discuss implementation aspects of the proposed framework. Finally, section 6.6 gives conclusion.

6.2 Related Work

Integrated Service (IntServ) [11] model is not suitable for mobile ad hoc networks due to high overhead and scalability problem. Differentiated Service (DiffServ) [9, 77] avoids the scalability problem in IntServ, but it is difficult implement service level agreement and to predefine boundary and interior routers in MANETs. Flexible QoS Model for MANETs (FQMM) [113] tries to combine the advantages of both the per-flow service granularity in IntServ [11] and service differentiation in DiffServ [9]. FQMM has the same scalability problem that exists in IntServ and it is difficult to code the PHB (Per Hop Behaviour) in the DS field of IP, if the PHB includes per-flow granularity. Service Differentiation in Stateless Wireless Ad hoc Networks (SWAN) [3] is a stateless network model which uses distributed control algorithms to deliver service differentiation in mobile wireless ad hoc networks. Although SWAN is claimed to be stateless, intermediate nodes have to remember about the traffic flow to distinguish between new and old flows for regulating flows. SWAN does not consider best-effort traffic for calculating available bandwidth and source-based admission control using probing packets is ineffective to cope up with dynamic environment in MANETs. INORA [28] is a QoS support mechanism that makes use of the INSIGNIA [52] in-band signaling and TORA routing protocol [83] for MANETs. INORA represents a QoS-signaling approach with a soft state resource reservation mechanism. The idea is based upon the property of TORA to provide multiple routes between a given source-destination pair. While INSIGNIA [52] does not take any help from the network with regard to redirecting the flow along routes which are able to provide the required QoS guarantees, INORA gives feedback to the routing protocol (TORA) on a per-hop basis to direct the flow along the route that is able to satisfy the QoS requirements of the flow. INORA is better than INSIGNIA in that it can search multiple paths with lesser QoS guarantees. Since data packets have to be transmitted as best-effort packets in case of admission control failure at the intermediate nodes, this model may not be able to provide QoS guarantees for real-time applications. Due to dependability on underlying network characteristics to provide QoS, a cross-layer approach to support QoS in MANETs was first proposed in [78]. Their model considers three layers namely Application layer, Network layer and MAC layer. QoS support for three different classes of application layer QoS requirements are supported by corresponding throughput and delay characteristics of the network. Their model lacks detail mapping across the protocol layers and no adaptive techniques are used for QoS in the MANETs. QPART [116] is a cross-layer framework for QoS support in MANETs. Although it incorporates mechanisms for network layer and MAC layer adaptations, it has similar scalability problem that exists in IntServ due to maintenance of per flow state information in the nodes. QPART overloads network layer with a lot of functionalities, including MAC layer adaptation. A cross-layer scheme by Chen et al. [18] is based on using a middleware to implement a data accessibility service at application layer. The middleware is mainly responsible to maintain information about the node and translating applications request to network layer metrics. A cross-layer framework for real-time video streaming by using adaptation across all layers of the protocol stack is proposed in [95]. It uses technique to jointly optimize error-resilient source coding, packet scheduling, stream-based routing, link capacity assignment and adaptive link layer techniques. Another approach by Qu et al. [88] proposed a cross-layer rate control based on prevailing network condition like interference level, number of hops to destination, and packet loss rate. Further, a cross-layer joint source-channel coding is used in conjunction with rate control to provide error-resilient video delivery over ad hoc networks. It was found that optimizing across multiple layers become very complex problem due to possibility of forming adaptation loops [46].

6.3 QoS Mapping

In wireless networks, especially in mobile ad hoc networks, layers can not function efficiently in isolation to provide complex network services like QoS. In order to guarantee the fulfillment of application's QoS requirements, a mapping of QoS parameters onto the involved protocol layers has to be performed. The protocol layers adapt their behaviour according to the underlying network conditions and the user requirements to provide efficient QoS support in the networks. Our framework mainly considers three layers of the protocol stack, namely - Application layer, Network layer and Link layer for QoS mapping. Firstly, based on the user QoS requirements, we have classified user traffic into four QoS service classes - class I, class II, class III and class IV. Where class I, class II, class III and class IV corresponds to delay sensitive, delay and throughput sensitive, throughput sensitive and best-effort service respectively. Here, we have assumed that the maximum delay constraint for traffic of class I category is smaller as compared to maximum delay constraint for traffic in class II category. Similarly, the minimum throughput requirement for class II is larger than the minimum throughput requirement in class III. The throughput requirement for class I category is smaller and constant, therefore we do not explicitly specify its throughput requirement. Based on the QoS class, application layer metrics are classified as - delay (maximum end-to-end delay), throughput (minimum throughput) and best-effort. Although, delay jitter is an important application layer metric for most of the real-time applications, here we have not considered it, because ensuring delay jitter through per-class based service provisioning is a very complex problem, especially in resource constrained and highly dynamic networks like MANETs. Application layer metrics are mapped to corresponding path metrics in network layer and finally Network layer metrics are mapped to relevant link layer metrics in the MAC layer. The network layer and MAC layer metrics are dependent on the quality of the underlying network which is determined from the current network load and network dynamics. Based on the quality of the underlying network in terms of network layer metric, admission control is performed at every node during route discovery between source and destination. It is very difficult to ensure QoS guarantee using admission control due to dynamic nature of MANETs. Although some level of throughput guarantee can be assured by admission control based on measurement of available channel bandwidth, but similar approaches based on channel delay measurements to provide delay guarantees is less effective. This is because the delay that will be experienced by a flow after the flow is admitted will be different from the current delay measured from the network. Further, node mobility and dynamic channel conditions make the measurement based admission control protocols to provide both delay and throughput guarantees very difficult. This may lead to wrong admission control decision at a node and hence affect the service quality of already admitted flows. Therefore, in our QoS framework, both MAC layer and network layer are designed to adapt their behaviour based on the dynamic network condition (which can be obtained through continuous monitoring) and service quality requirements of the admitted traffic. The overall QoS mapping at different layers are shown in Table 6.1. Following subsections discuss the QoS metrics at different layers of the protocol stack.

Table 6.1: Mapping of QoS Metrics across different protocol layers

QoS Class	Example Applications	Application layer QoS	Network layer QoS	MAC layer QoS
Class I	Voice over IP (VOIP)	Delay (Dmax) Throughput (Constant)	Path Delay, Path Stability	MAC delay, Link stability
Class II	Video conferencing, Video Telephony	Delay (Dmax), Throughput (Bmin)	Path Delay, Path Bandwidth, Buffer level, Path Stability	MAC delay, Link Bandwidth, Link stability
Class III	Multimedia Streaming, Video On-demand	Throughput (Bmin)	Path Bandwidth, Buffer level, Path Stability	Link Bandwidth, Link stability
Class IV	Email, FTP etc	Best-effort	Hop count, Path stability	Link stability

Table 6.2: Values for contention related parameters

QoS Class	CWmin (default)	CWmax (default)	CWmax (max)
Class I	7	15	63
Class II	15	31	127
Class III	31	63	255
Class IV	63	1023	1023

6.3.1 MAC QoS metrics

The link layer QoS metrics that we have considered are MAC delay, Link bandwidth and Link stability which represents the quality and the capacity of the underlying link. These metrics have direct impact on the QoS supported by the network. These metrics are *periodically measured* to keep the link QoS state up to date. In the proposed Cross-layer framework (as shown in Figure 6.1), the modules

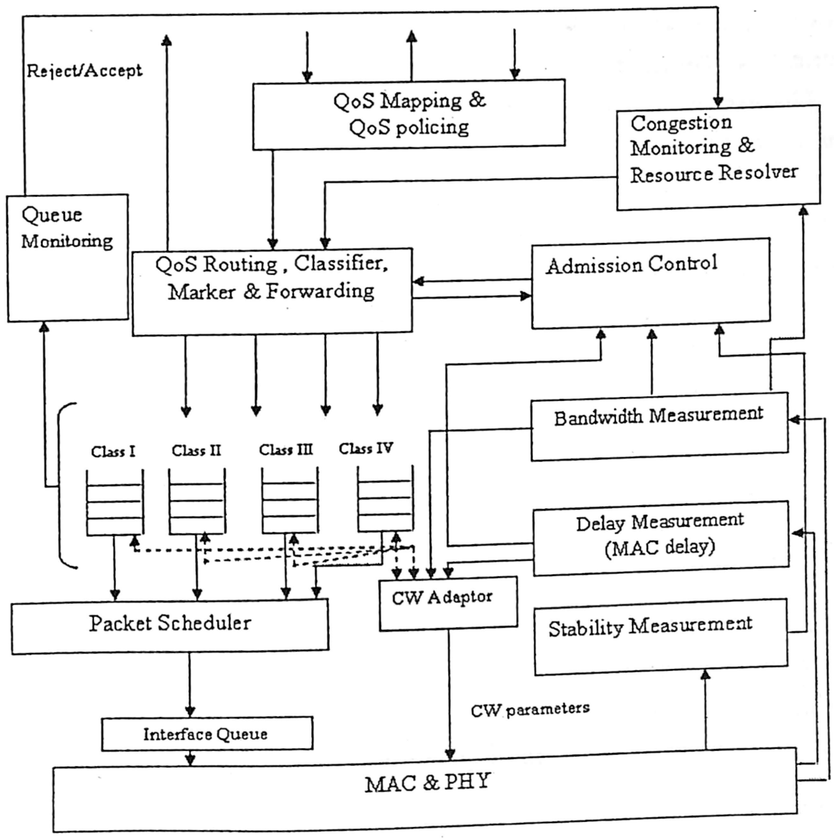


Figure 6.1: The proposed Cross layer QoS Framework for MANETs

- *Bandwidth Measurement*, *Delay Measurement* and *Stability Measurement* perform the link QoS measurement periodically. The link level QoS metrics are used by the routing protocol for admission control purpose. For a given network environment, MAC layer QoS metrics for a node are mainly determined from the MAC protocol and the underlying PHY layer. Therefore, in an IEEE 802.11 DCF based ad hoc network, to some extend especially when the network is under saturation condition, the link layer QoS metrics (e.g. Link bandwidth and MAC delay) can be adapted by adjusting MAC layer parameters like - $CWmin$ (Minimum Contention Window), $CWmax$ (Maximum Contention Window), maximum frame size, retry limit and backoff process. Previous studies have shown that MAC layer adaptation using CW related parameters give the most predictable results, although the other parameters can be used for achieving static service differentiation. In our framework, we have fixed the frame size, retry limit and backoff process, but allow the dynamic adaptation of Contention Window (CW) range, that is, $CWmin$ and $CWmax$ per traffic class based on dynamic network conditions and QoS requirement of the applications. This MAC layer control parameter is used to adapt the MAC behaviour according to the QoS requirements of the applications and underlying network load. Since in contention based MAC protocol like IEEE 802.11 DCF, during contention for the channel, the node with smallest backoff time always wins, the backoff process provides a distributed method to differentiate the service a node receives. Because, our aim is to provide per class based differentiated service, a static differentiation of contention window according to service requirements is proposed as shown in Table 6.2. The default values for $CWmin$ and $CWmax$ for each class shows a static differentiation of service (as provided by IEEE 802.11e EDCA [22, 69]). This type of fixed assignment of CW ranges can not ensure effective QoS in dynamic and changing network load conditions. Therefore, a contention window adaptation mechanism is used to dynamically adjust contention window based on periodic measurement of service level achieved by the different class of traffic. This adaptation works in situations where the total resource requirement of the admitted flows is below the network capacity. Contention window adaptation in case of overloaded situation will result in convergence to smaller contention window sizes for all higher priority traffic, which may result in higher number of collisions and hence throughput degradations. In the cross-layer framework, we have used a module called CW Adaptor to dynamically adapt the $CWmin$, $CWmax$ values for each of the different class of traffic. Further to keep delay and throughput for different QoS classes under control, a maximum limit for $CWmax$ is assigned per class based on their relative priorities (as shown in Table 6.2).

Estimation of available bandwidth at a node

Available link bandwidth at a node can only be determined by considering interference at neighboring nodes. Because, wireless channel is shared among nodes that may not be able to communicate directly, nodes need to passively determine the *network utilization* (U) to compute *available bandwidth* ($Bavail$), given the raw available bandwidth (B) in the network [87]. We can estimate $Bavail$ as shown below (Equation (6.1)).

$$Bavail = (1 - U) \times B \tag{6.1}$$

Though there are a number of metrics available to measure network utilization, we have used *channel busy time* due to the inherent problems in using other metrics as mentioned in [14]. We have defined the channel busy time, T_{busy}, as the total time within an interval, T_{total}, that a node is sending packets, receiving packets or sensing packet transmissions. We have chosen a suitable time interval for T_{total} to measure the channel busy time, so that it is large enough to make an accurate estimate and small enough to quickly adapt to changing network and traffic conditions. To further improve the estimated value, we have used an equally weighted sliding window to continuously measure the channel utilization. Therefore, using T_{busy} and T_{total}, we can estimate U as $\frac{T_{busy}}{T_{total}}$. In CSMA/CA based wireless networks, carrier sensing enables nodes to detect three states; transmitting, receiving and busy. If a node detects a carrier signal, it senses the channel as busy, but it can decode the packet if it is within the Reception Range (RxR) of the sender. By measuring the amount of time the channel is sensed busy (CS), sending (TX) or receiving (RX), a node can measure not only transmissions that occur within its RxR, but also to a range decided by the carrier sensing value of the receiver.

In MANETs, available bandwidth in a node is affected by transmission of nodes beyond its transmission range (e.g., hidden terminals). Therefore, to measure effective available bandwidth of a node, the sensing range should be adjusted in such a way that a transmission by the node will not interfere with any other potential sender in the neighborhood of that node. We have defined the minimum distance between two senders such that their packet transmissions do not affect the receivers in decoding their respective packets as the *Sender Interference Distance* (SID). From [14], we know that for IEEE 802.11 DCF, it is equal to $2 * RxR + RID$. Here, *Receiver Interference Distance*, RID, is the minimum separation between a receiver and another sender, such that the receiver's ability to successfully decode a packet from its sender is not affected. So, if a sender node can ensure that sufficient amount of bandwidth is available inside the area of radius SID, then initiating a data transmission will not result in congestion in the network. Based on this observation, channel sensing threshold is adjusted according to a given RxR, RID, *Transmission power* and *propagation model*. Here, we have assumed that no capture capability in the hardware interface of IEEE 802.11 PHY layer. Figure 6.2 shows the SID for two sender nodes with the help of an example.

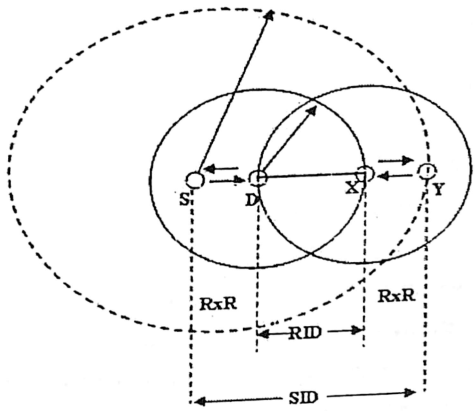


Figure 6.2: Example showing an area represented by the dashed line circle beyond which transmission by node S will not affect any potential transmitter Y

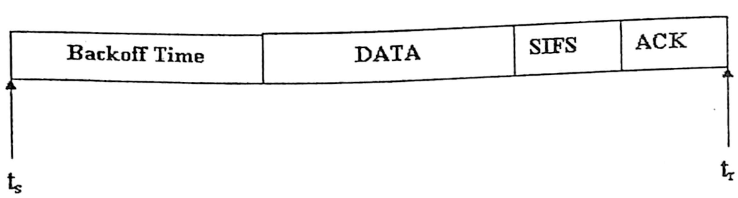


Figure 6.3: MAC delay (for basic access mechanism in IEEE 802.11 DCF) is measured as difference between t_r and t_s

Estimation of MAC delay

The measurement technique for MAC delay is very simple. A node computes the MAC delay (d) by subtracting the time (t_s) that a packet is passed to the MAC layer from the time (t_r) an ACK packet is received from the receiver for the same packet (as depicted in Figure 6.3). Because, the MAC delay depends on the size of the packet, so a standard sized packet should be used for measuring the MAC delay or measured delay can be normalized to a predefined fixed packet size. In order to minimize the biasness against transient changes in delay for a particular class of service, an *Exponentially Weighted Moving Average (EWMA)* estimator is used to measure the average of MAC delay (d^i_{avg}) for a service class as shown in Equation (6.2). Here d^i and d^{i-1}_{avg} are measured MAC delay and previously stored average MAC delay for a service class. η is a positive constant which determines how much effect the previously stored average MAC delay have on the current average MAC delay.

$$d^i_{avg} = (\eta - 1) \times d^i + \eta \times d^{i-1}_{avg} \tag{6.2}$$

To measure the link stability (s) for link to each neighbor of a node, we have used the method discussed in section 4.4 of chapter 4.

6.3.2 Network layer QoS metrics

The QoS metrics of an end-to-end path is also called network layer QoS metrics which depends on the QoS metrics of links on the computed path. Therefore, the network layer QoS metric is based on the quality of the selected path. There are basically two controlling parameters which determine the QoS metric of an end-to-end path. One is the routing protocol which decides what are the links on the computed path and the other is the admission control which decides whether a given link (therefore a given path) should be selected or not. In the path computation phase, the link layer metrics are accommodated and propagated through the intermediate nodes of the generated paths. Additive, concave and multiplicative functions are applied on QoS metric of individual links to represent the network layer metrics corresponding to an already explored path. During the route exploration, admission control is performed to avoid path which does not satisfy the QoS requirements. Suppose, P is a path between source s and destination d. Therefore, the estimation of the network layer QoS metrics for P is computed from the link layer QoS metrics as follows.

$$\text{Path Bandwidth}, B = \min\{b | b \text{ is the residual bandwidth of } e \in P\} \tag{6.3}$$

$$\text{Path Delay}, D = \sum_{e \in P} d, \quad \text{where } d \text{ is the MAC delay of } e \tag{6.4}$$

$$\text{Path Stability}, S = \prod_{e \in P} s, \quad \text{where } s \text{ is the link stability of } e \tag{6.5}$$

$$\text{Hop Count}, H = \sum_{e \in P} 1 \tag{6.6}$$

$$\tag{6.7}$$

At network layer, admitted flows are classified into different service classes according to their QoS class and hence network layer QoS metrics; whose packets are classified and treated differently at routing layer. Classified packets are buffered in separate queues based on their QoS requirements. Based on the required end-to-end QoS metrics in terms of throughput, delay, stability and hop count, our route stability based QoS routing protocol, namely RSQR can be adapted to compute an end-to-end QoS path for the requested user QoS service.

6.3.3 Application layer QoS metrics

Application layer QoS metrics are specified in terms of maximum end-to-end delay (D_{max}), minimum throughput (B_{min}) and best-effort (without any QoS). These application layer QoS metrics represent the class of service requested by the applications. In CLQM, transport layer does nothing on QoS mapping, therefore it simply pass on the QoS metrics to the network layer for finding an end-to-end path with desired quality. Therefore, the application layer QoS metrics are mapped to network layer QoS metric of an end-to-end path. Network layer uses the the application layer QoS metrics to perform admission control at each intermediate node of a path to ensure the QoS provisioning by the selected path. Besides using application layer QoS metrics namely delay and throughput, routing protocol uses path stability to select QoS path with longer life. Network layer metric Hop count is used for finding paths for best-effort service to optimize the use of network resources.

6.4 Cross-layer QoS Framework for MANETs (CLQM)

To implement the necessary cross-layer mapping for providing QoS support, we have proposed a cross-layer framework as depicted in Figure 6.1.

In CLQM, application layer, after receiving a connection request (with a specified service) from an user application, invokes transport layer service for initiation of a transport session which can provide the user requested service. The transport layer simply pass the QoS metrics that it receives from the upper layer to network layer for finding a route with desired QoS. These QoS metrics help the routing protocol to find an end-to-end path which can meet the needs of the flow. Accordingly, the routing protocol at the source node initiates a route discovery to search the best path which can satisfy the requested QoS. The route request packet of the routing protocol carry QoS related fields to enable the intermediate nodes to perform admission control while forwarding route request packets. Finally, the destination selects the best path according to the class of service requested by the source and sends a reply back to the source. Once source obtains a path with desired network layer QoS, the flow is admitted and data packets belonging to the flow can be sent using the selected path.

The service quality achieved in the network is finally determined by per-hop behaviour of differentiated service and the behaviour of the underlying MAC layer protocol. To achieve desired QoS within a service class, the QoS metrics at application layer should finally be mapped to link layer metrics at the MAC layer. In wireless media using random access MAC such as IEEE 802.11 DCF, service quality achieved by an application at the MAC layer is totally determined by the channel access parameters, mainly the contention window size. Therefore, contention window size in all nodes in the path from source to destination should be properly set to meet the application's demand. Due to variations in network load and topology, the mapping of QoS requirements to link layer metric needs to be adaptive. The link layer adaptation does not work in network overload situations, which may results in service quality degradation of existing flows. To maintain the QoS guarantees for the admitted flows, the network layer performs the QoS monitoring by periodically measuring the service level achieved by the QoS classes. Based on the estimated load and the service quality achieved by a class, the network layer of a node performs the contention window adaptation for the underlying MAC layer, or triggers the generation of Explicit Congestion Notification (ECN) to terminate or reroute some existing flows.

The proposed framework consists of the following main components as discussed below.

6.4.1 Admission control

Admission control is an important part in any QoS framework, especially for dynamic network like MANETs. Admission control prevents the network from reaching congested state by rejecting new flows if sufficient resources are not available. Providing admission control in wireless ad hoc network is particularly challenging because of the shared and open nature of wireless communications. Admission control depends on precise estimates of bandwidth available in the network and the bandwidth required by a new flow. The available link bandwidth, $Bavail$, of a node can be estimated by passive monitoring of the channel as discussed in section 6.3.1. The estimated bandwidth includes the overheads of lower layer protocols, which mean actual bandwidth available to user flow at the application layer will be somewhat lower than the estimated bandwidth. Actual channel capacity requirement of a data flow depends on both the protocol overhead at lower layers and the intra-flow contention at intermediate nodes of the selected route. The IEEE 802.11 DCF's overhead contribution (F) per data packet can be computed by the formula given in Equation (6.8) and Equation (6.9). Where Equation (6.8) is for channel access method using RTS/CTS and Equation (6.9) is for basic access method.

$$F = \frac{(L_{DIFS} + L_{RTS} + L_{CTS} + L_{DATA} + L_{MH} + L_{IPH} + L_{ACK} + 3 \times L_{SIFS} + L_{bo})}{L_{DATA}} \tag{6.8}$$

$$= \frac{(L_{DIFS} + L_{DATA} + L_{MH} + L_{IPH} + L_{ACK} + L_{SIFS} + L_{bo})}{L_{DATA}} \tag{6.9}$$

In the above formula F represents the weight factor that must be applied to a flow's throughput requirement, $Bmin$, to get the actual channel bandwidth requirement $B_{channel}$, that is,$B_{channel} = F \times Bmin$. In the above equations L_{RTS}, L_{CTS}, L_{DATA} and L_{ACK} represent the length of RTS, CTS, DATA and ACK frames respectively. Similarly, L_{DIFS}, L_{SIFS} and L_{bo} represent the number of bytes that can be transmitted during DIFS, SIFS and Backoff intervals respectively. L_{MH} and L_{IPH} represent length of MAC and IP headers respectively. In the situation, where admission control mechanism will avoid overloaded situations in the network, the number of collisions will be very less so backoff period can be taken as a fixed value with $(CWmin/2) * slottime$.

Due to self-interference of flows [93], the required bandwidth of a flow varies for each link of a route. A wireless transmission contends with transmissions in the nodes inside the carrier sensing range (CSR) of the sender. Therefore, the bandwidth requirement at a node in the route is determined by the contention count (CC) of the node. The computation of the contention count require either transmission of some high power signals or exchange of control packets among carrier sensing neighbors as discussed in CACP [117], these have drawbacks [94]. In [94], two methods were discussed to compute the contention count without using an extra high power transmission, but both the methods have the drawback of getting incorrect values for CC in some situations and increase the control overhead in the network. Further, none of the methods [94, 117] can support full admission control at the time of processing a RREQ message of a reactive routing protocol like AODV. To enable the admission control at the time of processing a RREQ by a node, here we have assumed that the carrier sensing range is same as the transmission range. Keeping this in mind, we can now calculate the per hop bandwidth requirements in a route knowing the minimum bandwidth requirement for the admitted flow and available bandwidth covering SID of a node.

We have defined a threshold called Channel Sensing Threshold (CHS Threshold) which is equal to the signal strength required to detect carrier at a distance of SID (SID is defined in section 6.3.1) from a node. For bandwidth admission control, a simple algorithm as discussed in Chapter 4 (Algorithm 4.1 in page 99) is applied. Besides, performing admission control for flows requesting minimum throughput in class II and class III, bandwidth admission control is very important in class I also. This is important to guarantee the delay assurance as well as the small constant throughput in class I. Therefore, a bandwidth admission control is performed based on a specific constant amount of throughput requirement for the class I flows. To control network overload condition, two thresholds for available bandwidths B_{resv} and B_m ($B_{resv} > B_m$) are maintained by each node. These two thresholds for available bandwidth indirectly helps in detecting probable network overload situations. Because, the amount of available bandwidth estimated by a node in MAC layer represents the channel utilization, and if channel utilization goes beyond a level, there is an indication of network overload. When a node finds its available bandwidth to go below B_{resv} (reserved bandwidth) it will not admit any new best-effort flow. similarly, if the available bandwidth goes below B_m (minimum reserved bandwidth), then no new flows (irrespective of its class) are admitted by the node. These two threshold are to be decided based on empirical study. Similarly, admission control due to delay and signal strength are performed as discussed in RSQR.

6.4.2 QoS Routing

We have proposed to modify the basic AODV protocol so that the route request (RREQ) and route reply (RREP) packets carry the network layer QoS metrics, both user requirements and accommodated values, to perform admission control at each intermediate node before forwarding RREQ packets. Any RREQ packet that can not pass the admission control criterion is simply dropped. Finally, out of the RREQ packets which reach the destination node and satisfy admission control criterion's are considered to find the best path which optimizes the QoS parameters of our interest. The selection procedure for finding the best QoS path is based on the service class requested by the applications. For the best effort service, the path with maximum value for Stability or with minimum Hop count is used. For other classes, the path with maximum value for stability should be used. Though best-effort service has no specific QoS requirements, during route discovery admission control is performed at each intermediate node to check if the current channel bandwidth observed in the node is more than an admission threshold (B_{resv}) or not. In the case, channel bandwidth is less than the admission threshold the RREQ packet for best-effort flow is dropped. This is required to reduce the possibilities of affecting existing real-time flows in the network.

6.4.3 Packet Classifier and Marker

After admitting a flow, the network layer mark the routing table with the service class requested by the flow. After that, any data packet belonging to the flow are marked with corresponding code in DSCP field of the IP packet. After marking, the data packet is placed in the queue based on its service class. These functions are similar to the functions performed by egress router in DiffServ Model. Any intermediate node, receiving a data packet for forwarding will check the DSCP code and store the packet using a class-based priority queue.

6.4.4 Packet Scheduler

Because, in the network four different classes of traffics are allowed, there should be some scheduling scheme to ensure the service level guarantees for each of the different classes of traffic. We have used a class-based weighted fair queuing (CBWFQ) scheduling for serving packets belonging to different

QoS classes as well as serving the network control packets. There are four queues maintained in each of the node. Due to their urgency in delivery, network control packets are enqueued in the same queue along with Class-I traffic. The classifier in the source node and the intermediate nodes classify the packets that need to be forwarded based on their DSCP fields and put them in the relevant queues. After packets are assigned to their respective queues, they receive differentiated service based on priority of the queue, which corresponds to the weights set for each queue. By defining a set of weights, a particular scheduling can be achieved. This gives service differentiation through intra node scheduling. One scheduling scheme can be based on dividing the available bandwidth among the different service classes in some proportion. Another scheduling scheme may be to serve the highest priority class first and remaining bandwidth can be used in a number of ways. In the framework, each node of a network must follows the same scheduling policy.

6.4.5 Contention Window Adaptor

The contention window parameters namely $CWmin$ and $CWmax$ provides inter node service differentiation among different class of traffic. The different service classes are assigned a non-overlapping ranges of contention window in default setting. This provides a proportional service differentiation in terms of delay and throughput (assuming frame sizes are same for all classes of traffic) [1]. This assured service differentiation will be lost due to priority inversion in higher load conditions as we discussed in section 3.4 of Chapter 3. These default contention window ranges are adjusted by the CW Adaptor based on dynamic network conditions and required service level of a QoS class. To ensure that the admitted QoS flows can receive the desired level of service quality, if enough network resources are available, a network load and QoS monitoring module is implemented in network layer of each node. Network load is monitored by using the MAC layer information such as available link bandwidth and MAC layer delay per class. Since, we have used per-class service differentiation, all packets belonging to a service class are put in a common queue and only one contention window range is kept for that class. Periodic updates of contention window range are performed for all the classes depending on the current service quality and network load conditions. To provide service differentiation across different classes of traffic, the non-overlapping contention window ranges are maintained while performing contention window adaptation. Therefore, while adjusting the CW range for lower priority class, an extra checking is performed such that the newly computed CW range does not overlap with higher priority QoS class. Network layer maintain a table called CWT (Contention Window Table), where one entry per QoS class is used to store the current $CWmin$ and $CWmax$ values of the QoS class. These entries are periodically updated through the CW adaptation runs in the network layer. MAC layer uses the corresponding entry in CWT, as a cross-layer information, while sending data frame of a QoS class. For convenience, we have used the notations $CWmin[I]$ and $CWmax[I]$ to represent $CWmin$ and $CWmax$ values for QoS class I. Similarly, $CWmin^{default}[I]$, $CWmax^{default}[I]$ and $CWmin^{max}[I]$ are used to represent CWmin (default), CWmax (default) and CWmax (max) values for class I of Table 6.2. The Contention window adaptation rules for different service classes are given below.

Class - I (Delay sensitive service):

This class of traffic corresponds to Guaranteed Service (GS) of IntServ model. Traffic in this class is generally CBR with very small peak rate with stringent delay requirement. We have assumed that each node ensure a maximum delay of D_I for traffic in class I and try to maintain it by periodically monitoring the observed delay in that class and adjusting the CW range accordingly. In this case, based on the per hop delay requirements of the class I, given as D_I, and the current MAC delay, D, measured in the node for Class I traffic (obtained from MAC layer), $CWmin[I]$ is adjusted as shown in Algorithm 6.1. In the algorithm, $CWmin^{prev}[I]$ represents the value of $CWmin$ for class I before applying CW adaptation, and $CWmin[I]$ is the new value of $CWmin$ obtained by applying the dynamic adaptation of contention window due to the difference between expected maximum delay, D_I, and the current measured delay, D, for class I. To avoid the possibility of unnecessary fluctuation in setting of $CWmin$, a threshold value namely *Contention Window Threshold*, $CWthresh_I$, is used in the algorithm. Here, α_1 is a small positive constant that should be selected appropriately to enable faster adaptation of $CWmin[I]$ to prevailing network condition.

Algorithm 6.1 Procedure for Contention Window Adjustment of Class I

$CWmin[I] := CWmin^{prev}[I] \times (1 - \alpha_1 \times \frac{(D-D_I)}{D_I})$ {* Here $CWmin[I]$ and $CWmin^{prev}[I]$ represent current and previous values of CWmin for class I *}
if $(abs(CWmin[I] - CWmin^{prev}[I]) < CWthresh_I)$ **then**
  $CWmin[I] := CWmin^{prev}[I]$
  return
end if
if $(CWmin[I] < CWmin^{prev}[I])$ **then**
  $CWmin[I] := max\{CWmin[I], CWmin^{default}[I]\}$
  $PF := \frac{CWmin[I]}{CWmin^{prev}[I]}$
  $CWmax[I] := max\{CWmax[I] \times PF, CWmax^{default}[I]\}$
else
  if $(CWmin[I] > CWmin\ prev[I])$ **then**
    $CWmin[I] := min\{CWmin[I], CWmax^{max}[I]/2\}$
    $PF := \frac{CWmin[I]}{CWmin^{prev}[I]}$
    $CWmax[I] := min\{CWmax[I] \times PF, CWmax^{max}[I]\}$
  end if
end if

Class - III (Throughput sensitive service):

This class of traffic corresponds to Assured Forwarding (AF) of DiffServ model. Traffic in this class has minimum throughput requirements but without stringent delay guarantees. In this case, based on difference between the current buffer occupancy, Q, and a threshold buffer level, Q_{III}, for class III (determined according to RED [30] queue management), $CWmin[III]$ is adjusted as shown in Algorithm 6.2. Here, we can detect throughput violation for a QoS class indirectly through the corresponding class's queue occupancy. In Algorithm 6.2, β_3 is a small positive constant which depends on value of Q_{III} and should be selected appropriately to enable proper adaptation of

$CWmin[III]$ to prevailing network condition. $CWthresh_{III}$ is a threshold to avoid unnecessary CW adaptation.

Algorithm 6.2 Procedure for Contention Window Adjustment of Class III

$CWmin[III] := CWmin^{prev}[III] - \beta_3 \times (Q - Q_{III})$ {* Here $CWmin[III]$ and $CWmin^{prev}[III]$ represent current and previous values of CWmin for class III *}
if $(abs(CWmin[III] - CWmin^{prev}[III]) < CWthresh_{III})$ **then**
  $CWmin[III] := CWmin^{prev}[III]$
  return
end if
if $(CWmin[III] < CWmin^{prev}[III])$ **then**
  $CWmin[III] := max\{CWmin[III], CWmin^{default}[III]\}$
  $PF := \frac{CWmin[III]}{CWmin^{prev}[III]}$
  $CWmax[III] := max\{CWmax[III] \times PF, CWmax^{default}[III]\}$
else
  if $(CWmin[III] > CWmin^{prev}[III])$ **then**
    $CWmin[III] := min\{CWmin[III], CWmax^{max}[III]/2\}$
    $PF := \frac{CWmin[III]}{CWmin^{prev}[III]}$
    $CWmax[III] := min\{CWmax[III] \times PF, CWmax^{max}[III]\}$
  end if
end if
if $(NAV[I] \neq Null)$ **then**
  $CWmin[III] := max\{CWmin[III], CWmax[I]\}$
  $CWmax[III] := max\{CWmax[III], CWmin[III] \times 2\}$
end if
if $(NAV[II] \neq Null)$ **then**
  $CWmin[III] := max\{CWmin[III], CWmax[II]\}$
  $CWmax[III] := max\{CWmax[III], CWmin[III] \times 2\}$
end if

Class - II (Delay and Throughput sensitive service):

This class of traffic corresponds to Controlled Load (CL) and Expedited Forwarding (EF) of IntServ and DiffServ models respectively. Traffic in this class has both minimum throughput and maximum delay requirements. In this class, based on difference between the current buffer occupancy, Q, and the threshold buffer level, Q_{II} (estimated using RED [30] like in class III) and based on per hop delay requirements, D_{II}, and current measured MAC delay, D, $CWmin[II]$ and $CWmax[II]$ are adjusted as shown in Algorithm 6.3. Here α_2 and β_2 are parameters specific to class II service, and are chosen for service class II based on characteristics of this class. Similar to other classes, $CWthresh_{II}$ is a threshold to avoid unnecessary CW adaptation for class II.

Algorithm 6.3 Procedure for Contention Window Adjustment of Class II

$CWmin[II] := min\{CWmin^{prev}[II] \times (1 - \alpha_2 \times \frac{(D-D_{II})}{D_{II}}, CWmin^{prev}[II] - \beta_2 \times (Q - Q_{II})\}$
{* Here $CWmin[II]$ and $CWmin^{prev}[II]$ represent current and previous values of $CWmin$ for class II *}
if $(abs(CWmin[II] - CWmin^{prev}[II]) < CWthresh_{II})$ **then**
  $CWmin[II] := CWmin^{prev}[II]$
  return
end if
if $(CWmin[II] < CWmin^{prev}[II])$ **then**
  $CWmin[II] := max\{CWmin[II], CWmin^{default}[II]\}$
  $PF := \frac{CWmin[II]}{CWmin^{prev}[II]}$
  $CWmax[II] := max\{CWmax[II] \times PF, CWmax^{default}[II]\}$
else
  if $(CWmin[II] > CWmin^{prev}[II])$ **then**
    $CWmin[II] := min\{CWmin[II], CWmax^{max}[II]/2\}$
    $PF := \frac{CWmin[II]}{CWmin^{prev}[II]}$
    $CWmax[II] := min\{CWmax[II] \times PF, CWmax^{max}[II]\}$
  end if
end if

Class - IV (Best-effort service):

This class of traffic corresponds to Best-Effort (BE) service of IntServ and DiffServ models. Traffic in this class has neither throughput nor delay guarantee requirements. Contention window adaptation for this class of service is required to properly utilize the available resources in the network without degrading the service quality of other higher quality service class. In this class, based on difference between the current available channel bandwidth, B_{avail}, and the channel overload threshold, B_{resv}, $CWmin[IV]$ is adjusted as shown in *Algorithm* 6.4. Here, γ is a small positive constant depends on how fast adaptation of $CWmin[IV]$ to prevailing network condition should take place and $CWthresh_{IV}$ is a threshold to avoid unnecessary CW adaptation. Because, the contention window adjustment of class-IV traffic should not degrade the performance of ongoing higher priority traffic, a checking is performed such that the $CWmin$ value of Best-effort traffic will not be smaller than the $CWmax$ values of any of the existing high priority traffic in the neighborhood of the node under consideration. To know whether a particular class of traffic exists or not, we have to maintain a Network Allocation Vector (NAV) for each traffic class separately. These NAVs are represented as $NAV[I]$, $NAV[II]$, $NAV[III]$ and $NAV[IV]$ respectively. Accordingly, $CWmin[IV]$ is finally set after checking the NAV for each of the higher priority traffic class, such that best-effort traffic can not affect the service quality of higher priority traffic. The detailed method for $CWmin[IV]$ adjustment is shown in Algorithm 6.4. To carry the service class information along with RTS, CTS and DATA frames in MAC layer, we have utilized the frame control field of the MAC header in IEEE 802.11 MAC frame format. We have used the reserved unused bit patterns in subtype sub fields of control and DATA frames as shown in Table 6.3.

Table 6.3: Using reserved values of subtype in Frame Control Field in MAC Header of IEEE 802.11 to carry traffic class

Type Value $b_3\ b_2$	Type Description	Subtype Value $b_7\ b_6\ b_5\ b_4$	Subtype Description
0 1	Control	0 0 0 0	RTS for Class I
0 1	Control	0 0 0 1	RTS for Class II
0 1	Control	0 0 1 0	RTS for Class III
0 1	Control	0 0 1 1	RTS for Class IV
0 1	Control	0 1 0 0	CTS for Class I
0 1	Control	0 1 0 1	CTS for Class II
0 1	Control	0 1 1 0	CTS for Class III
0 1	Control	0 1 1 1	CTS for Class IV
1 0	Data	1 0 0 0	DATA for Class I
1 0	Data	1 0 0 1	DATA for Class II
1 0	Data	1 0 1 0	DATA for Class III
1 0	Data	1 0 1 0	DATA for Class IV

6.4.6 Congestion Monitoring & Resource Resolver

Though admission control reduces the possibility of congestion or overload situation in the network, to handle dynamic load condition due to mobility and time varying channel conditions, periodic monitoring of the network is very important. In our framework, each node tries to set aside an amount of bandwidth to handle QoS violations due to temporary load fluctuations. Each node passively monitors the available bandwidth in MAC layer periodically, which is used by the network layer as a cross-layer information. When the available bandwidth of a node goes below a threshold called *reservation threshold* (B_{resv}), an alarm for a possible congestion situation is generated and rate control of best-effort traffic is triggered. If the available bandwidth goes below another threshold called *minimum threshold*, B_m, a priority-based marking of packets belonging to higher priority classes is enabled to trigger potential congestion situation. Destination monitors the marked packets. Once it receives a marked packet, it sends a congestion notification to the source. Source which receives a congestion notification, waits for a random time to hear more congestion notifications. If during this period more congestion notification packets about the same flow are received, the given flow is terminated or rerouted. Therefore, during congestion control a victim flow is selected based on the priority of the traffic class of the flow. Since, routing protocol in our framework does not perform resource reservation, therefore, termination of existing flows do not require extra control overhead. Bandwidth availability due to termination of existing flows can be detected and utilized without much delay. Another congestion control method based on RED (Random Early Detection) [30] technique is used in the network layer, which relies on continuous monitoring of the queue size and using dropped threshold for the different queues.

Algorithm 6.4 Procedure for Contention Window Adjustment of Class IV

$CWmin[IV] := CWmin^{prev}[IV] * (1 - \gamma_{IV} * (B_{avail} - B_{resv}))$ {* Here $CWmin[IV]$ and $CWmin^{prev}[IV]$ represent current and previous values of CWmin for class IV *}
if $(abs(CWmin[IV] - CWmin^{prev}[IV]) < CWthresh_{IV})$ **then**
    $CWmin[IV] := CWmin^{prev}[IV]$
    return
end if
if $(NAV[I] \neq Null)$ **then**
    $CWmin[IV] := max\{CWmin[IV], CWmax[I]\}$
end if
if $(NAV[II] \neq Null)$ **then**
    $CWmin[IV] := max\{CWmin[IV], CWmax[II]\}$
end if
if $(NAV[III] \neq Null)$ **then**
    $CWmin[IV] := max\{CWmin[IV], CWmax[III]\}$
end if

6.5 Implementation issues

To implement the QoS mapping from application layer to network layer and finally from network layer to link layer, following steps are taken. First of all, based on the QoS requirement of the application, traffic flows are categorized as Class I, II, III and IV. The application layer requests the transport layer to provide QoS services by passing the required QoS parameters. TCP is used as a transport layer service for class IV traffic, for all other classes UDP is invoked by the application layer. Therefore, UDP packet format should include a field to indicate about the class of service that is being served by the packet. Then transport layer requests network layer to compute routes satisfying the QoS requirements to transport packets from source to destination. Routing protocol of the network layer accordingly starts a route discovery process to find a QoS route for the requested traffic. Once the admission control for the traffic is performed and the flow is admitted, the routing table entry for the flow stores the per-hop maximum delay requirement and the minimum throughput of the admitted flow along with other typical routing table information. The packets of traffic flows are classified at the network layer of the source with the help of DSCP in IP header. A class-based weighted fair queuing (CBWFQ) scheduling at network layer is used for serving packets belonging to different QoS classes as well as serving the network control packets. Congestion in the network is detected through RED queue management procedure and regulating flow with ECN based congestion notification by setting CE bit in IP header. Destination monitors the ECN bits and reports the source about the potential congestion through ICMP QoS lost message. Source after receiving ICMP QoS lost packet waits for a random time period to hear more ICMP QoS lost packets. If during this time period more QoS lost packets are received for a given flow, the flow is terminated/rerouted. MAC layer adaptation is done by modifying IEEE 802.11 DCF sender side functionality to include the setting up of initial contention window ($CWmin$) and maximum contention window ($CWmax$) by accessing the Contention Window Table maintained by the network layer. This leads to a cross-layer interaction between network layer and MAC layer. The MAC layer starts the backoff procedure to content for the medium with other neighboring nodes in CSR. If the transmission encounters collision, retransmission is initiated with the doubling of contention window till the packet is transmitted successfully or the maximum retry limit is reached.

6.6 Conclusion

In this chapter, we have proposed a cross-layer QoS framework for supporting per-class service differentiation in mobile ad hoc networks. According to our model, flows are grouped into four QoS classes based on delay and throughput sensitivity of the underlying applications. The service differentiation among the QoS classes is achieved by

- a class-based scheduler in the network layer
- assigning non-overlapping contention window ranges across different QoS classes
- dynamic adaptation of contention window ranges based on changing network loads and achieved QoS in per-class basis

The overall functionalities can be realized through cross-layer interaction and adaptation of two protocol layers namely network layer and MAC layer.

Chapter 7

Conclusion and Future Direction

This dissertation makes five important contributions to the body of knowledge on QoS support in mobile ad hoc networks. In this chapter, we summarize the main contributions made in this dissertation and provide directions for future works.

7.1 Conclusion

Following conclusions are drawn from the contributions in this dissertation.

- In chapter 3, two dynamic priority based MAC schemes to provide enhanced level of service differentiation in mobile ad hoc networks are proposed.

  PQAMP is designed for supporting a minimum level of service to best-effort traffic while maximizing network utilization. In PQAMP, priorities are assigned to nodes based on their roles and type of traffic to handle, where each priority uses a non-overlapping range of contention window (in the same backoff stage) with lower range assigned to higher priority. By assigning the highest priority to nodes with relay traffic, PQAMP reduces the end-to-end delay for real-time traffic at the same time improving network utilization. Both analytical and simulation results show the effectiveness of the proposed models in supporting QoS for real-time traffic.

  SPQAMP assigns priorities to nodes based on the type of data frame they have to transmit, where each priority uses a non-overlapping contention window irrespective of their backoff stage. SPQAMP maintains the non-overlapping of backoff timer between real-time and best-effort traffic with the help of a backoff counter resetting mechanism for nodes with best-effort traffic. This helps SPQAMP to provide a strict notion of service differentiation, where the QoS support provided to real-time traffic is not affected by the amount of background/best-effort traffic present in the network. Both analytical and simulation results show improvements and the effectiveness of the proposed models in supporting QoS for real-time traffic.

- Route stability plays an important role in improving the performance of QoS routing in MANETs. In chapter 4, we have proposed a route stability based QoS aware routing for Mobile Ad Hoc Networks to provide QoS assurance to applications in terms of maximum delay and minimum throughput. We found that similar QoS aware routing available for MANETs lacks inclusion of a proper stability model which is very important to ensure that discovered path remain valid for a longer period of time. We have shown that use of a simple route stability model (RSM) in the proposed routing significantly reduces the number of route recoveries required during QoS data transmission. This results in significant performance improvements in terms of control overhead, average end-to-end delay and packet delivery ratio especially in highly mobile scenarios with marginal decrease in traffic admission ratio.

- In chapter 5, we have proposed a route stability based multipath QoS routing (SMQR) protocol for Mobile Ad Hoc Networks. SMQR uses a number of special route request forwarding rules to compute a maximum of three node-disjoint QoS routes which reduces path diminution problems without imposing excessive control overheads. Like RSQR, SMQR is also designed to support throughput and delay sensitive real-time applications in MANETs. Reliability of the multiple QoS-aware routes is improved through node-disjointness and stability properties of the discovered routes. Further, SMQR incorporates an effective QoS violations detection and a proactive route maintenance mechanism. These help the protocol to significantly reduce the QoS disruption due to frequent route recovery and hence improve the QoS support for real-time traffic.

- A cross-layer QoS framework for mapping application's QoS requirements across three layers of the protocol stack to support a stronger notion of per-class service guarantees in mobile ad hoc networks is proposed in chapter 6. In the network layer, QoS mapping is realized through a class based queuing and scheduling discipline. QoS mapping at MAC layer is achieved through assignment of non-overlapping contention windows to different service classes. Further, to maintain the QoS mapping despite the network dynamics, both MAC layer and network layer are made adaptive. MAC layer adaptation takes place in the form of dynamic contention window adaptation based on network condition and user QoS requirement. Similarly, network layer adaptation takes place in the form of reroute or termination of existing flows.

7.2 Future Direction

In the following, we outline some of the possible directions of future research works in this field of research.

- Some form of hybridization of both PQAMP and SPQAMP in the form of - inclusion of strict priority scheme in PQAMP or treating nodes to relay frames with higher priority in SPQAMP may give some interesting results. Further, development of an analytical model, based on queuing theory, which aims at studying throughput share among the priority classes as well as the average delay per priority class is also kept under future work.

- Extension of RSQR to detect potential link breaks or QoS violations before actual link breaks or QoS violation takes place. It will help to develop a proactive route maintenance scheme for RSQR to save route recovery time and reduce possibility of packet loss during active communication.

- Extending SMQR to use the node-disjoint paths in parallel to further improve the QoS performance and network utilization. It needs development of an effective packet scheduling scheme to perform load balancing and at the same time reducing packet reordering problem.

- Analysis and validation of the proposed cross-layer framework for service differentiation in MANETs.
- Extending the QoS protocols developed in this dissertation with energy-efficiency as an additional QoS parameter to improve network lifetime. The QoS protocols which we have developed in this dissertation have not considered optimization of energy consumption. Since most of the nodes in MANETs are driven by a limited battery source, energy consumption is tightly integrated with network lifetime and hence the QoS provisioning abilities of the mobile nodes.

Bibliography

[1] I. Aad and C. Casellucia, *Differentiation mechanisms for IEEE 802.11*, in Proc. of IEEE INFOCOM, pp.209-218, April 2001.

[2] A. M. Abbas and B. N. Jain, *Mitigating path diminution in disjoint multipath routing for mobile ad hoc networks*, International Journal of Ad Hoc and Ubiquitous Computing, vol.1, no.3, pp.137-146, May 2006.

[3] G. S. Ahn, A. T. Campbell, A. Veres and L. H. Sun, *SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks*, in Proc. of IEEE INFOCOM, vol.2, pp.457-466, June 2002.

[4] H. Arora, L. Greenwald, U. Rao and J. Novatnack, *Performance Comparison and Analysis of Two QoS schemes: SWAN and DiffServ*, Drexel University Res Day 2003, April 2003.

[5] H. Badis and K. Al. Agha, *QOLSR, QoS routing for Ad Hoc Wireless Networks Using OLSR*, European Transactions on Telecommunications, vol.15, no.4, pp.427-442, September 2005.

[6] H. Badis and K. Al. Agha, *Quality of Service for Ad Hoc Optimized Link State Routing Protocol (QOLSR)*, IETF MANET Working Group, Internet Draft, March 2006.

[7] R. O. Baldwin, N. J. Davis IV and S. F. Midkiff, *A Real-time Medium Access Control Protocol for Ad Hoc Wireless Local Area Networks*, Mobile Computing and Communications Review, vol.3, no.2, pp. 20-27, April 1999.

[8] G. Bianchi, *Performance analysis of the IEEE 802.11 distributed coordination function*, IEEE Journal on Selected Areas in Communications, vol.18, no.3, pp.535-547, March 2000.

[9] S. Blake, *An architecture for Differential Services*, IETF RFC 2475, December 1998.

[10] L. Bononi, L. Budriesi, D. Blasi, V. Cacace, L. Casone and S. Rotolo, *A Differentiated Distributed Coordination Function MAC protocol for cluster-based Wireless Ad hoc networks*, in Proc. of 1st ACM Int'l Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Network, pp.77-86, October 2004.

[11] R. Braden, D. Clark and S. Shenker, *Integrated Services in the Internet Architecture- an Overview*, IETF RFC 1633, June 1994.

[12] A. T-Cabrera, I. N-Perez, E. Casilari and F. J. G-Canete, *Ad Hoc Routing Based on the Stability of Routes*, in Proc. of 4th ACM International workshop on Mobility management and Wireless access (MobiWAC), pp.100-103, October 2006.

[13] R. Castenada, S. R. Das and M. K. Marina, *Query localization techniques for on-demand routing protocols for mobile ad hoc networks*, ACM/Kluwer Wireless Networks (WINET), vol.8, no.2-3, pp.137-151, March-May 2002.

[14] I. D. Chakeres, E. M. Belding-Royer and J. P. Macker, *Perceptive Admission Control for Wireless Network Quality of Service*, Ad Hoc Networks, vol.5, no.7, pp.1129-1148, September 2007.

[15] S. Chakrabarti and A. Mishra, *QoS Issues in Ad Hoc Wireless Networks*, IEEE Communications Magazine, vol.39, no.2, pp.142-148, February 2001.

[16] S. Chen and K. Nahrstedt, *Distributed Quality-of-Service Routing in Ad Hoc networks*, IEEE Journal on Sel. Areas in Communications, vol.17, no.8, pp.1-18, August 1999.

[17] H. Chen, B. Shi, L. Zuo and H. Gong, *A Distributed Entropy-based Long-life QoS Routing Algorithm in Ad hoc Network*, in Proc. of IEEE Canadian Conference on Electrical and Computer Engineering, vol.3, pp.1535-1538, May 2003.

[18] K. Chen, S. H. Shah and K. Nahrstedt, *Cross layer Design for Data Accessibility in Mobile Ad-hoc Networks*, Wireless Personal Communications, vol.21, no.1, pp.49-76, April 2002.

[19] L. Chen and W. Heinzelman, *QoS-aware Routing Based on Bandwidth Estimation for Mobile Ad Hoc Networks*, IEEE Journal on Sel. Areas in Communications, vol.23, no.3, pp.561-572, March 2005.

[20] Y. Chen, Q. Zeng and D. P. Agrawal, *Performance evaluation for IEEE 802.11e enhanced distributed coordination function*, Wireless Communication Mobile Computing, vol.4, no.6, pp.639-653, September 2004.

[21] P. Chevillat, *A Dynamic Link Adaptation Algorithm for IEEE 802.11a Wireless LANs*, in Proc. of IEEE ICC, vol.2, pp.1141-1145, May 2003.

[22] S. Choi, J. del Pedro, N. S. Shankar and S. Mangold, *IEEE 802.11e contention-based channel access (EDCF) performance evaluation*, in Proc. of IEEE International Conference on Communications (ICC), May 2003.

[23] S. Corson and J. Macker, *Mobile Ad hoc Networking (MANET):Routing Protocol Performance Issues and Evaluation Considerations*, RFC-2501, January 1999.

[24] E. Crawley, R. Nair, B. Rajagopalan and H. Sandrick, *A Framework for QoS Based Routing in the Internet*, RFC 2386, August 1998.

[25] B. P. Crow, I. Widjaja, J. G. Kim and P. T. Sakai, *IEEE 802.11 wireless local area networks*, IEEE Communications Magazine, vol.35, no.9, pp.116-126, September 1997.

[26] Z. Y. Demetrios, *A Glance at Quality of Services in Mobile Ad-Hoc Networks*, Technical Report, University of California Riverside, 2001.

[27] D. J. Deng and R. S. Chang, *A Priority Scheme for IEEE 802.11 DCF Access Method*, IEICE Trans. Commun., vol. E82-B, no.1, pp.96-102, January 1999.

[28] D. Dharmaraju, A. R. Chowdhury, P. Hovareshti and J. S. Baras, *INORA - A unified signaling and routing mechanism for QoS support in mobile ad hoc networks*, in Proc. of Int'l Conference on Parallel Processing Workshops (ICPPW), pp.86-93, August 2002.

[29] 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 Communications, pp.36-45, February 1997.

[30] S. Floyd and V. Jacobson, *Random Early Detection gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking, vol.1, Issue 4, pp. 397-413, August 1993.

[31] J. A. Freebersyser and B. Leinerr, *A DoD perspective on mobile ad hoc networks*, Ad Hoc Networking, Ed. C. E. Perkin, Addison-Wesley, pp.29-51, 2001.

[32] D. Gao, J. Cai and K. N. Ngan, *Admission control in IEEE 802.11e Wireless LANs*, IEEE Network Magazine, vol.19, no.4, pp.6-13, July-August, 2005.

[33] R. Garces and J. J. Garcia-Luna-Aceves, *Near-Optimum Channel Access Protocol Based on Incremental Collision Resolution and Distributed Transmission Queues*, in Proc. of IEEE INFOCOM, March-April, 1998.

[34] Y. Ge and J. Hou, *An Analytical Model for Service Differentiation in IEEE 802.11*, in Proc. of IEEE International Conference on Communications (ICC), vol.2, pp.1157-1162, May 2003.

[35] M. Gerharz, C. de Waal, M. Frank and P. Martini, *Link Stability in Mobile Wireless Ad Hoc Networks*, in Proc. of 27th Annual IEEE Conference on Local Computer Networks (LCN), pp.30-39, November 2002.

[36] T. Goff, N. A. Ghazaleh, D. Phatak and R. Kahvecioglu, *Preemptive routing in ad hoc networks*, Journal of Parallel and Distributed Computing, vol.63, no.2, pp.123-140, February 2003.

[37] A. J. Goldsmith and S. B. Wicker, *Design challenges for energy-constrained ad hoc wireless networks*, IEEE Wireless Communications, vol.9, no.4, pp.8-27, August 2002.

[38] S. J. Golestani, *A Self-Clocked Fair Queuing Scheme for Broadband Applications*, in Proc. of IEEE INFOCOM, June 1994.

[39] G. Holland, N. Vaidya and P. Bahl, *A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks*, in Proc. of ACM MOBICOM, June 2001.

[40] Y. Hwang and P. Varshney, *An Adaptive QoS Routing Protocol with Dispersity for Ad-hoc Networks*, in Proc. of 36th Hawaii International Conference on System Sciences (HICSS), pp.302-311, January 2003.

[41] IEEE 802.11 Working Group, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, 1997.

[42] IEEE Task Group E, *IEEE 802.11e-2005, Medium Access Control (MAC) Quality of Service Enhancements*, 2005.

[43] J. Jubin and J. Tornow, *The DARPA Packet Radio Network Protocols*, Proceedings of the IEEE, vol.75, no.1, pp.21-32, January 1987.

[44] J. Jun and M. L. Sichitiu, *Fairness and QoS in Multihop Wireless Networks*, in Proc. of 58th IEEE Conference on Vehicular Technology (VTC), pp.2936-2940, Fall 2003.

[45] A. Kamerman and L. Monteban, *WaveLAN II: A High-performance Wireless LAN for the Unlicensed Band*, Bell Labs Technical Journal, pp. 118-133, Summer 1997.

[46] V. Kawadia and P. R. Kumar, *A Cautionary Perspective on Cross-layer Design*, IEEE Wireless Communications, vol.12, no.1, pp.3-11, February 2005.

[47] V. Kone and S. Nandi, *QoS Constrained Adaptive Routing Protocol for Mobile Adhoc Networks*, in Proc. of 9th International Conference on Information Technology, pp.40-45, December 2006.

[48] A. Ksentini and M. Ibrahim, *Modeling and performance analysis of an improved DCF-based mechanism under noisy channel*, in Proc. of Fourth International Conference on Broadband Communications, Networks, and systems (BroadNets), pp.874-879, September 2007.

[49] S. Kumar, V. S. Raghavan and J. Deng, *Medium Access Control Protocols for Ad hoc Wireless Networks: A survey*, Ad Hoc Networks, vol.4, no.3, pp. 326-358, May 2006.

[50] T. Kuang and C. Williamson, *A bidirectional Multi-channel MAC protocol for improving TCP performance on Multihop Wireless Ad hoc Networks* , in Proc. of 7th ACM Int'l Symposium on Modeling, Analysis and Simulation of Wireless Mobile Systems (MSWiM), pp.301-310, October 2004.

[51] G-S. Kuo and P-C. Ko, *Dynamic RSVP Protocol*, IEEE Communications Magazine, vol.41, no.5, pp.130-135, May 2003.

[52] S. Lee and M. Gerla, *AODV-BR: Backup Routing in Ad Hoc Networks*, in Proc. of IEEE Wireless Communications and Networking Conference (WCNC), pp.1311-1316, August 2000.

[53] S. J. Lee and M. Gerla, *Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks*, in Proc. of the IEEE International Conference on Communications (ICC), pp.3201-3205, June 2001.

[54] M-G. Lee and S. Lee, *A Link Stability Model and Stable Routing for Mobile Ad-Hoc Networks*, Embedded and Ubiquitous Computing, LNCS, vol.4096, pp.904-913, 2006.

[55] S-B. Lee and A.T. Campbell, *INSIGNIA: In-band signaling Support for QoS in Mobile Ad Hoc Networks*, in Proc. of 5th International Workshop on Mobile Multimedia Communications (MoMuC), October 1998.

[56] R. Leung, J. Liu, E. Poon, A-L. C. Chan and B. Li, *MP-DSR: A QoS-aware Multipath Dynamic Source Routing Protocol for Wireless Ad-hoc Networks*, in Proc. of 26th IEEE Annual Conference on Local Computer Networks (LCN), pp.132-141, November 2001.

[57] J. Li and P. Mohapatra, *PANDA: A Novel Mechanism for Flooding Based Route Discovery in Ad Hoc Networks*, Wireless Networks, vol.12, no.6, pp.771-781, November 2006.

[58] X. Li and L. Cuthbert, *Stable Node-Disjoint Multipath Routing with Low Overhead in Mobile Ad hoc Networks*, in Proc. of 12th IEEE International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp.184-191, October 2004.

[59] X. Li and L. Cuthbert, *Multipath QoS routing of supporting Diffserv in Mobile Ad hoc Networks*, in Proc. of the 6th International conference on Software Engineering, Artificial Intelligence, networking and Parallel/distributed Computing and 1st ACIS international Workshop on Self-Assembling Wireless Networks (SNPD/SAWN), May 2005.

[60] Z. Li, S. Nandi and A.K. Gupta, *Achieving MAC fairness in wireless ad-hoc networks using adaptive transmission control*, in Proc. of Ninth Int'l Symposium on Computers and communications, vol.1, pp.176-181, June-July, 2004.

[61] Z. Li, A. Das, A.K. Gupta and S. Nandi, *Full auto rate MAC protocol for wireless ad hoc networks*, IEE Proceedings on Communications, vol.152, no.3, pp.311-319, June 2005.

[62] W-H. Liao, Y-C. Tseng, S-L. Wang and J-P. Sheu, *A Multi-Path QoS Routing Protocol in a Wireless Mobile Ad Hoc Network*, Telecommunication Systems, vol.19, no.3-4, pp.329-347, March 2002.

[63] W. Liu and Y. Fang, *Courtesy Piggybacking: Supporting Differentiated Services in Multihop Mobile Ad Hoc Networks*, IEEE Transactions on Mobile Computing, vol.3, no.4, pp.380-393, October-December, 2004.

[64] G. Lim, K. Shin, S. Lee and H. Yoon, *Link Stability and Route Lifetime in Ad Hoc Wireless Networks*, in Proc. of Int'l Conference on Parallel Processing Workshops (ICPPW), pp.116-123, August 2002.

[65] C. R. Lin and M. Gerla, *MACA/PR: An Asynchronous Multimedia Wireless Network*, in Proc. of IEEE INFOCOM, vol.1, pp.118-125, April 1997.

[66] C-R. Lin, *On-Demand QoS Routing in Multihop Mobile Networks*, in Proc. of IEEE INFOCOM, pp.1735-1744, April 2001.

[67] A. Lindgren, A. Almquist and O. Schelen, *Evaluation of Quality of Service schemes for IEEE 802.11 Wireless LANs*, in Proc. of 26th Annual IEEE Conference Local Computer Networks (LCN), pp.348-351, November 2001.

[68] B. Lu and U. W. Pooch, *A Game Theoretic Framework for Bandwidth Reservation in Mobile Ad Hoc Networks*, in Proc. of First International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE), October 2004.

[69] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz and L. Stibor, *IEEE 802.11e wireless LAN for quality of service*, in Proc. of European Wireless, February 2002.

[70] B. S. Manoj, R. Ananthapadmanabha and C. S. R. Murthy, *Link life Based Routing Protocol for Ad Hoc Wireless Networks*, in Proc. of 10th International Conference on Computer Communications and Networks, pp. 573-576, October 2001.

[71] M. K. Marina and S. R. Das, *On-demand Multipath Distance Vector Routing in Ad Hoc Networks*, in Proc. of International Conference for Network Protocols (ICNP), pp.14-23, November 2001.

[72] M. Mirhakkak, N. Schultz and D. Thompson, *Dynamic QoS for Mobile Ad Hoc Networks*, The MITRE corporation, April 2000.

[73] P. Mohapatra, J. Li and C. Gui, *QoS in Mobile Ad Hoc Networks*, IEEE Wireless Communications Magazine, vol.10, no.3, pp.44-52, June 2003.

[74] C. S. R. Murthy and B. S. Manoj, *Ad Hoc Wireless Networks: Architectures and Protocols*, Prentice Hall, New Jersey, USA, 2004.

[75] A. Nafaa, *Provisioning of Multimedia services in 802.11-based Networks: Facts and Challenges*, IEEE Wireless Communications, vol.14, no.5, pp.106-112, October 2007.

[76] A. Nasipuri, R. Castaneda, S. R. Das, *Performance of Multipath Routing for On-Demand Protocols in Mobile Ad Hoc Networks*, Mobile Networks and Applications, vol.6, no.4, pp.339-349, August 2001.

[77] K. Nichols, V. Jacobson and L. Zhang, *A Two-bit Differentiated Services architecture for the Internet*, RFC-2638, July 1999.

[78] N. Nikaein and C. Bonnet, *A Glance at Quality of Service Models for Mobile Ad Hoc Networks*, in Proc. of DNAC, December 2002.

[79] No Ad Hoc Routing Agent (NOAH), available at `http://icapeople.epfl.ch/widmer/uwb/ns-2/noah`

[80] NS-2, available at `http://www.isi.edu/nsnam/ns/`

[81] A. Pal, *Distributed MAC Layer Protocols for Real-Time Communication in Ad-Hoc Wireless Networks*, M.S. Thesis, Ohio State University, 2001.

[82] A. Pal, A. Dogan and F. Ozguner, *MAC layer protocols for real-traffic in ad-hoc networks*, in Proc. of IEEE International Conference on Parallel Processing (ICPP), August 2002.

[83] V. D. Park and M. S. Corson, *A highly adaptive distributed routing algorithm for mobile wireless networks*, in Proc. of IEEE INFOCOM, vol.3, pp.1405-1413, April 1997.

[84] D. D. Perkins and H. D. Hughes, *A survey on quality-of-service support for mobile ad hoc networks*, Wireless Communications and Mobile Computing, vol.2, no.5, pp.503-513, September 2002.

[85] C. E. Perkins, E. M. Belding-Royer and S. R. Das, *Ad hoc on-demand distance vector (AODV) routing*, IETF Internet Draft, MANET working group, January 2002.

[86] C. E. Perkins, E. M. Belding-Royer, *Quality of Service for Ad hoc On-Demand Distance Vector Routing*, Internet Draft, October 2003.

[87] R. S. Prasad, M. Murray, C. Dovrolis and K. Claffy, *Bandwidth Estimation: Metrics, Measurement Techniques and Tools*, IEEE Networks, vol.17, no.6, pp.27- 35, November-December, 2003.

[88] Q. Qu, Y. Pei, J. W. Modestino, X. Tian and B. Wang, *Cross-Layer QoS Control for Video Communications over Wireless Ad Hoc Networks*, EURASIP Journal on Wireless Communications and Networking, vol.5, pp.743-756, May, 2005.

[89] T. B. Reddy, I. Karthigeyan, B. S. Manoj and C. S. R. Murthy, *Quality of Service provisioning in ad hoc wireless networks: a survey of issues and solutions*, Ad Hoc Networks, vol.4, no.1, pp.83-124, January 2006.

[90] I. Rubin and Y-C. Liu, *Link Stability Models for QoS Ad Hoc Routing Algorithms*, in Proc. of IEEE Conference on Vehicular Technology (VTC), pp.3084-3088, October 2003.

[91] B. Sadeghi, V. Kanodia, A. Sabharwal and E. Knightly, *Opportunistic Media Access for Multirate Ad-hoc Networks*, in Proc. of the 8th ACM International Conference on Mobile Computing and Networking (MobiCom), September 2002.

[92] P. Sambasivam, A. Murthy and E. M. Belding-Royer, *Dynamically Adaptive Multipath Routing based on AODV*, in Proc. of the 3rd Annual Mediterranean Ad Hoc Networking Workshop (MedHocNet), June 2004.

[93] K. Sanzgiri, I. D. Chakeres and E. M. Belding-Royer, *Determining Intra-Flow Contention along Multihop Paths in Wireless Networks*, in Proc. of 1st Annual International Conference on Broadband Networks, (BroadNets), pp.611-620, October 2004.

[94] K. Sanzgiri, I. D. Chakeres and E. M. Belding-Royer, *Pre-Reply Probe and Route Request Tail: approaches to calculate intra-flow contention in multihop wireless networks*, Mobile Networks and Applications, vol.11, no.1, pp.21-35, February 2006.

[95] E. Setton, T. Yoo, X. Zhu, A. Goldsmith and B. Girod, *Cross-Layer Design of Ad Hoc Networks for Real-Time Video Streaming*, IEEE Wireless Communications, vol.12, no.4, pp.59-64, August 2005.

[96] S.-T. Sheu and T.-F. Sheu, *A bandwidth allocation/sharing/extension protocol for multimedia over IEEE 802.11 ad hoc wireless LANs*, IEEE Journal on Sel. Areas in Communications, vol.19, no.10, pp.2065-2080, October 2001.

[97] J.-P. Sheu, C. H. Liu, S.-L. Wu and Y.-C. Tseng, *A priority MAC protocol to support real-time traffic in ad hoc networks*, Wirless Networks, vol.10, no.1, pp.61-69, January 2004.

[98] P. Sinha, *QoS Issues in Ad-Hoc Networks*, Ad Hoc Networks: Technologies and Protocols, Springer Science + Business Media Inc., pp. 229-247, 2006.

[99] R. Sivakumar, P. Sinha and V. Bharghavan, *CEDAR: a Core-Extraction Distributed Ad Hoc Routing algorithm*, IEEE Journal on Selected Areas in Communications, vol.17, no.8, pp.1454-65, August 1999.

[100] J. L. Sobrinho and A. S. Krishnakumar, *Quality-of-Service in Ad Hoc Carrier Sense Multiple Access Wireless Networks*, IEEE Journal on Sel. Areas in Communications, vol.17, no.8, pp.1353-1368, August 1999.

[101] K. N. Sridhar and M. C. Chan, *Stability and Hop-Count based Approach for Route Computation in MANET*, in Proc. of 14th International Conference on Computer Communications and Networks (ICCCN), October 2005.

[102] K. N. Sridhar and L. Jacob, *Performance evaluation and enhancement of a link stability based routing protocol for MANETs*, International Journal of High Performance Computing and Networking (IJHPCN), vol.4, no.1-2, pp.66-77, June 2006.

[103] A. K. Talukdar, B. R. Badrinath and A. Acharya, *MRSVP: A Resource Reservation Protocol for an Integrated Services Network with Mobile Hosts*, Wireless Networks, vol.7, no.1, pp.5-19, January 2001.

[104] O. Tickoo, S. Raghunath and S. Kalyanaraman, *Route Fragility: A Novel Metric for Route selection in Mobile Ad Hoc Networks*, in Proc. of the 11th IEEE International Conference on Networks (ICON), pp.537-542, September-October, 2003.

[105] C-K. Toh, *Associativity-Based Routing for Ad-Hoc Mobile Networks*, Wireless Personal Communications, vol.4, no.2, pp.1-36, March 1997.

[106] T. M. Trung, J. Mo and S-L. Kim, *A Flow-Based Media Access Control (F-MAC) for Wireless Ad-Hoc Networks*, IEICE Trans. on Communications, vol. E89-B, no.3, March 2006.

[107] N. H. Vaidya, S. Bahl and S. Gupta, *Distributed fair scheduling in a wireless LAN*, in Proc. of 6th Annual Intl. Conference on Mobile Computing and Networking, August 2000.

[108] A. Veres, A. T. Campbell, M. Barry and L-H. Sun, *Supporting Service Differentiation in Wireless Packet Networks Using Distributed Control*, IEEE Journal on Selected Areas in Communications, vol.19, no.10, pp.2081-2093, October 2001.

[109] H. Q. Wang, K. C. Yow and S. Nandi, *Providing Quality of Service to Real-time Traffic through Multipath Discovery*, in Proc. of First International Conference on Mobile Computing and Ubiquitous Networking (ICMU), vol.2, January 2004.

[110] Y-H. Wang, C-H. Tsai, H-Z. Lin and C-A. Wang1, *Interference-aware QoS Multipath Routing for Ad Hoc Wireless Networks*, International Journal of Computers and Applications, vol.29, no.4, 2007.

[111] S.-L. Wu, S.-Y. Ni, Y.-C. Tseng and J.-P. Sheu, *Route Maintenance in a Wireless Mobile Ad Hoc Network*, Telecommunication Systems, vol.18, no.1-3, pp.61-84, September 2001.

[112] K. Wu and J. Harms, *QoS Support in Mobile Ad Hoc Networks*, Crossing Boundaries- an Interdisciplinary Journal, vol.1, no.1, Fall 2001.

[113] H. Xiao, W. K. G. Seah, A. Lo and KC Chua, *A Flexible Quality of Service Model for Mobile Ad-Hoc Networks*, in Proc. of IEEE Conference on Vehicular Technology (VTC), May 2000.

[114] Y. Xiao, *Performance analysis of priority schemes for IEEE 802.11 and IEEE 802.11e Wireless LANs*, IEEE Trans. on Wireless Communications, vol.4, no.4, pp.1506-1515, July 2005.

[115] Q. Xue and A. Ganz, *Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks*, Journal of Parallel and Distributed Computing, vol.63, no.2, pp.154-165, February 2003.

[116] Y. Yang and R. Kravets, *Distributed QoS Guarantees for Realtime Traffic in Ad Hoc Networks*, in Proc. of IEEE SECON, pp.118-127, October 2004.

[117] Y. Yang and R. Kravets, *Contention-Aware Admission Control for Ad Hoc Networks*, IEEE Trans. on Mobile Computing, vol.4, no.4, pp.363-377, July-August, 2005.

[118] Z. Ye, S. V. Krishnamurthy and S. K. Tripathi, *Framework for Reliable Routing in Mobile Ad Hoc Networks*, in Proc. of IEEE INFOCOM, vol.1, pp.270-280, March-April, 2003.

[119] S. Yin, Y. Xiong, Q. Zhang and X. Lin, *Traffic-aware Routing for Real Time Communications in Wireless Multi-hop Networks*, Wireless Commun. Mobile Computing, vol.6, no.6, pp.825-843, September 2006.

[120] L. Zhang, S. Deering and D. Estrin, *RSVP:A new resource reservation protocol*, IEEE Networks, vol.7, no.5, pp.8-18, September 1993.

[121] Q. Zhang and Y.-Q. Zhang, *Cross-Layer Design for QoS Support in Multihop Wireless Networks*, Proceedings of the IEEE, vol.96, no.1, pp.64-76, January 2008.

[122] H. Zhu, M. Li, I. Chlamtac and B. Prabhakaran, *A Survey of Quality of Service in IEEE 802.11 Networks*, IEEE Wireless Communications, vol.11, no.4, pp.6-14, August 2005.

[123] W. Zhu, M. Song and S. Olariu, *Integrating Stability Estimation into Quality of Service Routing in Mobile Ad-hoc Networks*, in Proc. of 14th IEEE International Workshop on Quality of Service (IWQoS), pp.122-129, June 2006.

Publications based on the Thesis Works

1. N. Sarma and S. Nandi, Enhancing QoS Support in Mobile Ad Hoc Networks, *International Conference on Telecommunications and Networking (TENE 2005), Springer Netherlands*, pp.268-274, December 2005.

2. N. Sarma and S. Nandi, QoS Support in Mobile Ad Hoc Networks, *Third IEEE/IFIP International Conference on Wireless and Optical Communications Networks (WOCN 2006)*, Bangalore, India, April 2006 (IEEE Computer Society Press).

3. M. Jims, N. Sarma and S. Nandi, Priority Based Fairness Provisioning QoS-Aware MAC Protocol, *15th International Conference on Advanced Computing and Communications*, December 2007 (IEEE Press).

4. N. Sarma and S. Nandi, Route Stability based QoS Routing (RSQR) in MANETs, *10th International Symposium on Wireless Personal Multimedia Communications, WPMC-2007*, December 2007.

5. N. Sarma, S. Nandi and R. Tripathi, Enhancing Route Recovery for QAODV Routing in Mobile Ad Hoc Networks, *10th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN 2008)*, pp.39-44, May 2008 (IEEE Press).

6. N. Sarma and S. Nandi, A Route Stability based Multipath QoS Routing (SMQR) in MANETs, *1st International Conference on Emerging Trends in Engineering and Technology (ICETET-08)*, pp.193-198, 16-18th July 2008 (IEEE Press).

7. N. Sarma and S. Nandi, Route Stability based QoS Routing in Mobile Ad Hoc Networks, accepted for publication in the *Special Issue of the Wireless Personal Communications Journal, Springer*.

8. N. Sarma and S. Nandi, A Priority based QoS-Aware MAC Protocol (PQAMP) for Mobile Ad Hoc Networks, *4th ACM International Workshop on QoS and Security for Wireless and Mobile Networks (Q2SWinet 2008)*, pp.79-82, October 27-28, 2008.

9. N. Sarma and S. Nandi, A Strict Priority based QoS-Aware MAC Protocol (SPQAMP) for Mobile Ad Hoc Networks, *5th International Conference on Distributed Computing and Internet Technologies (ICDCIT 2008)*, Dec 10-12, 2008 (LNCS vol no. 5375).

10. N. Sarma and S. Nandi, A Cross-layer QoS mapping framework for Mobile Ad Hoc networks, accepted for publication in *IETE Technical Review*, Vol. 25, No. 6, Nov-Dec 2008.

11. N. Sarma and S. Nandi, Enhanced Service Differentiation using Priority based MAC Protocol for MANETs, under review for *Wireless Networks, Springer/ACM*.

12. N. Sarma and S. Nandi, Providing QoS using Priority based MAC Protocol in MANETs, submitted to *Performance Evaluation, Elsevier*.