NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

QUALITY OF SERVICE SCHEMES FOR MOBILE AD-HOC NETWORKS

by

Leonardo da Silva Mattos

March 2001

Chairman of Committee and Supervisor:	Murali Tummala
Committee Member:	John McEachen
Committee Member:	Robert Ives

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 2001	3. REPORT TYPE AND DATES COVERED Engineer's Thesis	
4. TITLE AND SUBTITLE: Quality of Service Schemes for Mobile Ad-hoc Networks			5. FUNDING NUMBERS
6. AUTHOR(S) Mattos, Leonardo da Silva.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) SPAWARSYSCEN, D841 (Attn: Dr. North) 53560 Hull Street, San Diego, CA 92152 - 5001			10.SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed here are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis proposes schemes to provide Quality of Service (QoS) in mobile ad-hoc networks (MANETs). To achieve QoS, independently of the routing protocol, each mobile node participating in the network must implement traffic conditioning, traffic marking and buffer management (Random Early Drop with in-out dropping) or queue scheduling (Priority Queuing) schemes. In MANETs, since the mobile nodes can have simultaneous multiple roles (*ingress, interior* and *destination*), it was found that traffic conditioning and marking must be implemented in all mobile nodes acting as source (*ingress*) nodes. Buffer management and queue scheduling schemes must be performed by all mobile nodes.

By utilizing the Network Simulator (NS2) tool, this thesis focused on the empirical performance evaluation of the QoS schemes for different types of traffic (FTP/TCP, CBR/UDP and VBR/UDP), geographical areas of different sizes and various mobility levels. Key metrics, such as throughput, end-to-end delay and packet loss rates, were used to measure the relative improvements of QoS-enabled traffic sessions. The results indicate that in the presence of congestion, service differentiation can be achieved under different scenarios and for different types of traffic, whenever a physical connection between two nodes is realizable.

14. SUBJECT TERMS Joint Tactical Radio System, Network Simulator 2, Dynamic Source Routing, Quality of Service, Resource Reservation Protocol, Differentiated Services, Mobile Ad-hoc Network			15. NUMBER OF PAGES 178
			16. PRICE CODE
17.SECURITYCLASSIFICATION OF REPORT Unclassified	18.SECURITYCLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

THIS PAGE INTENTIONALLY LEFT BLANK

QUALITY OF SERVICE ANALYSIS IN MOBILE AD-HOC NETWORKS

Leonardo da Silva Mattos
Lieutenant Commander, Brazilian Navy
B.S., Escola Naval (Brazil), December 1988
B.S.E.E., Universidade de São Paulo (Brazil), December 1994

Submitted in partial fulfillment of the
requirements for the degrees of

ELECTRICAL ENGINEER
and
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2001

Author: [illegible]
Leonardo da Silva Mattos

Approved by: [illegible]
Murali Tummala, Chairman of Committee and Supervisor

[illegible]
John McEachen, Committee Member

[illegible]
Robert Ives, Committee Member

[illegible]
Jeffrey B. Knorr, Chairman
Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis proposes schemes to provide Quality of Service (QoS) in mobile ad-hoc networks (MANETs). To achieve QoS, independently of the routing protocol, each mobile node participating in the network must implement traffic conditioning, traffic marking and buffer management (Random Early Drop with in-out dropping) or queue scheduling (Priority Queuing) schemes. In MANETs, since the mobile nodes can have simultaneous multiple roles (*ingress, interior* and *destination*), it was found that traffic conditioning and marking must be implemented in all mobile nodes acting as source (*ingress*) nodes. Buffer management and queue scheduling schemes must be performed by all mobile nodes.

By utilizing the Network Simulator (NS2) tool, this thesis focused on the empirical performance evaluation of the QoS schemes for different types of traffic (FTP/TCP, CBR/UDP and VBR/UDP), geographical areas of different sizes and various mobility levels. Key metrics, such as throughput, end-to-end delay and packet loss rates, were used to measure the relative improvements of QoS-enabled traffic sessions. The results indicate that in the presence of congestion, service differentiation can be achieved under different scenarios and for different types of traffic, whenever a physical connection between two nodes is realizable.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE LEFT INTENTIONALLY BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ABBREVIATIONS

ACK	Acknowledgment
AODV	Ad Hoc On-Demand Distance Vector Routing
API	Application Program Interface
ARP	Address Resolution Protocol
AS	Assured Service
ATM	Asynchronous Transfer Mode
BPS	Bits per Second
BER	Bit Error Rate
CBQ	Class Based Queuing
CBR	Constant Bit Rate
CEDAR	Core Extraction Distributed Ad-Hoc Algorithm
CLP	Cell Loss Priority
COTS	Commercial-off-the-shelf
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
CTS	Clear-to-Send
DAMA	Demand Assignment Multiple Access
DARPA	Defense Advanced Research Projects Agency
DCF	Distributed Coordination Function
DIFFSERV	Differentiated Services
DMR	Digital Modular Radio
DOD	Department of Defense
DSCP	Differentiated Service Code Point
DSDV	Destination Sequenced Distance Vector Routing
DSR	Dynamic Source Routing Protocol
DSSS	Direct Sequence Spread Spectrum
EF	Expedited Forwarding
FDMA	Frequency Division Multiple Access

FIFO	First-In First-Out
FTP	File Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
INTSERV	Integrated Services
IP	Internet Protocol
ISI	Information Sciences Institute
JTRS	Joint Tactical Radio System
LAN	Local Area Network
LBNL	Lawrence Berkeley National Laboratory
LOS	Line of Sight
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
NPS	Naval Postgraduate School
NAM	Network Animator
NS2	Network Simulator 2
OTCL	Object Tool Command Language
PARC	Palo Alto Research Center
PQ	Priority Queuing
QDR	Quadrennial Defense Review
QoS	Quality Of Service
RED	Random Early Discard
RF	Radio Frequency
RFC	Request For Comments
RREP	Route Reply
RREQ	Route Request
RSVP	Resource Reservation Protocol
RTP	Real-time Transport Protocol
SDR	Software Defined Radio
SIFS	Short Interframe Space
TCL	Tool Command Language

TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TORA	Temporally Ordered Routing Algorithm
TOS	Type of Service
TTL	Time to Live
UCB	University of California at Berkeley
UCLA	University of California at Los Angeles
UDP	User Datagram Protocol
UHF	Ultra High Frequency
USC	University of Southern California
VINT	Virtual InterNetwork Testbed
VTC	Video Teleconferencing
WAN	Wide Area Network
WFQ	Weight Fair Queuing
WLAN	Wireless Local Area Network
ZRP	Zone Routing Protocol

THIS PAGE LEFT INTENTIONALLY BLANK

EXECUTIVE SUMMARY

The Joint Tactical Radio System (JTRS) acquisition program was established to develop a new family of tactical digital radios that will be interoperable with existing tactical radios, will be capable of performing wireless data internetworking, and will be based on a common communications system architecture. The most ambitious objective of JTRS is to exploit the radios to work effectively in a mobile ad hoc network (MANET) environment. Quality of Service (QoS), among other issues, is one of the main requirements to achieve the desired efficiency.

QoS support in networks means providing the applications with enough network resources in order to achieve acceptable performance. Delay-sensitive applications such as real-time voice/video or important file transfers, should receive special treatment from the network compared to other low priority traffic. By doing this, the network offers QoS support to the high priority traffic, improving the perceived quality of the information contained in the traffic.

The wireless bandwidth constraints and dynamic, multi-hop topologies of mobile ad-hoc networks do not allow direct application of QoS protocols widely available for fixed Internet Protocol (IP) and cellular networks. Because of MANETs' dynamic nature and bandwidth limitations, the use of any type of signaling-based protocol, such as Resource Reservation Protocol (RSVP), to guarantee bandwidth is not recommended. By adapting DiffServ function blocks and using the IntServ/RSVP's per-flow approach from fixed IP-network, this thesis proposes schemes to provide QoS for MANETs in a flexible manner.

To achieve this objective, the mobile nodes participating in a MANET must implement traffic conditioning and buffer management or packet scheduling schemes. Traffic packets are conditioned and marked as high or low priority before being sent to the wireless channel. By utilizing the buffer management or the packet scheduling scheme in each node's buffer, the mobile nodes are able to prioritize the service of packets pre-defined as high priority.

By extensive simulation under different MANET scenarios, this thesis evaluated the performance of the proposed QoS schemes, in terms of throughput, end-to-end delay and packet loss rates. Different traffic patterns, such as file transfer using a reliable connection service and constant and variable bit rate streams using a datagram delivery service, were evaluated. Results indicated that in the presence of congestion, service differentiation was achieved using either packet scheduling or buffer management schemes for all types of traffic under a variety of geographical areas (400 to 2500 km^2) and mobility levels. High priority traffic sessions obtained relative throughput improvements compared to sessions with low priority. Additionally, simulation results indicated that in MANETs, the built-in congestion control mechanisms of the reliable connection service can degrade the overall throughput performance but keep the loss rates and the end-to-end delay at low levels. On the other hand, for real-time traffic using datagram service, traffic differentiation is achieved, but the absence of any congestion control mechanisms along with the mobile node's limited buffer size causes loss rates and end-to-end delays to increase significantly sometimes.

ACKNOWLEDGMENTS

I must take this opportunity to acknowledge the constant support and understanding of the two most important people in my life, my wife Patricia and my daughter Rebeca. Their love has been the inspiration to me throughout the course of my life. I don't have words to express the gratitude I have for their patience and support.

I also would like to acknowledge the omnipresent support of my father, Anoildo Mattos, who has always encouraged me to pursue high education.

My great appreciation to the Telecommunication Department of the Brazilian Navy who has placed faith in my ability to conduct advanced research and lead technical staffs in years to come.

I would like to thank my advisor Prof. Murali Tummala, for being there for me whenever needed. His guidance, advice, listening, careful revisions of this work, and his friendship above all, made me feel very proud of being a NPS student.

The committee members, Professor John McEachen and Prof. Robert Ives not only have I been honored to attend their classes and to receive their advices, but their professionalism and thoroughness made me work harder and become more professional.

A special hand goes out to Xiao Hannan from National University of Singapore. I followed in the footsteps of this "brilliant guy" and this thesis would not have been possible had it not been for his C++ codes and continuous support, information exchange and guidance along this journey.

My special thanks to my colleagues and friends, Major (USMC) Kevin Shea, Captain (USMC) Tyrone Theriot and LT Leonidas (Hellenic Navy) for sparing the time to discuss several topics and to contribute significantly on the research of the material for my thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The quality of service (QoS) issues in mobile ad-hoc networks (MANET) are better understood by taking into consideration the unique requirements of typical military traffic applications that demand *QoS guarantees.* According to [1], scalability, ease-of-use, mobility and *QoS guarantees* are requirements that must be supported by the future networked radio nodes in the envisioned mobile ad-hoc network environment of military operations.

Presently, military systems tend to achieve QoS by relying on application-specific, dedicated systems [2]. For example, QoS can be provided by reserving a frequency channel exclusively for a certain mission-critical application. While this approach provides some degree of QoS, it might lead to poor utilization of scarce wireless bandwidth.

Commercial systems, such as cellular telephones or mobile IP networks, obtain QoS capability through virtual circuit switching and by relying on the fixed infrastructure. In both cellular and mobile IP networks, the receiving node is typically one hop away. Cellular systems use a circuit switching approach in which bandwidth reservation is achieved with the help of explicit signaling established in the network prior to communication. Mobile IP communications might utilize the fixed infrastructure and protocols, such as RSVP, to provide resource reservation. In contrast, the dynamic nature of the ad-hoc multi-hop wireless networks requires more responsive protocols (with low overhead), and the virtual circuit approach is not attractive in MANETs.

In addition, today's military systems need to transmit multimedia traffic, such as voice, video, text, and images. It is known that the specific characteristics of these types of traffic demand better network response to guarantee performance metrics, such as delay and throughput. While providing *QoS guarantees* to multimedia flows in fixed or one-hop (cellular like) wireless networks is a problem that has been addressed to some extent in the past, for MANETs the solutions are in early stages of research and development.

The motivation behind this thesis is the need to address the QoS issues in MANETs in order to overcome the challenges imposed by the dynamic, rapidly changing, multi-hop and bandwidth-constrained wireless ad-hoc environment.

B. OBJECTIVES

The objectives of this thesis are to empirically analyze the QoS-related issues in all layers of the existing protocol stack and to propose and develop schemes and algorithms to provide QoS for mobile nodes in a MANET during congestion, independent of the routing protocol being used. The intended QoS protocols must allow service differentiation between high priority traffic sessions and low priority traffic sessions in terms of throughput, delay, and other metrics. Service differentiation is achieved by implementing traffic conditioning, traffic marking, Priority Queuing (PQ), and Random Early Drop with IN/OUT (RED/RIO) queuing disciplines in the mobile nodes. Since analytical (mathematical) modeling of the QoS problem is complex due to the dynamic behavior of MANETs, several simulations were run under different scenarios for different types of traffic in order to verify the performance of the proposed QoS schemes.

C. THESIS OUTLINE

This thesis is organized as follows. Chapter II presents an overview of Software Defined Radios (SDR) and the Joint Tactical Radio Systems (JTRS) program, which is the main motivator of this thesis. It also points to the increasing need for investigating MANET protocols for military use, especially the protocols related to Quality of Service. Chapter III reviews the basis of mobile ad-hoc networks, emphasizing the QoS-related issues of each layer of the protocol stack. Chapter IV describes the challenges in providing QoS in MANETs. It also reviews two QoS protocols for fixed IP networks, highlighting the limitations and advantages of applying them to MANETs. Chapter V presents the schemes and algorithms investigated to provide QoS in MANETs. It discusses the assumptions made to implement the QoS model in MANETs. Based on that, the details of the algorithms utilized by the model are discussed. Chapter VI describes the network simulation tool and presents the details of the parameters used to simulate certain military scenarios. Chapter VII shows the graphical simulation results for different scenarios and traffic types. Chapter VIII concludes the thesis by reviewing the results achieved and by suggesting future extensions to the proposed model. Appendix A contains an example of a script file program generated by the user. Appendix B contains a segment of an output trace file generated by one of the simulations. Examples of node movement and traffic pattern files are shown in Appendix C and Appendix D, respectively.

THIS PAGE INTENTIONALLY LEFT IN BLANK

II. SOFTWARE DEFINED RADIOS (SDR)

In the modern war environment, wireless mobile networks are envisioned to provide seamless connectivity among radio nodes and provide a communications backbone that can be deployed on short notice. Future radio nodes are planned to be all digital, hence programmable and flexible. Legacy systems typically consist of single band or single mode radios that have limited or no networking capability. Consequently, legacy systems require complex technology solutions to be integrated into networks. The main goal of the Joint Program Office (JPO) is to migrate the existing legacy systems to systems compliant with an open architecture in which each mobile node is represented by a software-defined radio (SDR) belonging to an integrated mobile ad-hoc network (MANET).

This chapter starts with an overview of the SDR technology. Section B summarizes the JTRS program and introduces the Digital Modular Radio (DMR), which is the present US Navy effort related to the JTRS program. In Section C, we present the current areas of research in the context of JTRS.

A. OVERVIEW

In principle, the Software Defined Radio (SDR) is a flexible hardware platform that integrates functional modules and software, as required, to realize a specified radio node operating over a defined frequency band. SDR is an implementation technique that increases the speed, flexibility and economy with which wireless systems and equipment can be developed, deployed, upgraded and debugged. In SDR, information channel processing is accomplished by software-programmable, hardware re-configurable processor elements, such as digital signal processing (DSP) chips, microprocessor chips, field-programmable arrays (FPGA), and other programmable devices [1].

By using the SDR technology, tactical radios will be able to perform more than one radio node function associated with a particular frequency range and waveform. SDR will enable emulation of numerous legacy air interfaces and will act as a bridge between incompatible air interfaces. This will certainly lengthen the useful life of legacy systems, diminish the barriers to communications among military services and allied forces, and ease the transition from legacy radios to SDRs.

The ultimate objective of the SDR technology is to provide an efficient and comparatively inexpensive mechanism for the production of multi-mode, multi-band, multi-functional wireless devices that can be enhanced using software upgrades. Consequently, SDR provides the baseline upon which an advanced radio capable of meeting increasing networking, security, and life cycle cost concerns can be built. Figure 2.1 is the functional interface diagram of a SDR [1].

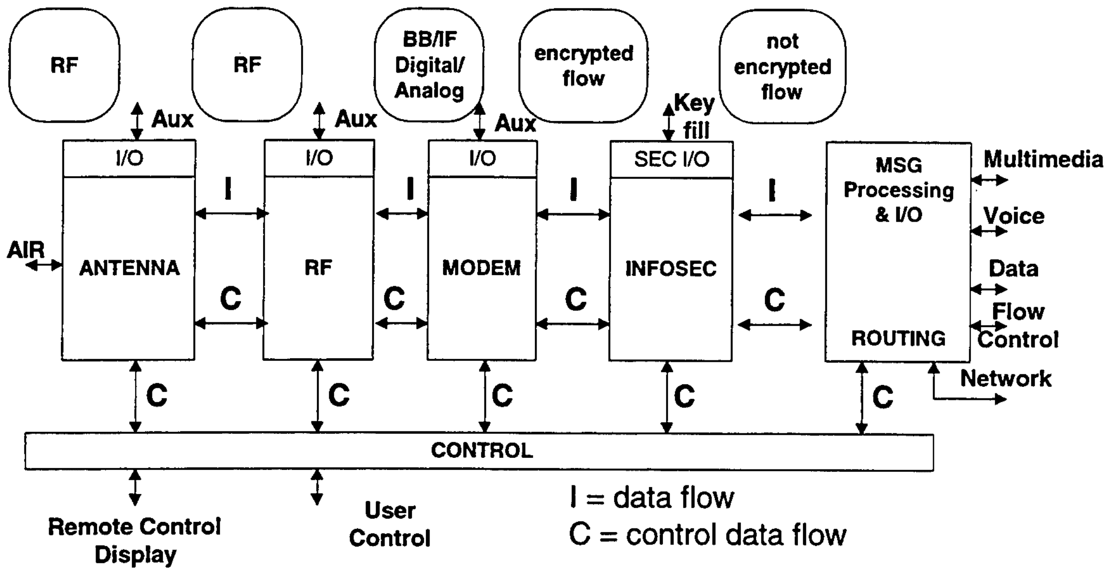


Figure 2.1 Functional Interface Diagram in SDR Architecture (After Ref. [1]).

B. JOINT TACTICAL RADIO SYSTEMS (JTRS)

JTRS is a joint acquisition program with the Army assigned as the lead service. The Joint Tactical Radio System acquisition program was a result of the 1997 Quadrennial Defense Review (QDR), which called for all the services to combine and integrate all tactical radio development. The JTRS Joint Project Office (JPO) is represented by all of the services.

As outlined in the JTRS Mission Need Statement and the JTRS Joint Operational Requirements Document [1], the goal of the JTRS project is to develop a family of affordable, high-capacity tactical radios to provide both line-of-site and beyond-line-of-site Command, Control, Communications, Computer and Intelligence (C4I) capabilities to the war fighters. This family of radios aims to cover an operating spectrum from 2 MHz to 2 GHz and will be capable of transmitting voice, video and data. The radios must be interoperable, affordable and scalable.

One of the most critical decisions by JPO in the system engineering process of JTRS was to mandate the adoption of the SDR technology. SDR forms the framework upon which the entire JTRS design will be assembled. The essential premise behind the JTRS project is to leverage commercial-off-the-shelf (COTS) and software defined radio (SDR) technologies to produce a new family of tactical radios that are multi-functional with advanced wireless data networking capabilities to meet the needs of modern information warfare.

Based upon an open architecture framework, JTRS allows multiple vendors to participate in the design of the hardware platform and the development of software functions.

1. Digital Modular Radio (DMR)

The DMR is a Navy effort to procure a modular, scalable, software programmable, re-configurable digital radio to satisfy near term Ultra High Frequency (UHF) and Very High Frequency (VHF) communications requirements until the JTRS is available [3]. In a DMR, many components traditionally realized in hardware are implemented in software. Thus, a DMR is a SDR.

DMR covers a frequency band from 2 MHz to 2 GHz and is capable of operating on four RF channels simultaneously. The RF channels can operate on any combination of supported modulation waveforms, and new waveforms may be added via software upgrades. DMR is based on an open systems architecture, which is designed to allow any information channel to be connected to any other information channel. This enables features such as bridging between RF waveforms, simulcasting and routing data between wireline and wireless channels [3].

DMR is currently under development. One of its desired features is to act as a line-of-site (LOS) wireless networking radio, capable of transmitting voice, video and data to/from any mobile node (ship, tank, helicopter, aircraft, etc.) in a MANET. These features are envisioned to offer automatic network formation and maintenance, automatic relaying to extend LOS range, independent encryption and *QoS guarantees* to each user service (voice, data, video), robustness against jamming, denial of service, and spoofing attempts, and robustness to random node failures.

DMR is intended to allow for prototyping of various network algorithms and implementation of the eventual joint wireless standards as established by JTRS.

C. MOBILE NETWORK PROTOCOLS FOR JTRS

One of the technical challenges that must be addressed in developing JTRS compliant implementations is the ability to network the radios in a mobile ad-hoc environment. In addition to supporting legacy network protocols, the JTRS architecture aims to support emerging wideband networking capabilities for voice, data, and video. The new war fighting paradigms demand mobile, flexible networks that automatically adapt to the war fighter's needs. These paradigms, as expressed in [1], require mobile networking capabilities far beyond what is possible with the currently available technology. As a result, JTRS networking protocols must support a variety of services, including automatic neighbor and link quality discovery, automatic neighbor reconfiguration, *QoS guarantees*, *precedence and priority marking*, and automatic relaying of traffic.

JTRS has motivated the industry and the researchers to develop a networking structure that is more robust and responsive than the current IP protocol stack largely used in the fixed IP networks. The current areas of research include: routing protocols, *quality of service (QoS)*, medium access control (MAC), low power design, mobility management, and security. The development and implementation of mobile networking protocols and its mapping into the technical architecture of the next generation of radios is one the most complex technical challenges to the JTRS effort [1]. In support of this effort, this thesis will address the *QoS issues* in MANETs.

III. MOBILE AD-HOC NETWORK (MANET)

This chapter provides the necessary background on MANETs and presents the details of the protocol stack required to provide QoS capability in mobile nodes. The first section looks at the characteristics and applications of MANETs. In the following sections, the MANET protocol stack is described in a layer-by-layer fashion. From the physical to the application layer, each section presents the main issues that must be addressed to provide QoS in MANETs.

A. BACKGROUND

The basic idea behind mobile ad-hoc networking (MANET) was first championed by the DARPA packet radio networks or mobile packet radio in the 1970s. Since then, the technology has evolved significantly and applicable commercial radio technologies, such as IEEE 802.11 Wireless Local Area Network (WLAN) standard, have begun to appear. Previously, most of the interest in MANETs has been from the military side, and commercial interest is a recent phenomenon due to the demand for mobile computing [4].

Unlike Mobile IP or cellular networks where there is always reliance on pre-established fixed backbones, MANETs are intended to be autonomous and function independent of any fixed infrastructure, with the exception of a few possible gateways to provide access to a larger network or the Internet. Figure 3.1 shows the conceptual difference between two layers of mobile networks.

The Mobile IP layer consists of hosts temporarily attached to routers on a fixed network. Hosts in this layer are one hop away from a fixed router, and their connections may be wired or wireless. In cellular networks, although they are not IP based, mobile nodes and base stations are also only one hop away from each other. On the other hand, the mobile nodes in a MANET do not require any routing support as they form their own mobile infrastructure in parallel to the fixed one. The focus of this thesis is on the issues related to MANETs, where mobile nodes can be separated by more than one hop and hosts and mobile routers are distinguished only logically.

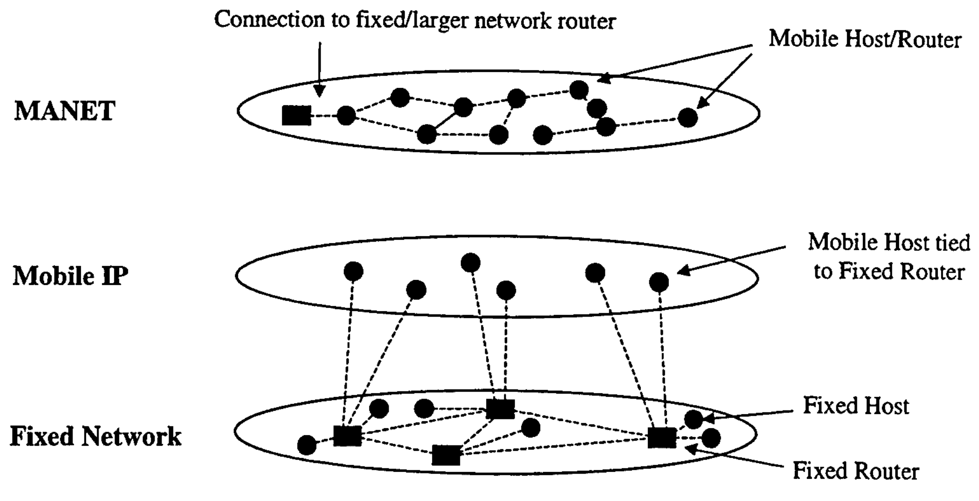

Connection to fixed/larger network router
Mobile Host/Router
MANET
Mobile IP
Mobile Host tied to Fixed Router
Fixed Host
Fixed Network
Fixed Router

Figure 3.1 Mobile IP (Mobile Host/Fixed Router) and Mobile Ad-hoc (Mobile Host/Mobile Router) Networks (After Ref. [4]).

In principle, in the case of MANET, a single mobile node can simultaneously perform two roles during traffic exchange: host and router [5]. In the role of a host, a node can be the source or the destination of different types of traffic. As a router, it is responsible for relaying packets to the intended destinations and to maintain the routing paths. If a MANET has interfaces with a fixed network infrastructure, it typically operates as a "stub", carrying traffic that is either sourced or terminated within the MANET, but not permitting external traffic to "transit" through the stub network. The physical layer among MANET mobile nodes is assumed to be wireless, and it will be further explored in Section C.

Essentially, a MANET can be treated as an autonomous system of mobile nodes [5]. In terms of applications, it is well suited for enabling peer-to-peer operation in mobile, forward-deployed military networks or for situations where it is not feasible to provide the necessary fixed infrastructure, like in emergency situations (search-and-rescue, fire fighting, etc.). The management requirements for organizing and controlling the network are distributed among the nodes themselves.

Networking in MANETs presents new challenges since both the users and the infrastructure are in constant transition. As shown in Figure 3.2, forward-deployed military MANETs are envisioned as relatively large, dynamic, and heterogeneous networks with several mobile nodes per domain. Depending on the application scenario, the nodes may be located in manned or unmanned aircraft, ships, trucks, cars, and perhaps even carried by people as handheld devices or manpacks.

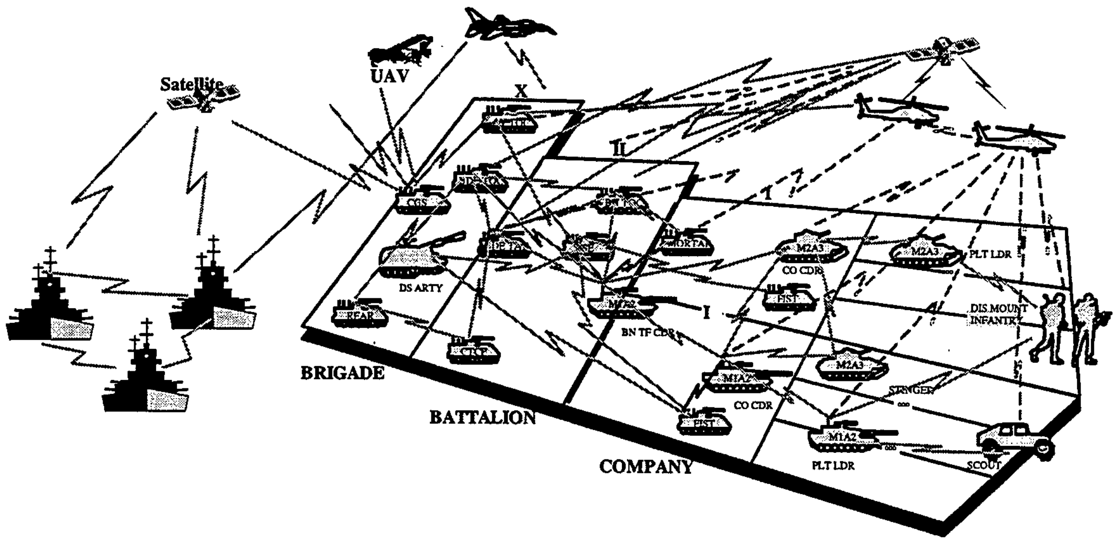


Figure 3.2 Typical MANET Military Application Scenarios.

MANETs have four unique characteristics that differentiate them from the fixed multi-hop networks [4]: dynamic topology, bandwidth constraints, energy constraints and limited physical security. The first characteristic implies that nodes can move arbitrarily, changing the topology randomly and rapidly depending on the scenario. The second means that wireless links have significantly lower capacity than wired links, which intensifies congestion problems and requires special consideration for the bandwidth-delay characteristics. Also, the effective throughput of wireless communication channels is often much less than a radio's maximum transmission rate due to multiple access, fading, noise, and interference effects. The third refers to the fact that some or all nodes in a MANET may rely on batteries for energy, making power conservation a critical design criterion. Finally, wireless networks are generally more prone to information and physical security threats than are fixed, hardwired networks. Thus, security threats must be taken into account in the design and selection of the protocols and in the development of applications.

B. PROTOCOL STACK

The Internet Engineering Task Force (IETF) MANET Working Group has taken the lead in the development of a reference protocol stack (depicted in Figure 3.3) for the mobile nodes. The protocols that support mobile networking must be compatible with the TCP/IP suite [5]. In this thesis, the emphasis will be on the wireless side (see Figure 3.3) of the mobile node's protocol stack.

The traffic can be generated in two different ways. First, it can be generated by a given mobile node itself. Second, it can be generated by another node (in which case this node is simply relaying the packet traffic) or the traffic might have originated in a fixed node and was passed along to a mobile node via the wired side of the protocol stack. In both cases, the MANET routing protocol updates the IP header fields of the packet and the physical address of the "Radio-Frequency (RF) Ethernet" of the next-hop mobile node. The packet is then forwarded to the wireless MAC protocol, which is finally sent via the wireless RF network interface when the channel becomes available.

The main focus of the MANET working group is the development and standardization of routing protocols that reside at the network-layer. The MANET routing protocols must provide effective operation over a wide range of mobile network scenarios, support traditional connectionless IP service and react efficiently to topological changes and traffic demands while maintaining effective routing in a mobile networking context. The standardization process of the routing protocols must guarantee compatibility and interoperability with Internet standards in the other layers, both existing and under development. Backward compatibility with the traditional wired IP routing is an upfront requirement and extends the existing fixed infrastructure [5].

As displayed in Figure 3.3, the QoS issues are present across the protocol layers. This thesis aims to emphasize only the upper-layer (application) QoS related problems while highlighting how different aspects of other layers influence the QoS. The following sections will cover the main issues and the assumptions made in this thesis in each layer of the envisioned MANET protocol stack in order to create an appropriate model for simulation in typical military scenarios.

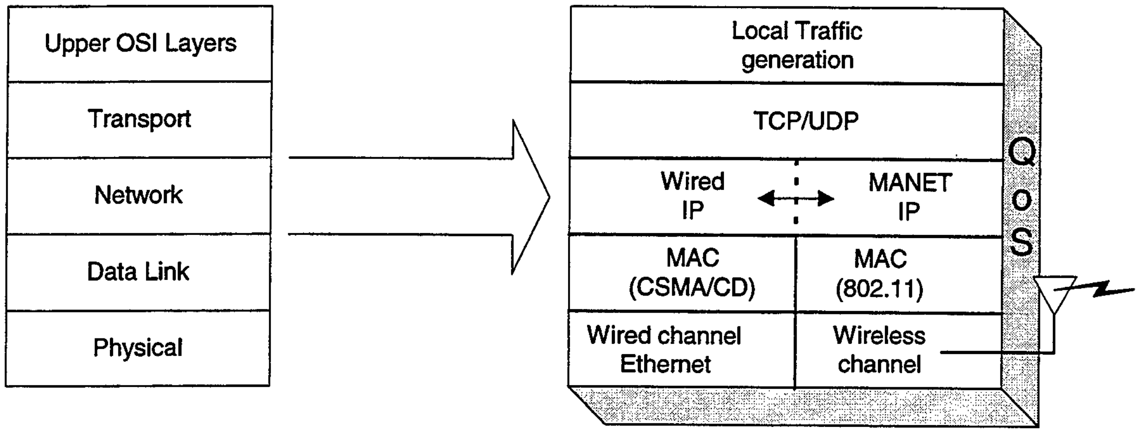


Figure 3.3 Mobile Node Protocol Stack.

C. PHYSICAL LAYER

Unlike wired channels that are stationary and predictable, wireless channels are time varying and do not offer easy analysis. Modeling the wireless physical layer has been one of the most difficult challenges of mobile radio systems design, and is often done empirically, based on measurements made specifically for an intended communication system or spectrum allocation [6]. The main issues are related to the propagation phenomena of radiowaves, which is dependent on several factors such as: frequency of operation; transmitter and receiver characteristics; antenna gains and coverage patterns; distance between sending and receiving nodes; height of the nodes and obstacles between them; terrain and environment features; and presence of noise and multipaths, among others.

The weight given to each of these factors will be a function of the specific scenario in which the mobile ad-hoc network is established. While factors, such as frequency and receiver characteristics, are likely to remain constant during traffic exchange, others, such as noise and distance, are likely to vary with time and location.

Several mathematical propagation models exist in the literature to address these issues [6]. By taking the appropriate factors into consideration, all the models aim to predict, with certain probability, the average received signal power of a selected traffic signal under a specific environment at a given distance from the transmitter.

At the physical layer of each mobile node device, there is a receiving threshold level. A traffic packet is considered to be received if its signal power is above the receiving threshold. However, often times, although the packet is received, it may contain errors due to channel noise effects (Gaussian noise, slow and fast fading, jamming, etc.), which lead to erroneous packet reception.

Clearly, for tactical communications, good performance over a wide range of channel conditions is essential. Thus, to minimize the effects introduced by channel noise, several approaches can be used, such as: forward error correction (FEC) by coding the information (e.g., convolutional, cyclic, block codes); retransmission of the erroneous packets by using automatic repeat request (ARQ); special modulation schemes; and smart antennas at the receivers. These error-correction or error-minimization approaches are in general associated with link quality monitoring schemes and are applicable based on the status of the link. As link conditions deteriorate, FEC or ARQ can potentially be applied. The down side of using FEC or ARQ is the overhead in a bandwidth constrained environment, which can reduce efficiency (bps/Hz).

Although it is known that the propagation phenomena of electromagnetic waves will have a considerable effect on the overall performance of a MANET and ultimately on the desirable traffic QoS, the analysis of these effects and error-correction methods are outside the scope of this thesis. Here, a simple physical layer propagation model based on the two-way reflection approach for a specific UHF frequency range (300 MHz – 3GHz) is adopted.

1. Simple Physical Layer Model

The receiving power using a free space propagation model for a single and clear (unobstructed) direct line-of-sight (LOS) path between two communicating mobile nodes, as defined by the well-known Friis formula, is given by:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{3.1}$$

where P_t is the transmitter power, d is the distance between the nodes, G_t and G_r are the transmitter and receiver antenna gains, respectively, L is the system loss factor not related to propagation ($L \geq 1$), and λ is the wavelength. Friis equation shows that the received power falls off inversely to the square of the separation distance (i.e., 20 dB/decade).

In a UHF mobile radio channel, a single direct path between transmitter and receiver is not the only physical means of propagation. Consequently, in most cases, the free space propagation model is innacurate when used alone. It is shown [6] that the two-ray reflection model provides a more reasonable model than the free-space model, especially at longer distances. Figure 3.4 displays the geometry of this model.

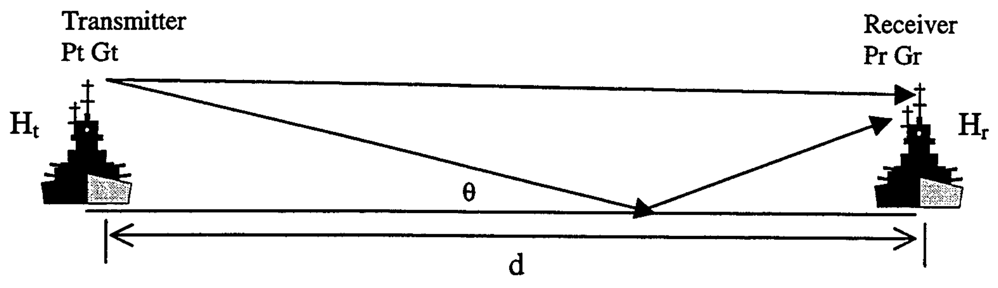


Figure 3.4 Two-way Reflection Propagation Model in an UHF Frequency Range.

When the antenna heights (H_t, H_r) and the range (d) are such that the Earth can be considered flat, the grazing angle for values below 10° is given by:

$$\theta = \tan^{-1}(H_t+H_r)/d \tag{3.2}$$

In this case, it can be shown [6] that the surface reflection coefficient approaches –1. The received power is then given by:

$$P_r(d) = \frac{P_t G_t G_r H_t^2 H_r^2}{d^4 L} \tag{3.3}$$

For most shipboard installations, where the antenna heights are on the order of tens of meters and the distances are on the order of kilometers, these assumptions are reasonable. From equation (3.3), at large distances ($d >> \sqrt{H_t H_r}$), the received power falls off inversely to the fourth power of the separation distance, which is a more rapid path loss (40 dB/decade) than is experienced in free space. Also, it is important to notice that at these ranges, the received power becomes independent of frequency.

On the other hand, the two-ray model does not give good results for short distances due to the oscillation caused by the constructive and destructive combination of the two rays (the reflection coefficient is no longer equal to –1). In this case, the first ellipsoid of Fresnell will not touch the surface, and hence the free-space model can be used. Therefore, a cross-over distance d_c must be calculated so that when $d < d_c$, equation (3.1) is used, and when $d > d_c$, equation (3.3) is used, where d_c is given by:

$$d_c = (4\pi H_t H_r) / \lambda \tag{3.4}$$

where λ is the wavelength. Figure 3.5 shows the behavior of the average received power as a function of the separation distance between the mobile nodes, according to the two-ray propagation model.

For the sake of simplicity, fading, i.e., the rapid fluctuations of the received signal strength over very short distances (few wavelengths) or short time durations is not considered. When a mobile node moves away from the transmitter over larger distances (hundreds of meters), the local average received signal will gradually decrease and, moreover, will fluctuate less. It is this average signal level, not the instantaneous signal level, that will be predicted and used by the physical layer model in this work.

D. MEDIUM ACCESS CONTROL (MAC) LAYER

The MAC layer protocols to be used in MANET are required to provide the following basic services for the upper layers [5]:

- Link status sensing and neighbor discovery;
- Reliable, in-order control packet delivery;
- Link and network layer address resolution and mapping; and
- Security authentication.

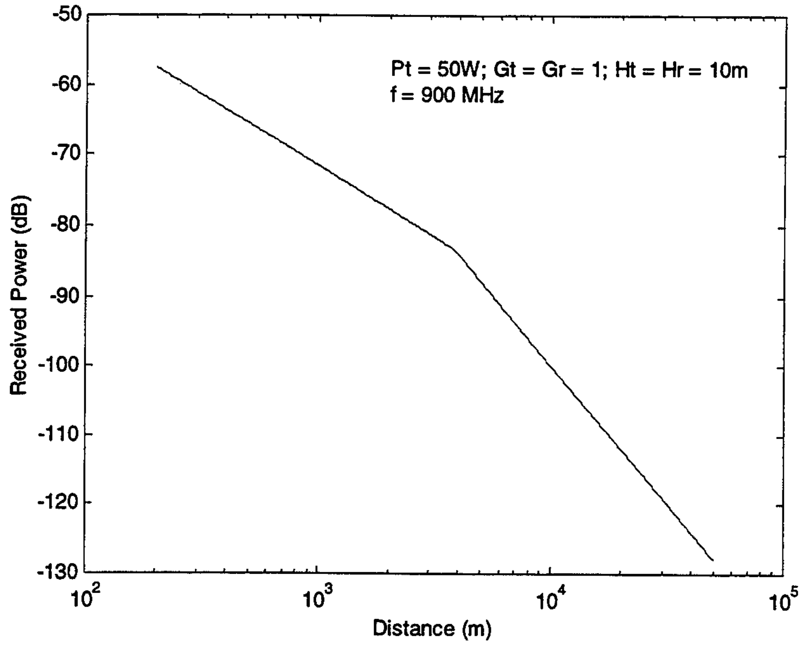


Figure 3.5 Average Received Power versus Range.

Link status sensing and neighbor discovery are closely related to the "hidden terminal" problem. As depicted in Figure 3.6, the broadcast nature of the wireless medium allows node B to communicate with both nodes A and C. However, assuming that the dotted circles are the ranges of each node, A and C cannot directly communicate with each other, i.e., they are "hidden" from each other. When B transmits to C, D cannot detect the transmission using the carrier sense mechanism, and hence, if D also transmits, a collision will occur at node C.

A simple solution suggested in the IEEE 802.11 standard to avoid this problem is based on request-to-send (RTS) and clear-to-send (CTS) messages. When node B wants to transmit a packet to node C, it first sends an RTS to C. On receiving RTS, node C responds by sending a CTS message, provided node C is able to receive that packet. When a node, such as D or A, overhears a CTS message, it keeps quiet for the duration of the transfer, which is known because it is included in both RTS and CTS messages. Thus, the area covered by the transmission range (indicated by the vertical bars in Figure 3.6) of both the sender (B) and the receiver (C) is reserved for the duration of the data transfer from B to C, to prevent collisions.

Reliability in data transfer is achieved by means of acknowledgements (ACK). When node C receives a data packet from node B, node C sends an ACK to B. If node B fails to receive the ACK message, it retransmits the data packet.

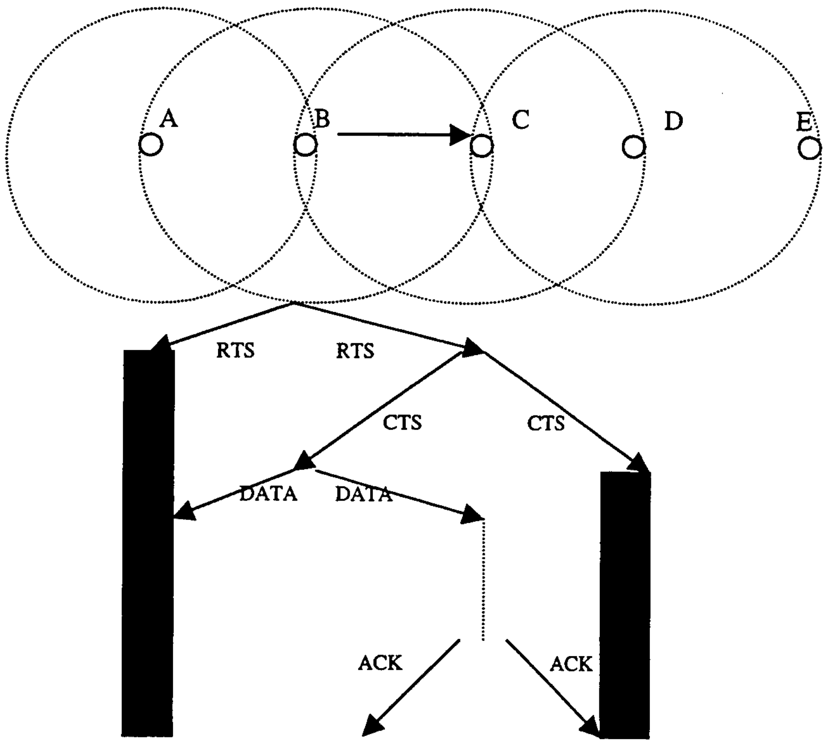


Figure 3.6 Hidden Terminal Problem (After Ref. [7]).

1. IEEE 802.11

IEEE802.11 [8] is a commercially adopted MAC layer standard for wireless communication within local area networks (LAN). IEEE802.11 adopts a Distribution Coordination Function (DCF), which uses RTS-CTS exchange to overcome the "hidden terminal" problem and ACKs to achieve reliability. It also uses a Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) scheme based on dynamically selected backoff intervals.

Due to the DCF algorithm, when a specific node *i* wishes to transmit a packet, it chooses a "backoff" interval equal to B_i slots, where B_i is randomly chosen from the uniform interval *[0, C_w]*. C_w is the so-called contention window and it is reset to a value C_{wmin} at the start, and after each successful transmission of a data packet by node *i*. If the transmission medium is not idle, node *i* waits until it becomes idle. While the medium is idle, B_i is decremented by 1 after each slot time. If the medium becomes busy while B_i is non-zero, then B_i is frozen while the medium is busy. B_i is decremented again when the medium becomes idle. Eventually, when B_i reaches 0, node *i* transmits a RTS for the intended destination of the packet. The destination node, on receiving the RTS, sends a CTS packet. Finally, on receiving CTS, node *i* transmits the data packet. However, it is possible that two nodes may choose their back-off intervals such that they both transmit their RTS simultaneously, causing a collision between them. In this case, one of the nodes will not receive a CTS and will then double its C_w, will pick a new B_i uniformly distributed over *[0, C_w]*, and will repeat the above procedure.

RTS-CTS and ACK messages, although serving useful purposes, can potentially waste a large amount of network capacity by reserving the wireless shared medium over a large area (see Figure 3.6). This has a direct impact on the TCP and UDP traffic performance in multi-hop scenarios as will be shown later in Section D. Also, the use of back-off intervals to avoid congestion results in unfairness when one node backs off more than another node, i.e., one node may transmit several packets before the other node transmits its first packet.

Several modifications of the IEEE802.11 have been proposed to address the MAC fairness issue. Most of them provide fairness by trying to control the bandwidth used by each node, which invariably causes an increase in the back-off intervals. This further results in a trade-off between fairness and throughput, since larger back-off intervals mean better fairness but less throughput. Although these modifications seem to directly impact the QoS of a wireless LAN, the present results are still not proved to be effective in MANET, where the shared nature of the wireless channel in a mobile multi-hop environment and the hidden terminal problem creates difficulty in determining a suitable definition of fairness [9].

2. Other MAC Protocols

Reference [10] presents a comparison of other types of MAC protocols for wireless networks. Frequency Division Multiple Access (FDMA) is a fairly simple MAC protocol that assigns each network member a unique carrier frequency. There is no need for any channel contention, which potentially reduces delay and complexity of the protocol. On the other hand, the waste of bandwidth when a network member is allocated a frequency but has no data to send along with the scalability problem of having to add a receiver at each node when a new member joins the network are some of the major disadvantages of this scheme.

Time Division Multiple Access (TDMA) is based on pre-assigned time slots for each of the network members and uses only a single carrier frequency for all nodes. In this case, the scalability problems are solved by reducing the transmission range or the data rate [10]. However, there is still waste of bandwidth if the node has no data to send during its assigned slot. Also, the channel access delay is potentially larger than in FDMA. Compared to CSMA/CA, the TDMA approach offers the most efficient means of communications. However, in situations where nodes are mobile, management of TDMA networks can be quite complex [10].

Demand Assigned Multiple Access (DAMA) and other variations of the basic TDMA/FDMA have also been proposed to mitigate some of the major drawbacks of each approach.

Table 3.1 shows that as the nodes move apart, the signal-to-noise ratio at the receiver (E_b/N_0) decreases up to a point where the bit error rate (BER) would reach unacceptable levels. Decreasing the data rate, if possible, allows longer ranges. From Table 3.1, it can be concluded that the selection of the best MAC approach to be used in a MANET is usually driven by an appropriate link-layer signal management algorithm that can control the trade-off between bandwidth (throughput) and range requirements [10].

In this thesis, IEEE802.11 with CSMA/CA was adopted as the MAC layer protocol for the mobile ad-hoc network protocol stack, although variations of this standard or a FDMA/TDMA based protocol may be tried in the near future. It is important to notice that, by adopting IEEE802.11, a protocol that is tailored for small areas and stub networks must be adapted for use in a network environment encompassing larger geographical areas.

(kbps)	10nmi	15nmi	20nmi	25nmi	30nmi
4608	24 (dB)	16 (dB)	8 (dB)	1 (dB)	-6 (dB)
1536	29	21	13	6	-1
576	33	25	17	10	3
64	43	35	27	20	13

Table 3.1 Maximum Available E_b/N_0 at 100W Transmitting Power (From Ref. [10]).

E. NETWORK LAYER

The network layer is responsible for optimal path determination and packet forwarding functions. Routing protocols are responsible for determining the routes in the network while the routed protocols (such as IP) perform the packet forwarding functions. The network layer (see Figure 3.3) utilizes the link status, in-order packet delivery and address resolution/mapping services of the MAC layer. This Section reviews the various routing protocols reported in the literature. Three of the MANET routing protocols being considered by the IETF MANET Working Group will be introduced. Optimum performance is the main goal of a routing protocol. To evaluate a routing protocol, we need appropriate quantitative and qualitative metrics [5].

As qualitative metrics, demand-based operation, proactive operation and sleep period operation are the most significant. Demand-based operation consists of letting the routing algorithm adapt to the traffic pattern on a need basis, in order to more efficiently utilize energy and bandwidth resources at the expense of increased route discovery delay. The proactive operation is to be considered in situations where the latency of the demand-based operation is unacceptable. The sleep period operation is required to guarantee energy savings for the nodes and hence is desirable to be included in a MANET routing protocol [5].

The most significant quantitative metrics are the end-to-end data throughput and delay, route acquisition time (particularly important in "on-demand" routing algorithms), and efficiency (as determined by the overhead due to the control traffic). Also, it is important to consider the networking context to evaluate the performance of a routing protocol. In MANETs, the size of the network, the average number of neighbors, the speed with which the network topology can change, the link capacity and the traffic patterns (uniform, bursty, non-uniform), among others, should be considered [5].

As mentioned earlier, the routing algorithms and much of the protocol suite need to be redesigned to function efficiently and effectively in the MANET environment. Conventional routing protocols associated with fixed IP networks are unable to meet the unique requirements of a MANET due to considerable overhead and slow reaction to topological changes [5].

1. Conventional Routing Protocols

Conventional routing protocols use either distance vector or link-state algorithms to determine the optimum path to the destination [11].

Distance vector algorithms require each router to maintain a table with routes to all possible destinations along with an associated metric. Each router periodically produces broadcasts of this information to neighboring routers. The routers review their tables and neighbor updates to produce an internal routing table. The overhead associated with this technique is constant, regardless of the amount of topology changes (node movement) in the network. This type of routing is closely associated with the Distributed Bellman-Ford routing algorithm. A version still being used today is the Router Information Protocol (RIP). This protocol is table-driven and each router maintains a table on how to reach all possible destinations in the network. For each entry, the next hop/router to the destination is stored along with a metric to reach the destination. The metric can be based on distance, total delay, or the cost of sending the message [12]. Each node shares its internal information with its neighbors to achieve optimum routing. After a prolonged period, each node will have a consistent table to reach all other nodes.

Link-state routing is based on Dijkstra's algorithm. Network topology information is again used to make routing decisions, but in this case, the algorithm is driven by changes in the link status of the nodes. Each router actively tests the status of its link to each of its neighbors and sends this information to its other neighbors, which then propagate it throughout the autonomous system. Then, each router takes this information and builds a complete routing table.

Both algorithms allow a host to find the next hop neighbor to reach the destination via the optimum (shortest) path; however, the overhead associated with these techniques is considerable and exhibits slow convergence in the presence of intensive topological changes, common in MANETs.

In [12], the authors conducted a simulation study to analyze RIP in a MANET and highlight its shortfalls. According to the study, the periodic routing updates do not allow RIP to scale well to large networks. The study revealed an increase in overhead directly proportional to node mobility. In MANETs, excessive overhead leading to inefficient use of the limited wireless bandwidth is unacceptable. Clearly, more responsive protocols are needed to meet the requirements of MANETs.

2. Classification of MANET Routing Protocols

Due to different approaches taken by the researchers in the area, MANET routing protocols can be classified in several different ways. They can be table-driven versus on-demand, proactive versus reactive, symmetric versus asymmetric, and unicast versus multicast.

Table-driven versus on-demand is the most common classification [13]. Table-driven algorithms can be interpreted as adaptations of the conventional distance vector and link-state techniques. The routing updates, types of tables, distributions, and techniques have been adapted to increase efficiency in MANET. In contrast, on-demand protocols attempt to reduce overhead and are more responsive to MANET by having the sender node dictate requirements. On-demand means that routes are created on an as-required basis by the sender node. This "lazy routing" approach reduces overhead by eliminating unnecessary periodic updates and by letting the changes in the network dictate overhead [13]. Current routing updates are not maintained at every node, because the routes are created on an as-required basis and expire with a time metric.

Table driven protocols are also known as proactive, i.e., the routes are determined independent of the traffic pattern so that when a packet needs to be forwarded, the route is already known and can be immediately used. On the other hand, on-demand protocols are reactive because routes are established and maintained only if needed. There are also hybrid protocols, such as Zone Routing Protocol (ZRP) and Ad Hoc On-Demand Distance Vector Routing (AODV), which aim to combine proactive and reactive behavior, according to the context.

The routing protocols may also be classified according to the capabilities of the nodes. Symmetric protocols assume that all the nodes have the same responsibilities and capabilities. Asymmetric protocols, such as Core-Extraction Distributed Ad-Hoc Routing Protocol (CEDAR), may assume that the transmission ranges or the battery life at different nodes may differ, or only some nodes can route packets and act as leaders (cluster heads) of nearby nodes.

Dynamic Destination-Sequenced Vector (DSDV), Wireless Routing Protocol (WRP), Global State Routing (GSR), Fisheye State Routing (FSR), Hierarchical State Routing (HSR), Zone-Based Hierarchical Link State Protocol (ZHLS), and Cluster-Head Gateway Switch Routing Protocol (CGSR) are examples of table driven or proactive MANET protocols. Cluster Based Routing Protocol (CBRP), Dynamic Source Routing Protocol (DSR), Associative Based Routing (ABR), Signal Stability Routing (SSR) and Temporally Ordered Routing Algorithm (TORA) are examples of on-demand or reactive MANET protocols.

References [14] and [15] provide a detailed simulation performance analysis of AODV and ZRP, respectively. In the following Sections, we will present an overview of three of the most widely studied protocols, keeping in mind that no single protocol works well in all environments. The intention of this coverage is to provide an understanding of how the routing protocols affect QoS in MANETs.

a. Zone Routing Protocol (ZRP)

The ZRP protocol, developed by Haas and Pearlman [16], incorporates a localized zone approach to routing. The approach is to incorporate a hybrid protocol that exploits the benefits of both a reactive and a proactive protocol. ZRP limits the scope of the proactive procedure to only the node's local neighborhood. Global searches for non-local nodes then use an efficient reactive scheme that queries only selected network nodes, as opposed to querying all of the network nodes.

As shown in Figure 3.7, each mobile node has a proactive routing zone around it that is dictated by an adjustable zone routing radius. The zone routing radius is directly related to hop counts from the node. In Figure 3.7, nodes D, C, F, B, and E are in Zone A with a zone routing radius of 2. Routes outside the zone are determined by an on-demand protocol query which "bordercasts" the out-of-zone query to the peripheral nodes (D, F, and E), which in turn leverage the zone structure of the network to reduce query detection time. The intent behind this MANET routing approach is to utilize the routing knowledge in a localized region and obtain a route to a distant node on-demand. Intrazone Routing Protocol (IARP), Interzone Routing Protocol (IERP), and routing optimization are the main algorithms implemented within ZRP and explained in detail in [15].

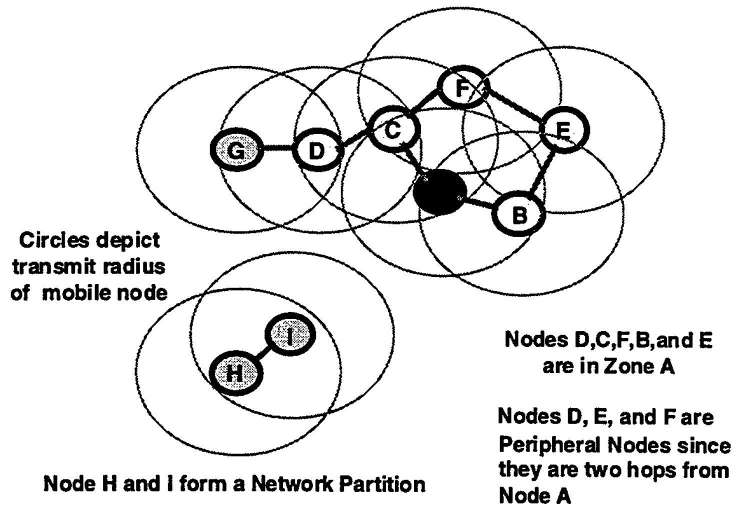
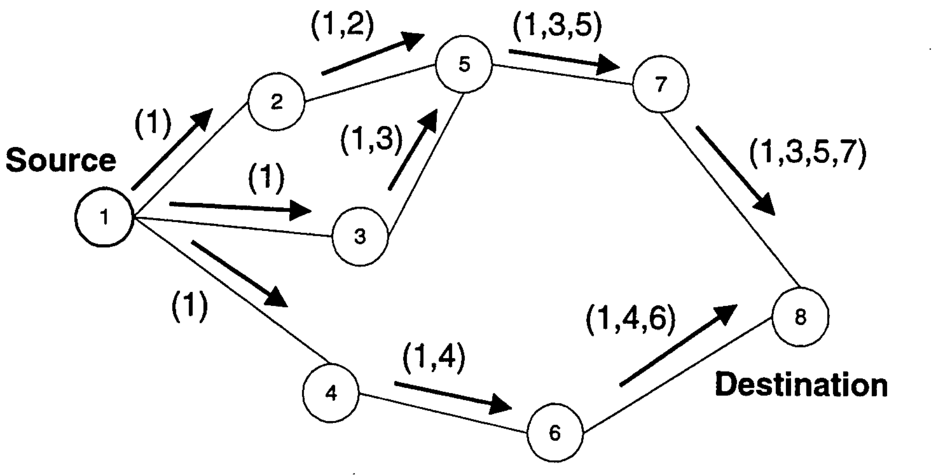

G
D
C
F
E
B
Circles depict transmit radius of mobile node
I
H
Nodes D,C,F,B,and E are in Zone A
Nodes D, E, and F are Peripheral Nodes since they are two hops from Node A
Node H and I form a Network Partition

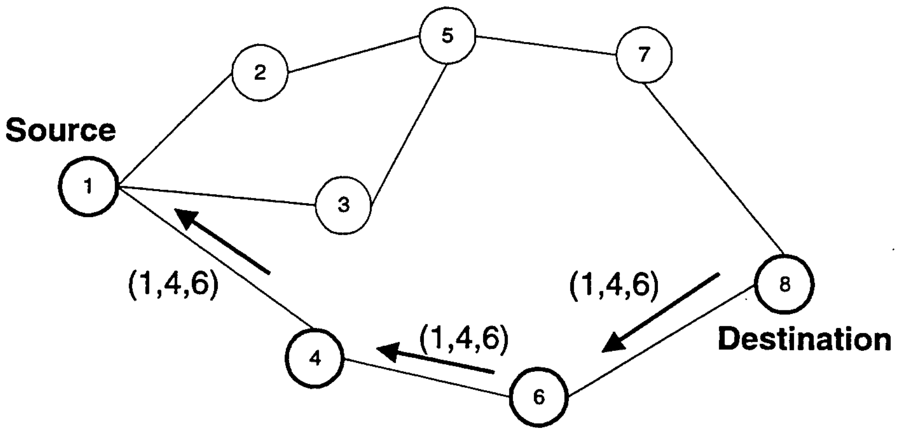
Figure 3.7 ZRP Example with a Zone Routing Radius of 2 (From Ref. [15]).

b. Dynamic Source Routing (DSR)

Broch, Johnson and Maltz developed Dynamic Source Routing (DSR) in 1998 [17]. DSR is a pure on-demand protocol based on source routing. The source specifies the complete path to the destination in the packet header and each node along this path simply forwards the packet to the next hop indicated in the path. DSR utilizes a route cache approach, where the source routes acquired by the nodes are cached (see Figure 3.8.a).



a) Building route record during route discovery



b) Propagation of route reply with source route record

Figure 3.8 Creation of Route Cache in DSR (After Ref. [17]).

A source first checks its route cache to determine the route to the destination. If a route is found, the source uses this route. If a route is not found, the source uses a route discovery protocol to discover a route. In route discovery, the source floods a query packet or route request packet (RREQ) through the ad hoc network. Either the destination or another host that can complete the query from its route cache returns a route reply (RREP) (see Figure 3.8.b). Each query packet has a unique identifier (ID).

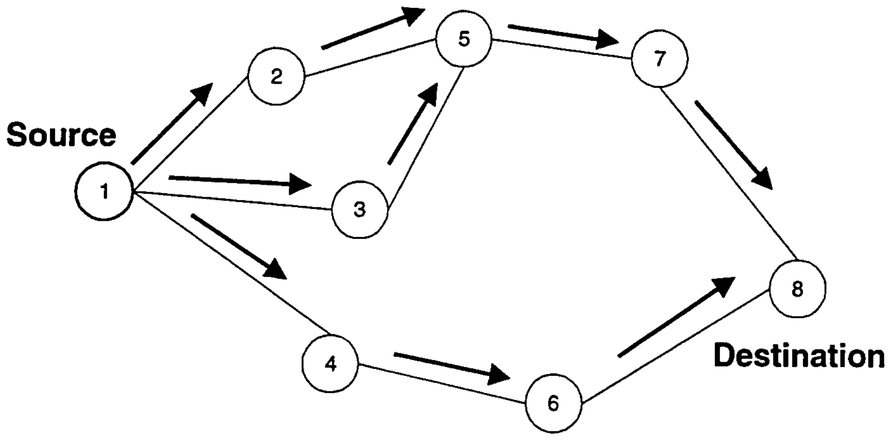
When receiving a query packet, if a node has already seen this ID (a duplicate ID), or it finds its own address already recorded in the list, it discards the copy and stops flooding. Otherwise, it appends its own address on the list and broadcasts the query to its neighbors. If a node can complete the query from its route cache, it may send a reply packet to the source without propagating the query packet further. Any node participating in route discovery can learn routes from passing data packets and gather this routing information into its route.

In DSR, no periodic control messages are used for route maintenance, and there is little or no routing overhead when a single or few sources communicate with infrequently accessed destinations. The on-demand, flooding-based nature of DSR's route discovery process eliminates the need for periodic router advertisement and link-status packets, which significantly reduces the overhead of DSR during periods when the network topology is stable.

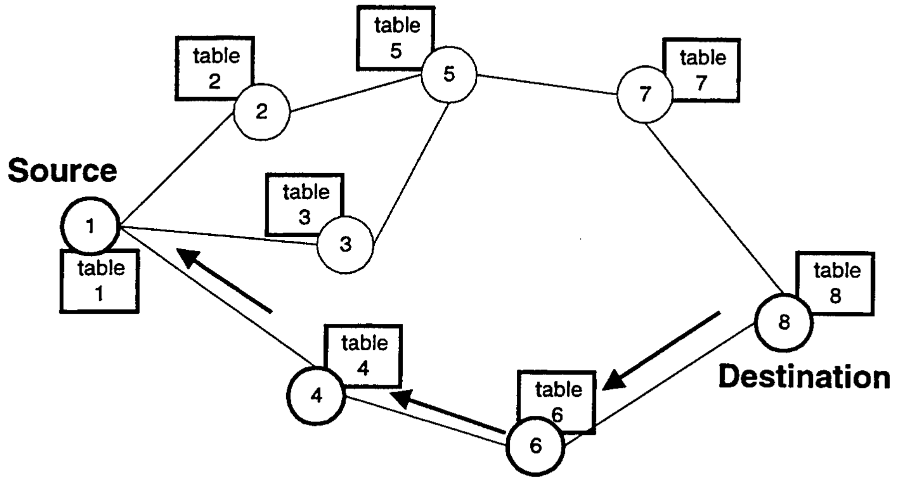
c. *Ad-Hoc On-Demand Distance-Vector Routing Protocol (AODV)*

Perkins and Hoyer developed the AODV routing protocol in 1999 [18]. It is considered to be a hybrid protocol, because it combines features of a pure on-demand protocol (DSR) with a table-driven protocol (DSDV). Specifically, AODV uses the same features as DSR for route discovery, and from DSDV it uses the hop-by-hop routing, sequence numbers, periodic update packets and loop free routing [18].

The process of finding a path to the destination is quite similar to DSR. The source node first broadcasts a route request packet (RREQ) (See Figure 3.9.a). Nodes receiving this packet update their information for the source node and set up backwards pointers to the source node in the route tables. In addition to the source node's IP address, current sequence number, and broadcast ID, the RREQ also contains the most recent sequence number for the destination of which the source node is aware. A node receiving the RREQ may send a route reply (RREP) if it is either the destination or if it has a route to the destination with a corresponding sequence number greater than or equal to that contained in the RREQ. If this is the case, it "unicasts" a RREP back to the source. Otherwise, it rebroadcasts the RREQ. Nodes keep track of the RREQ's source IP address and broadcast ID. If they receive a RREQ, which they have already processed, they discard the RREQ and do not forward it.



a) Propagation of route request packet (RREQ)



b) Path taken by the route reply (RREP) packet

Figure 3.9 Route Discovery in AODV (After Ref. [18]).

As the RREP propagates back to the source, nodes set up forward pointers to the destination (see Figure 3.9.b). Once the source node receives the RREP, it may begin to forward data packets to the destination. If the source later receives a RREP containing a greater sequence number or contains the same sequence number with a smaller hop count, it may update its routing information for that destination and begin using the new best route.

AODV maintains routing tables (see Figure 3.9.b) at the nodes so that data packets do not have to contain routes in its headers, which could increase the overhead when data packets are small. Similar to DSR, as long as the route remains active, AODV will continue to maintain the route. A route is considered active as long as there are data packets periodically traveling from the source to the destination along that path. Unused routes expire even if the topology does not change. Once the source stops sending data packets, the links will time out and eventually be deleted from the intermediate node routing tables. If the source moves, then it can reinitiate route discovery to the destination. If one of the intermediate nodes moves, then the moved nodes' neighbor realizes the link failure and sends a link failure notification to its upstream neighbors until it reaches the source, upon which the source can reinitiate route discovery if needed. If a link break occurs while the route is active, the node upstream of the break propagates a route error (RERR) message to the source node to inform it of the now unreachable destination(s). After receiving the RERR, if the source node still desires the route, it can reinitiate route discovery.

d. DSR vs AODV

By extensive use of simulation under different scenarios, [19] establishes a detailed performance comparison between DSR and AODV. It presents the relative merits of the aggressive use of source routing and caching in DSR and the more conservative routing table and sequence number driven approach in AODV.

When DSR is used in large networks (more than 20 active source nodes) the packet header size grows with route length. The resultant overhead implies degradation in performance. Also, if the network topology changes a lot (higher mobility cases), a cached route in DSR may become invalid, forcing the sender host to try several stale routes before finding a usable one. On the other hand, in small (less than 20 nodes) and lower mobility networks, the advantage of DSR can be significant because the above mentioned problems will not be in evidence and also because route caching can potentially speed up route discovery and reduce propagation of routing requests [19].

AODV outperforms DSR, in terms of throughput and end-to-end delay in more "stressful" situations (higher mobility and higher traffic load). On the other hand, DSR outperforms AODV for low loads (less sources) with small (less than 20) number of nodes.

In this thesis, simulations are developed to test the QoS issues under typical tactical scenarios, which are generally composed of no more than 20 nodes, and the relative mobility between nodes is generally low. Thus, DSR is adopted as the routing protocol in the proposed model.

3. Impact of Routing Protocol on QoS

From the review of the main features of ZRP, DSR and AODV, it becomes clear that delivering QoS in mobile networks is intrinsically tied to the performance of the underlying routing protocols [20]. In MANET, efficient routing means efficient tracking of the network changes without causing too much overload in the bandwidth constrained environment.

There are two known approaches that utilize routing protocols in order to provide QoS in MANETs. The first one tries to embed end-to-end minimum *QoS guarantees* (delay, bandwidth) in the computation of the routing algorithm. The idea is to implement routing integrated with a resource management scheme in which an application requesting a connection specifies the minimum required bandwidth. The routing protocol would then find the optimum route that can best satisfy that requirement. Core-Extraction Distributed Ad-Hoc Routing (CEDAR) is an example of use of this approach [21]. The second approach is presented in [22]. In this case, an extension to AODV is proposed that takes routing into account to satisfy QoS requirements. Extensions to the RREQ and RREP messages are mapped into QoS pre-defined parameters (such as maximum allowed delay or minimum bandwidth along a route). A node that receives a RREQ message with a QoS extension must be able to meet that service requirement in order to either rebroadcast the RREQ or unicast a RREP to the source, thus reserving resources along the established path.

While there are indications that both approaches can be successfully applied to provide routing with QoS in MANETs, it is also true that routing protocols should not be burdened with the computation associated with providing QoS functionality at the network layer. Rather, since routing and forwarding are two different tasks, especially in MANETs, this thesis prefers to separate the two issues by implementing QoS schemes that can easily and flexibly adapt to any given routing protocol.

F. TRANSPORT LAYER

In this Section, the focus is in understanding the interaction between TCP/IP (or UDP/IP) and the IEEE802.11 MAC layer in order to establish upper bound metrics for typical types of traffic in multi-hop wireless networks. These values are important because they impose practical limits on the application layer performance of a MANET under any scenario, thus serving as reference for QoS analysis.

It is common knowledge that, in the transport layer, User Datagram Protocol (UDP) provides unreliable data packet delivery, while Transmission Control Protocol (TCP) offers reliable ordered delivery and also congestion avoidance/flow control mechanisms. The use of either in MANET will depend on the characteristics (requirements) of the traffic (voice, video, images, data) to be transmitted. In multi-hop wireless networks, both UDP and TCP perform in a much less predictable way than in wired networks. The main reason for that is the interaction with the MAC layer [23].

In a multi-hop configuration, such the one shown in Figure 3.10, where neighboring nodes are separated from each other by a distance equal to the transmission range, packet transmission can occur on at most one hop among three consecutive hops because of the contention for the shared wireless medium. Since each node has a finite buffer, increasing the number of hops from 1 to N results in increased delay, eventual packet dropping (buffer overflow) and decreased throughput, as shown in Figure 3.11. When the number of hops is large enough, the throughput stabilizes due to "effective pipelining" [23].

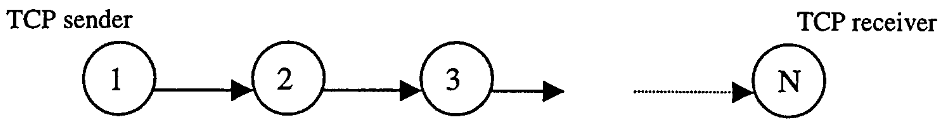


Figure 3.10 Fixed Multi-hop Wireless Network – String Topology, No Movement, Variable Number of Hops (After Ref. [23]).

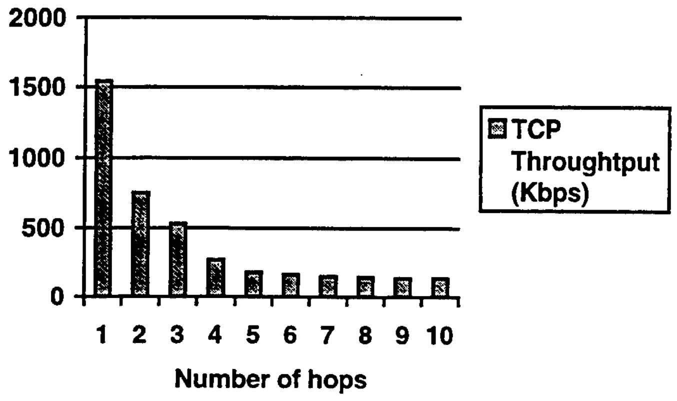


Figure 3.11 TCP Throughput Using 2Mbps Channel over 802.11 MAC Layer in the Network Shown in Figure 3.10 (After Ref. [23]).

The result depicted in Figure 3.11 establishes an upper bound on throughput:

$$Throughput = \sum_{n=1}^{N} f(n) \cdot T(n) \qquad (3.5)$$

where *f(n)* is the fraction of time during which the path length between sender and receiver contains "*n*" hops and *T(n)* is the maximum throughput (from Figure 3.11), when path length is "*n*".

If mobility is added, the effects of link failure and route changes generally will cause even more degradation on TCP performance. This is because the TCP sender assumes that all losses are caused by congestion and thus, it reacts by reducing the congestion window size or by backing-off its retransmission timeout, which tends to decrease throughput even more (see Figure 3.12).

However, it is important to notice that, sometimes, increasing mobility may improve TCP performance instead of degrading it, since at higher speeds, the network configuration when timeout occurs may be more favorable than at lower speeds [23].

Improving the interaction between TCP/UDP and the MAC layer in order to improve performance is a major problem in MANETs and has been addressed by several different approaches that are outside the scope of this thesis.

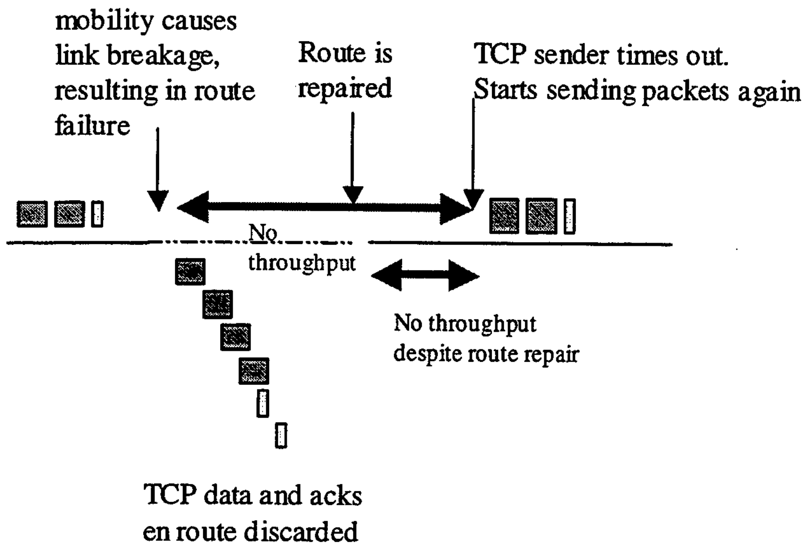

mobility causes link breakage, resulting in route failure
Route is repaired
TCP sender times out. Starts sending packets again
No throughput
No throughput despite route repair
TCP data and acks en route discarded

Figure 3.12 Degradation of TCP Throughput due to Mobility (From Ref. [24]).

G. APPLICATION LAYER

Referring back to Figure 3.3, it is in the application layer that the traffic is generated and received. Also, it is where the performance metrics (throughput, end-to-end delay, etc.) are collected, and hence the high level QoS mechanisms can be effectively applied.

Each node in a MANET can potentially represent a source of different types of traffic. For example, a mobile node can be a battleship or a Marine soldier in the field. Accordingly, the types of information generated by different sources can be vastly different. In the battleship case, the mobile node can be a router connected to a radio on one interface and to a fixed, shipboard LAN on another interface, as shown in Figure 3.13. The separation between the router and radio equipment is logical. It is likely that most of the functions of these blocks can be integrated into one device, say a software defined radio (SDR).

Part of the traffic generated by the different sources belonging to the shipboard LAN will have destinations located outside the ship. It is assumed that protocols in this wired network will be able to organize the type of traffic to be sent to the wireless medium. When forwarding or relaying packets to the wireless medium, each mobile node will be responsible for processing packets in order to provide QoS. This will be explored in the following chapters.

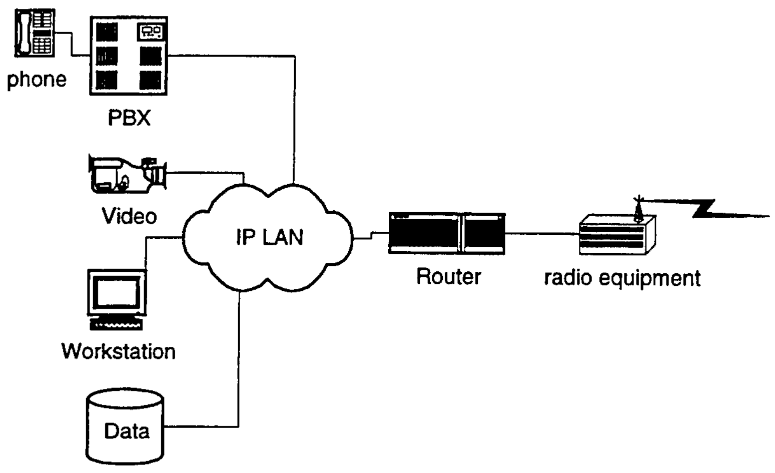


Figure 3.13 Traffic Application in a MANET Mobile Source Node.

1. Types of Traffic

There are basically two types of traffic that can be sent by a mobile node: congestion-controlled and non-congestion-controlled traffic.

Congestion-controlled refers to traffic for which the source "backs-off" in response to congestion. Reliability is an important issue for this type of traffic. In this case, TCP and its flow control and congestion avoidance mechanisms are used. The nature of this traffic allows accepting a variable amount of delay in the delivery of the packets. Interactive traffic and transfer of large files (using FTP) are good examples of this type.

Non-congestion-controlled refers to traffic for which a relatively smooth data rate and delivery delay are desirable. Examples are real-time video and audio. In this case, UDP is used because typically no retransmissions are feasible for real-time data packets and it is important to maintain a smooth delivery flow. The concern in this case is how much the quality of the received traffic will deteriorate due to lost packets. Typically, real-time traffic contains a fair amount of redundancy, which implies that the loss of a few packets will not be noticeable.

In summary, independent of the type, some types of traffic such as real-time multimedia (voice, video, image) and mission critical data, have specific quality of service to be provided by the network. In MANETs, where variable queuing delays and congestion losses are more likely, it is difficult to meet these requirements.

H. SUMMARY

This chapter provided a background on MANETs and an overview of the main issues related to each layer of the MANET protocol stack. The models adopted to represent the physical, MAC and network layers were presented in detail. Additionally, the QoS-related topics of each layer were described.

THIS PAGE INTENTIONALLY LEFT IN BLANK

IV. QUALITY OF SERVICE ANALYSIS

This chapter presents the theoretical basis upon which the work reported in this thesis has been developed. Specifically, the first section presents an overview of the QoS problem and the scope of the analysis. The second section describes the two most important QoS protocols available for fixed IP networks and discusses how they can be adapted for MANETs.

A. OVERVIEW OF THE PROBLEM

1. Defining QoS in IP networks

The idea of providing quality of service (QoS) in IP networks, mainly in the Internet, first appeared when some types of multimedia and time-sensitive traffic started to be transmitted over IP networks. The Internet Engineering Task Force (IETF) community has realized that the so-called "best-effort" service offered by the routers along the path between a sender and a receiver was no longer satisfactory for these more demanding types of traffic.

"Best-effort" type of service is directly related to the "first-in first-out" (FIFO) queuing scheme implemented in the routers. After determining the next hop to send the data packets, the intermediate routers forward the arriving packets in a sequential manner: independent of the traffic type, whichever packet arrives first is also served first. Also, if more packets have arrived than the routers could handle and the queue is filled up, newly arriving packets are dropped. Of course, in a meshed IP network, where every router can communicate with every other, this simple FIFO scheme had an acceptable performance until the need for serving bandwidth and delay-sensitive types of traffic arose.

In essence, the inadequacy of best-effort service motivated the need for QoS support; that is, providing applications with enough network resources in order to achieve acceptable performance [25]. Two widely used approaches that support the provisioning of QoS in IP networks are resource reservation and traffic prioritization.

The resource reservation approach strives to establish QoS guarantees by reserving bandwidth for a specific type of traffic or flow. This reservation will only be required if the available bandwidth in the network cannot accommodate all the flows, which is likely to happen during congestion. This implies that every router along the path must be able to meet the reservation requirements ahead of time. Integrated Services (IntServ) and Resource Reservation Protocol (RSVP) [26] are the protocols developed to implement the resource reservation capability for fixed IP networks and they will be further explored in Section B.

Traffic prioritization techniques provide different packet forwarding treatment to different types of traffic. In order to achieve this, the routers in the network establish priority differentiation under the assumption that high priority traffic will be served first. Also, packets belonging to the high priority traffic are dropped less than those of the low priority traffic. Differentiated Services (DiffServ) [27] is the protocol used to provide this capability and its details will be discussed in Section B.

2. Challenges in MANETs

Providing quality of service (QoS) in a MANET environment is even more challenging compared to the fixed IP networks. Network resources in the wireless channel are quite limited: available channel capacity is low and the channel quality is poor. As a consequence, providing QoS under these conditions is a difficult task. The previously mentioned schemes of resource reservation and traffic prioritization can be applied to MANETs as well. Nevertheless, close attention must be paid to the bandwidth demands and processing needs.

As seen in the previous chapter, the time-varying, multi-hop nature of the path between the sender and the receiver, associated with the MAC contention for the shared medium impose strict bounds on the achievable throughput (i.e., link capacity). The dynamics of the network, manifested by the frequent variations in the network topology and poor quality of the routes linking the communicating nodes, make a virtual-circuit signaling-based resource reservation approach somewhat unattractive. In addition to that, the dynamic wireless links are prone to errors due to impairments in the channel, such as fading, jamming, Gaussian noise and obstacles. Error correction coding schemes are sometimes used to mitigate the effects of the channel impairments. However, the error correction schemes result in overhead and further reduce the available bandwidth.

Taking the above-mentioned limitations into account, the question that arises is: *Is it possible to provide QoS guarantees 100% of the time in MANETs?* The answer is obviously negative and a simple example can corroborate it. At a given time, let two communicating nodes reach a "stable" state in which the channel is error-free and the relative path between them does not change. In a MANET, it is probable that a brief instant later, this "stability" would be compromised due to the presence of random errors or the mobility of any of the nodes along the path. As a result, the *QoS guarantees* established during the "stable" period would be no longer possible. First, the nodes would try to re-establish the path by extensive use of signaling, and then try to correct errors by using some kind of channel coding. Both solutions would involve a large number of control packets (overhead) in the channel, thus increasing contention, reducing the available bandwidth, and decreasing the overall system throughput.

Since the characteristics of MANETs extensively limit the robustness of the possible solutions, it is clear that *flexibility* must be the primary attribute of a protocol in order to support QoS in MANETs. Although strict *QoS guarantees* are not possible at all times, the protocol should be able to provide better forwarding treatment for high priority (mission-critical) traffic whenever there is an established route between the communicating nodes. On the other hand, by emphasizing flexibility, it is important to notice that the protocols in the other layers of the stack are allowed to be loosely coupled, which often is contrary to the extreme need for efficiency in MANETs [28]. This was evident from the discussion in the previous chapter, where it was mentioned that the MAC algorithm and the routing protocol, if coupled together, would provide better QoS support. Despite the fact that a solution that tightly couples all the inter-layer protocol functions in a logical fashion is desirable, this thesis adopts a more flexible approach as there are still a lot of open, unresolved issues in each MANET protocol layer.

B. INTSERV AND DIFFSERV IN MANETS

QoS support is especially important during network congestion, which is likely to happen in MANETs frequently because of the limited available bandwidth. There are situations, however, in which bandwidth constraint is not a limitation and QoS guarantees are not needed. For example, if there is not enough traffic to exhaust the resources causing congestion, there is no need to provide QoS.

Keeping the above in mind, there is a number of solutions to the problem of congestion management [25]. One very simple solution is to over-engineer the network such that there is no need to have a contention for resources. Unfortunately, in MANETs, this solution is not applicable in general. Another approach is based on reserving end-to-end resources along the path of a flow related to a specific application. A third method is based on identifying data packets carrying high priority (real-time or mission critical) application data and giving them special forwarding treatment in the nodes along the path. Integrated Services (IntServ) and Resource Reservation Protocol (RSVP) [26] implement the second approach while the third method is realized by Differentiated Services (DiffServ) [27].

The second and third approaches have been successfully applied to fixed IP networks, and they both provide better performance in terms of throughput, delay and service differentiation than the best-effort model. Thus, understanding how the QoS features provided by these protocols in fixed IP networks could be adapted for MANETs is the key before presenting the details of the QoS schemes proposed in this thesis.

1. Integrated Services (IntServ) with RSVP

Integrated Services (IntServ) can be defined as a virtual circuit mechanism, which strongly relies on the RSVP signaling protocol to setup and maintain the virtual connections (see Figure 4.1) corresponding to each flow (i.e., application session between sender and receiver). All routers along the path must be RSVP enabled and must apply resource management schemes to support the QoS specifications of the connection. Also, the so-called soft states established in each intermediate router must be refreshed periodically in order to maintain the connection [26].

RSVP signaling is illustrated in Figure 4.1. The source initiates a PATH message that specifies the traffic characteristics of the source's media flow. If the receiver can support the specified traffic flow, then it returns a RESV message to the source. The intermediate routers can either accept or reject the QoS reservations (buffer capacity or link bandwidth) contained in the RESV messages.

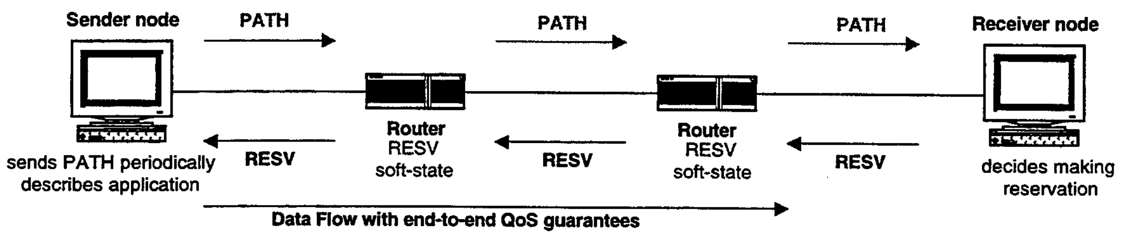


Figure 4.1 RSVP Protocol (After Ref. [29]).

The per-flow granularity provided by IntServ tends to cause scalability problems in fixed IP networks since the amount of state information that must be kept by each router along the path increases proportional to the number of flows. This problem is less likely to happen in the current MANETs since the limited bandwidth generally limits the number of flows in the network. However, the use of a pure RSVP-like signaling scheme, which requires extensive use of control packets (overhead) to maintain the routes, is not practical in MANETs due to the frequent changes in topology and link capacities [20,30].

2. Differentiated Services (DiffServ) [27]

Differentiated Services (DiffServ) uses relative-priority schemes in order to provide a different packet forwarding treatment (service) for heterogeneous application requirements or end-user expectations. Unlike IntServ, no signaling is necessary. Instead, a service level agreement is generally established prior to the traffic exchange, so the actions to which the traffic streams will be submitted are pre-defined. The traffic streams in this case are no longer micro flows, but instead, they are grouped forming "aggregates" of multiple flows. As shown in Figure 4.2, traffic entering a QoS-enabled fixed IP network is first classified, and then conditioned at the boundary of the network in an *ingress* router or *ingress* node.

Classification consists of distinguishing a packet based on some identifying field present in the header of the packet. For example, the identifying field can be formed using the flow ID field in IP version 6 or the type of service field in IP version 4 (named DS field in DiffServ), or even a multi-field composed of source/destination addresses and source/destination ports. The classifier, based on one of these fields, "steers" the packets to a logical instance of a traffic conditioner (see Figure 4.2).

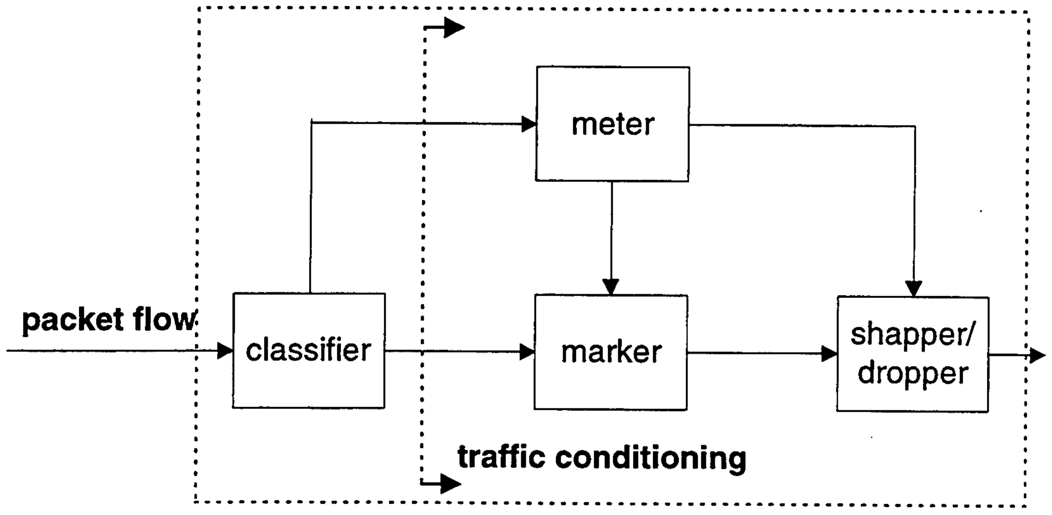


Figure 4.2 Traffic Classification and Conditioning in DiffServ.

The traffic conditioning function may contain three main elements: meter, marker and shaper. All of them need not to be present. By metering, a traffic stream is compared against a specific pre-defined traffic profile. The resultant state of the meter (*IN* or *OUT* profile) defines the actions to be performed by the marking and shaping blocks. For instance, packets may be marked as "premium" (or simply high priority) or "best-effort". Shaping can be used to adapt (or drop) a traffic stream, making it conforming to a specific traffic profile.

In fixed IP networks, routers located at the boundary of a network (*ingress* routers) execute classification and conditioning. After conditioning, routers in the interior part of the network provide the appropriate queuing discipline or scheduling to guarantee the differentiated forward treatment (named per-hop-behavior or PHB) for the now distinct types of traffic. The combination of these processes is illustrated in Figure 4.3.

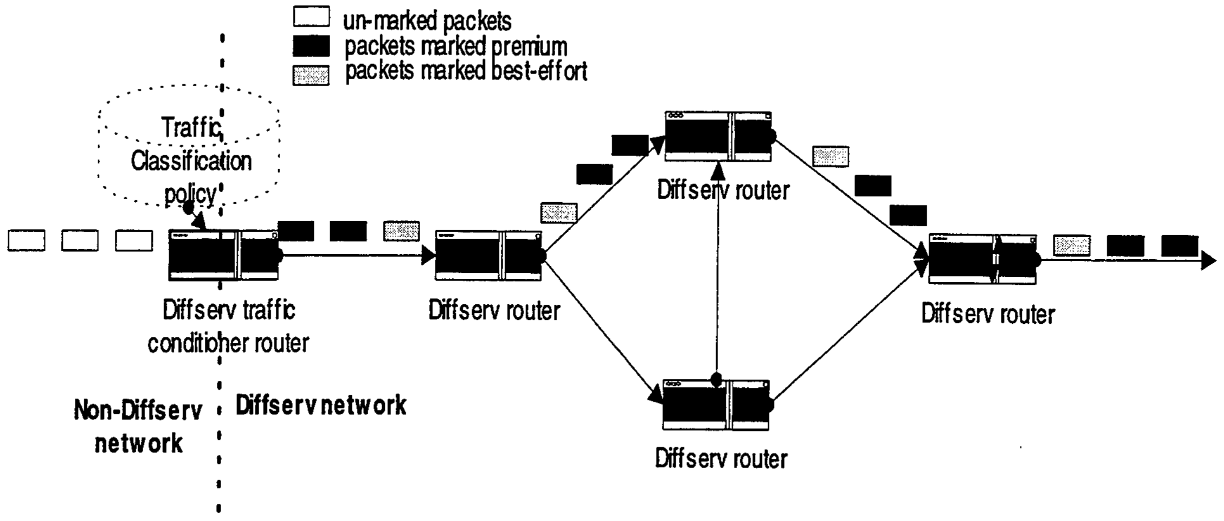


Figure 4.3 Overall Results of DiffServ Protocol.

There are specific points that need to be addressed in MANETs relative to DiffServ. The relative-priority mechanism and the absence of any kind of signaling make DiffServ a *flexible* approach for use in MANETs. Originally, DiffServ was developed for fixed IP networks with the basic idea of softening the hard signaling requirements of QoS models like ATM and IntServ/RSVP.

Also, unlike in fixed IP networks, MANET nodes may execute a router/switch and host function. The dual role of MANET nodes makes the distinction between *ingress* nodes and *interior* nodes impractical. DiffServ is lightweight in interior routers as it avoids the per-flow states and signaling at every hop that is common in IntServ/RSVP. In MANETs, it is important to keep the QoS protocol lightweight in the nodes acting as routers. Since most MANET nodes serve the roles of router and host, the processing available for QoS service is limited. The power budget of MANET nodes is another concern. Requiring the intermediate nodes to carry out additional processing could lead to draining of the battery power [30].

In the DiffServ architecture for fixed IP networks, currently two types of service are defined: Expedited Forwarding (EF) [31] and Assured Forwarding (AF)[32]. EF is supposed to provide low loss, low latency, low jitter and end-to-end assured bandwidth like a "virtual-leased-line". Obviously, as in IntServ/RSVP, this virtual-leased line is quite difficult to implement and maintain in a MANET due to its dynamics. On the other hand, AF is a means of providing different levels (probabilities) of forwarding assurances for IP packets belonging to an application. The assurances can, for example, be translated as expected (not fixed) throughput. As implied in [32], AF is not required to provide specific QoS guarantees. Therefore, since it is not easy, if not impossible, to provide precise quantitative QoS in MANETs (given its constraints), AF has a potential application in MANETs.

With all the flexibility and desirable features, the use of DiffServ in MANETs is still not straightforward. It is desirable to incorporate, for example, the flow granularity of IntServ/RSVP and to make some modifications in DiffServ function blocks. In the following chapter, the details of these modifications, leading to a *flexible* QoS model, are presented.

C. SUMMARY

In this chapter, the main concepts related to QoS in a fixed IP network were presented and then extended to the MANET environment. In addition, we described two QoS protocols available for fixed IP networks and discussed how they could be adapted for MANETs.

V. QUALITY OF SERVICE IMPLEMENTATION

The QoS model adopted in this thesis is based on the work presented in [30] and [33]. By adapting the fixed IP-network's DiffServ function blocks (Figures 4.2 and 4.3) and using the IntServ/RSVP's per-flow approach, the model aims to provide QoS for MANETs in a flexible manner. In this chapter, details of the three main mechanisms – traffic conditioning, buffer management, queue scheduling – used to achieve the goal of providing QoS in MANETs are presented.

First, it is important to mention that although a MANET can be treated as an IP packet switched network, its dynamics do not allow an exact analytical model for the queuing schemes. Traffic flows and service times do not necessarily follow a Poisson or exponential distribution. Also, partitioning/merging of traffic and non-independent service distributions make a mathematical analysis based on queuing theory an infeasible approach [24]. Thus, this thesis will strongly rely on simulations to evaluate the performance of the proposed QoS model.

The aimed flexibility of the QoS model is achieved with simplicity of the algorithms and transparency with respect to other layers of the protocol stack. As previously discussed, *flexibility* is a highly desired characteristic because not only several underlying protocols (MAC and routing) are still under research, but also power/processing constraints of mobile nodes generally require lightweight algorithms. Thus, the general idea is to define a traffic profile for each traffic session (or flow) and to apply mechanisms that benefit traffic in conformance with those profiles. As a traffic flow is generated in the source node (before entering the network), packets are conditioned and then marked (tagged) as being "*IN*" or "*OUT*". If congestion occurs, each node will preferentially drop packets that are tagged as being "*OUT*"(buffer management) or serve packets that are tagged as being "*IN*" (priority scheduling).

A. TRAFFIC CONDITIONING

Traffic conditioning is the process in which the traffic characteristics are altered in order to conform to pre-defined requirements [27]. Traffic conditioning is different from DiffServ in which the nodes (routers) along the path have pre-established and fixed roles (*ingress, interior, destination*). In MANETs, the nodes will have dynamic roles, which means that a node acting as an *ingress* node during one connection may be an *interior* or *destination* node in another connection [30]. As depicted in Figure 5.1, Node 2 is an *interior* node in Connection 1, a *destination* (sink) node in Connection 2 and an *ingress* (or source) node in Connection 3.

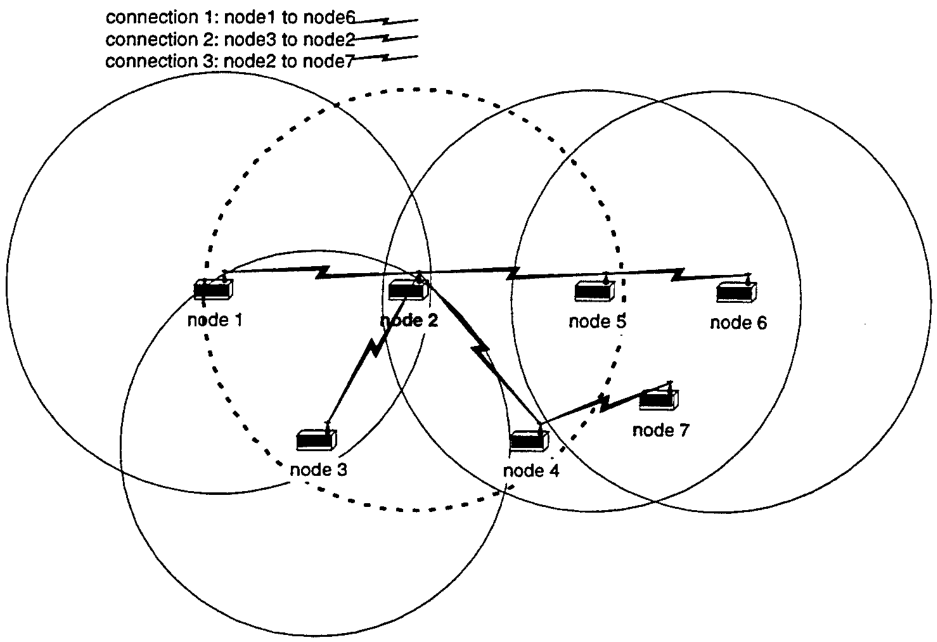

connection 1: node1 to node6
connection 2: node3 to node2
connection 3: node2 to node7
node 1
node 2
node 5
node 6
node 7
node 3
node 4

Figure 5.1 Example of Dynamic Roles of Mobile Nodes in MANETs.

In Figure 5.1, in the role of an *interior* node, Node 2 must act like a DiffServ *interior* router, receiving and forwarding data packets to the next-hop node according to a pre-defined queue scheduling/management scheme derived from the *Type of Service* (TOS) field in the IP header of the packet. As a *destination* node, it must act as a receiver or sink, receiving traffic and sending acknowledgments if TCP is used. Finally, as an *ingress* (source) node, Node 2 must generate traffic, and is also responsible for implementing traffic conditioning.

As shown in Figure 5.2, traffic conditioning is composed of four main elements: meter, traffic profiler, marker and shaper/dropper. Each block will be described in the following sections.

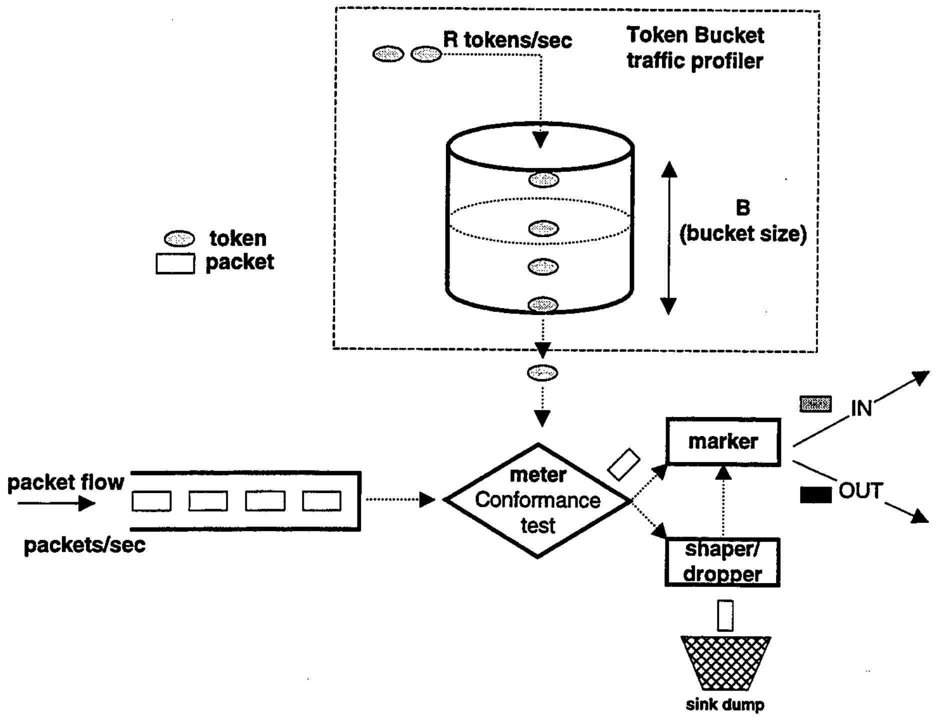


Figure 5.2 Block Diagram Illustrating Traffic Conditioning at an *Ingress* (Source) Node.

1. Metering and Traffic Profiler

The meter function block is responsible for comparing the temporal characteristics of a traffic flow (traffic stream) to be sent by a source node with a pre-defined traffic profile implemented by a token bucket algorithm (see Figure 5.2).

A token bucket traffic profiler consists of two parameters: a token rate R_i and a bucket size B_i. R_i specifies the continually sustainable rate or average target data rate to be supported for flow "*i*". B_i specifies the amount by which the data rate can exceed R_i for short periods of time. During any time period T, the amount of data sent cannot exceed $R_i \times T + B_i$. Over the long run, R_i will be the average data rate (or target rate) allowed by the token bucket. However, if there is an idle or slow period, the bucket builds up, so a traffic rate of at most B_i above the stated rate can be accepted. Thus, B_i is a measure of the degree of burstiness of the data flow *"i"* that is allowed [33].

The metering and the conformance test of Figure 5.2 are implemented according to the pseudo code in Figure 5.3. The logic behind the algorithm is simple. If a packet arrives and there are insufficient tokens available, then the packet is violating the average target rate for this flow. The treatment for such packets can be either marking them as *OUT* or sending them to the shaper/dropper (to be explained in a later section).

```
Initialize Tokens = Bucket_size($B_i$) and $R_i$=token_generation_rate for flow i
Function get_update_tokens /*gets present number of tokens in the bucket*/
Tokens = Tokens + (present_time – lastupdate_time)*$R_i$
If Tokens > $B_i$, then Tokens=$B_i$ /*bucket cannot overflow*/
Function consume_tokens /* consume tokens of the bucket*/
Tokens = Tokens – packet_size
Initialize Metering and Conformance Test
get_update_tokens
if (Tokens $\geq$pktsize) /* if there are enough tokens in the bucket*/
iph->prio = IN ; /* mark conforming packet as IN, associated with high priority*/
consume_tokens;
else /* if there are not enough tokens */
iph->prio =OUT; /*mark non-conforming packet as OUT, associated with low priority*/
or sent to shaper/dropper
```

Figure 5.3 Pseudo Code of Meter/Marker Block.

At this point, it is essential to realize that the parameters (token rate "R_i" and bucket size "B_i") defining the traffic profile for a specific flow " *i* " should not be fixed because in MANETs, the available bandwidth of a wireless link between nodes will certainly vary with time [34]. In fact, the number of hops, among other parameters (such as routing load, noise and propagation phenomena), imposes an upper bound on the throughput (or bandwidth capacity). Mobility causes variations in the number of hops, which causes variations in the maximum available link bandwidth (see Figure 3.10), hence causing variations in the maximum allowed throughput of traffic that can be sent by a source node. Since the traffic to be sent is confronted (by a meter) with a traffic profile in order to check for conformance (*IN* or *OUT* conformance), it is not prudent to keep the profile parameters fixed. In order to keep the traffic profile *adaptive* to the dynamics of the network while keeping the differentiation between traffic flows predictable, the token bucket profiler parameters (R_i, B_i) must be defined as a function of the relative percentage of the link capacity, i.e.,

$$R_i = T_i \times f(N_h) \times R, \qquad 1 \leq \mathrm{i} \leq \mathrm{N} \tag{5.1}$$

$$B_i = L_i \times f(N_h) \times B, \qquad 1 \leq \mathrm{i} \leq \mathrm{N} \tag{5.2}$$

where R_i is the token generation rate, B_i is token bucket size, T_i and L_i are parameters related to the relative target rate of the i^{th} flow, $f\,(N_h)$ returns the available bandwidth as a function of the number of hops (N_h) between sender and destination nodes at a specific time, and R and B are proportionality constants representing other factors (such as routing load, noise, fading, etc.). Assuming a quasi-static (no relative mobility) scenario, we can set $R = B = 1$, leaving the number of hops as the main driving factor for setting the profiler parameters.

It is reasonable to assume that the required up-to-date information about the number of hops between sender nodes and destination nodes can be provided by the routing protocol adopted in the MANET. The meter block and its associated meter traffic profiler can then utilize the up-to-date information about the number of hops to regulate the allocation of the available network bandwidth among individual flows.

In summary, the meter and traffic profiler blocks do not provide a strict guarantee, but rather an *expectation* of the service that will be provided to a flow during times of congestion [33]. Thus, the key detail provided by this QoS model is that it permits different users (traffic flows) to have different expectations. Simulation results for traffic conditioning are presented in Chapter VII.

2. Marking

A marker basically executes a tagging algorithm associated with the results obtained by the traffic meter. Generally, the distinction between marking and metering is much more functional than logical since one routine can execute both functions at the same time (see Figure 5.3).

The idea of using tags to identify packets is not exclusive to DiffServ. ATM, for example, uses a similar concept by marking the so-called Cell Loss Priority (CLP) bit. In the literature [27], the Type of Service (TOS) field in the IP header embedding this tag is usually named DiffServ Code Point (DSCP). This thesis adopts the same nomenclature.

IP packets conforming to a traffic profile have their header DS (DiffServ) field marked as *IN* (meaning *in* conformance). Similarly, non-conforming IP packets are marked as *OUT*. The idea of marking the packets at the *ingress* nodes, as part of conditioning, attempts to classify and facilitate the identification and further processing (forwarding) of the packets by the intermediate nodes acting as routers.

The result of marking or tagging gives the intermediate mobile nodes a better handle on how much of the traffic, at any instant, is "valued" traffic, and how much is just "opportunistic" traffic for which a more relaxed service can be tolerated. In other words, the marking process is directly associated with establishing priorities for the packets. Clearly, the *IN* packets have higher priority than *OUT* packets.

3. Shaping/Dropping

As depicted in Figure 5.2, after the conformance test, packets marked as OUT can be made to conform or can be dropped even before transmitting. The idea here is to police the traffic flow so that a flow known to exceed the allocated capacity can be either dropped early or adjusted (shaped) to fit in. The shaping/dropping function can also be executed by a token bucket-like algorithm.

B. BUFFER MANAGEMENT

1. Random Early Discard (RED)

Random Early Discard, Random Early Drop, or Random Early Detection are equivalent names for the same algorithm. The idea behind RED is quite simple: packets are randomly dropped with increased probability as the queue size grows. In order to maintain the network in a region of low delay and high throughput, RED's main goal is to avoid congestion by detecting it early rather than reacting to it [35].

In a simple FIFO (first-in first-out) algorithm, queues drop all packets at the tail of the queue when it reaches its size limit. This implies that by only dropping newly arriving packets, FIFO discarding algorithm is biased against burst sources as compared to smooth sources with the same average traffic. If interactive traffic, which in general is a typical bursty traffic, is assumed to be largely used in MANET, this behavior of FIFO should be avoided.

Proceeding further with a FIFO scenario, typically TCP sources use packet drops as an implicit signal of network congestion and reduce their information transmission rate according to a "fast-recovery" or "slow-start" algorithm [36]. With a traffic burst, queues fill up quickly, and several packets are dropped. The likely result is that many TCP connections are affected and enter the "fast-recovery" or slow-start" algorithm. This causes a severe drop in network traffic, so the network could be unnecessarily underutilized for a period of time. Because many TCP connections potentially entered "slow-start" at about the same time, they will also come out of this "lazy" state at the same time by ramping up their sending rates, which results in another big queue build up, and the above cycle repeats. This cyclic behavior is known as "global synchronization" [36] and its effect tends to be more harmful in a bandwidth constrained environment, such as in MANETs.

One possible solution for this problem is to increase the buffer (queue) size. As these larger buffers fill up, the delay suffered by all connections would increase dramatically [29], indicating that increasing the queue size may not be acceptable to many applications.

RED aims to minimize the effects of global synchronization by utilizing a packet-discard strategy, which discards incoming packets before the queue is completely filled up (see Figure 5.4). It basically anticipates the onset of congestion (congestion avoidance) and "tells" one application session at a time to slow down. Then, by measuring the effect of the first "slow-down", RED determines if it is necessary to slow down another connection.

To achieve the above objective, the RED algorithm defines a minimum queue size threshold (θ_{min}), a maximum queue size threshold (θ_{max}), and a time-based average queue length (L_{av}), as shown in Figure 5.4. If $L_{av} < \theta_{min}$, then packets are queued and no packets are dropped (normal stage); if $L_{av} > \theta_{max}$, then all packets are dropped (congestion control stage); and finally, if $\theta_{min} < L_{av} < \theta_{max}$, then packets are randomly dropped with linearly increasing probability proportional to L_{av} (congestion avoidance stage). The purpose of using an average and not the actual queue size is to filter out transient congestion at the node router. Also, the optimal values for θ_{min} and θ_{max} depend on the desired average queue size. If the typical traffic is fairly bursty, then θ_{min} must be correspondingly large to allow the link utilization to be maintained at an acceptably high level [35].

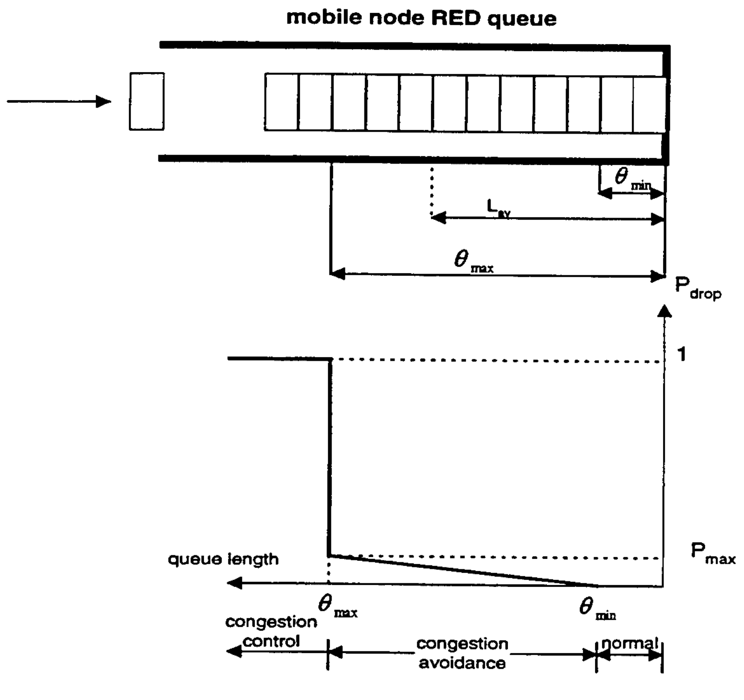


Figure 5.4 RED Queue Management Scheme.

The overall result of this probabilistic queue management scheme is that it gracefully instructs some TCP sources to reduce their sending rates so that an interior node's queue does not overflow. This allows the node to support new TCP connections, handle periodic bursts of data, and maintain high network utilization during periods of congestion, which is highly likely in MANETs.

2. Random Early Discard (RED) with IN/OUT (RIO)

The idea of RED with RIO is an extension of the RED concept and hence utilizes the same mechanisms. In RIO, there are two sets of parameters, one for *IN* packets and the other for *OUT* packets. The parameters $\theta_{\min}^{in}$, $\theta_{\max}^{in}$ and $P_{\max}^{in}$ define the normal operation $[0, \theta_{\min}^{in})$, congestion avoidance $[\theta_{\min}^{in}, \theta_{\max}^{in})$, and congestion control $[\theta_{\max}^{in}, \infty)$ stages for the *IN* packets. For the OUT packets, $\theta_{\min}^{out}$, $\theta_{\max}^{out}$ and $P_{\max}^{out}$ define the corresponding phases [33]. The algorithm works in a similar fashion to RED and is displayed in pseudocode in Figure 5.5.

As a packet arrives at a mobile node, the node checks the packet's tag. Then, the node calculates the average queue length for the *IN* packets (L_{av}^{in}) and the average total queue size for all arriving packets (*IN* and *OUT*). The probability of dropping a packet will then depend on which phase of operation the node is operating in. Figure 5.6 shows that during periods of congestion, nodes adopting this buffer (queue) management scheme will preferentially (with a high probability) drop *OUT* packets.

The packet scheduling (serving) scheme in the queue is FIFO (first-in first-out), meaning that the serving and dropping order of the packets in the queue will not be changed as shown in Figure 5.7. At this point, it should be clear that in MANETs, routing control packets must *always* have the highest priority and must be inserted at the head of the queue in order for them to be not dropped. This is to prevent routing perturbations or disruption of routing management functions, which are very important in MANETs due to mobility of the nodes. As a consequence, by using RED/RIO, the MANET nodes will offer different levels of service based only on differentiated probability of packet dropping, but still keep the FIFO serving (scheduling scheme).

```
For each packet arrival
    If it is an IN Packet
        Calculate the average IN queue size = $L_{av}^{in}$;
        If $\theta_{\min}^{in} < L_{av}^{in} < \theta_{\max}^{in}$
            Calculate the Probability $P^{in}$;
            Drop packet with probability $P^{in}$;
        Else if $L_{av}^{in} > \theta_{\max}^{in}$ Drop packet
    Calculate the average total queue size = $L_{av}^{total}$;
    If it is an OUT packet
        If $\theta_{\min}^{out} < L_{av}^{total} < \theta_{\max}^{out}$
            Calculate the Probability $P^{out}$;
            Drop packet with probability $P^{out}$;
        Else if $L_{av}^{total} > \theta_{\max}^{out}$ Drop packet
```

Figure 5.5 RED/RIO Algorithm (After Ref. [33]).

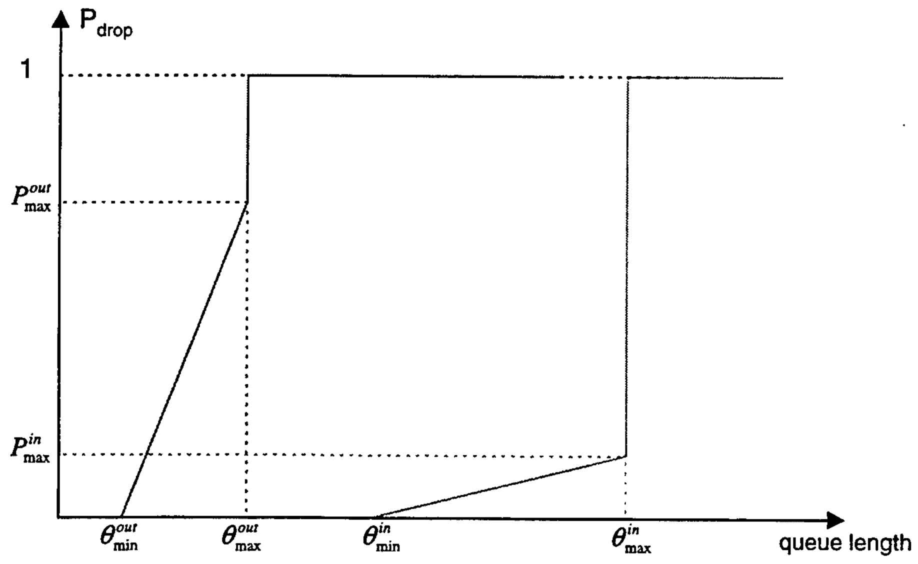


Figure 5.6 RED/RIO Queue Dropping Scheme.

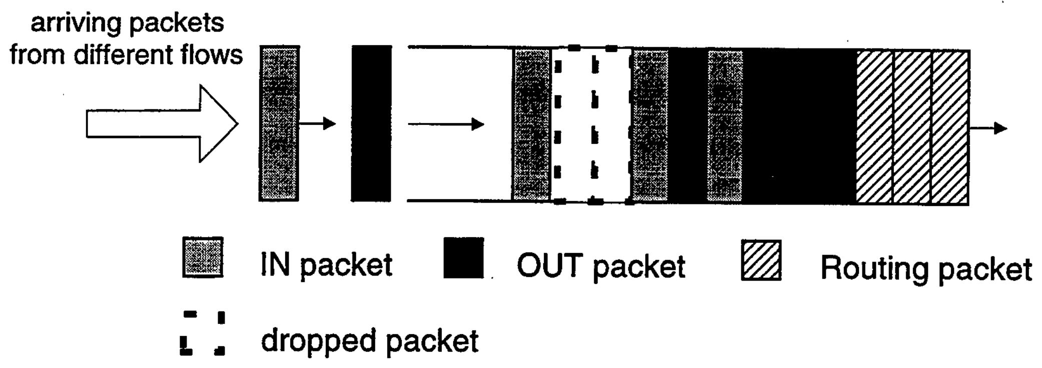


Figure 5.7 Arrangement of Packets in a Mobile Node's Queue Using RED/RIO Management Scheme (After Ref. [37]).

C. QUEUE SCHEDULING WITH PRIORITY

As previously mentioned, RED/RIO buffer (queue) management provides service differentiation (QoS), without changing the basic FIFO scheduling algorithm. This means that RED/RIO can be seen as an intelligent discard policy, instead of a packet-scheduling scheme. Scheduling means determining the order in which the queued packets are transmitted. There are several scheduling methods in the literature that are distinguished by the way the decisions to select a packet to transmit are made. Priority Queuing (PQ), Class-Base Queuing (CBQ) and Weighted Fair Queuing (WFQ) [38] are representative examples. All of them aim to provide some sort of QoS by providing different service (forward) treatment to different types of traffic.

This thesis adopted the PQ method, which provides QoS by implementing precedence-ordered queue service or simply priority queuing. When a packet is selected for output on a logical link, the packet of highest precedence that has been queued for that link is sent.

1. Priority Queuing

The main purpose of implementing a priority scheduling in the queues of intermediate and source (*ingress*) nodes is to create different priorities to serve users with different needs. When a packet with a higher priority arrives, it is inserted ahead of all packets with lower priority. Thus, higher priority packets in the queue will always be served first. In the case of network congestion, the last packet in the queue will be dropped. Consequently, the result of using this algorithm is to build up a queue of lower priority packets, which will cause packets in this class to be preferentially dropped due to queue overflow as shown in Figure 5.8.

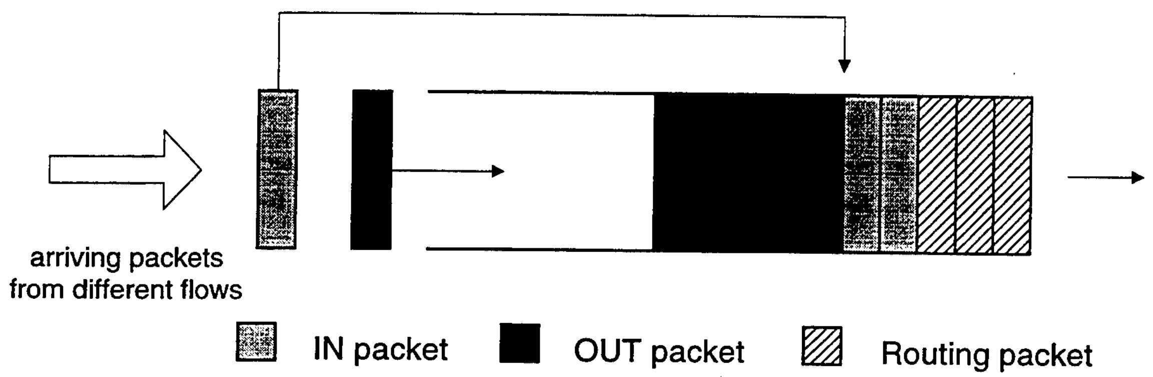


Figure 5.8 Arrangement of Packets in a Mobile Node's Queue Using a Priority Scheduling Mechanism (After Ref. [37]).

In the mobile nodes, there will be no separation of traffic from different flows into different queues. The packets of all flows are placed in just one queue. Since different flows can have very different profiles, this will result different quantities of "IN" packets in the service queue. This attribute helps to reduce the processing time in each node because they do not need to keep track of the status of several different queues. Besides, as pointed out in [33], separate queues for different types of packets will likely cause packet reordering, resulting in performance degradation in TCP or jitter in real-time traffic.

D. SUMMARY

In this chapter, details of the three main schemes – traffic conditioning, buffer management, queue scheduling – used to achieve the goal of providing QoS in MANETs were presented.

VI. SIMULATION

The simulation software used in this thesis was the Network Simulator 2 (NS2), version 2.1b6 running on a Linux RedHat 6.2 platform. Although there were other popular software packages available to simulate wireless networks, NS2 was, at the time of this work, the only tool that embedded four of the main MANET routing protocols (DSDV, DSR, AODV and TORA) proposed by the IETF. Also, NS2 is freeware software that can be downloaded on the Internet from the University of California (UCB) at Berkeley. Thus, to achieve a common ground in evaluating and comparing MANET protocol (not only routing) performance results, researchers recently agreed on the adoption of NS2 [39]. In this chapter, the main contributions of scripts and functions added to the NS2 structure to allow simulations of QoS in wireless ad-hoc networks will be presented.

A. NETWORK SIMULATOR 2 (NS2) [40]

NS2 version 2.1b6 was created by the Virtual InterNetwork Testbed (VINT) project funded by Defense Advanced Research Projects Agency (DARPA). The VINT project is a collaborative project that includes the University of California at Berkeley (UCB), University Southern California (USC)/Information Sciences Institute (ISI), Xerox Palo Alto Research Center (PARC) and the Lawrence Berkeley National Laboratory (LBNL). The purpose of this on-going project is to build a network simulator that allows the study of scale and protocol interaction in the context of current and future network protocols. The simulator is object oriented, written in C++ programming language, and uses Object Tool Command Language (OTcl) as a command and configuration interface (interpreter).

Basically, NS2 offers an open platform where users can manipulate the existing C++ classes and OTcl files and also add new functions according to their simulation needs. Following this idea, this thesis used and adapted several C++ and OTcl functions already existing under the NS2 architecture and manipulated other functions made available by different researchers, mainly from [30] and [33]. The main simulation goal was to properly integrate these functions and make the necessary changes in order to provide the desired QoS capabilities for the MANET scenarios under study. Close reproduction of real case military scenarios and types of traffic was also one of the major concerns that drove the simulation set-up parameters and the manipulation of the C++ and OTcl functions.

The Tool Command Language (Tcl), which is a high-level transcript language, is used in NS2 to encapsulate the actual instance of the OTcl interpreter. Tcl scripts, written by the user, are then called by the simulator and provide the links to access and communicate with functions defined in the OTcl interpreter [40].

In the simulation of MANETs, the user first generates two Tcl scripts: one defining the traffic pattern to be associated with the source nodes and another characterizing the mobility scenarios of the nodes. Then, with a third Tcl script containing the main program, the user can completely define each of the mobile nodes. That is, the protocol stack – physical, MAC, routing and queuing parameters and algorithms – associated with each mobile node is configured before running the simulation.

Figure 6.1 depicts an overview of how a simulation is performed in NS2 from the user input in the form of Tcl scripts for data processing. The Tcl script is used to bridge the OTcl script created by the user with the C++ code resident in the NS2 simulator in order to implement the different layers and algorithms of the protocol stack and the actual simulation. The details of the interaction between Tcl, OTcl and C++ in NS2 can be found in [40] and are outside the scope of this thesis.

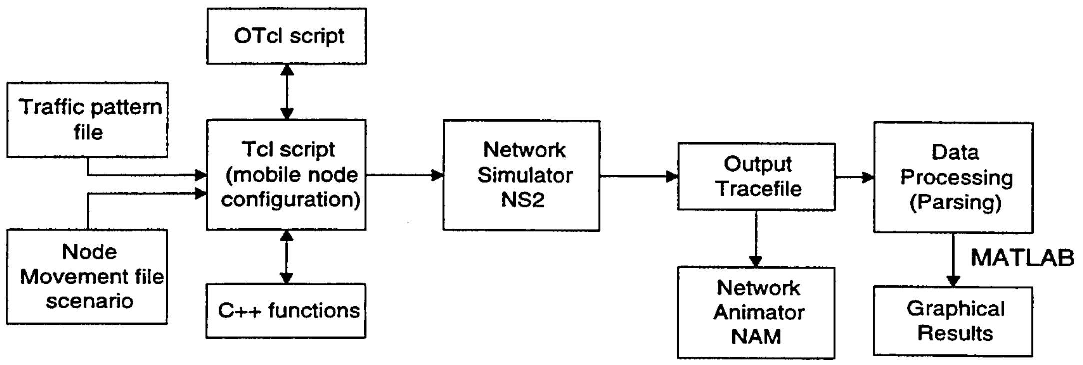


Figure 6.1 Simulation Flow in NS2.

The NS2 simulator performs the simulation and creates an output file containing results of the simulation. In the Tcl script, the user can select the granularity of the output trace file information by choosing the level (MAC, routing or application layer) of tracing to be executed. The user can also add the network animator (NAM) to the Tcl script to view the movement of the MANET nodes during the simulation time, which helps in the debugging process. Finally, the output trace file is parsed using a Perl or a Linux shell script in order to obtain the desired metrics, such as throughput, end-to-end delay, packet losses, etc. The results of the data processing are further manipulated in MATLAB to allow graphical interpretation.

Appendix A contains one example of a Tcl script file generated by the user, and Appendix B contains an output trace file generated by one of the simulations.

B. QOS SIMULATION STRUCTURE IN NS2

The NS2 QoS model proposed by this thesis contains two different levels of software programming: the user level in OTcl and the specific classes of functions resident in the simulator in C++ programming language.

The user level in OTcl is implemented by generating a mobile node movement file and a traffic pattern file (with traffic conditioning) and by defining all protocols to be used in each layer. Each specific user-file generation will be discussed in further detail in the following sections. The user has flexibility in changing the various parameters for each simulation in NS2.

Figure 6.2 depicts the hierarchical simulation design for NS2 in the two software-programming levels. Some of the C++ functions, such as FIFO, RED/RIO, MAC 802.11, and token bucket profiler were resident within NS2 and were appropriately modified according to the QoS simulation needs. Others, such as meter, marker and priority queuing were externally created in C++ and ported to the NS2 structure obeying the pre-existing class hierarchy. All these C++ functions define the corresponding Tcl classes to be used at the upper simulation level.

At the Tcl level, the C++ related subroutines are called within the appropriate Tcl scripts (see Figure 6.2). Finally in the main Tcl program, the remaining Application

Program Interface (API) configuration parameters, such as the physical layer model (two-ray ground), the type of antenna (omnidirectional), the type of ad-hoc routing protocol (this thesis adopted DSR), the height of the nodes, the simulation duration, etc. are defined and changed according to the scenario under consideration.

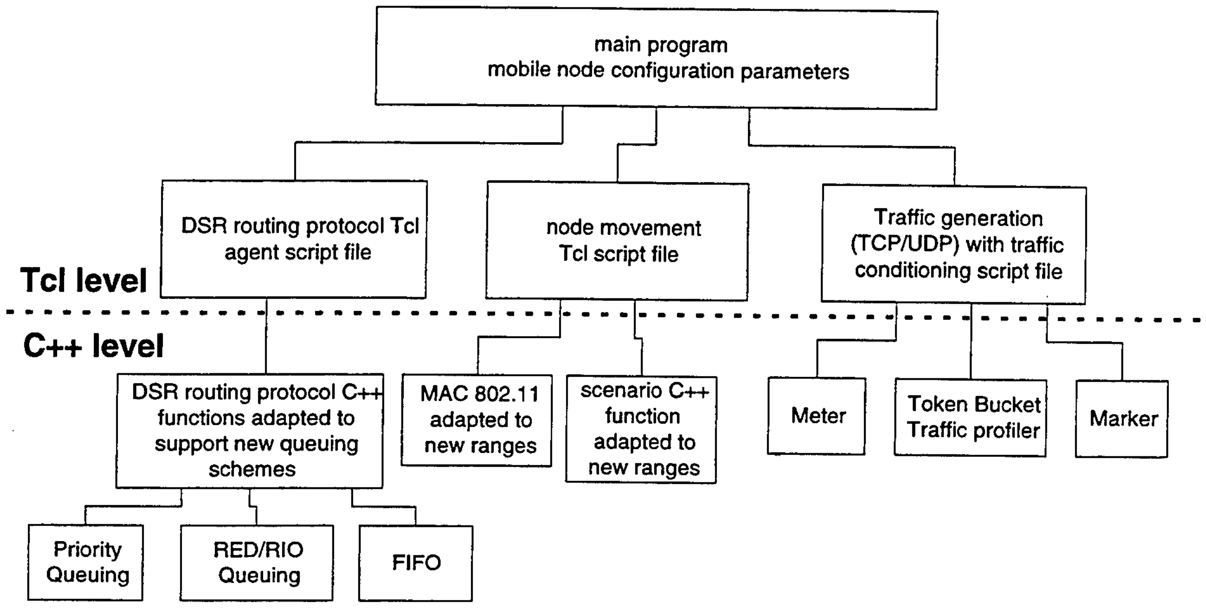

main program
mobile node configuration parameters
DSR routing protocol Tcl agent script file
node movement Tcl script file
Traffic generation (TCP/UDP) with traffic conditioning script file
Tcl level
C++ level
DSR routing protocol C++ functions adapted to support new queuing schemes
MAC 802.11 adapted to new ranges
scenario C++ function adapted to new ranges
Meter
Token Bucket Traffic profiler
Marker
Priority Queuing
RED/RIO Queuing
FIFO

Figure 6.2 Simulation Structure in NS2 to Implement QoS in MANETs.

1. Protocol Stack

Figure 6.3 shows the mobile node mechanism and implementation in NS2. The network stack for a mobile node consists of a routing agent, a link layer (LL), an Address Resolution Protocol (ARP) module connected to the LL, an interface queue (IFq), a MAC layer and a network interface. All network components are then connected to the channel [40].

Packets are generated (sent) by the application and are received by the routing agent. This agent determines the path that the packet must travel to reach the destination and stamps it with this information.

The packet is then sent down to the link layer. The Address Resolution Protocol (ARP) is used to determine the hardware addresses of the neighboring nodes and map IP addresses to the correct interfaces for dissemination. When the hardware address of a packet's next hop is known, the packet is sent down to the correct interface queue, where it awaits a signal from the Medium Access Control (MAC) protocol.

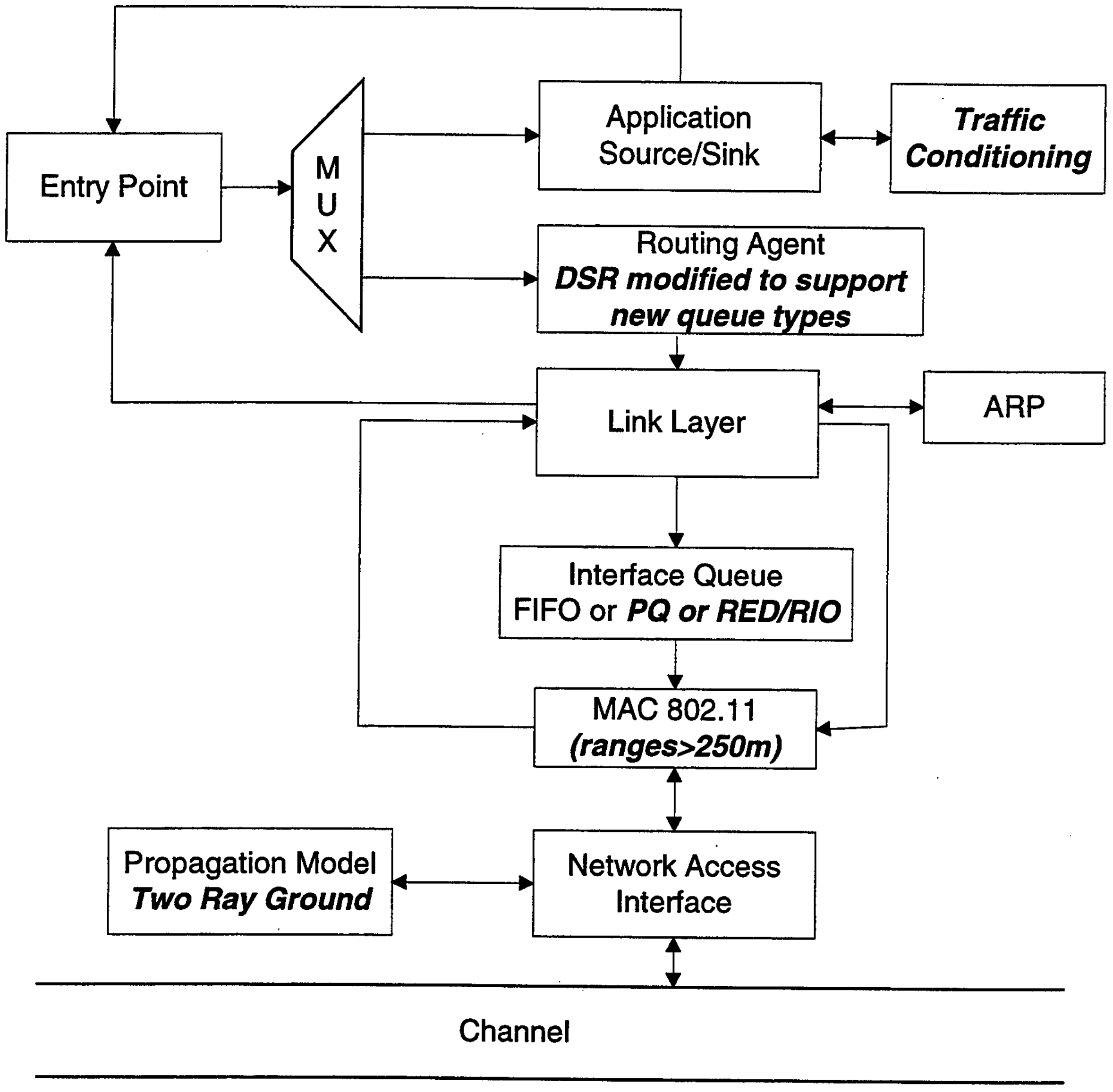


Figure 6.3 Schematic of a Mobile Node Protocol Stack in NS2 (After Ref. [40]).

When the MAC layer determines that it can send a packet onto the channel, it takes the packet from the head of the interface queue and moves it over to the network access interface. The network access interface sends the packet onto the radio channel. The packet is copied and delivered to all network access interfaces. The time of arrival of the packet is calculated for each interface based on the distance between the nodes and the speed of light. Each network access interface stamps the packet with the receiving interface information and then invokes the propagation model.

The radio propagation model uses the transmitting and receiving stamps to determine the power with which the interface at the receiving node will receive the packet. The receiving node, after checking if it successfully received the packet, passes the packet to its MAC layer. From there up to the application layer, the above procedure is then carried out in reverse order.

Figure 6.3 shows in bold italics the functionally added to the protocol stack in this thesis in order to provide the QoS and to simulate the intended military scenarios. The details of these additions are discussed in the following sections.

2. Mobile Node Movement

For node movement, the user-controlled parameters are: the number of nodes in the network *(–n)*, the pause time in between node movements *(–p)*, the maximum speed of each node in meters per second *(–s)*, the total simulation time in seconds *(–t)*, the coordinates along the x axis in meters *(–x)* and the coordinates along the y axis in meters *(–y)*.

Each mobile node movement makes use of a routing agent for the purpose of calculating routes and reachability to other nodes within the MANET. In addition, there is a C++ program (*setdest.cc*) in the NS structure that allows the user to program movement scenarios using a Random Waypoint Algorithm, described in [41]. This algorithm generates a file with embedded Tcl commands, such as:

$ns at $time $node_i setdest <x> <y> <speed> (6.1)

which induces random movement of the mobile nodes. It does so by defining at each simulation time the future destination (X and Y location) of the node and the speed with which it will move to reach this destination. Hence, the destination and speed values are generated in a random fashion.

The command line

./setdest –n 10 –p 0 –s 20 –t 1000 –x 10000 –y 10000>scenario_example (6.2)

details the use of the *"setdest.cc"* program to generate a typical mobile node movement file in NS2 with 10 nodes, average pause time between movement equal to 0 seconds (nodes always moving without stopping), maximum moving speed of 20 m/s, simulation duration of 1000 seconds and a topology size of 10 km × 10 km. This file is saved as *"scenario_example"*. Appendix C contains a node movement file generated by the user in the simulation.

a. *Expanding the range*

The NS2 program *"setdest.cc"* calculates at each simulation time the reachability of the nodes based on the default 250-meter WLAN range of the commercial "Lucent WaveLAN DSSS (direct sequence spread spectrum) radio interface". The important modification made here was the changing of this default value to larger ranges (up to 10 km), consistent with typical Navy/Marine Corps battlefield scenarios.

It was necessary to modify the IEEE 802.11 MAC C++ code in order to change the standard timing parameters of the Distributed Coordination Function (DCF), such as SIFS (short interframe space) and the slot time. This was required because, by increasing the range, the propagation delay is increased which causes the timeout of the standard 802.11 RTS (request-to-send) frames before getting the CTS (clear-to-send) back. Avoiding timeout of RTS means not dropping the packet before sending it [42]. The receiving threshold and transmitted power of each mobile node's network access interface had to be set according to larger ranges. In order to do that, the two ray ground propagation (equation (3.3)) was used to change the default values.

3. Traffic Pattern Generation

The traffic pattern generation program (*cbrgen.tcl*) embedded in NS2 allows the user to create a specific traffic pattern file, where the following parameters can be selected: the type of traffic (*–type,* Transmission Control Protocol (TCP) or constant bit rate (CBR)), total number of nodes in the network *(–nn)*, the maximum number of connections set up between them in the network *(–mc)*, a random seed and, in case of CBR connections, the rate of packet distribution in packets per second *(-rate)*. The CBR traffic is associated with User Datagram Protocol (UDP) while TCP is associated with File Transfer Protocol (FTP).

Either CBR/UDP or FTP/TCP traffic connections can be created. Each traffic pattern file generated is unique. Command line

ns cbrgen.tcl –type cbr –nn 10 –seed 1.0 –mc 8 –rate 2.0>traffic_example (6.3)

details the generation of a CBR traffic pattern in NS2 with 10 nodes, a seed value of 1, and a maximum of 4 connections with each source node transmitting at a rate of 2 packets per second. The resultant file, which will be called by the main program, is "*traffic_example*".

To simulate variable bit rate (VBR) traffic, exponential ON/OFF sources were defined and associated with UDP. In this case, packets are sent at a fixed rate during ON periods, and no packets are sent during OFF periods. Both ON/OFF periods are selected based on an exponential distribution. In all cases (VBR/UDP, CBR/UDP and FTP/TCP), packets have a constant size.

a. Traffic Conditioning

Like in the node movement file, the traffic pattern file can contain arbitrary Tcl code to configure the traffic load in the simulation. Based on the C++ functions inserted in the NS2 architecture, the corresponding Tcl classes are called by the traffic profile script (see Figure 6.2) in order to provide the necessary DiffServ conditioning functionality at the "***ingress (source) nodes***".

Figure 6.4 contains a example of a modified traffic generation file used in one of the QoS simulations. Following the Tcl command lines in this example, first, a regular NS2 TCP agent is created, where the TCP window size, the packet size, and the ID (or class) of the flow are defined. Second, the "DiffTC" (DiffServ Traffic Conditioning) Tcl class is created. This class defines the traffic profiler to be used (Token Bucket – TB) and its initial parameters (bucket size and token generation rate). The corresponding meter/marker Tcl functions are then associated with this pre-defined traffic profile (per Figure 5.2). Note that the underlying C++ functions for the meter, marker and token bucket profiler allow the association and the definition of binding parameters at upper simulation levels. After that, the conditioning function is associated with the source node and with the application (FTP) to be used as traffic type. A sink is then created at the receiving node, and the time when the traffic exchange will begin is defined.

```
# node 0 connecting to node 1 at time 52.090227846563891
# definition of TCP agent
set tcp_(0) [new Agent/TCP/Reno]
$tcp_(0) set window_ 8
$tcp_(0) set packetSize_ 1460
$tcp_(0) set class_ 0
# definition of the DiffServ Traffic Conditioning (DiffTC)
# TB=Token Bucket, Target Rate (Ri)=100kbps, Token Bucket Size (Bi)= 0
set difftc(0) [new DiffTC TB 100kbps]
set meter_ [$difftc(0) getmeter]
$meter_ tbsize 0
# attaching DiffTC to the source (ingress) node
$difftc(0) attach-conditioner $node_(0) $tcp_(0)
# defining the sink (destination node) and linking to the source node
set sink_(0) [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $sink_(0)
$ns_ connect $tcp_(0) $sink_(0)
set ftp_(0) [$tcp_(0) attach-source FTP]
$ns_ at 52.090227846563891 "$ftp_(0) start"
# adapting DiffTC parameters
$difftc(0) AdptPara 1.5Mbps 100000
```

Figure 6.4 DiffServ Traffic Conditioning (part of Tcl traffic file in NS2).

Finally, at the end of the node-to-node connection section of the file and during a specific time in the simulation, the paramaters (token generation rate and bucket size) of the trafic profiler can be modified to adapt to a new traffic conditioning situation.

4. DSR Routing Protocol Adapted

NS2 allows the selection of one of the four available ad-hoc routing protocols (DSDV, DSR, AODV, TORA) during the node configuration in the main Tcl program. In this thesis, as explained in Chapter III, DSR is adopted.

The original definition of the DSR routing agent in NS2, which also includes the link layer functionality, allows the use of only one type of interface queue: First-In First-Out (FIFO) with priority given only to the routing messages. However, to implement the DiffServ structure, Chapter V showed that the *interior* and the *ingress* nodes should be able to implement buffer (queue) management (RED/RIO) or priority queuing (PQ) in order to provide the desired service differentiation or per-hop behavior. Therefore, the NS2 functions *dsr.tcl, dsragent.cc* and *dsragent.h* were modified to allow configuration of these two new interface queuing (IFq) schemes.

5. Interface Queuing

After implementing the changes in the routing/link layer levels, the mobile nodes can be configured with one of the three different types of Interface Queue (IFq): FIFO, RED/RIO or PQ. While FIFO was originally in the NS2 structure, the other two have been added as part of this work.

a. Buffer Management (RED/RIO)

The buffer management algorithm based on Random Early Drop with IN/OUT (RED/RIO) was one of the C++ functions already embedded in NS2 for the wired networks and was based on the work presented in [35] and [33]. The original NS2 version of this queuing scheme allows the user to define the main IN/OUT threshold and dropping probability parameters during the simulation time (main Tcl program).

Two modifications were made to this program. First, access of the IN/OUT DSCP (DiffServ Code Point) field in the IP header of the packets, which is set by the marker during the conditioning at the ingress node, was implemented. Second, the MANET routing messages were set to have the highest priority among all packets. The first modification guarantees that the RED/RIO queuing scheme will be able to access the specific field in the IP header that contains information about priority of the data packets. The second modification is made because the existing RED/RIO in NS2 is not able to prioritize routing packets. We know that in MANETs the mobility of the nodes constantly causes link breakages, which causes exchange of routing packets in order to establish the new routes. If priority were not given to the routing packets or if they were treated as regular data packets, the new routes would not be rapidly established and hence a connection between the nodes would not be possible.

b. Priority Queuing (PQ)

The C++ function to execute this queuing scheme was derived from the existing FIFO in NS2. Two modifications were made to this function. Like in RED/RIO, a function that accesses the IN/OUT DSCP field in the IP header of the packets was first created. Second, another function that inserts IN packets ahead of the OUT packets in the queue was created. The main effect of these modifications is to guarantee that the queues in the mobile nodes will be able to distinguish packets having different priorities by appropriately placing them in the queue. The FIFO scheme in NS2 already provides priority to routing packets.

6. Trace Files

The completion of all simulations in NS2 generates an output trace file. NS2 has no resident statistic production capability as is available in OPNET. Each output trace file must be parsed using either a Linux shell *(awk/grep)* or Perl script commands to collect specific data from the output file for further data processing. The output trace files generated by the simulation can exceed several gigabytes of storage capacity; therefore, only a maximum of 10 active connections were used. The data for each metric (end-to-end delay, throughput and losses) is parsed from the output file using a Linux or Perl script and dumped into MATLAB to produce graphs. Appendix B contains an output trace file generated in the simulation.

C. SUMMARY

This chapter has provided an overview of the implementation of NS2 version 2.1b6 and has presented the QoS model. Several NS2 resident features of the mobile node protocol stack were maintained in our simulations. Traffic conditioning capability was added to the traffic pattern generation (see Figure 6.4). Modifications were made to the existing Interface Queues (IFq), to the MAC layer (IEEE 802.11) and to the propagation model in order to implement the QoS-enabled protocol stack and to simulate typical military scenarios. Statistics were collected on each simulation through parsing of the output file and further processing in MATLAB.

VII. RESULTS

In this chapter, the results of simulations in NS2 are presented. The objective was to use throughput, average end-to-end delay and loss rate as the main performance metrics in the evaluation of the proposed QoS algorithms. Throughput is a measure of the actual number of bits per second received by the destination nodes. Average end-to-end delay is the average of the total time taken by a packet from the moment it was sent by the sender node's application until it is received by the receiver node's application. This includes all possible delays caused by buffering during route discovery and media access and propagation. The loss rate represents the percentage of packets that are lost due to buffer overflow or routing (destination unreachable) in comparison with the total number of packets that were sent. In this thesis, the adopted physical layer model is assumed to be error-free; hence errors due to the wireless channel propagation effects are not considered.

Various network sizes (number of nodes), pause times, node velocities and packet rates were used during the simulation and further details will be provided later in the chapter. FTP/TCP, CBR/UDP and VBR/UDP performance results are compared to show how the QoS algorithms in typical MANET scenarios can affect different types of traffic.

Section A describes the scenario used in the simulation. Section B presents a comparison of results of TCP and UDP types of traffic under multi-hop wireless networks. Section C presents the impact of controlling the traffic profiler parameters in the traffic conditioning function block of the sender nodes. Section D investigates the effectiveness and limitations of the QoS algorithms when providing service differentiation for different types of traffic. Finally, Section F demonstrates how a simple voice application is affected by packet dropping and how it can take advantage of the QoS algorithms in order to provide a better signal quality for the end user.

A. SIMULATION SCENARIOS

The network configuration used in this simulation is typical of a tactical deployment envisioned by the JTRS. The network implementation encompassed 10 (or

50) nodes with a maximum of 8 connections over a variety of geographic environments: 20 km × 20 km, 40 km × 40 km and 50 km × 50 km. A larger number of connections could be used; however, the processing time and available space on the computer hard drive were limiting factors. The default parameters used in the simulations are listed in Table 7.1.

Parameter	Range of Values
Transmitter range	10 kilometers (km)
Simulation time	1000, 2000 seconds (s)
Number of nodes	10 or 50
Pause time	0, 20, 50, 120, 200, 300, 600, 900,1000
Environment size	20 km × 20 km, 40 km × 40 km 50 km × 50 km
Traffic type	FTP/TCP, CBR/UDP or VBR/UDP
Packet size	1460 bytes (TCP), 1000 bytes (UDP)
Transmitter power	50 watts (W)
Node velocity	Random number between 0 to 20m/s
Antenna heights	10 m
Wireless Channel Bandwidth	2 Mbps

Table 7.1 Parameters Used During NS2 Simulations.

1. Configuration

Each simulation was configured using the parameters listed in Table 7.1. Network environment sizes, pause times, simulation time and packet rates were varied as needed. Three different network environment sizes were chosen to provide a realistic view of the performance of the QoS algorithms with respect to different battlespaces. Nine different pause times were used to represent a high mobility to a low mobility environment. In terms of mobility, the velocity parameter was set to 20.0 m/s (72 km/h), which means that for each node, NS2 selects a random velocity in the uniform interval from 0 to 20.0 m/s (average 10 m/s) to define the node movement at a specific simulation time. Constant Bit Rate (CBR) or Variable Bit Rate (VBR) traffic using User Datagram Protocol (UDP) and File Transfer Protocol (FTP) using Transmission Control Protocol (TCP) were used in each simulation. The selection of the type of protocol (CBR/UDP, VBR/UDP or FTP/TCP) for each simulation depends on the specific type of traffic behavior to be investigated. Traffic, such as data, requiring reliable transmission use FTP/TCP; TCP has built-in congestion control mechanisms. On the other hand, loss-tolerant traffic, such as voice and video, generally uses CBR/UDP or VBR/UDP. In CBR/UDP and VBR/UDP, different sending packet rates can be selected to reflect different applications and traffic loads.

The results of a simulation session are stored in an output trace file (see Appendix C for an example of output file). The trace file is parsed using a Perl or Linux script to extract the required metrics: throughput, average end-to-end delay, and loss rate. The extracted data is then exported to MATLAB for further manipulation and graphical presentation.

B. TCP AND UDP OVER MULTI-HOP WIRELESS NETWORKS

The first step before implementing the QoS simulations is to understand the performance behavior of TCP and UDP types of traffic over 802.11 in a fixed multi-hop wireless network. The simulation results presented in Figure 7.1 show the dependence of the throughput metric on the number of hops between the sender and destination nodes. The channel bandwidth and CBR/UDP sending rate are both 2 Mbps. FTP/TCP window size is 8 packets and the packet size is 1460 bytes. Although the wireless channel has a 2 Mbps bandwidth, the maximum throughput that can be achieved with both types of traffic is always below 1.5 Mbps (see Figure 7.1). The main reason for that is the contention imposed by the RTS/CTS and the DCF protocol embedded in the MAC layer [42].

The results shown in Figure 7.1 are valid for a fixed wireless network (see Figure 3.10). It is clear that in MANETs, the node movement besides the number of hops will also play an important role to reduce the overall throughput. Thus, the values shown in Figure 7.1 can serve as upper bounds for the throughput metric and they will be used as reference values in simulations presented later.

Although the throughput achieved with both types of traffic is quite similar, in TCP this is accomplished without any packet loss due to the embedded congestion/flow control mechanisms [36]. On the other hand, in UDP there is no congestion/flow control scheme and packets are generated at a constant rate (2 Mbps) at the sender node. Since the buffer of each node has limited size (50 packets in this simulation), packets waiting to be transmitted will add to the queue (buffer) up to a point when overflow starts to occur and packets begin to be dropped. As a result, CBR/UDP traffic transmission rate should always be kept below the upper bounds of Figure 7.1 to guarantee low loss rates.

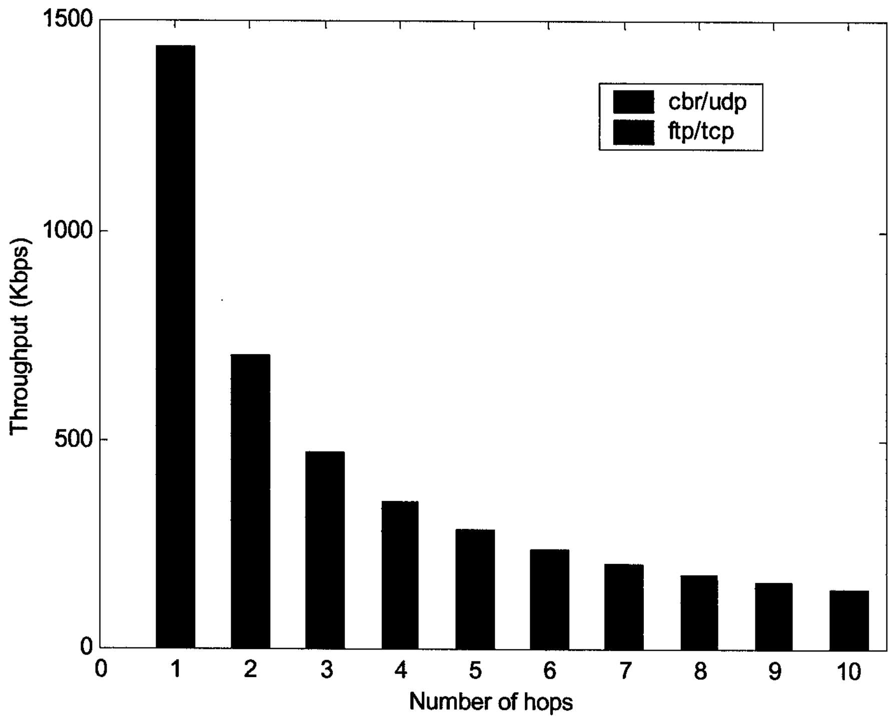


Figure 7.1 CBR/UDP and FTP/TCP Throughput over MAC802.11.

C. TRAFFIC CONDITIONING

As mentioned in Chapter IV, traffic conditioning is implemented in DiffServ in order to define packet flows that need better forward treatment within the network. Details of the main function blocks implementing traffic conditioning were presented in Chapter V. In the QoS model implemented in this thesis, traffic conditioning is performed at the ingress (source) nodes. The idea is to identify (mark) packets needing priority-based treatment at the source nodes, as shown in Figure 5.2.

1. Impact of Traffic Profiler Parameters

In order to understand how the traffic profiler parameters affect service differentiation, the following simulation scenario was created. Using the "string" fixed-node wireless topology depicted in Figure 7.2, FTP/TCP traffic was generated and conditioned at the source node (Node 0). Priority Queuing (PQ) was implemented in all the nodes. There were one or two hops between the sender and the destination depending on the receiver node being Node 1 or Node 2, respectively. Two FTP/TCP sessions were generated at Node 0. A token bucket profiler, with a target rate (token generation rate) R_i, is pre-defined and associated with each TCP session.

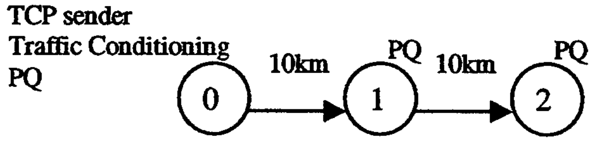


Figure 7.2 String Topology for Traffic Conditioning Simulation

There were two main goals in these TCP simulations. The first goal was to show that traffic differentiation implemented by the QoS model is highly dependent on the number of hops between the sender and the receiver. The second goal was to demonstrate that the traffic profile parameters in the traffic-conditioning block should take this information (number of hops) into account in order to maintain consistent traffic differentiation.

Figure 7.3 shows results of FTP/TCP throughput when traffic is sent through one or two hops. The target rate values were pre-defined to make the throughput of the first TCP session much larger (about 6 times) than that of the second one. Based on the a priori knowledge of the total available bandwidth for each case (see Figure 7.1), the target rates were then appropriately chosen. For the one-hop case, the two TCP sessions were given about 85% and 15% of the total available bandwidth (approximately 1300 kbps). For the two-hop case, the two TCP sessions were also given about 85% and 15% of the total available bandwidth (approximately 700 kbps).

Figure 7.3 shows that traffic differentiation is consistently achieved in both cases. In addition, the sessions with lower target rate (dotted lines) tend to have slightly greater throughput than the pre-specified target rates while the higher target rate sessions (solid lines) tend to have slightly lower throughput than the pre-specified target rates.

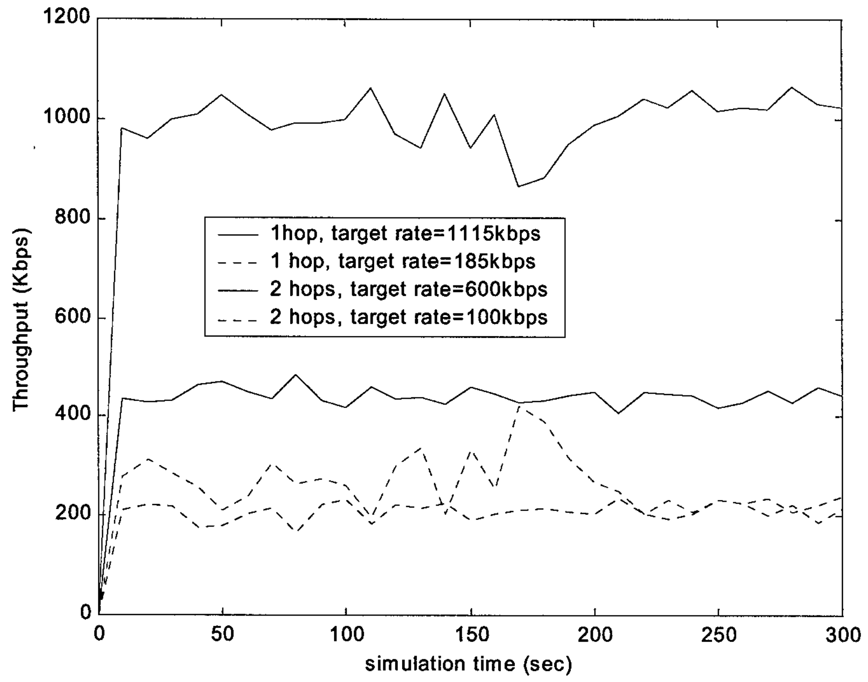


Figure 7.3 FTP/TCP over 1 and 2 Hops. Target Rates were Pre-defined Based on the a priori Knowledge of the Total Bandwidth Available for Each Case.

Figure 7.4 shows the throughput performance when the target rates are defined without knowing a priori the total available bandwidth for each case. The results demonstrate that traffic differentiation becomes difficult to achieve when the traffic profile parameters for the two-hop case are used in the one-hop case. In fact, the distinction between the solid and dotted lines for the one hop case in Figure 7.4 is much more subtle than that between the solid and dotted lines for the two hop case. This is because for the one-hop separation case, there is plenty of bandwidth available (about 1300 kbps), which is much more than the sum of the pre-defined target rates. This implies that the TCP traffic, especially the session with lower target rate, tends to absorb the excess available bandwidth, thus increasing its throughput. This leads to the conclusion that ideally, the traffic-conditioning block must be informed of the actual number of hops and then must adjust its traffic profile parameters accordingly to consistent relative traffic differentiation. It is assumed that the MANET routing protocol (DSR or any other) can supply this information when establishing a path between the sender and the receiver. If this information is not provided, assuming one-hop traffic profile parameters, if not ideal, at least can guarantee traffic differentiation (see solid and dotted lines for the two hop case in Figure 7.4).

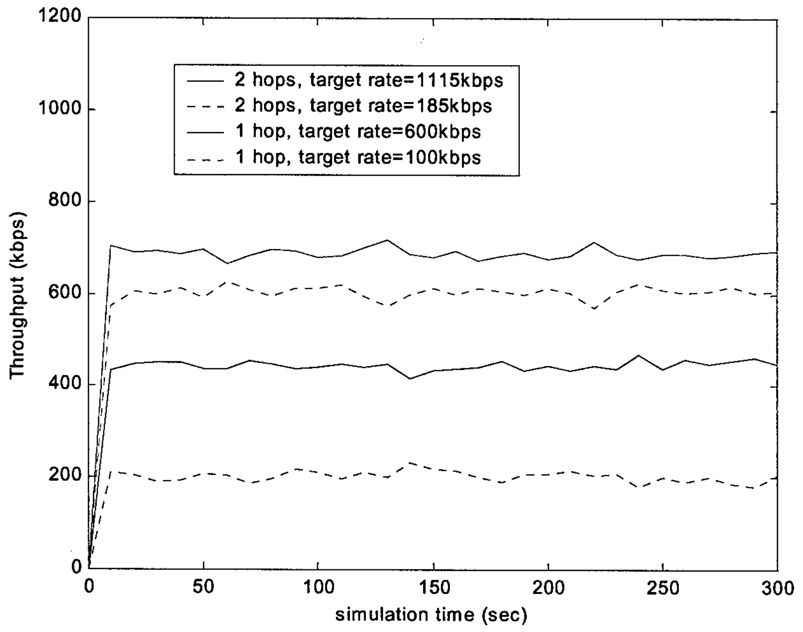


Figure 7.4 FTP/TCP over 1 and 2 Hops. Target Rates were Pre-Defined Without Knowing a priori the Total Bandwidth Available for Each Case.

D. SERVICE DIFFERENTIATION IN MANETS

In this section, the goal is to present the simulation performance results of the proposed QoS model for typical military scenarios with different types of traffic (FTP/TCP, CBR/UDP, TCP/UDP combined and VBR/UDP) being generated by the source nodes. To verify the effectiveness of the QoS model, the offered traffic load was defined so that congestion was always present in the network; i.e., in all simulations, the sending rates needed to guarantee that the available wireless bandwidth was not enough to accommodate the entire traffic load. It is important to note that without congestion, there is no need to provide QoS–based service differentiation schemes since the traffic requirements are potentially satisfied for all sources.

The scenarios for all simulations in this case had 10 nodes. Of a total of 90 possible pairs of connections that could be established between these 10 nodes, 8 traffic pairs were defined. Four of these eight pairs were selected to have low priority (packets were OUT of conformance) and the other four were selected to have high priority (packets were marked as IN conformance). Twelve movement scenarios, with a geographical size of 20 km $\times$ 20 km and 10 nodes moving with velocities randomly chosen from a uniform distribution [0 to 20 m/s], were created for these pause times: 0, 20, 50, 120, 200, 300, 600, 900 and 1000 seconds. The different pause times characterize the behavior of the system under different mobility levels. A zero pause time indicates a high mobility level. Simulations were run for RED/RIO and PQ queuing schemes in order to compare their performance with that of the best-effort FIFO scheme.

1. FTP/TCP Analysis

Results depicted in Figure 7.5 and 7.6 clearly show that while the best-effort FIFO scheme does not provide any traffic differentiation, both RED/RIO and PQ do. Figure 7.5 shows that PQ provides a better throughput differentiation than RED/RIO for all mobility levels.

It is important to mention that, for this type of traffic (FTP/TCP), the loss rates for all mobility cases were quite low. In fact, the average loss rate was below 0.5% and only slightly higher (2%) for the low priority packets in the RED/RIO scheme. Note that a packet loss is only computed when a packet that is sent by the source node is not received at the destination. In other words, although packet losses may have occurred, TCP retransmissions guarantee good reliability in terms of packet delivery at the end-receiver. The average percentage of packets that were retransmitted was below 5%, even for the high mobility cases.

As depicted in Figure 7.6, the average end-to-end delay for the low and high priority sessions in the FIFO and RED/RIO schemes does not differ significantly. This is because the position of the packets in the queues for these schemes stays the same, independent of whether the packets are low or high priority (see Figure 5.7). On the other hand, for the PQ scheme, low priority packets stay longer in the queues than the high priority ones (see Figure 5.8), which explains the significant difference in the delay metric.

Two major facts justify the low loss rates (below 0.5%) for TCP connections. First, the selection of a limited geographical size (20 km × 20 km) combined with a small number of nodes (10) and a transmission range of 10 km reduces the losses caused by unreachability of the nodes (routing losses). Second and most important, the flow control/congestion avoidance of TCP seems to adapt well to the RTS/CTS and DCF schemes of the 802.11 MAC layer.

It is known that when congestion or link breakage occurs, TCP reduces its sending rate, which causes degradation in the throughput performance. However, as shown in Figure 7.5, this degradation occurs for all traffic sessions. Thus, it does not cause any prejudice to the desired traffic differentiation. The "logical association" between TCP's flow/congestion control and IEEE802.11's RTS/CTS and DCF helps to keep buffer (queue) sizes small and hence low losses.

From the geographical size (20 km × 20 km), it is obvious that the number of hops between the sender and the receiver during this simulation was 1, 2 or 3, at maximum. From Figure 7.1, the upper bounds on throughput are approximately 1300 kbps, 700 kbps and 450 kbps for 1, 2 and 3 hops, respectively. The performance degradation observed in the resultant average throughput (about 160 kbps) of all sessions is mainly caused by two factors: TCP's reaction to congestion and link breakage (node mobility) and the higher contention in the MAC layer due to more than just one traffic source trying to access the medium.

2. CBR/UDP Analysis

The same twelve scenario (movement) files as in the previous subsection were used here. The results are shown in Figures 7.7 through 7.9. The sending rates of the

CBR/UDP sources were defined as 160 kbps (average throughput resulting from TCP simulation) in order to guarantee congestion in the simulations.

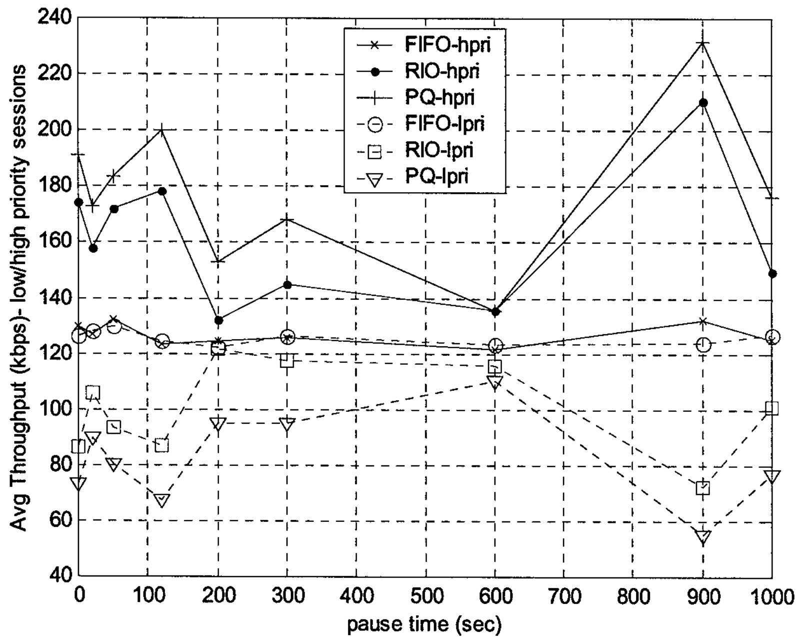


Figure 7.5 FTP/TCP: Average Throughput for Low and High Priority Sessions.

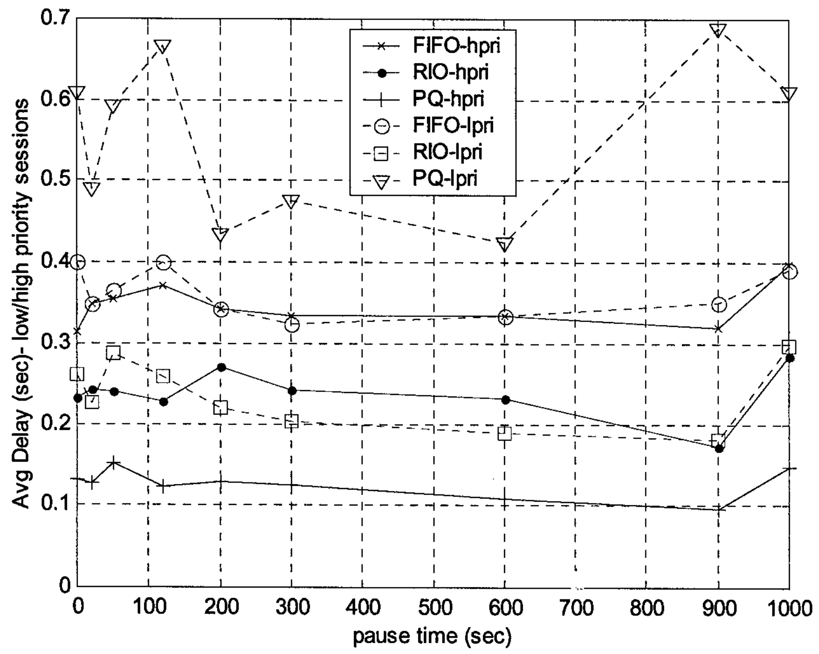


Figure 7.6 FTP/TCP: Average End-to-end Delay for Low and High Priority Sessions.

Figure 7.7 shows that both RED/RIO and PQ provide satisfactory traffic differentiation between the low and high priority types of traffic for the throughput metric. As expected, the FIFO scheme provides no differentiation.

However, as shown in Figure 7.8, the drawback of traffic differentiation in CBR/UDP is a severe increase in the loss rates, which are mainly due to buffer overflows. Unlike TCP, UDP does not offer any flow control. Also, the applications keep generating and sending traffic to the source nodes' buffer at a constant rate. These two factors combined with the contention for the media at the MAC layer cause the buffers to rapidly fill up and overflow, resulting in severe packet dropping. Although high priority packets are dropped less than the low priority ones, the loss rates are still significant for these packets (average of 16%).

The average end-to-end delay results shown in Figure 7.9 confirm the traffic differentiation for the PQ scheme in which high priority packets are delayed less than the low priority ones. Figure 7.9 also shows that no differentiation is realized for the FIFO scheme as expected. For the RED/RIO algorithm, low priority packets are delayed less than the high priority ones. The explanation for this is directly related to the loss rates depicted in Figure 7.8 and the probability-based dropping mechanism shown in Figure 5.6. Due to the high overall loss rates of CBR/UDP traffic, the few low priority packets that reach the destination are the ones below the $\theta_{\max}^{out}$ threshold. Since this value is less than the $\theta_{\min}^{in}$ threshold, the end-to-end delay for the "surviving" (not dropped) low priority packets (OUT) is expected to be lower than that for the high priority ones (IN).

3. TCP and UDP Combined

Two simulations were generated for this case: one with CBR/UDP having priority and the other with FTP/TCP having priority. The same twelve scenario (mobility) files as in the previous subsections were used.

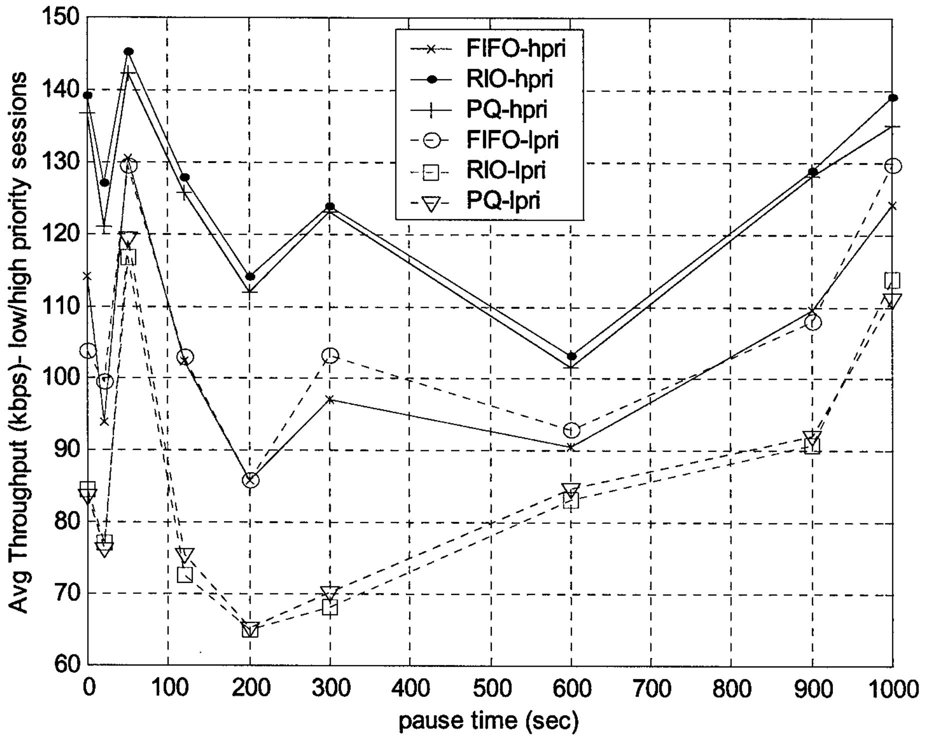


Figure 7.7 CBR/UDP: Average Throughput for Low and High Priority Sessions.

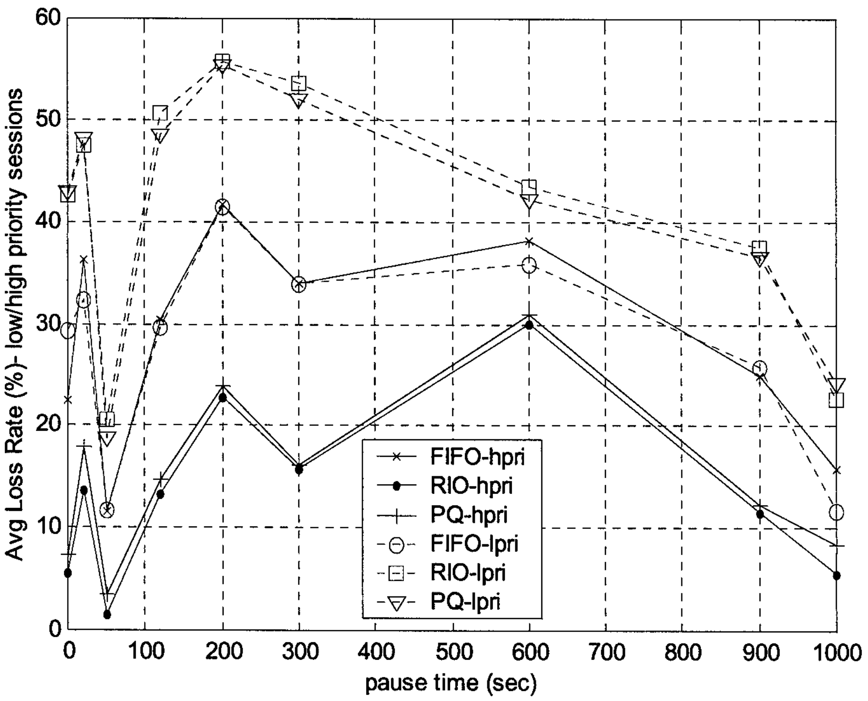


Figure 7.8 CBR/UDP: Average Loss Rate, Buffer Size = 50 packets.

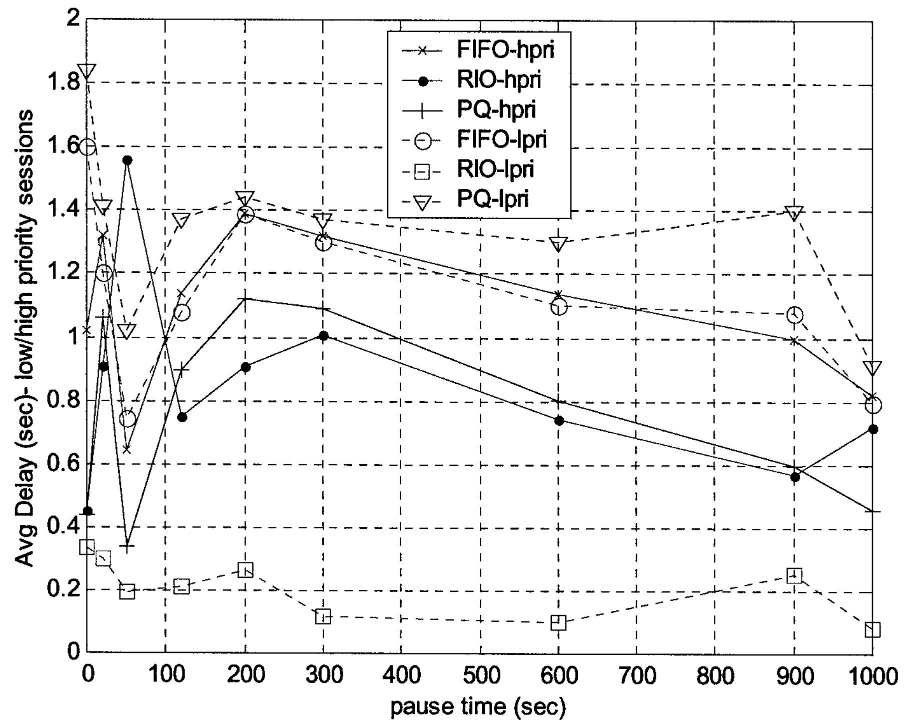


Figure 7.9 CBR/UDP: Average Delay for Low and High Priority Sessions.

a. UDP with priority

Eight sessions between randomly selected pairs of nodes were created. Four of these were FTP/TCP and the others were CBR/UDP. Each source node generated FTP/TCP and/or CBR/UDP traffic at the same time. CBR/UDP was defined to have high priority packets and its sending rate was kept at 160 kbps.

Figure 7.10 shows that, except in two high mobility cases (pause times of 200 and 600 seconds), UDP throughput was always higher than that of TCP. When PQ and RED/RIO were used to benefit the UDP traffic, TCP throughput performance degraded while that of UDPs improved. When RED/RIO buffer management was used, the threshold parameters ($\theta_{\min}^{out}$ and $\theta_{\max}^{out}$) and the dropping probability ($P_{\max}^{out}$) for the low priority TCP packets were selected in order to reduce the interaction between TCP flow/congestion control and the IEEE802.11 MAC layer. This resulted in a better throughput differentiation for RED/RIO compared to PQ.

Figure 7.11 shows that the average TCP end-to-end delay is lower than UDP's, even when TCP is defined as low priority. TCP's flow/congestion control and its interaction with the MAC 802.11 layer seems to be responsible for maintaining small queue sizes, and hence reduced average delays. In this simulation, the TCP and UDP sessions were randomly selected and all nodes were free to move, making it impossible to predict whether or not the high priority UDP sessions and the low priority TCP sessions had common intermediate nodes. It is clear that if this were the case, the PQ scheme applied in these common intermediate nodes, would certainly benefit the high-priority UDP sessions, causing less delay for this type of session.

Improvements in the end-to-end delay of the UDP high priority packets are obtained for both PQ and RED/RIO when compared to FIFO. For PQ, this improvement is mainly due to the re-positioning of the UDP packets at the head of the queues. On the other hand, for RED/RIO, the improvement is due to the more aggressive dropping of the TCP packets, which cuts waiting time for the UDP packets.

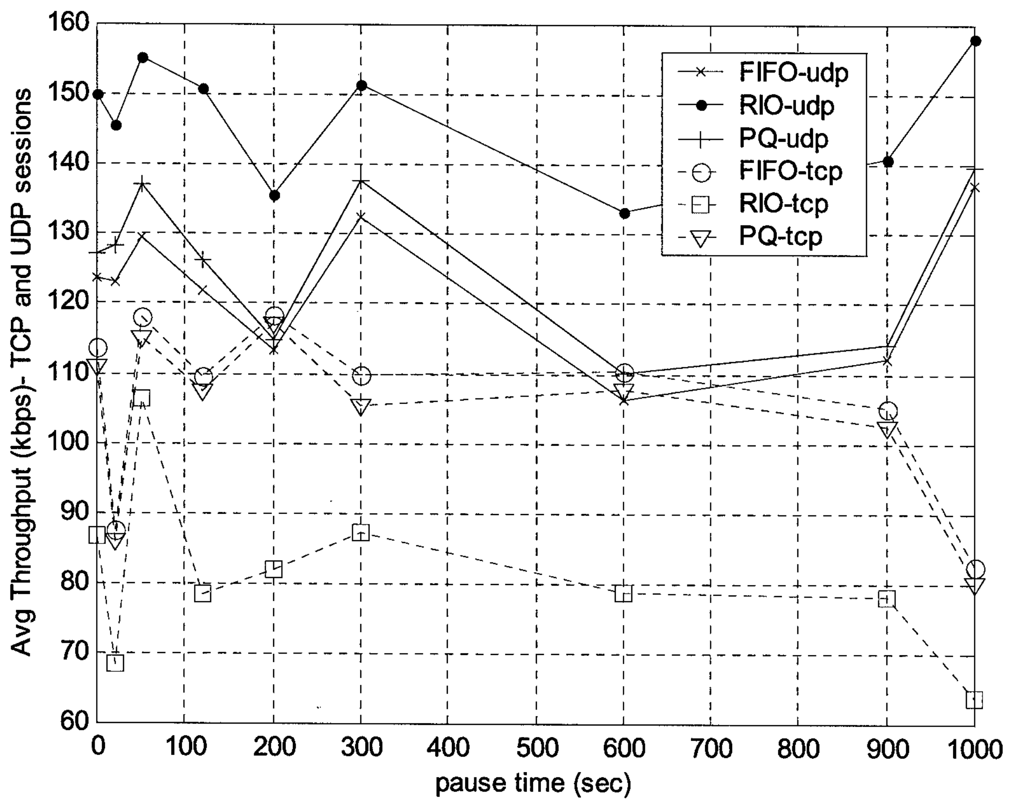


Figure 7.10 Average Throughput for TCP and UDP (High Priority) Combined Sessions.

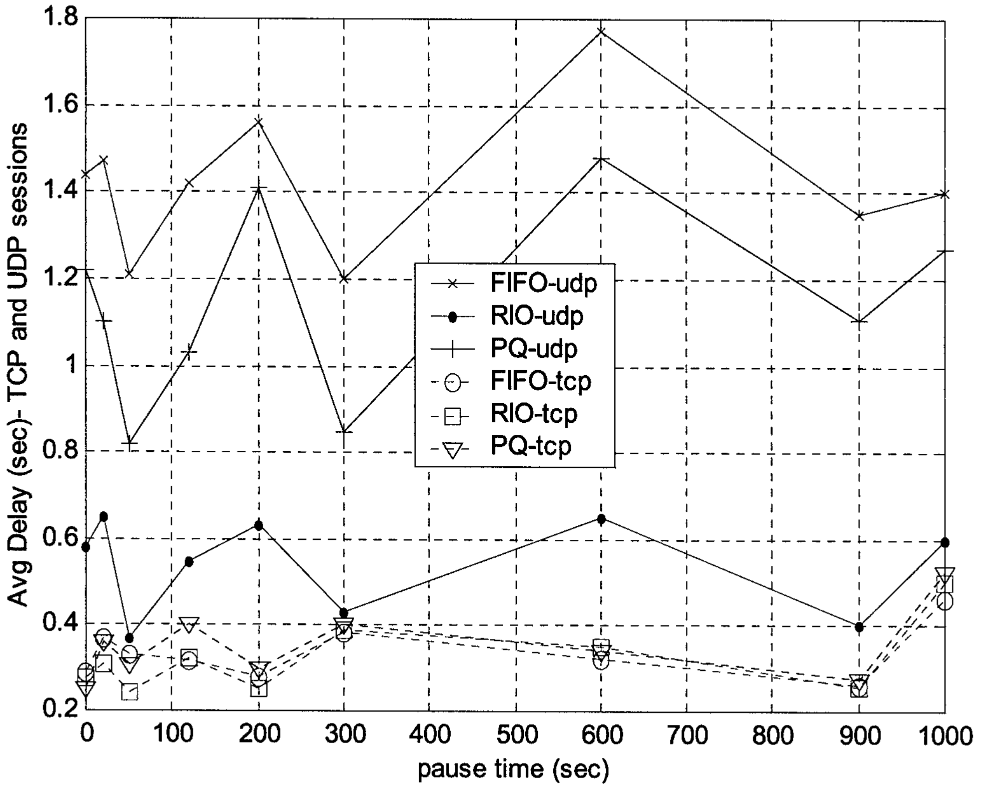


Figure 7.11 Average Delay for TCP and UDP (High Priority) Combined Sessions.

b. TCP with priority

Eight sessions between the same pairs of nodes as in the previous subsection were created. Four of these were TCP and the others were UDP. Each source node generates both TCP and UDP. CBR/UDP's sending rate was kept at 160 kbps, and now the FTP/TCP traffic was given priority.

Figure 7.12 shows that throughput differentiation between TCP and UDP traffic is clearly achieved at all mobility levels, and is more evident for the PQ scheme. Also, when PQ and RED/RIO were used to benefit the TCP traffic, the UDP throughput performance degraded while that of TCP improved. Compared to Figure 7.10, the improvements obtained when TCP was given priority were much more significant than when UDP was given priority.

For the end-to-end delay metric, the same trend could be verified. Results shown in Figure 7.13 demonstrate that end-to-end delays are low for TCP compared to UDP, even when not using any priority scheme (FIFO). TCP flow control and its interaction with the MAC802.11 layer were the factors responsible for this behavior. Low priority UDP packets that were kept in a mobile node's queue for a long time before being served increased the overall end-to-end delay significantly.

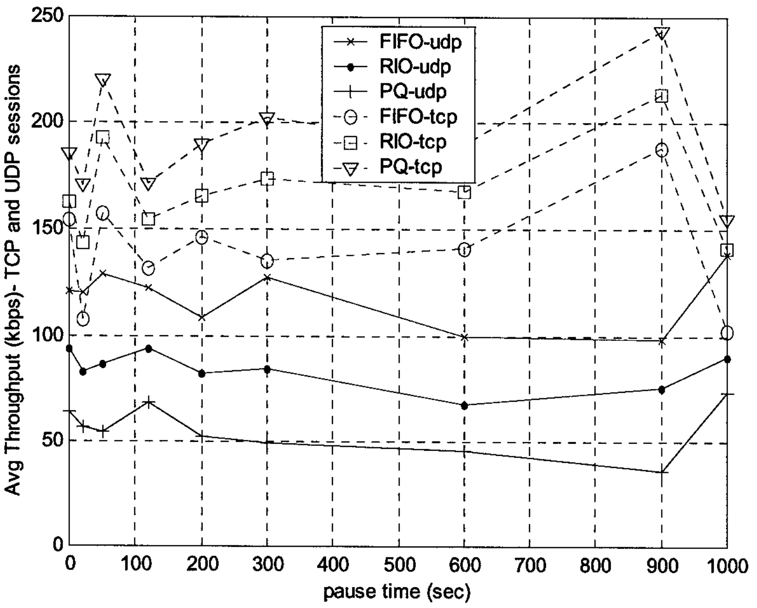


Figure 7.12 Average Throughput for TCP (High Priority) and UDP Combined Sessions.

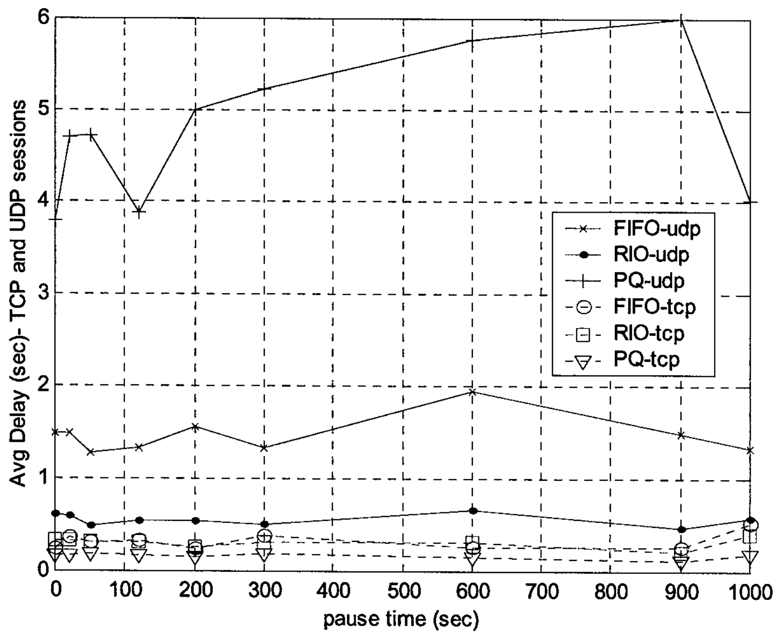


Figure 7.13 Average Delay for TCP (High Priority) and UDP Combined Sessions.

4. VBR (on/off Exponential) / UDP

Is this section, results for variable bit rate sources are presented. To simulate variable bit rate (VBR) traffic, exponential ON/OFF sources were defined and associated with the transport layer protocol UDP. In this case, packets are sent at a fixed rate during ON periods, and no packets are sent during OFF periods. Both ON/OFF periods are selected based on an exponential distribution.

The reason for using VBR was to more closely represent real multimedia types of traffic, such as voice. Since the traffic sources are not synchronized (packets are sent at different times), this simulation provides an insight into a real scenario where the offered network load varies much more than in the CBR/UDP case.

For this simulation, the average "on" (burst) and "off" (idle) times for the traffic generator was chosen to be 0.5 seconds. The sending rate during the "on" time was 300 kbps, which resulted in an average sending rate of approximately 140 kbps for each source. The same previous movement scenarios were used. The rates and the average "on" and "off " times were selected such that congestion was guaranteed to occur in all simulated scenarios. Figure 7.14 depicts the average throughput for the low and high priority sessions for the VBR/UDP type of traffic. Note that service differentiation is clearly achieved using both RED/RIO and PQ schemes.

Figure 7.15 shows the average loss rate for the VBR/UDP traffic. By comparing these results with those in Figure 7.8 for the CBR/UDP traffic, it is seen that the loss rates are lower for VBR/UDP. The high priority sessions show an average loss rate of 10%, instead of 16% as in the CBR/UDP case. The reason behind smaller losses for VBR/UDP is the less aggressive rate of traffic generation. Of course, UDP still does not offer any flow control. However, the "off" (idle) periods of time help the mobile nodes' buffers to alleviate packet dropping due to buffer overflow during the "on" times.

Results for the end-to-end delay metric are depicted in Figure 7.16. The trend is similar to the one obtained for the CBR/UDP traffic (see Figure 7.9). However, due to the less aggressive rate of traffic generation, VBR packets wait for a relatively short duration in the queues and hence are not delayed as much as in the CBR/UDP traffic.

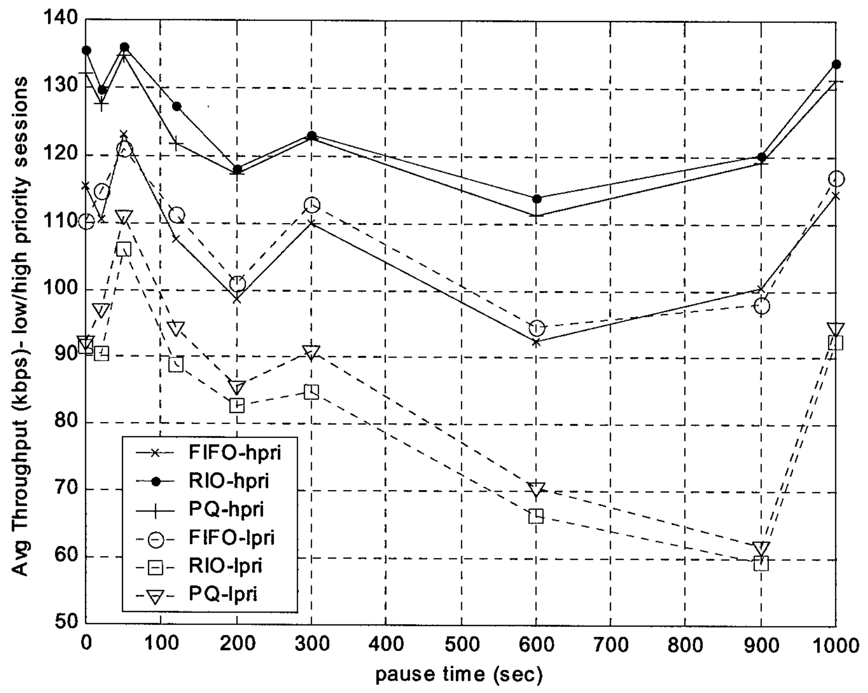


Figure 7.14 VBR/UDP: Average Throughput for Low and High Priority Sessions.

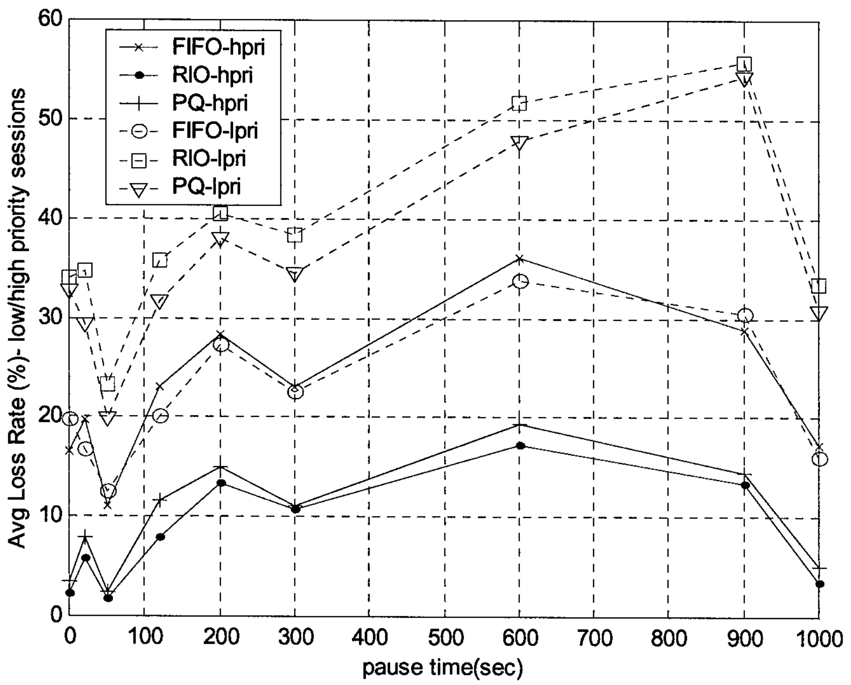


Figure 7.15 VBR/UDP – Average Loss Rate, Buffer Size = 50 packets.

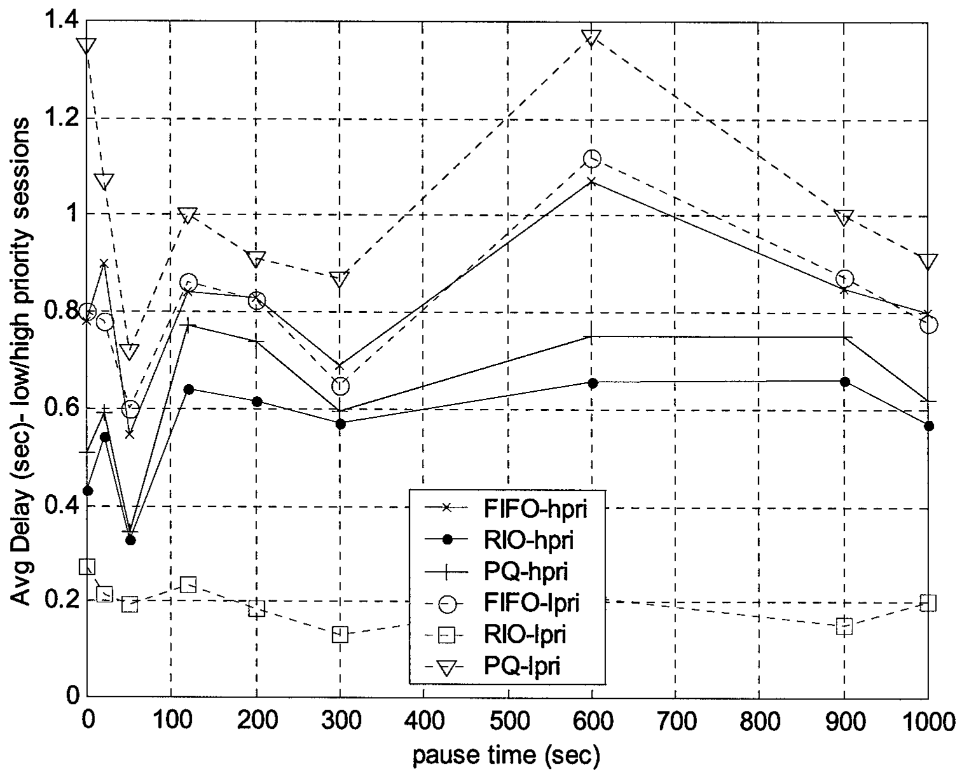


Figure 7.16 VBR/UDP: Average Delay for Low and High Priority Sessions.

5. FTP/TCP Throughput Over Large Ranges

In the following simulations, throughput performance of FTP/TCP in scenarios of different geographical sizes is presented. The ultimate goal is to show that the proposed QoS scheme is flexible and works well even for geographical situations in which connecting nodes are far apart. In addition, the following simulations will show that the QoS model reported in this thesis does not guarantee a connection between the nodes. However, if a physical wireless connection is established and congestion is presented, the model will be able to provide service differentiation in terms of throughput for a variety of scenarios. QoS results for CBR/UDP follow the same trends as these for FTP/TCP and will be in a following section.

a. 10 nodes, Geographical Area 20 km ×20 km

Figure 7.17 presents results of the instantaneous throughputs (measured at the receiver nodes) as a function of the simulation time. In this simulation, 10 nodes were used; node speeds were randomly selected between 0 and 20 m/s; a pause time of zero (highest mobility level) was adopted; and the geographical area was 20 km × 20 km. Four FTP/TCP sessions were generated in each simulation. In the first simulation, all four sessions had equal priority. In the following four simulations, one session per simulation was selected to have packets marked as high priority.

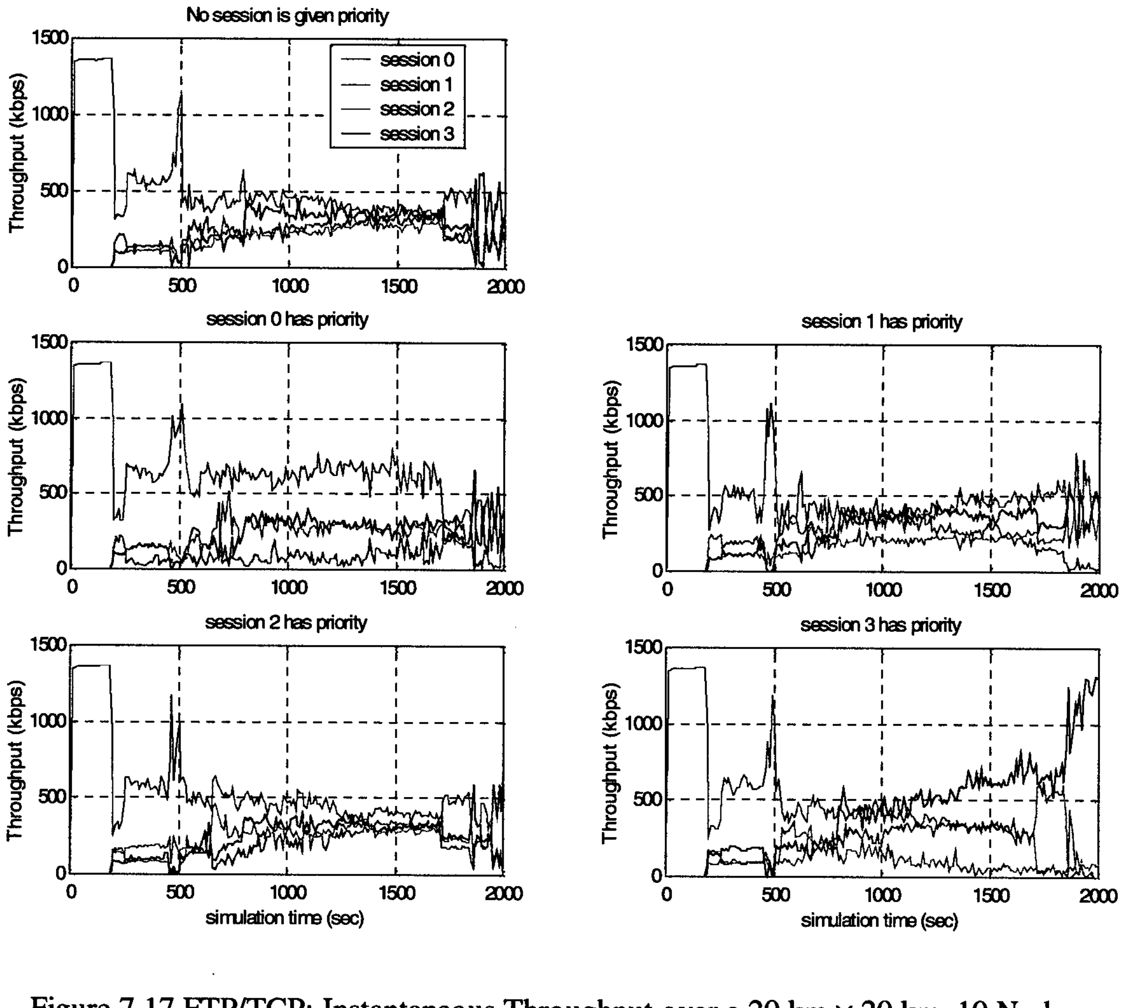


Figure 7.17 FTP/TCP: Instantaneous Throughput over a 20 km × 20 km, 10 Nodes, and High Mobility Scenario.

Figure 7.18 complements the results of Figure 7.17. In Figure 7.18, the average throughput for each session was computed and compared with the case of equal priority (packets of all sessions are marked as OUT) in order to demonstrate the relative improvement in throughput when priority is provided for a specific session (only packets of this session are marked as IN). When no session has priority, the TCP and 802.11 interaction forces the connections to take turns in "capturing" the channel. As a result, throughput is distributed with single-hop connections having a clear advantage over the others. By using the proposed QoS algorithms (traffic conditioning associated with queuing schemes, PQ or RED/RIO), it is noted that all sessions achieve better throughput when given priority. The percentage improvement is dependent on a combination of factors, mainly mobility and interference from neighboring nodes trying to access the channel.

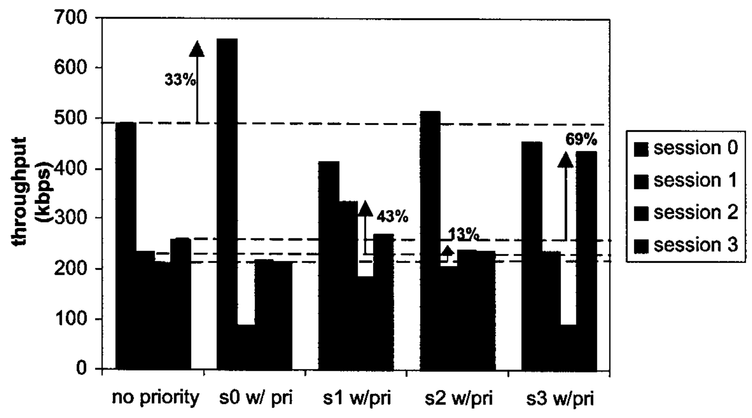


Figure 7.18 FTP/TCP Relative Improvement in the Average Throughput over a 20 km × 20 km, 10 Nodes, and High Mobility Scenario.

b. 10 nodes, Geographical Area 40 km × 40 km

Figure 7.19 shows plots of the instantaneous throughput as a function of the simulation time. The only difference from the simulation parameters of Figure 7.17 was the larger geographical area of 40 km × 40 km instead of 20 km × 20 km. As shown in Figure 7.19, in this larger scenario, there were periods of time (from 0 to about 1100 seconds) when no connection was possible due to the unreachability of the nodes attempting to establish FTP/TCP sessions.

As expected, Figure 7.20 shows that the average throughput for each session tends to be lower when compared to the 20 km × 20 km case. However, relative throughput performance improvement can still be achieved as shown in Figure 7.20.

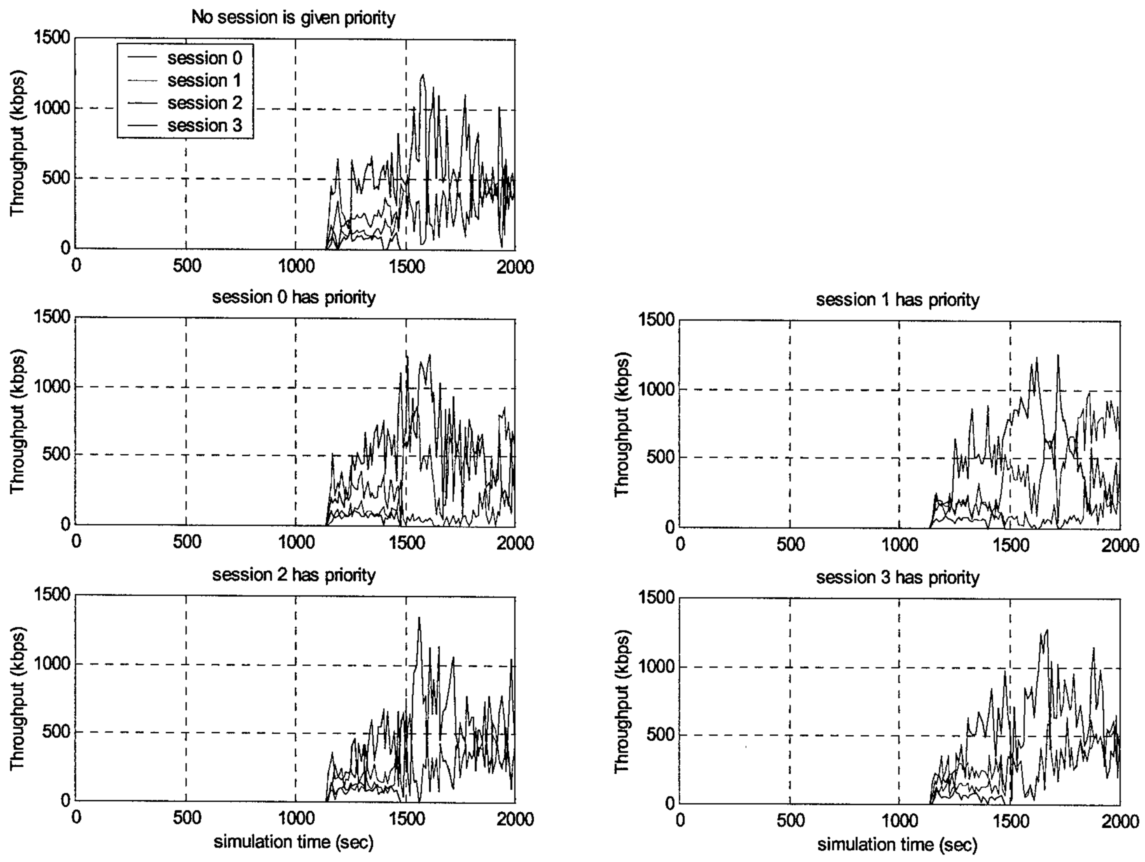


Figure 7.19 FTP/TCP Instantaneous Throughput over a 40 km × 40 km, 10 Nodes, and High Mobility Scenario.

c. 50 nodes, Geographical Area 50 km × 50 km

For this simulation, the number of nodes and the geographical area were increased. The remaining simulation parameters were kept the same. By populating the geographical area with more nodes, the probability of not having a connection, even with a large area, was reduced. Results depicted in Figure 7.21 support this observation.

Finally, Figure 7.22 shows that relative improvements in throughput are achieved for this case, following the same trends as before.

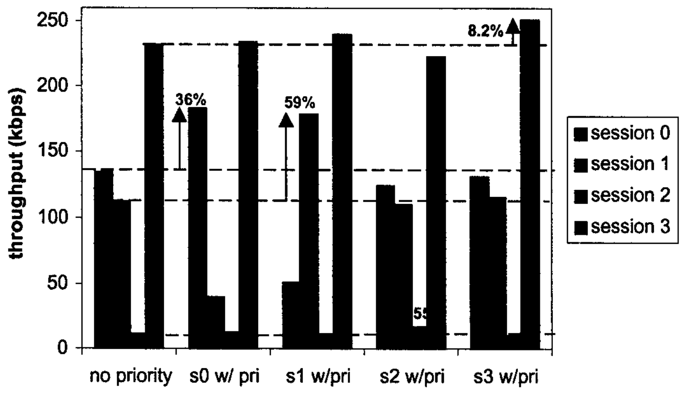


Figure 7.20 FTP/TCP Relative Improvement in the Average Throughput over a 40 km × 40 km, 10 Nodes, and High Mobility Scenario.

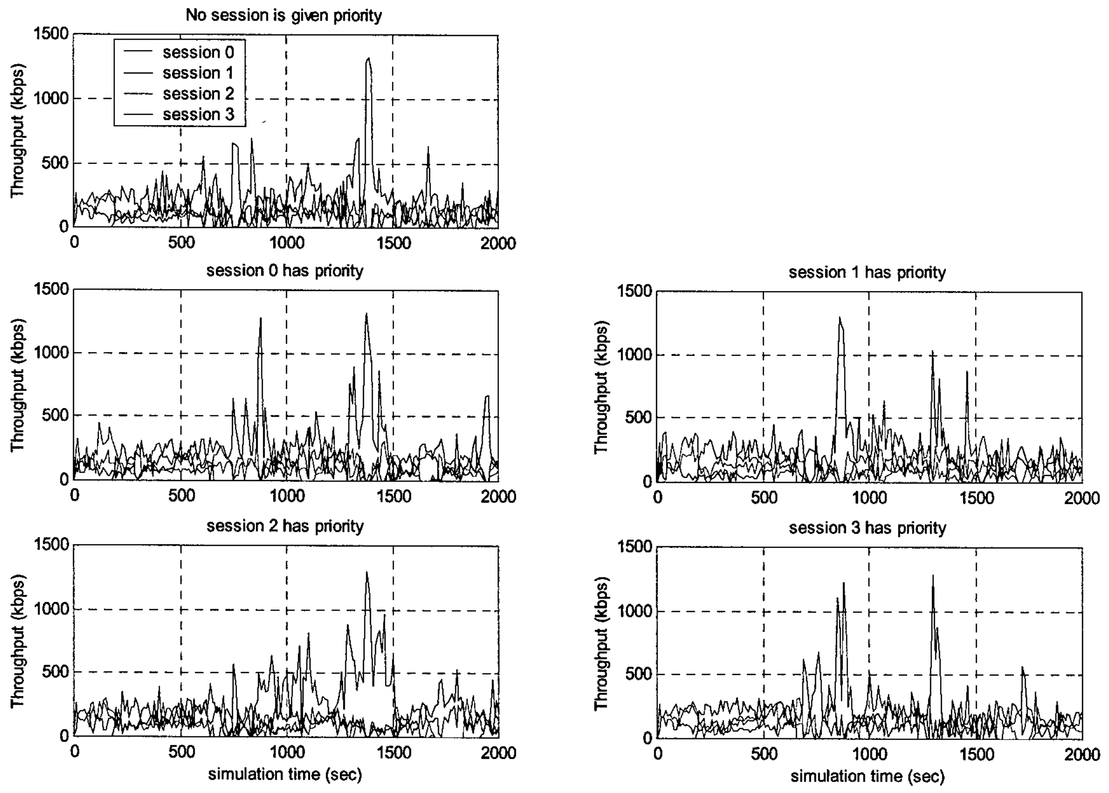


Figure 7.21 FTP/TCP Instantaneous Throughput over a 50 km × 50 km, 50 nodes, and High Mobility Scenario.

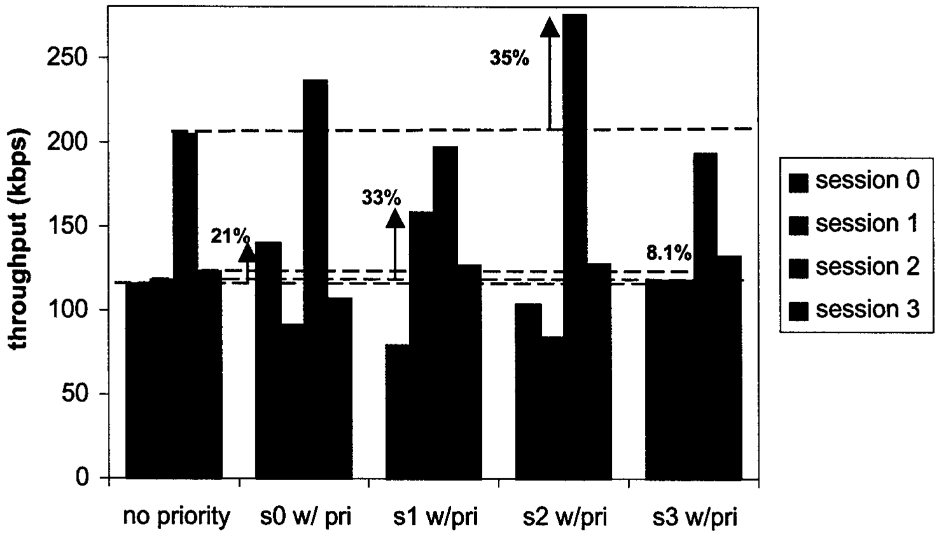


Figure 7.22 FTP/TCP Relative Improvement in the Average Throughput over a 50 km × 50 km, 50 nodes, and High Mobility Scenario.

6. CBR/UDP Throughput Over Large Ranges

In the following simulations, throughput performance of CBR/UDP traffic in scenarios of different geographical sizes will be presented. The goal is to show that, for this type of traffic, the QoS model works well in providing service differentiation in terms of throughput.

Concurrent CBR/UDP and FTP/TCP types of traffic were inserted in the network. In the first simulation, for each scenario, none of the traffic sessions had priority. In the second simulation, one CBR/UDP session was selected to have priority. RED/RIO buffer management was implemented in all mobile nodes. The threshold parameters ($\theta_{\min}^{out}$ and $\theta_{\max}^{out}$) and the dropping probability ($P_{\max}^{out}$) for the low priority TCP packets were selected in order to reduce the interaction between the TCP flow/congestion control and the IEEE802.11 MAC layer.

a. 10 nodes, Geographical Area 20 km × 20 km

In this simulation, 10 nodes were used; node speeds were randomly selected to be 0 and 20 m/s; a pause time of zero (highest mobility level) was adopted; and the geographical area was 20 km × 20 km. One CBR/UDP 160 kbps session and three FTP/TCP sessions were used in each simulation. In the first simulation, the packets of all four sessions were marked as OUT, i.e., no priority was given. In the following simulation, the CBR/UDP session was selected to have packets marked as IN (high priority).

As depicted in Figure 7.23, a relative improvement of 30% for the CBR/UDP is achieved when priority was given for this type of traffic. Two of the TCP sessions had their throughput relatively decreased (12% and 75%) while the sum of the throughput of all four sessions stayed approximately the same in both simulations.

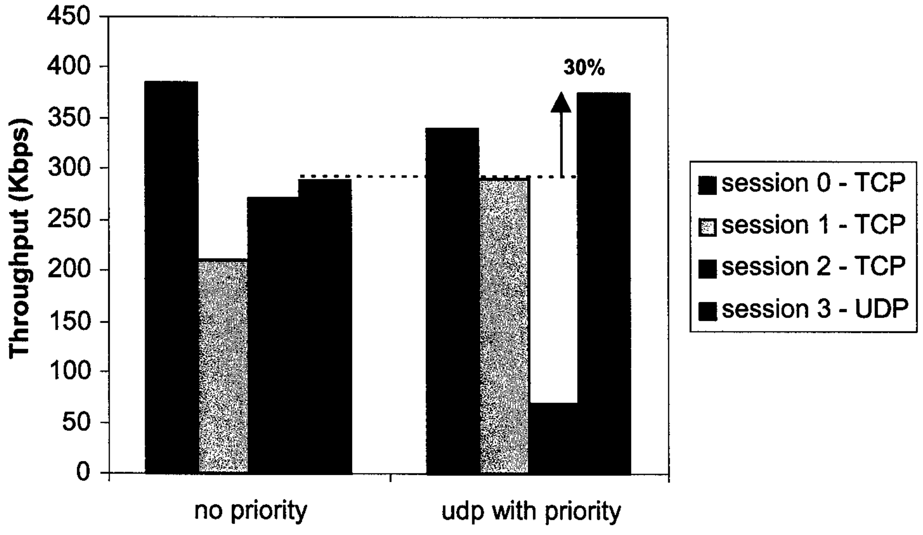


Figure 7.23 CBR/UDP Relative Improvement in the Average Throughput over a 20 km × 20 km, 10 Nodes, and High Mobility Scenario.

b. 10 nodes, Geographical Area 40 km × 40 km

In this larger geographical scenario, there were periods during which no connection was possible due to the unreachability of the nodes attempting to establish either TCP or UDP sessions. Thus, the average throughput for each session tended to be lower when compared to the 20 km × 20 km case (see Figure 7.23). However, relative throughput performance improvements could still be achieved for the high-priority CBR/UDP traffic as shown in Figure 7.24.

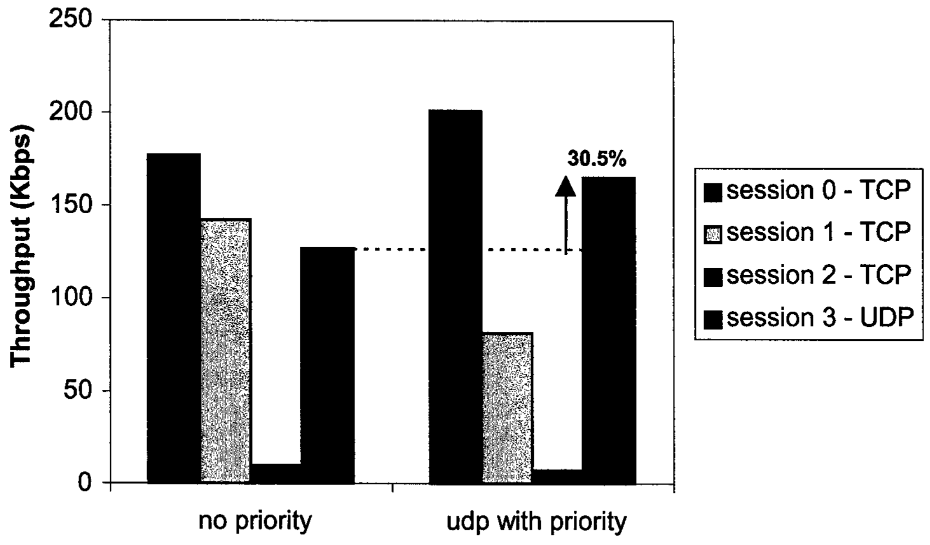


Figure 7.24 CBR/UDP Relative Improvement in the Average Throughput over a 40 km × 40 km, 10 Nodes, and High Mobility Scenario.

c. 50 nodes, Geographical Area 50 km × 50 km

In this simulation, the number of nodes and the size of the geographical area were further increased. The remaining simulation parameters were kept the same. By populating the area with more nodes, the probability of not having a connection was reduced. Figure 7.25 shows relative improvement in the throughput of the CBR/UDP traffic for this case following the same trend as before.

E. VOICE APPLICATION

The main goal of this section is to show the effects of packet dropping in a simple voice application. By doing this, it is possible to demonstrate the improvement in signal quality that can be achieved at the destination for high priority sessions when compared to the low priority ones.

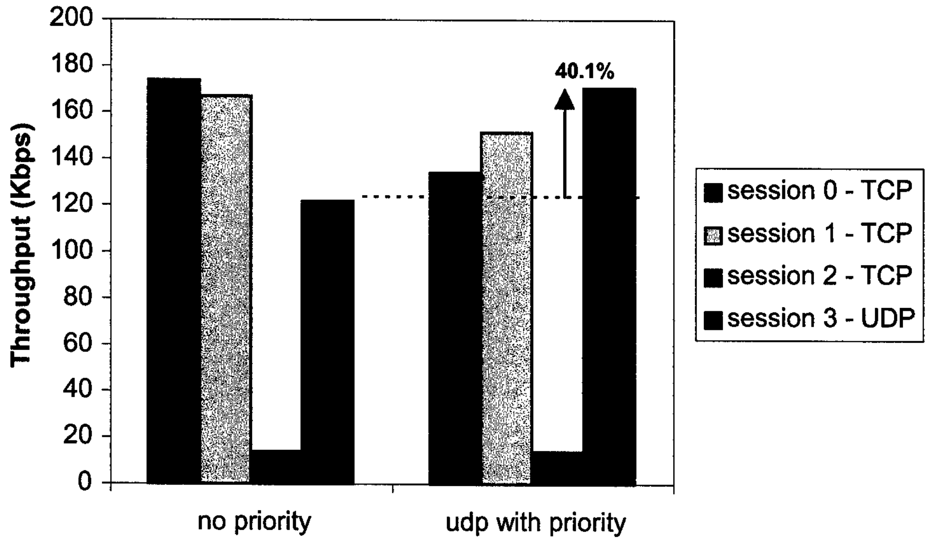


Figure 7.25 CBR/UDP Relative Improvement in the Average Throughput over a 50 km × 50 km, 50 Nodes, and High Mobility Scenario.

In fixed IP networks, real-time applications such as voice are normally associated with UDP. This is because TCP flow/congestion control causes degradation in throughput performance and adds delay, although it improves the connection's reliability. It is assumed that loss of some UDP packets is tolerated by voice traffic due to the redundancy present in the signal and the human ear's psycho-acoustic capabilities that compensate for some loss of quality. Therefore, by using QoS protocols available for fixed IP networks, such as DiffServ or IntServ with RSVP, priority or bandwidth reservation can be applied to the UDP real-time traffic.

In MANETs, however, previous simulation results showed that in a congested wireless ad-hoc environment, UDP traffic could sometimes face heavy packet losses, mainly due to buffer overflows. If these losses happen in bursts, severe degradation in the digitized (sampled) voice traffic can occur. As a result, the burst size (the number of packets dropped consecutively) is an important factor in determining the quality of the voice traffic at the receiver end.

1. Packet Burst Loss

In order to investigate the packet-dropping problem, a 60-second speech message was recorded. The digitized speech was read into MATLAB in the form of a vector containing the sampled speech signal values. Next, some of the previously simulated VBR/UDP sessions were selected and their packet loss behavior was measured. The VBR/UDP sessions were simulated for 1000 seconds in a congested environment in order to allow the realistic effects of node movement and contention for channel access to play a role in the packet loss behavior.

Table 7.2 shows the frequency of occurrence of different sizes (B) of burst packet losses for several sessions. Each session has a different level of total packet loss, which can be due to these sessions having different priorities. The loss rates in the first three rows of Table 7.2 can be associated with high priority sessions. A value of 12 for $B \leq 5$ packets and loss rate = 1% means that, for the specific simulated VBR/UDP session, which has a total loss rate of 1%, a burst error of 1 to 5 packets occurs 12 times during the simulation time (1000 seconds). If the packet rate is 20 packets/second, a total of 20,000 packets are sent during 1000 seconds. Thus, having a burst of 5 packets means that we lose about 250 milliseconds of the original speech. In other words, if the voice is sampled at 8000 samples/second, a packet holds 400 samples. Losing 5 packets means losing 2000 samples, which is equivalent to 250 milliseconds.

	$1 \leq B \leq 5$ pkts	**$5 < B \leq 10$ pkts**	**$10 < B \leq 20$ pkts**	**$B > 20$ pkts**
Loss rate = 1%	12	7	0	1
Loss rate = 3%	16	0	0	3
Loss rate = 5%	17	3	0	4
Loss rate = 10%	291	29	31	3
Loss rate = 20%	344	41	24	30
Loss rate = 30%	976	129	46	14

Table 7.2 Frequency of Burst Packet Loss for Different Levels of Session Loss Rates.

The impact of different sizes of burst errors can be demonstrated by mapping the 60-second digitized speech onto simulated intervals. First, a simulated interval, having burst errors less than 5 packets ($1 \leq B \leq 5$ packets), is selected. After applying the burst error pattern to the digitized speech, it is reconstructed in MATLAB, and the resulting voice packet sequence is played back. The procedure is repeated for the other error cases.

It is observed that while small burst sizes are almost imperceptible, long burst sizes made comprehension of some segments of the original speech difficult. The spreading of the burst errors among the sampled speech can improve the quality of the reconstructed signal. This indicates that the use of interleaving of packets or flow control techniques applied to VBR/UDP sources may help to decrease the effects of burst errors.

F. SUMMARY

Several simulations were performed to evaluate the proposed QoS algorithms. Different types of traffic (FTP/TCP, CBR/UDP and VBR/UDP) were tested using various mobility levels and geographical areas of different size (20 km $\times$ 20 km, 40 km $\times$ 40 km and 50 km $\times$ 50 km). Simulation results have shown that in the presence of congestion, differentiation in average throughput and delay between high and low priority sessions was achieved for all types of traffic. In TCP, retransmissions and flow/congestion control guaranteed low loss rates. On the other hand, in UDP, this differentiation was followed by significant packet loss rates due to buffer overflows.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSION

A. SUMMARY

The objective of this thesis was to develop algorithms and schemes that could be applied in mobile nodes to provide Quality of Service (QoS) in a mobile ad-hoc network (MANET) environment.

The issues related to QoS in the different layers of the protocol stack were thoroughly investigated in Chapter III. In Chapter IV, two QoS protocols (IntServ/RSVP and DiffServ) for fixed IP networks were reviewed and the limitations/advantages of their application to MANETs were presented. The algorithms and schemes for traffic conditioning (metering, marking and shaping), buffer management (RED/RIO) and packet scheduling with priority (Priority Queuing) were detailed in Chapter V. In Chapter VI, NS2 was introduced as the simulation tool used to evaluate the proposed algorithms and schemes. The modifications to the original NS2 functions were also detailed in this chapter. In Chapter VII, simulations were performed under different military scenarios and for different types of traffic in order to verify the advantages and limitations of the suggested approaches.

B. SIGNIFICANT RESULTS

The MAC layer based on the IEEE 802.11 standard was originally designed to be used in LAN environments (ranges less than 250 m). In this work, by changing the Short Interframe Space (SIFS) parameter, we were able to apply this protocol for larger ranges (up to 10 km), which allowed simulation of more realistic military scenarios.

In fixed IP networks, DiffServ traffic conditioning is implemented in the *ingress* nodes, generally located at the network boundaries. In addition to that, the queuing and buffer management schemes to provide the desired differentiated per-hop-behavior (PHB) are restricted to the *interior* nodes. In MANETs, since the mobile nodes can have simultaneous multiple roles (*ingress, interior* and *destination*), it was found that traffic conditioning must be implemented in all mobile nodes acting as source (*ingress*) nodes and the queuing and buffer management schemes must be performed by all mobile nodes.

Based on the results of Chapter VII, the traffic-conditioning function must have knowledge of the actual number of hops between the communicating nodes and must adjust its traffic profile parameters accordingly to achieve consistent relative traffic differentiation. If this information is not provided by the routing protocol, using one-hop traffic profile parameters for the one-hop case, if not ideal, can at least guarantee traffic differentiation

The buffer management (RED/RIO) scheme provided service differentiation by using different dropping probabilities for the low and high priority packets. Priority Queuing (PQ) used packet scheduling to achieve the same objective. In the presence of congestion, traffic conditioning associated with both Priority Queuing (PQ) or buffer management (RED/RIO) applied on a per-packet basis in each mobile node resulted in a flexible scheme that can be used to provide QoS in MANETs, despite the mobility levels and geographical areas.

Results for FTP/TCP traffic were more satisfactory due to the interaction of TCP flow/congestion control in conjunction with the MAC layer functionality. The percentage of TCP retransmissions was found to be very low in all simulations. Thus, although some performance degradation can be expected, the reliability obtained for this type of traffic indicates its potential use for real-time traffic when congestion is present in the medium.

Service differentiation was also obtained for VBR/UDP and CBR/UDP. Nevertheless, the resulting high loss rates make the direct application of the QoS model for this type of traffic less adequate. The results for VBR/UDP, when compared to CBR/UDP, showed that some form of flow control for UDP traffic could decrease the loss rates, keeping the benefits of the desired traffic differentiation.

The demonstration of speech packet transmission indicated that while small error burst sizes were almost imperceptible, long error burst sizes made comprehension of some segments of the original speech difficult. Spreading of the burst errors among the sampled speech could have improved the quality of the reconstructed signal. This indicates that the use of interleaving of packets or flow control techniques may be applied to VBR/UDP sources to decrease the effects of burst errors.

C. SUGGESTIONS FOR FUTURE WORK

In this work, only two traffic classes (IN and OUT) were defined to implement service differentiation. This can be extended by introducing more granularity in the levels of differentiation. Additional levels may allow better distinction among classes with specific requirements.

Here, we adopted a simple physical layer model in which errors due to propagation effects (Gaussian noise and fading) in the wireless channel were ignored. A model that includes these effects would better represent the physical layer and would allow analysis of the impact of packet loss (due to errors in the medium) on the QoS.

An important issue for further investigation is the study of the influence of the IEEE 802.11 MAC layer on the QoS algorithms presented here. All the QoS analysis was performed in the presence of congestion. The CSMA/CA and the DCF algorithms embedded in IEEE 802.11 do not offer priority for any mobile node trying to access the medium. Modifications in the 802.11 MAC layer [24] or utilization of other MAC protocols, such as TDMA/FDMA and DAMA, could improve QoS by reserving the channel for a pre-defined period for a given mobile node.

CBR and VBR types of traffic using UDP were investigated in the presence of congestion in MANETs. The high loss rates due to buffer overflow indicate that some sort of flow control at the application layer level might help improve the UDP performance under congestion and hence would improve the desired traffic differentiation. A detailed investigation of the use of Real-time Transport Protocol (RTP) or RTP-like protocols could provide the desired flow control and thereby enhance the QoS performance.

The proposed QoS model assumed that the sender node, by using a traffic-conditioning scheme, determined the traffic generation rates. An interesting point to be investigated is the receiver-based conditioning using a similar approach. In the sender-based scheme adopted in this thesis, the sender node marks the packets without knowing if congestion actually exists in the network. Some new proposals in the literature [33] successfully addressed receiver-based QoS schemes for fixed TCP/IP networks by using the Explicit Congestion Notification bit in the TCP acknowledgment packets. This scheme can be extended to MANET environments to provide better control of the traffic generation rates and hence better utilize the available bandwidth.

APPENDIX A. TCL MAIN SCRIPT SIMULATION PROGRAM FILE

This appendix contains an example of a Tcl script used in one of the simulations. A Tcl script file resident in NS2 was modified for the specific purpose of running the different queue schemes with the main parameters listed previously in Table 7.1.

```
# Copyright (c) 1997 Regents of the University of California.
# All rights reserved.
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
#     1. Redistributions of source code must retain the above copyright
#     notice, this list of conditions and the following disclaimer.
#     2. Redistributions in binary form must reproduce the above
#     copyright notice, this list of conditions and the following
#     disclaimer in the documentation and/or other materials provided
#     with the distribution.
#     3. All advertising materials mentioning features or use of this
#     software must display the following acknowledgement: This product
#     includes software developed by the Computer Systems Engineering
#     Group at Lawrence Berkeley Laboratory.
#     4. Neither the name of the University nor of the Laboratory may
#     be used to endorse or promote products derived from this software
#     without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS"'
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
# THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR
# CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
# EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
# PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
# PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
# OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
# (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
# OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

#$Header: /usr/src/mash/repository/vint/ns-2/tcl/ex/wireless-test.tcl,
# v1.2 1999/04/22 18:53:50 Haldar Exp $
#
#
======================================================================
# Default Script Options
#
======================================================================
set dir [pwd]
catch "cd /ns-allinone-2.1b6/ns-2.1b6/simulation"
source fqmm.tcl ;# hannan files
source fqmmmisc.tcl ;# hannan files
catch "cd $dir"

Class Test/WirelessRIO - superclass RIOTest
```

```
set opt(chan)           Channel/WirelessChannel
set opt(prop)           Propagation/TwoRayGround
set opt(netif)          Phy/WirelessPhy
set opt(mac)            Mac/802_11
# three different queue schemes (FIFO, RED/RIO and PQ)
#set opt(ifq)           Queue/DropTail/PriQueue
#set opt(ifq)           Queue/RIO/PRIO
set opt(ifq)            Queue/DropTail/PinoutQueue
set opt(ll)             LL
set opt(ant)            Antenna/OmniAntenna
set opt(x)              40000 ;# X dimension of the topography
set opt(y)              40000 ;# Y dimension of the topography
# cp is traffic file and sc is movement file
set opt(cp)             "./leo-n10-tcp3-cbr1-diff"
set opt(sc)             "./leo-10np0stop2000-v20-40kmx40km-1"
set opt(ifqlen)         50    ;# max packets the queue can hold
set opt(seed)           0.0
set opt(start)          0.0
set opt(stop)           2000.0;# simulation time
set opt(tr)             out-test.tr ;# output trace file
set opt(rp)             dsr   ;# routing protocol script
set opt(lm)             "off" ;# log movement
set opt(nn)             10    ;# number of nodes
set opt(SessionNum)     4     ;# number of traffic sessions

#==========================================================================
set AgentTrace              ON
set RouterTrace             ON
set MacTrace                ON
LL set mindelay_            50us
LL set delay_               25us
LL set bandwidth_           0     ;# not used
LL set off_prune_           0     ;# not used
LL set off_CtrMcast_        0     ;# not used
Agent/Null set sport_       0
Agent/Null set dport_       0
Agent/CBR set sport_        0
Agent/CBR set dport_        0
Agent/TCPSink set sport_    0
Agent/TCPSink set dport_    0
Agent/TCP set sport_        0
Agent/TCP set dport_        0

#Queue/RIO/PRIO set Prefer_Routing_Protocols    1
#Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 10 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
# average height of ship antenna
Antenna/OmniAntenna set Z_ 10
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

# Initialize the SharedMedia interface with parameters to make
# it work like the 914MHz Lucent WaveLAN DSSS radio interface
Phy/WirelessPhy set CPThresh_ 10.0
# change to adapt to ranges of 10km
Phy/WirelessPhy set CSThresh_ 1.0e-19
Phy/WirelessPhy set RXThresh_ 5.0e-11
Phy/WirelessPhy set Rb_ 2*1e6
# change Pt power to 50W
Phy/WirelessPhy set Pt_ 50
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0
#=======================================================================
Node/MobileNode instproc getq {i} {
      $self instvar ifq_
      return $ifq_($i)
}
Test/WirelessRIO instproc usage { argv0 }  {
      puts "Usage: $argv0"
      puts "\tmandatory arguments:"
      puts "\t\t\[-x MAXX\] \[-y MAXY\]"
      puts "\toptional arguments:"
      puts "\t\t\[-cp conn pattern\] \[-sc scenario\] \[-nn nodes\]"
      puts "\t\t\[-seed seed\] \[-stop sec\] \[-tr tracefile\]\n"
}
Test/WirelessRIO instproc create-god { nodes } {
      global ns_ god_ tracefd
      set god_ [new God]
      $god_ num_nodes $nodes
}
Test/WirelessRIO instproc log-movement {} {
 global logtimer ns_ ns env
 set ns $ns_
 catch "source /ns-allinone-2.1b6/ns-2.1b6/tcl/mobility/timer.tcl"
 Class LogTimer -superclass Timer
 LogTimer instproc timeout {} {
 global opt node_;
 for {set i 0} {$i < $opt(nn)} {incr i} {
     $node_($i) log-movement
 }
      $self sched 0.1
 }
      set logtimer [new LogTimer]
      $logtimer sched 0.1
 }
#=======================================================================
# Main Program
#=======================================================================
Test/WirelessRIO instproc run {} {
      global opt argc argv env
      global ns_ god_ tracefd topo chan prop node_
      $self getopt $argc $argv
      set opt(vstarttime) [expr $opt(start)+20]
      $self instvar nf f
#
# Source External TCL Scripts
#
```

```
catch "source /ns-allinone-2.1b6/ns-2.1b6/tcl/lib/ns-mobilenode.tcl"
if { $opt(rp) !="" } {
catch "source /ns-allinone-2.1b6/ns-2.1b6/tcl/mobility/$opt(rp).tcl"
}
catch "source /ns-allinone-2.1b6/ns-2.1b6/tcl/lib/ns-cmutrace.tcl"
# do the get opt again in case the routing protocol file added some
# more options to look for
      $self getopt $argc $argv
      if { $opt(x) == 0 || $opt(y) == 0 } {
            usage $argv0
            exit 1
      }
      if {$opt(seed) > 0} {
            puts "Seeding Random number generator with $opt(seed)\n"
            ns-random $opt(seed)
      }
#
# Initialize Global Variables
#
      set ns_           [new Simulator]
      set chan    [new $opt(chan)]
      set prop    [new $opt(prop)]
      set topo    [new Topography]
      set tracefd [open $opt(tr) w]
      set nf [open nam-out-test.nam w]
      $ns_ namtrace-all-wireless $nf $opt(x) $opt(y)
      $ns_ trace-all $tracefd
      $topo load_flatgrid $opt(x) $opt(y)
      $prop topography $topo
#
# Create God
#
      $self create-god $opt(nn)
#
# log the mobile nodes movements if desired
#
      if { $opt(lm) == "on" } {
            $self log-movement
      }
#
#  Create the specified number of nodes $opt(nn) and "attach" them
#  to the channel.
#  Each routing protocol script is expected to have defined a proc
#  create-mobile-node that builds a mobile node and inserts it into the
#  array global $node_($i)
#
      if { [string compare $opt(rp) "dsr"] == 0} {
            for {set i 0} {$i < $opt(nn) } {incr i} {
                  dsr-create-mobile-node $i
            }
      } elseif { [string compare $opt(rp) "dsdv"] == 0} {
            for {set i 0} {$i < $opt(nn) } {incr i} {
                  dsdv-create-mobile-node $i
            }
      }
#enable node trace in nam
```

```
      for {set i 0} {$i < $opt(nn)} {incr i} {
            $node_($i) namattach $nf
# 20 defines the node size in nam, must adjust it according to your
scenario
            $ns_ initial_node_pos $node_($i) 20
      }
#
# Source the Connection and Movement scripts
#
      if { $opt(cp) == "" } {
            puts "*** NOTE: no connection pattern specified."
             set opt(cp) "none"
      } else {
            puts "Loading connection pattern..."
            source $opt(cp)
      }
      if { $opt(sc) == "" } {
            puts "*** NOTE: no scenario file specified."
                  set opt(sc) "none"
      } else {
            puts "Loading scenario file..."
            source $opt(sc)
            puts "Load complete..."
      }
#
# Tell all the nodes when the simulation ends
#
      for {set i } {$i < $opt(nn) } {incr i} {
            $ns_ at [expr $opt(stop)+0.000000001] "$node_($i) reset";
      }
      $ns_  at  [expr  $opt(stop)+0.01]  "puts  \"NS  EXITING...\"  ;  $ns_
halt"
      $ns_ at $opt(stop) "$self stop"
# information to be inserted in the output trace file
      puts $tracefd "M 0.0 nn $opt(nn) x $opt(x) y $opt(y) rp $opt(rp)"
      puts $tracefd "M 0.0 sc $opt(sc) cp $opt(cp) seed $opt(seed)"
      puts $tracefd "M 0.0 prop $opt(prop) ant $opt(ant)"

# set the rioq parameters at each node
      for { set i 0} {$i < $opt(nn)} {incr i} {
            set rioq_($i) [$node_($i) getq 0]
            $rioq_($i) set limit_ 50
            $rioq_($i) set in_minthresh_ 20
            $rioq_($i) set in_maxthresh_ 35
            $rioq_($i) set in_linterm_ 500
            $rioq_($i) set out_minthresh_ 0
            $rioq_($i) set out_maxthresh_ 3
            $rioq_($i) set out_linter_ 1
      }
      puts "Starting Simulation..."
      $ns_ run
}
global ourrun
set ourrun [new Test/WirelessRIO]
$ourrun run
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. OUTPUT TRACE FILE

This appendix contains an example of an output trace file generated by the NS2 simulator. Only the initial part of the output trace file is shown. The user can select the granularity of each trace file. In this case, the routing (RTR) and the application (AGT) layers were monitored. The user must then parse this output file using either Linux shell scripts or Perl scripts to collect the required statistics.

```
M 0.0 nn 10 x 40000 y 40000 rp dsr
M 0.0 sc ./leo-10np0stop2000-v20-40kmx40km-1
cp ./leo-n10-tcp3-cbr1-diff seed 0.0
M 0.0 prop Propagation/TwoRayGround ant Antenna/OmniAntenna
s 6.989893944 _2_ AGT  --- 0 tcp 1460 [0 0 0 0] ------- [2:1 4:0 32 0]
[0 0] 0 2
r 6.989893944 _2_ RTR  --- 0 tcp 1460 [0 0 0 0] ------- [2:1 4:0 32 0]
[0 0] 0 2
s 6.998200635 _2_ RTR  --- 1 DSR 24 [0 0 0 0] ------- [2:255 4:255 32
0] 1 [1 1] [0 1 0 0->0] [0 0 0 0->0]
r 6.998666865 _1_ RTR  --- 1 DSR 24 [0 ffffffff 2 800] ------- [2:255
4:255 32 0] 1 [1 1] [0 1 0 0->0] [0 0 0 0->0]
r 6.998688330 _7_ RTR  --- 1 DSR 24 [0 ffffffff 2 800] ------- [2:255
4:255 32 0] 1 [1 1] [0 1 0 0->0] [0 0 0 0->0]
s 7.034132271 _2_ RTR  --- 2 DSR 24 [0 0 0 0] ------- [2:255 4:255 32
0] 1 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.034598501 _1_ RTR  --- 2 DSR 24 [0 ffffffff 2 800] ------- [2:255
4:255 32 0] 1 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.034619966 _7_ RTR  --- 2 DSR 24 [0 ffffffff 2 800] ------- [2:255
4:255 32 0] 1 [1 2] [0 2 0 0->0] [0 0 0 0->0]
f 7.035446229 _1_ RTR  --- 2 DSR 32 [0 ffffffff 2 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.035944459 _2_ RTR  --- 2 DSR 32 [0 ffffffff 1 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.035967004 _7_ RTR  --- 2 DSR 32 [0 ffffffff 1 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.035968006 _6_ RTR  --- 2 DSR 32 [0 ffffffff 1 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.035969597 _5_ RTR  --- 2 DSR 32 [0 ffffffff 1 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.035971520 _4_ RTR  --- 2 DSR 32 [0 ffffffff 1 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0] [0 0 0 0->0]
f 7.036488954 _5_ RTR  --- 2 DSR 44 [0 ffffffff 1 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.037042409 _6_ RTR  --- 2 DSR 44 [0 ffffffff 5 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.037046517 _8_ RTR  --- 2 DSR 44 [0 ffffffff 5 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.037047930 _4_ RTR  --- 2 DSR 44 [0 ffffffff 5 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.037056252 _3_ RTR  --- 2 DSR 44 [0 ffffffff 5 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
```

```
r 7.037060323 _1_ RTR  --- 2 DSR 44 [0 fffffff 5 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.037062157 _9_ RTR  --- 2 DSR 44 [0 fffffff 5 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
s 7.039747103 _4_ RTR  --- 4 DSR 52 [0 0 0 0] ------- [4:255 2:255 255
5] 4 [0 2] [1 2 4 2->4] [0 0 0 0->0]
f 7.042053983 _3_ RTR  --- 2 DSR 60 [0 fffffff 5 800] ------- [2:255
4:255 32 0] 4 [1 2] [0 2 0 0->0] [0 0 0 0->0]
f 7.042389806 _7_ RTR  --- 2 DSR 32 [0 fffffff 2 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.042687931 _5_ RTR  --- 4 DSR 52 [db 5 4 800] ------- [4:255 2:255
255 5] 4 [0 2] [1 2 4 2->4] [0 0 0 0->0]
f 7.042687931 _5_ RTR  --- 4 DSR 52 [db 5 4 800] ------- [4:255 2:255
255 1] 4 [0 2] [1 2 4 2->4] [0 0 0 0->0]
f 7.043051990 _6_ RTR  --- 2 DSR 44 [0 fffffff 1 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.043869840 _5_ RTR  --- 2 DSR 44 [0 fffffff 6 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.043877643 _9_ RTR  --- 2 DSR 44 [0 fffffff 6 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
f 7.043935334 _9_ RTR  --- 2 DSR 60 [0 fffffff 5 800] ------- [2:255
4:255 32 0] 4 [1 2] [0 2 0 0->0] [0 0 0 0->0]
s 7.044807167 _4_ RTR  --- 3 DSR 44 [0 0 0 0] ------- [4:255 2:255 255
1] 3 [0 2] [1 2 3 2->4] [0 0 0 0->0]
r 7.045048792 _0_ RTR  --- 2 DSR 32 [0 fffffff 7 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.045051573 _2_ RTR  --- 2 DSR 32 [0 fffffff 7 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.045052653 _1_ RTR  --- 2 DSR 32 [0 fffffff 7 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0 [0 0 0 0->0]
r 7.045057272 _9_ RTR  --- 2 DSR 32 [0 fffffff 7 800] ------- [2:255
4:255 32 0] 2 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.045800801 _0_ RTR  --- 2 DSR 60 [0 fffffff 9 800] ------- [2:255
4:255 32 0] 4 [1 2] [0 2 0 0->0] [0 0 0 0->0]
f 7.046993129 _8_ RTR  --- 2 DSR 60 [0 fffffff 5 800] ------- [2:255
4:255 32 0] 4 [1 2] [0 2 0 0->0] [0 0 0 0->0]
f 7.047149434 _0_ RTR  --- 2 DSR 44 [0 fffffff 7 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.047613655 _3_ RTR  --- 2 DSR 60 [0 fffffff 8 800] ------- [2:255
4:255 32 0] 4 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.047614691 _5_ RTR  --- 2 DSR 60 [0 fffffff 8 800] ------- [2:255
4:255 32 0] 4 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.047620512 _4_ RTR  --- 2 DSR 60 [0 fffffff 8 800] ------- [2:255
4:255 32 0] 4 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.047624406 _6_ RTR  --- 2 DSR 60 [0 fffffff 8 800] ------- [2:255
4:255 32 0] 4 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.048966003 _9_ RTR  --- 2 DSR 44 [0 fffffff 0 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.048978803 _6_ RTR  --- 2 DSR 44 [0 fffffff 0 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.048981387 _7_ RTR  --- 2 DSR 44 [0 fffffff 0 800] ------- [2:255
4:255 32 0] 3 [1 2] [0 2 0 0->0] [0 0 0 0->0]
r 7.052771176 _1_ RTR  --- 4 DSR 52 [db 1 5 800] ------- [4:255 2:255
255 1] 4 [0 2] [1 2 4 2->4  [0 0 0 0->0]
f 7.052771176 _1_ RTR  --- 4 DSR 52 [db 1 5 800] ------- [4:255 2:255
255 2] 4 [0 2] [1 2 4 2->4] [0 0 0 0->0]
```

```
s 7.054209633 _4_ RTR  --- 5 DSR 60 [0 0 0 0] ------- [4:255 2:255 255
8] 5 [0 2] [1 2 5 2->4] [0 0 0 0->0]
r 7.055404921 _1_ RTR  --- 3 DSR 44 [db 1 4 800] ------- [4:255 2:255
255 1] 3 [0 2] [1 2 3 2->4] [0 0 0 0->0]
f 7.055404921 _1_ RTR  --- 3 DSR 44 [db 1 4 800] ------- [4:255 2:255
255 2] 3 [0 2] [1 2 3 2->4] [0 0 0 0->0]
D 7.055404921 _1_ IFQ  ARP 4 DSR 52 [db 1 1 800] ------- [4:255 2:255
255 2] 0 [0 0] [1 0 4 2->2] [0 0 0 0->0]
r 7.061111029 _2_ RTR  --- 3 DSR 44 [db 2 1 800] ------- [4:255 2:255
255 2] 3 [0 2] [1 2 3 2->4] [0 0 0 0->0]
s 7.061111029 _2_ RTR  --- 0 tcp 1492 [0 0 0 0] ------- [2:1 4:0 32 1]
[0 0] 0 2
r 7.068290718 _1_ RTR  --- 0 tcp 1492 [db 1 2 800] ------- [2:1 4:0 32
1] [0 0] 1 2
f 7.068290718 _1_ RTR  --- 0 tcp 1492 [db 1 2 800] ------- [2:1 4:0 32
4] [0 0] 1 2
r 7.075391591 _4_ RTR  --- 0 tcp 1492 [db 4 1 800] ------- [2:1 4:0 32
4] [0 0] 2 2
r 7.075391591 _4_ AGT  --- 0 tcp 1492 [db 4 1 800] ------- [2:1 4:0 32
4] [0 0] 2 2
s 7.075391591 _4_ AGT  --- 6 ack 40 [0 0 0 0] ------- [4:0 2:1 32 0] [0
0] 0 2
r 7.075391591 _4_ RTR  --- 6 ack 40 [0 0 0 0] ------- [4:0 2:1 32 0] [0
0] 0 2
s 7.075391591 _4_ RTR  --- 6 ack 72 [0 0 0 0] ------- [4:0 2:1 32 1] [0
0] 0 2
r 7.077232463 _1_ RTR  --- 6 ack 72 [db 1 4 800] ------- [4:0 2:1 32 1]
[0 0] 1 2
f 7.077232463 _1_ RTR  --- 6 ack 72 [db 1 4 800] ------- [4:0 2:1 32 2]
[0 0] 1 2
r 7.078572152 _2_ RTR  --- 6 ack 72 [db 2 1 800] ------- [4:0 2:1 32 2]
[0 0] 2 2
r 7.078572152 _2_ AGT  --- 6 ack 72 [db 2 1 800] ------- [4:0 2:1 32 2]
[0 0] 2 2
s 7.078572152 _2_ AGT  --- 7 tcp 1460 [0 0 0 0] ------- [2:1 4:0 32 0]
[1 0] 0 2
r 7.078572152 _2_ RTR  --- 7 tcp 1460 [0 0 0 0] ------- [2:1 4:0 32 0]
[1 0] 0 2
s 7.078572152 _2_ AGT  --- 8 tcp 1460 [0 0 0 0] ------- [2:1 4:0 32 0]
[2 0] 0 2
r 7.078572152 _2_ RTR  --- 8 tcp 1460 [0 0 0 0] ------- [2:1 4:0 32 0]
[2 0] 0 2
s 7.078572152 _2_ RTR  --- 7 tcp 1492 [0 0 0 0] ------- [2:1 4:0 32 1]
[1 0] 0 2
s 7.078572152 _2_ RTR  --- 8 tcp 1492 [0 0 0 0] ------- [2:1 4:0 32 1]
[2 0] 0 2
r 7.085971841 _1_ RTR  --- 7 tcp 1492 [db 1 2 800] ------- [2:1 4:0 32
1] [1 0] 1 2
f 7.085971841 _1_ RTR  --- 7 tcp 1492 [db 1 2 800] ------- [2:1 4:0 32
4] [1 0] 1 2
r 7.093072714 _4_ RTR  --- 7 tcp 1492 [db 4 1 800] ------- [2:1 4:0 32
4] [1 0] 2 2
r 7.093072714 _4_ AGT  --- 7 tcp 1492 [db 4 1 800] ------- [2:1 4:0 32
4] [1 0] 2 2
s 7.093072714 _4_ AGT  --- 9 ack 40 [0 0 0 0] ------- [4:0 2:1 32 0] [1
0] 0 2
```

```
r 7.093072714 _4_ RTR  --- 9 ack 40 [0 0 0 0] ------- [4:0 2:1 32 0] [1
0] 0 2
s 7.093072714 _4_ RTR  --- 9 ack 72 [0 0 0 0] ------- [4:0 2:1 32 1] [1
0] 0 2
r 7.094893587 _1_ RTR  --- 9 ack 72 [db 1 4 800] ------- [4:0 2:1 32 1]
[1 0] 1 2
f 7.094893587 _1_ RTR  --- 9 ack 72 [db 1 4 800] ------- [4:0 2:1 32 2]
[1 0] 1 2
r 7.096433276 _2_ RTR  --- 9 ack 72 [db 2 1 800] ------- [4:0 2:1 32 2]
[1 0] 2 2
r 7.096433276 _2_ AGT  --- 9 ack 72 [db 2 1 800] ------- [4:0 2:1 32 2]
[1 0] 2 2
s 7.096433276 _2_ AGT  --- 10 tcp 1460 [0 0 0 0] ------- [2:1 4:0 32 0]
[3 0] 0 2
r 7.096433276 _2_ RTR  --- 10 tcp 1460 [0 0 0 0] ------- [2:1 4:0 32 0]
[3 0] 0 2
s 7.096433276 _2_ AGT  --- 11 tcp 1460 [0 0 0 0] ------- [2:1 4:0 32 0]
[4 0] 0 2
r 7.096433276 _2_ RTR  --- 11 tcp 1460 [0 0 0 0] ------- [2:1 4:0 32 0]
[4 0] 0 2
s 7.096433276 _2_ RTR  --- 10 tcp 1492 [0 0 0 0] ------- [2:1 4:0 32 1]
[3 0] 0 2
s 7.096433276 _2_ RTR  --- 11 tcp 1492 [0 0 0 0] ------- [2:1 4:0 32 1]
[4 0] 0 2
r 7.103792965 _1_ RTR  --- 8 tcp 1492 [db 1 2 800] ------- [2:1 4:0 32
1] [2 0] 1 2
f 7.103792965 _1_ RTR  --- 8 tcp 1492 [db 1 2 800] ------- [2:1 4:0 32
4] [2 0] 1 2
r 7.110893837 _4_ RTR  --- 8 tcp 1492 [db 4 1 800] ------- [2:1 4:0 32
4] [2 0] 2 2
r 7.110893837 _4_ AGT  --- 8 tcp 1492 [db 4 1 800] ------- [2:1 4:0 32
4] [2 0] 2 2
s 7.110893837 _4_ AGT  --- 12 ack 40 [0 0 0 0] ------- [4:0 2:1 32 0]
[2 0] 0 2
r 7.110893837 _4_ RTR  --- 12 ack 40 [0 0 0 0] ------- [4:0 2:1 32 0]
[2 0] 0 2
```

APPENDIX C. TCL NODE MOVEMENT FILE (LARGE RANGES)

This appendix contains an example of a node movement file used in one of the simulations. A command line input into NS2 generated the node movement file based upon specific parameters established by the user. As mentioned in Chapter VI, the transmission range of each node was expanded to 10 km in order to allow larger scenarios. The command line below was used to generate the movement file showed in this appendix:

./setdest –n 10 –p0 –s 20 –t 2000 –x 40000 –y 40000 > leo-10np0stop2000-v20-40kmx40km-1

```
#
# nodes: 10, pause: 0.00, max speed: 20.00  max x = 40000.00, max y:
# 40000.00
#
$node_(0) set X_ 22502.137003443873
$node_(0) set Y_ 33416.636842130094
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 33415.435324249636
$node_(1) set Y_ 13221.524307375223
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 14159.045784060874
$node_(2) set Y_ 11082.505271689157
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 23666.111110236248
$node_(3) set Y_ 36329.450734058104
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 29078.519542600196
$node_(4) set Y_ 2677.978278312096
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 8780.925966699409
$node_(5) set Y_ 21022.730105854444
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 9024.907745421819
$node_(6) set Y_ 1624.485310837787
$node_(6) set Z_ 0.000000000000
$node_(7) set X_ 22724.620690638527
$node_(7) set Y_ 12699.967721132738
$node_(7) set Z_ 0.000000000000
$node_(8) set X_ 8357.500344240176
$node_(8) set Y_ 24508.293057666247
$node_(8) set Z_ 0.000000000000
$node_(9) set X_ 30881.441937471234
$node_(9) set Y_ 24394.670474170449
$node_(9) set Z_ 0.000000000000
$ns_ at 0.000000000000 "$node_(0) setdest 1226.681024342604
16827.977213505870 13.906521160309"
```

```
$ns_ at 0.000000000000 "$node_(1) setdest 13802.307309693548
15378.966263325747 17.143000678718"
$ns_ at 0.000000000000 "$node_(2) setdest 4824.844842906078
11167.279002210789 4.229100031662"
$ns_ at 0.000000000000 "$node_(3) setdest 36968.471777056322
9105.189781670031 15.462334302119"
$ns_ at 0.000000000000 "$node_(4) setdest 30905.258854572676
24685.596219211533 5.407839093727"
$ns_ at 0.000000000000 "$node_(5) setdest 19103.306148637959
29266.457104100376 0.672287289331"
$ns_ at 0.000000000000 "$node_(6) setdest 38264.944755438468
38926.538598963089 19.167133652137"
$ns_ at 0.000000000000 "$node_(7) setdest 4030.616937006699
22578.863846147186 1.482341113181"
$ns_ at 0.000000000000 "$node_(8) setdest 27414.181094759460
30141.683941113301 15.641012514880"
$ns_ at 0.000000000000 "$node_(9) setdest 36994.702924416037
9972.083479557983 0.403524888846"
$god_ set-dist 0 1 16777215
$god_ set-dist 0 2 16777215
$god_ set-dist 0 3 1
$god_ set-dist 0 4 16777215
$god_ set-dist 0 5 16777215
$god_ set-dist 0 6 16777215
$god_ set-dist 0 7 16777215
$god_ set-dist 0 8 16777215
$god_ set-dist 0 9 16777215
$god_ set-dist 1 2 16777215
$god_ set-dist 1 3 16777215
$god_ set-dist 1 4 16777215
$god_ set-dist 1 5 16777215
$god_ set-dist 1 6 16777215
$god_ set-dist 1 7 16777215
$god_ set-dist 1 8 16777215
$god_ set-dist 1 9 16777215
$god_ set-dist 2 3 16777215
$god_ set-dist 2 4 16777215
$god_ set-dist 2 5 16777215
$god_ set-dist 2 6 16777215
$god_ set-dist 2 7 1
$god_ set-dist 2 8 16777215
$god_ set-dist 2 9 16777215
$god_ set-dist 3 4 16777215
$god_ set-dist 3 5 16777215
$god_ set-dist 3 6 16777215
$god_ set-dist 3 7 16777215
$god_ set-dist 3 8 16777215
$god_ set-dist 3 9 16777215
$god_ set-dist 4 5 16777215
$god_ set-dist 4 6 16777215
$god_ set-dist 4 7 16777215
$god_ set-dist 4 8 16777215
$god_ set-dist 4 9 16777215
$god_ set-dist 5 6 16777215
$god_ set-dist 5 7 16777215
$god_ set-dist 5 8 1
```

```
$god_ set-dist 5 9 16777215
$god_ set-dist 6 7 16777215
$god_ set-dist 6 8 16777215
$god_ set-dist 6 9 16777215
$god_ set-dist 7 8 16777215
$god_ set-dist 7 9 16777215
$god_ set-dist 8 9 16777215
$ns_ at 36.628104672255 "$god_ set-dist 2 6 1"
$ns_ at 36.628104672255 "$god_ set-dist 6 7 2"
$ns_ at 44.968810103258 "$god_ set-dist 1 2 2"
$ns_ at 44.968810103258 "$god_ set-dist 1 6 3"
$ns_ at 44.968810103258 "$god_ set-dist 1 7 1"
$ns_ at 176.788288614934 "$god_ set-dist 1 4 1"
$ns_ at 176.788288614934 "$god_ set-dist 2 4 3"
$ns_ at 176.788288614934 "$god_ set-dist 4 6 4"
$ns_ at 176.788288614934 "$god_ set-dist 4 7 2"
$ns_ at 233.224928518294 "$god_ set-dist 0 5 2"
$ns_ at 233.224928518294 "$god_ set-dist 0 8 1"
$ns_ at 233.224928518294 "$god_ set-dist 3 5 3"
$ns_ at 233.224928518294 "$god_ set-dist 3 8 2"
$ns_ at 263.468355592235 "$god_ set-dist 0 9 2"
$ns_ at 263.468355592235 "$god_ set-dist 3 9 1"
$ns_ at 263.468355592235 "$god_ set-dist 5 9 4"
$ns_ at 263.468355592235 "$god_ set-dist 8 9 3"
$ns_ at 398.974361211903 "$god_ set-dist 1 6 2"
$ns_ at 398.974361211903 "$god_ set-dist 4 6 3"
$ns_ at 398.974361211903 "$god_ set-dist 6 7 1"
$ns_ at 429.219521220472 "$god_ set-dist 1 2 3"
$ns_ at 429.219521220472 "$god_ set-dist 2 4 4"
$ns_ at 429.219521220472 "$god_ set-dist 2 7 2"
$ns_ at 497.477763260797 "$god_ set-dist 0 3 16777215"
$ns_ at 497.477763260797 "$god_ set-dist 0 9 16777215"
$ns_ at 497.477763260797 "$god_ set-dist 3 5 16777215"
$ns_ at 497.477763260797 "$god_ set-dist 3 8 16777215"
$ns_ at 497.477763260797 "$god_ set-dist 5 9 16777215"
$ns_ at 497.477763260797 "$god_ set-dist 8 9 16777215"
$ns_ at 532.906139678420 "$god_ set-dist 1 4 16777215"
$ns_ at 532.906139678420 "$god_ set-dist 2 4 16777215"
$ns_ at 532.906139678420 "$god_ set-dist 4 6 16777215"
$ns_ at 532.906139678420 "$god_ set-dist 4 7 16777215"
$ns_ at 535.910980047556 "$god_ set-dist 1 2 2"
$ns_ at 535.910980047556 "$god_ set-dist 1 6 1"
$ns_ at 585.150684797371 "$god_ set-dist 0 5 1"
$ns_ at 589.488598839237 "$god_ set-dist 5 8 2"
$ns_ at 646.517128127917 "$god_ set-dist 0 3 2"
$ns_ at 646.517128127917 "$god_ set-dist 0 9 3"
$ns_ at 646.517128127917 "$god_ set-dist 3 5 3"
$ns_ at 646.517128127917 "$god_ set-dist 3 8 1"
$ns_ at 646.517128127917 "$god_ set-dist 5 9 4"
$ns_ at 646.517128127917 "$god_ set-dist 8 9 2"
$ns_ at 773.638869589614 "$god_ set-dist 1 2 1"
$ns_ at 783.328155691574 "$god_ set-dist 1 4 2"
$ns_ at 783.328155691574 "$god_ set-dist 2 4 3"
$ns_ at 783.328155691574 "$god_ set-dist 4 6 2"
$ns_ at 783.328155691574 "$god_ set-dist 4 7 1"
$ns_ at 891.792272257138 "$god_ set-dist 2 6 2"
```

```
$ns_ at 912.779175509088 "$god_ set-dist 0 3 16777215"
$ns_ at 912.779175509088 "$god_ set-dist 0 8 16777215"
$ns_ at 912.779175509088 "$god_ set-dist 0 9 16777215"
$ns_ at 912.779175509088 "$god_ set-dist 3 5 16777215"
$ns_ at 912.779175509088 "$god_ set-dist 5 8 16777215"
$ns_ at 912.779175509088 "$god_ set-dist 5 9 16777215"
$ns_ at 916.508318469820 "$god_ set-dist 8 9 1"
$ns_ at 964.878461047888 "$god_ set-dist 0 1 2"
$ns_ at 964.878461047888 "$god_ set-dist 0 2 3"
$ns_ at 964.878461047888 "$god_ set-dist 0 4 4"
$ns_ at 964.878461047888 "$god_ set-dist 0 6 3"
$ns_ at 964.878461047888 "$god_ set-dist 0 7 3"
$ns_ at 964.878461047888 "$god_ set-dist 1 5 1"
$ns_ at 964.878461047888 "$god_ set-dist 2 5 2"
$ns_ at 964.878461047888 "$god_ set-dist 4 5 3"
$ns_ at 964.878461047888 "$god_ set-dist 5 6 2"
$ns_ at 964.878461047888 "$god_ set-dist 5 7 2"
$ns_ at 1051.246752375747 "$god_ set-dist 3 8 2"
$ns_ at 1067.692132184842 "$god_ set-dist 0 3 4"
$ns_ at 1067.692132184842 "$god_ set-dist 0 8 6"
$ns_ at 1067.692132184842 "$god_ set-dist 0 9 5"
$ns_ at 1067.692132184842 "$god_ set-dist 1 3 2"
$ns_ at 1067.692132184842 "$god_ set-dist 1 8 4"
$ns_ at 1067.692132184842 "$god_ set-dist 1 9 3"
$ns_ at 1067.692132184842 "$god_ set-dist 2 3 3"
$ns_ at 1067.692132184842 "$god_ set-dist 2 8 5"
$ns_ at 1067.692132184842 "$god_ set-dist 2 9 4"
$ns_ at 1067.692132184842 "$god_ set-dist 3 4 3"
$ns_ at 1067.692132184842 "$god_ set-dist 3 5 3"
$ns_ at 1067.692132184842 "$god_ set-dist 3 6 1"
$ns_ at 1067.692132184842 "$god_ set-dist 3 7 2"
$ns_ at 1067.692132184842 "$god_ set-dist 4 8 5"
$ns_ at 1067.692132184842 "$god_ set-dist 4 9 4"
$ns_ at 1067.692132184842 "$god_ set-dist 5 8 5"
$ns_ at 1067.692132184842 "$god_ set-dist 5 9 4"
$ns_ at 1067.692132184842 "$god_ set-dist 6 8 3"
$ns_ at 1067.692132184842 "$god_ set-dist 6 9 2"
$ns_ at 1067.692132184842 "$god_ set-dist 7 8 4"
$ns_ at 1067.692132184842 "$god_ set-dist 7 9 3"
$ns_ at 1075.017963804004 "$god_ set-dist 0 1 1"
$ns_ at 1075.017963804004 "$god_ set-dist 0 2 2"
$ns_ at 1075.017963804004 "$god_ set-dist 0 3 3"
$ns_ at 1075.017963804004 "$god_ set-dist 0 4 3"
$ns_ at 1075.017963804004 "$god_ set-dist 0 6 2"
$ns_ at 1075.017963804004 "$god_ set-dist 0 7 2"
$ns_ at 1075.017963804004 "$god_ set-dist 0 8 5"
$ns_ at 1075.017963804004 "$god_ set-dist 0 9 4"
$ns_ at 1139.232074304606 "$god_ set-dist 0 8 4"
$ns_ at 1139.232074304606 "$god_ set-dist 0 9 3"
$ns_ at 1139.232074304606 "$god_ set-dist 1 8 3"
$ns_ at 1139.232074304606 "$god_ set-dist 1 9 2"
$ns_ at 1139.232074304606 "$god_ set-dist 2 8 4"
$ns_ at 1139.232074304606 "$god_ set-dist 2 9 3"
$ns_ at 1139.232074304606 "$god_ set-dist 4 8 4"
$ns_ at 1139.232074304606 "$god_ set-dist 4 9 3"
$ns_ at 1139.232074304606 "$god_ set-dist 5 8 4"
```

```
$ns_ at 1139.232074304606 "$god_ set-dist 5 9 3"
$ns_ at 1139.232074304606 "$god_ set-dist 6 8 2"
$ns_ at 1139.232074304606 "$god_ set-dist 6 9 1"
$ns_ at 1139.232074304606 "$god_ set-dist 7 8 3"
$ns_ at 1139.232074304606 "$god_ set-dist 7 9 2"
$ns_ at 1150.990470512652 "$node_(1) setdest 4085.614384907528
26920.970763950318 10.377826797404"
$ns_ at 1174.117702349961 "$god_ set-dist 0 3 4"
$ns_ at 1174.117702349961 "$god_ set-dist 0 6 3"
$ns_ at 1174.117702349961 "$god_ set-dist 0 8 5"
$ns_ at 1174.117702349961 "$god_ set-dist 0 9 4"
$ns_ at 1174.117702349961 "$god_ set-dist 1 3 3"
$ns_ at 1174.117702349961 "$god_ set-dist 1 6 2"
$ns_ at 1174.117702349961 "$god_ set-dist 1 8 4"
$ns_ at 1174.117702349961 "$god_ set-dist 1 9 3"
$ns_ at 1174.117702349961 "$god_ set-dist 2 3 4"
$ns_ at 1174.117702349961 "$god_ set-dist 2 6 3"
$ns_ at 1174.117702349961 "$god_ set-dist 2 8 5"
$ns_ at 1174.117702349961 "$god_ set-dist 2 9 4"
$ns_ at 1174.117702349961 "$god_ set-dist 3 5 4"
$ns_ at 1174.117702349961 "$god_ set-dist 5 6 3"
$ns_ at 1174.117702349961 "$god_ set-dist 5 8 5"
$ns_ at 1174.117702349961 "$god_ set-dist 5 9 4"
$ns_ at 1242.646280867531 "$god_ set-dist 3 4 1"
$ns_ at 1242.646280867531 "$god_ set-dist 4 8 3"
$ns_ at 1242.646280867531 "$god_ set-dist 4 9 2"
$ns_ at 1263.703445200898 "$god_ set-dist 0 8 4"
$ns_ at 1263.703445200898 "$god_ set-dist 1 8 3"
$ns_ at 1263.703445200898 "$god_ set-dist 2 8 4"
$ns_ at 1263.703445200898 "$god_ set-dist 5 8 4"
$ns_ at 1263.703445200898 "$god_ set-dist 6 8 1"
$ns_ at 1263.703445200898 "$god_ set-dist 7 8 2"
$ns_ at 1270.499211908984 "$node_(8) setdest 269.986353926073
17660.650674094795 11.277947588824"
$ns_ at 1388.613684494774 "$god_ set-dist 0 6 5"
$ns_ at 1388.613684494774 "$god_ set-dist 0 8 6"
$ns_ at 1388.613684494774 "$god_ set-dist 0 9 5"
$ns_ at 1388.613684494774 "$god_ set-dist 1 6 4"
$ns_ at 1388.613684494774 "$god_ set-dist 1 8 5"
$ns_ at 1388.613684494774 "$god_ set-dist 1 9 4"
$ns_ at 1388.613684494774 "$god_ set-dist 2 6 5"
$ns_ at 1388.613684494774 "$god_ set-dist 2 8 6"
$ns_ at 1388.613684494774 "$god_ set-dist 2 9 5"
$ns_ at 1388.613684494774 "$god_ set-dist 5 6 5"
$ns_ at 1388.613684494774 "$god_ set-dist 5 8 6"
$ns_ at 1388.613684494774 "$god_ set-dist 5 9 5"
$ns_ at 1388.613684494774 "$god_ set-dist 6 7 3"
$ns_ at 1388.613684494774 "$god_ set-dist 7 8 4"
$ns_ at 1388.613684494774 "$god_ set-dist 7 9 3"
$ns_ at 1427.856229493271 "$god_ set-dist 0 6 6"
$ns_ at 1427.856229493271 "$god_ set-dist 1 6 5"
$ns_ at 1427.856229493271 "$god_ set-dist 2 6 6"
$ns_ at 1427.856229493271 "$god_ set-dist 3 6 2"
$ns_ at 1427.856229493271 "$god_ set-dist 4 6 3"
$ns_ at 1427.856229493271 "$god_ set-dist 5 6 6"
$ns_ at 1427.856229493271 "$god_ set-dist 6 7 4"
```

```
$ns_ at 1447.621754380096 "$god_ set-dist 0 2 1"
$ns_ at 1464.295227738686 "$god_ set-dist 0 3 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 0 4 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 0 6 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 0 7 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 0 8 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 0 9 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 1 3 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 1 4 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 1 6 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 1 7 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 1 8 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 1 9 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 2 3 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 2 4 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 2 6 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 2 7 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 2 8 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 2 9 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 3 5 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 4 5 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 5 6 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 5 7 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 5 8 16777215"
$ns_ at 1464.295227738686 "$god_ set-dist 5 9 16777215"
$ns_ at 1581.567256397830 "$god_ set-dist 3 6 16777215"
$ns_ at 1581.567256397830 "$god_ set-dist 3 8 16777215"
$ns_ at 1581.567256397830 "$god_ set-dist 3 9 16777215"
$ns_ at 1581.567256397830 "$god_ set-dist 4 6 16777215"
$ns_ at 1581.567256397830 "$god_ set-dist 4 8 16777215"
$ns_ at 1581.567256397830 "$god_ set-dist 4 9 16777215"
$ns_ at 1581.567256397830 "$god_ set-dist 6 7 16777215"
$ns_ at 1581.567256397830 "$god_ set-dist 7 8 16777215"
$ns_ at 1581.567256397830 "$god_ set-dist 7 9 16777215"
$ns_ at 1802.672479208529 "$god_ set-dist 8 9 2"
$ns_ at 1825.839285268703 "$god_ set-dist 1 2 2"
$ns_ at 1863.073268129671 "$god_ set-dist 6 8 16777215"
$ns_ at 1863.073268129671 "$god_ set-dist 8 9 16777215"
$ns_ at 1939.974735722339 "$node_(0) setdest 16930.270723538084
27060.065522786557 19.260632738973"
$ns_ at 1959.625369305903 "$node_(3) setdest 26908.922001653675
18252.105664547144 1.569960117988"
# Destination Unreachables: 93
# Route Changes: 195
# Link Changes: 34
# Node | Route Changes | Link Changes
#    0 |            45 |            6
#    1 |            35 |           10
#    2 |            36 |            6
#    3 |            33 |            8
#    4 |            33 |            4
#    5 |            37 |            3
#    6 |            44 |           11
#    7 |            27 |            6
#    8 |            53 |            9
#    9 |            47 |            5
```

APPENDIX D. TCL TRAFFIC GENERATION WITH CONDITIONING FILE

This appendix contains an example of a traffic pattern file (with conditioning embedded) that was used in one of the simulations. As mentioned in Chapter VI, the conditioning is executed in the source nodes.

```
#
# nodes: 10, tcp conn: 3, cbr conn: 1 cbr send rate: 20.0, seed: 0.0
#
#
# 0 connecting to 1 at time 0
#
set tcp_(0) [new Agent/TCP/Reno]
$tcp_(0) set window_ 8
$tcp_(0) set packetSize_ 1460
$tcp_(0) set class_ 0
set difftc(0) [new DiffTC TB 0]
set meter_ [$difftc(0) getmeter]
$meter_ tbsize 0
$difftc(0) attach-conditioner $node_(0) $tcp_(0)
set sink_(0) [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $sink_(0)
$ns_ connect $tcp_(0) $sink_(0)
set ftp_(0) [$tcp_(0) attach-source FTP]
$ns_ at 0 "$ftp_(0) start"
#
# 0 connecting to 2 at time 0
#
set tcp_(1) [new Agent/TCP/Reno]
$tcp_(1) set window_ 8
$tcp_(1) set packetSize_ 1460
$tcp_(1) set class_ 1
set difftc(1) [new DiffTC TB 0]
set meter_ [$difftc(1) getmeter]
$meter_ tbsize 0
$difftc(1) attach-conditioner $node_(0) $tcp_(1)
set sink_(1) [new Agent/TCPSink]
$ns_ attach-agent $node_(2) $sink_(1)
$ns_ connect $tcp_(1) $sink_(1)
set ftp_(1) [$tcp_(1) attach-source FTP]
$ns_ at 0 "$ftp_(1) start"
#
# 2 connecting to 3 at time 0
#
set tcp_(2) [new Agent/TCP/Reno]
$tcp_(2) set window_ 8
$tcp_(2) set packetSize_ 1460
$tcp_(2) set class_ 2
set difftc(2) [new DiffTC TB 0]
set meter_ [$difftc(2) getmeter]
$meter_ tbsize 0
$difftc(2) attach-conditioner $node_(2) $tcp_(2)
set sink_(2) [new Agent/TCPSink]
$ns_ attach-agent $node_(3) $sink_(2)
```

```
$ns_ connect $tcp_(2) $sink_(2)
set ftp_(2) [$tcp_(2) attach-source FTP]
$ns_ at 0 "$ftp_(2) start"
#
# 3 connecting to 4 at time 0
#
set udp_(0) [new Agent/UDP]
$udp_(0) set class_ 3
set difftccbr(0) [new DiffTC TB 0]
set meter_ [$difftccbr(3) getmeter]
$meter_ tbsize 0
$difftccbr(0) attach-conditioner $node_(3) $udp_(0)
set null_(0) [new Agent/LossMonitor]
$ns_ attach-agent $node_(4) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 1000
$cbr_(0) set interval_ 0.05
$cbr_(0) set random_ 1
#$cbr_(0) set maxpkts_ 80000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 0 "$cbr_(0) start"
# conditioning - higher target rate for cbr/udp traffic
$difftccbr(0) AdptPara 1.5Mbps 36000000
```

LIST OF REFERENCES

1. *"Operational Requirements Document (ORD) for Joint Tactical Radio System (JTRS),"* JTRS Joint Program Office, 23 March 1998.

2. Backer, J. Dennis and Hauser, P. James, *"Architecture for a Mobile Wide Area Subnet with Combined Packet-Switching and Virtual-Circuit Switching Capability,"* Naval Research Laboratory, Washington, DC, 1998.

3. *"Digital Modular Radio AN/USC-61 (V) Primer,"* Space and Naval Warfare System Center, San Diego, CA, April 19, 2000.

4. Corson, Scott M. et al, "Internet Based Mobile Ad Hoc Networking," *IEEE Internet Computing*, July/August 1999.

5. Corson, Scott S., Macker, J., "Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," *IETF RFC 2501*, January 1999.

6. Rappaport, Theodore S., *"Wireless Communications – Principles and Practice,"* Upper Saddle River, NJ: Prentice Hall, 1996.

7. Ko, Young-Bae, Vaidya, Nitin H., *"Medium Access Control Protocols Using Directional Antennas in Ad-Hoc Networks,"* Tech. Rep. 99-010, CS Dept., Texas A&M University, May 1999.

8. IEEE, *"IEEE std 802.11 – wireless LAN medium access control (MAC) and physical layer (PHY) specifications,"* 16 November, 1997.

9. Vaidya, Nitin H., *"Mobile Ad Hoc Networks; Routing, MAC, and Transport Issues,"* MobiComm Tutorial, 15 July, 2000, pg 1- 431.

10. Goodwin, G. et. al., *"HDR LOS Communication Network for USN and USMC Applications,"* MILCOM 97.

11. Johnson, David B., "Routing in Ad Hoc Networks," *Proceedings of IEEE Workshop*, pg 1-4, 1994.

12. Gerla, M., Lee, S. J., Toh, C. K., "A Simulation Study of Table-Driven and On-Demand Routing Protocols for Mobile Ad Hoc Networks," *IEEE Network*, Vol 13 Issue 4, pg 48-54, Jul-Aug 1999.

13. Misra, Padmini, *"Routing Protocols For Ad Hoc Mobile Wireless Networks,"* Computer and Information Systems Paper 788-99, Ohio State University, 18 November 1999.

14. Theriot, Ty, *"Simulation and Performance Analysis of the AODV protocol for Tactical Mobile Ad-hoc Networks,"* Master's Thesis, Naval Postgraduate School, Monterey, California, December 2000.

15. Shea, Kevin, *"Simulation and Performance Analysis of the ZRP protocol for Tactical Mobile Ad-hoc Networks,"* Master's Thesis, Naval Postgraduate School, Monterey, California, September 2000.

16. Haas, Zygmunt J, Pearlman, Marc R., *"The Zone Routing Protocol (ZRP) for Ad Hoc Networks,"* Internet Draft, draft-ietf-manet-zone-zrp-02.txt, June 1999.

17. Broch J., Johnson D. B., Maltz, D. A., *"The Dynamic Source Routing (DSR) Protocol for Mobile Ad Hoc Networks,"* Internet Draft, draft-ietf-manet-dsr-01.txt, Dec 1998.

18. Perkins, Charles E., Royer, Elizabeth M., "Ad-hoc On-Demand Distance Vector Routing," *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications,* New Orleans, LA, February 1999, pp. 90-100.

19. Das, Samir R., Perkins, Charles E., Royer, Elizabeth M., "Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks," *Proceedings of the IEEE Conference on Computer Communications (INFOCOM),* Tel Aviv, Israel, March 2000, pg. 3-12.

20. Lee, S.B., Campbell, A.T., "INSIGNIA: In-band signaling support for QoS in mobile ad-hoc networks," *5th Int. Workshop on Mobile Multimedia Communications (MoMuc),* Berlin, Germany, October 1998.

21. Sinha, P., Sivakumar, R. and Bharghavan, V., "CEDAR: A core-extraction distributed ad-hoc routing algorithm," in *IEEE Infocom 99,* New York, March 1999.

22. Das, Samir R., Perkins, Charles E., Royer, Elizabeth M., *"Qos for AODVs,"* Internet Draft, draft-ietf-manet-aodvqos-00.txt, July 2000.

23. G. Holland and N.H Vaidya, "Analysis of TCP performance over mobile ad-hoc networks," in *Proceedings of IEEE/ACM Mobicom99,* pages 219-230, Seattle, WA, USA, August 1999.

24. Vaidya, Nitin H., Bahl, Paramvir, Gupta, Seema, "Distributed Fair Scheduling in a Wireless LAN," *MobiCom 2000,* Boston, August 2000.

25. Cansever, Derya H., Levesque, Allen H., Michelson, Arnold M. "Quality of Service Support in Mobile Ad-Hoc Networks," in *Proceedings of Military Communications Conference, MILCOM99.*

26. R.Braden, D.Clark, and S.Shenker, "Integrated Services in the Internet Architecture - an Overview," *IETF RFC1633,* June 1994.

27. Blake, S., "An architecture for Differentiated Services," *IETF RFC2475,* December 1998.

28. Corson, M. Scott, Campbell, Andrew T., "Towards Supporting Quality of Service in Mobile Ad-Hoc Networks," *First Conference in Open Architecture and Network Programming,* San Francisco, CA, USA, April 1998.

29. Stallings, William, *"High-Speed Networks – TCP/IP and ATM Design Principles,"* Upper Saddle River, NJ: Prentice Hall, 1996.

30. H. Xiao, W.K.G. Seah, A.Lo., and K.C.Chua., "A Flexible Quality of Service Model for Mobile Ad-Hoc Networks," in *Proceedings of IEEE VTC2000-spring,* Tokyo, Japan, May 2000.

31. Jacobson, V., Nichols, K., Poduri, K., "An Expedited Forwarding PHB," *IETF RFC2598,* June 1999.

32. Heinanen, J., Baker, F., Weiss, W., Wroclawski, J., "Assured Forwarding PHB Group ," *IETF RFC2597,* June 1999.

33. D.D. Clark and W. Fang, "Explicit Allocation of Best-effort Packet Delivery Service," in *IEEE/ACM Trans. On Networking,* 6(4):362-373, August 1998.

34. H. Xiao, W.K.G. Seah, A.Lo., and K.C.Chua, "On Service Differentiation in Multi-hop Wireless Networks," in *ITC Specialist Seminar on Mobile Systems and Computing*, Lillehamer, Norway, March 2000.

35. Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance," Lawrence Berkeley Laboratory, University of California, *IEEE/ACM Transactions on Networking*, August 1993.

36. Stevens, Richard W., *"TCP/IP Illustrated Volume 1 – The Protocols,"* Addison-Wesley Professional Computing Series, MA, July 1999.

37. H. Xiao, W.K.G. Seah, A.Lo., and K.C.Chua, *"On Service Prioritization in Mobile Ad-Hoc Network,"* submitted to ICC2001, Helsinki, Finland, June 2001.

38. Zhang,H. "Service Disciplines for Guaranteed Performance Services in Packet Switching Networks," in *Proceedings of the IEEE*, October 1995.

39. Joseph P. Marker and M. Scott Corson, "Mobile Ad-hoc Networks and the IETF," in *Proceeding of the Minneapolis Work Group meeting, Mobile Computing and Communications Review*, Volume 3, Number 2.

40. K. Fall and K. Varadhan, *"ns Notes and Documentation,"* VINT Project, UC-Berkeley and LBNL, August 2000.

41. Broch J., Hu, Y. C., Jetcheva J., Johnson D. B., Maltz, D. A., "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1998)*, Dallas, TX, October 1998.

42. Bob O'Hara and Al Petrick, *"IEEE802.11 Handbook – A Designer's Companion,"* IEEE Press, The Institute of Electrical and Electronics Engineers, Inc.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library... 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Engineering and Technology Curricular Office, Code 34.. 1
   Naval Postgraduate School
   Monterey, CA 93943-5109

4. Chairman, Code EC.. 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

5. Dr. Murali Tummala, Code EC/Tu 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

6. Dr. John McEachen, Code EC/Me 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

7. Dr. Robert Ives (LCDR-USN), Code EC/Ir 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

8. Dr. Richard North.. 1
   SPAWARSYSCEN, D841
   53560 Hull Street
   San Diego, CA 92152-5001

9. LCDR Robert Parker.. 1
   Technical Director, PMW 176-1
   SPAWAR Systems Center
   San Diego, CA 92152

10. Michael J. Geile.. 1
Nova Engineering, Inc.
5 Circle Freeway
Cincinnati, OH 45246

11. LCDR Leonardo da Silva Mattos 1
Rua Eurico Cruz, 24/401
Jardim Botânico, Rio de Janeiro, RJ
Brazil, CEP: 22461-200

12. Diretoria de Telecomunicações da Marinha... 1
Rua Primeiro de Março, 118, 4°andar
Centro, Rio de Janeiro, RJ
Brazil, CEP: 20010-000