Protocols for Supporting Quality of Service in Mobile Ad Hoc Networks

by

Lei Chen

A Thesis Submitted in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor Wendi B. Heinzelman

Department of Electrical and Computer Engineering

The College

School of Engineering and Applied Sciences

University of Rochester

Rochester, New York

2006

This thesis is dedicated to the memory of my grandfather
Mingju Jiang

Acknowledgment

I would like to begin with thanking Professor Wendi B. Heinzelman, my thesis advisor and mentor for the past five years. Wendi provides me with the freedom of choosing research topics, endless source of ideas, and encouragement. Her enthusiasm in research, systematic organization in work, and optimistic attitude towards life positively affects my study and life. I thank her for guidance not only in academic but also in life attitude. Her assistance and instruction during my time at University of Rochester has been invaluable. She will be my example in personality, intelligence, and career.

I would also like to thank Professor Gaurav Sharma for his sincere support for being my thesis committee member and actively giving me feedback on conference speak practices. I would like to thank Professor Kai Shen for being my committee member and offering so many valuable suggestions on my thesis improvement.

I would like to thank all my colleagues at University of Rochester, especially my labmates for their feedback, advice, and friendship.

My special sincere thanks have to go to my parents, Tangming Chen and Mingmin Sun, for everything they having been giving me. They have provided all the support that they could offer and have stood by everything I have done. I know that I could not accomplish anything without their encouragement. I would like to extend my thanks to my sister, Lu Chen, who has took care of me so much in the past years. I am so lucky and so proud to have such a wonderful family.

Finally, I would like to thank my wonderful husband, Hongwei Liu. Hongwei shows his encouragement when I am depressed and frustrated. He taught me how to be patient and persistent while I was facing difficulties.

Curriculum Vitae

The author was born in Zhejiang province, P. R. China on June 10th, 1976. She joined in Zhejiang University in 1995, where she received B.E. degree in Information Science and Electronic Engineering in 1999. Then, she came to the USA studying at the State University of New York at Buffalo in 2000. She transferred to the Department of Electrical and Computer Engineering at University of Rochester in 2001. She received Master of Science degree from University of Rochester in 2003. She is currently working towards her Ph.D. degree in the area of wireless communications and networking. Her research interests include wireless communications and multimedia communications.

Abstract

Mobile ad hoc networks (MANETs) are self-organizing, infrastructureless networks that provide flexibility and convenience in setting up a dynamic network. Since the 1970's, MANETs have attracted a great deal of research aimed at improving their basic performance. However, with the development of real-time applications, incorporating Quality of Service (QoS) into the network architecture becomes essential.

My thesis is that supporting QoS in MANETs necessitates more harmonious collaboration among network layers, requiring a design that supports cross-layer interactions rather than a traditional independent layered network architecture. The work described in this dissertation enables this goal by proposing a general architecture that supports cross-layer interactions, as well as by designing protocols at several layers of the stack that can exploit this cross-layer information to improve the QoS performance.

Specifically, we begin by developing a QoS architecture for cross-layer information sharing, defining explicitly what information must be shared among the layers to provide support for QoS in terms of bandwidth and packet delivery rate. Using this architecture, we develop protocols for the transport, network and MAC layers that can improve QoS performance. At the transport layer, we develop a User Data Protocol (UDP) with Congestion Control (UDPC), in which network status is monitored using feedback sent by destination hosts, and this status is passed to real-time applications to adjust their transmission coding rate. This approach avoids wasting capacity and energy on data packets that cannot eventually be transmitted to the destination. In order to support QoS in the routing layer, we propose a bandwidth estimation based QoS-aware routing protocol with admission and adaptation schemes, which considerably improves packet delivery ratio, decreases transmission latency, and reduces energy consumption without impacting overall throughput. Finally, we investigate current distributed MAC protocols and develop a dual-channel MAC (DMAC) protocol to improve QoS at the MAC layer. We present the design considerations and show that using DMAC improves fairness. Using these different techniques that exploit cross-layer information sharing greatly improves QoS for mobile ad hoc networks.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless communication has shown its numerous advantages over wired communication since Guglielmo Marconi successfully transmitted signals across the English Channel for the first time in 1898. Since then, fueled by digital and Radio Frequency (RF) fabrication developments, portable mobile devices, such as cellular phones, personal digital assistants (PDA) and laptops, have brought great demands on wireless communication. Various wireless communication networks have been developed, such as cellular networks [2], wireless LANs (WLAN) [3], Bluetooth networks [4], Ultra-wideband (UWB) networks [5], Mobile Ad Hoc Networks (MANETs) [6], and WiMax [7]. Among these, cellular networks, Bluetooth networks, and WLANs are the most widely used. However, cellular networks and WLANs are centralized networks, which means that costly infrastructure and centralized administration are required. Using Bluetooth technology, hosts can connect to each other in an ad hoc fashion, but this technology is only targeted at low power short-range wire replacement. Therefore, a distributed, self-organized and multi-hop network—a Wireless Mobile Ad Hoc Network—is a different type of network that has obtained tremendous attention in recent years.

A MANET is a distributed network that does not require centralized control, and every host works not only as a source and a sink but also as a router. This type of dynamic network is especially useful for military communications or emergency search-and-rescue operations, where an infrastructure cannot be supported. Furthermore, the simplicity of building an ad hoc network enables sharing data in a meeting or in inhospitable terrain conveniently. At the same time, the rapid development of coding tech-

nologies, such as MPEG-4 [8] and H.264 [9], makes low data rate video over wireless feasible. Enabling such multi-media applications as video and audio communication in MANETs requires quality of service (QoS) support.

The challenges of supporting QoS in ad hoc networks are how to reserve bandwidth and how to guarantee the specified delay for real-time application data flows. For wireless transmissions, the channel is shared among neighbors. Therefore, the available bandwidth depends on the neighboring traffic status, as does the delay. Due to this characteristic, supporting QoS cannot be done by the host itself, but cooperation from the hosts within a node's interference range is needed. This requires an innovative design to coordinate the communication among the neighbors in order to support QoS in MANETs. Furthermore, the distributed organization of MANETs brings additional challenges to collaboration for supporting QoS.

In order to offer QoS support in MANETs, a network architecture, which should define each layer's functions and features, is required. For supporting multi-media transmission, the conventional layered network might not be the most efficient, because real-time applications can compress data according to the network status using different coding rates. We want to fully utilize this rate adaptation feature to best optimize data transmission. Therefore, a cross-layer design concept, which has been demonstrated as an efficient design for wireless multimedia delivery, is adopted in my QoS architecture. To fill the functions and features for this cross-layer QoS architecture, UDPC, QoS routing, and a dual-channel MAC are developed.

1.1 Overview of Mobile Ad Hoc Networks

A MANET is an autonomous collection of distributed mobile users. Every host in a MANET works as a source and a sink, and also relays packets for other hosts and is thus a router as well. This type of network can be used in fire/safety/rescue/disaster recovery operations, conference and campus settings, car networks, personal networking, etc.

MANETs have similar characteristics to other wireless communication networks, which are mainly attributed to the wireless channel's properties. A wireless channel is error-prone, which means that link bandwidth and packet delay are unpredictable due to multi-path fading, interference, and shadowing. Besides this common charac-

teristic, MANETs have their own features: they are autonomous and infrastructureless; they utilize multi-hop routing; they support a dynamic network topology; the nodes are energy constrained; the bandwidth is limited; and they are self-organizing and self-administering. Therefore, many widely used network protocols cannot directly be applied to MANETs.

Research has been done in the physical layer to deal with the rapid link changes, in the MAC layer to offer fair access to the medium [10][11][12][13] and minimize the hidden and exposed node problems [10][14][15][16][17], in the network layer to offer a path from source to destination [18], and in the transport layer to handle packet loss and delay. With all these efforts, data packets can be delivered from source to destination, but further research on offering QoS in MANETs is still needed.

In order to support QoS in MANETs, the network is expected to guarantee a set of measurable metrics, such as delay, delay variance (jitter), bandwidth, packet delivery rate, etc. However, the hidden node problem, the need to share channel resources, the distributed organization of the network and the dynamic topology of MANETs bring challenges to offering QoS.

Recently, much work has been done on protocols that offer some QoS support in MANETs. In the link layer, joint source-channel coding [19] has been proposed to deal with the trade-off between source coding and channel coding. In the MAC layer, IEEE 802.11e [20], the black burst contention scheme [21], and MACA/PR [22] have been studied. In the network layer, CEDAR [23], ticket-based QoS routing [24], OLSR based QoS routing [25], AQOR [26], ADQR [27], TDR [28], and TDMA supported QoS routing [29][30] have been investigated. In the transport layer, while real-time applications are often built on top of UDP, most work on QoS focuses on TCP [31][32].

Communication among hosts in MANETs faces two limitations: bandwidth and energy. Thus it is necessary to design an architecture that accounts for these limitations. The traditional layered network design provides the ability to design and implement each function independently [33]. However, as communication over MANETs is limited by bandwidth and energy constraints, the layered architecture is not efficient enough to provide optimized performance, as interaction among layers is not allowed. An inter-layer architecture has been studied in INSIGNIA [34] and iMAQ [35] for offering QoS in MANETs.

1.2 Research Motivation

As described in the previous section, QoS is defined as a set of requirements, such as delay, delay jitter, bandwidth, and packet delivery rate, which must be met in transporting a data packet in order to support application functions. In order to facilitate QoS support in MANETs, it is important to understand the metrics that are used to specify QoS.

- Bandwidth. Bandwidth is concave, which means that the end-to-end bandwidth is determined by the bottleneck bandwidth along the path.
- Delay and jitter. Delay and jitter are additive. End-to-end delay and jitter are the accumulation of each single hop delay/jitter.

Supporting more than one QoS constraint is an NP-complete problem [36]. However, delay is associated with network load and degree of congestion. When bandwidth is sufficient, delay is relatively short, but when congestion occurs, delay increases dramatically. The relationship between bandwidth and delay has been studied in [34]. Therefore, while in this dissertation we only study the bandwidth constraint, solving the bandwidth problem inherently helps in solving the delay problem.

Data transmission is the collaboration of each network layer. Thus, a service differentiation model SWAN has been presented in [37] and a QoS model has been discussed in [38]. However, these designs are not comprehensive enough to include all network layers into consideration. Especially for MANETs, where energy and bandwidth are two scarce resources, the requirements for efficiency are more strict. Hence, a cross-layer design QoS architecture could be a solution. Cross-layer design has been studied in application-specific sensor network protocols [39], application-controlled routing [40][41], and protocol frameworks for active wireless networks [42], and this has proven to be an efficient approach. However, rather than fusing layers, it may be sufficient to just enable the sharing of information among the layers of the OSI protocol stack. Thus, the QoS architecture problem becomes which information should be exchanged among the layers to support real-time transmissions in MANETs.

Based on an overview design of a QoS architecture, each layer's function should be determined. In the transport layer, UDP is used for transporting real-time data packets.

End-to-end throughput is greatly decreased beyond the maximal achievable when congestion occurs, which would not happen in wired networks in the single chain topology, as shown in Figure 4.2. Therefore, we argue for the necessity of using congestion control for best-effort traffic to improve the performance in MANETs. The idea of UDPC is proposed to demonstrate the requisite of incorporating a congestion control scheme in best-effort transmission.

In the network layer, an individual node's total available bandwidth is not a static value. On the contrary, it depends heavily on the neighborhood data transmission activities. Therefore, for supporting QoS, incorporating bandwidth estimation into the routing setup procedure is a key to finding the optimum path. Therefore, we study QoS routing based on bandwidth estimation for supporting QoS in MANETs.

The MAC protocol determines the most fundamental performance of data transmission, such as fairness, stability and packet loss rate. Therefore, it is crucial to obtain QoS support from the MAC layer. We utilize the idea of separating the control channel and the data channel for supporting QoS, which is borrowed from the cellular system. We study the protocol design issues, the performance and the trade-off to support QoS at the MAC layer.

1.3 Contributions

This dissertation includes a general information sharing cross-layer architecture and several protocols that address QoS support in MANETs. Specific contributions are as follows:

- A QoS architecture using cross-layer design is proposed, which extends from the application layer to the MAC layer and specifies each layer's features and the information that should be shared among the layers. A performance comparison between the traditional layered design used in MANETs and the proposed QoS cross-layer design in terms of network performance has been made. These results show that using a QoS cross-layer architecture is very effective for offering QoS in MANETs.

- A User Data Protocol with Congestion Control (UDPC) is designed for supporting QoS in the transport layer. The adaptive rate control with feedback to the

application enables efficient congestion avoidance in the transport layer.

- A QoS routing protocol based on bandwidth estimation is designed. This protocol uses two different approaches to estimate available bandwidth, and it supports admission control or feedback through cross-layer design to applications.

- A dual-channel MAC protocol using separate control and data channels is investigated to study the feasibility of improving QoS. This approach is shown to improve fairness.

1.4 Dissertation Structure

This dissertation begins with an overview of related work. A cross-layer framework for supporting QoS in MANETs is described in chapter 3. UDP with congestion control is analyzed and simulated in chapter 4. A survey on existing QoS-aware routing protocols is presented in chapter 5, and a study of routing design issues to support QoS in MANETs is conducted. After the literature study of QoS-aware routing protocols, a novel QoS-aware routing protocol based on bandwidth estimation is proposed in chapter 6. A dual channel CSMA MAC protocol is studied in chapter 7. The dissertation ends with conclusions and future work in chapter 8.

Chapter 2

Background and Related Work

Recent research on MANETs has been conducted in many areas, such as address and service discovery, energy management, scalability, security and QoS. This chapter only covers the basic concepts used in MANETs and the related work for QoS. This chapter first presents the characteristic of MANETs, based on which a corresponding quality of service goal is discussed and defined. Then, related work is detailed.

2.1 Design Goals

Mobile ad hoc networks (MANETs) distinguish themselves from other types of networks by their physical characteristics, organization format and dynamic topology:

- Physical characteristics: Wireless channels are inherently error-prone, due to such effects as multi-path fading, interference and shadowing, causing unpredictable link bandwidth and packet delay.
- Organization format: The distributed nature of MANETs means that channel resources cannot be assigned in a pre-determined way.
- Dynamic topology: As hosts in a MANET are mobile, links are created and destroyed in an unpredictable way.

Therefore, the network status changes quickly, causing hosts to have imprecise knowledge of the current status. Due to these characteristics of MANETs, QoS guarantees are not possible. Hence, **Soft QoS** [24] is proposed, which is detailed in sec-

tion 2.1.1. To support Soft QoS, any bandwidth reservation scheme should be designed to meet the Soft QoS requirements. Therefore, RSVP [43] cannot be applied directly to MANETs, and **soft-state resource management** is proposed instead, which is detailed in section 2.1.2.

Applications requiring QoS support are mostly real-time applications. Adaptation had become a widely embedded feature for most audio and video coding technology. Audio and video applications can optimize their performance according to the network status. However, some minimum bandwidth, which is used to deliver the basic layer data for video and audio packets, should be guaranteed. Thus, both **admission control** and **adaptive feedback service** are required in the QoS support design.

2.1.1 Soft QoS

In wired networks, a source-destination pair remains fixed during a transmission session and thus a virtual circuit can be created. Hence the resources are strictly reserved. The packet delay and packet delivery ratio are guaranteed during a session holding time, providing what is called **Hard QoS**.

However, Hard QoS cannot be applied to MANETs, because resources cannot be guaranteed during a data transmission session due to the unstable links, unreliable wireless channel and unguaranteed resources.

- Unstable links: Hosts in MANETs join, leave and move around the network in a random fashion. Therefore, links are created and destroyed in an unpredictable way. Thus, a stable link cannot be guaranteed in MANETs.
- Unreliable channel: The wireless channel is error-prone, has multi-path fading, and is time-varying, resulting in unreliable communication.
- Unguaranteed resources: The channel resources are shared among hosts in MANETs. No centralized resource allocation is used in MANETs, and thus resource sharing is based on competition and coordination among hosts. Therefore, resources cannot be strictly guaranteed.

Therefore, **Soft QoS** or better than best effort is proposed to meet the characteristics of MANETs. Soft QoS is defined such that a transient time when QoS is not met is allowed during network partitions or when routing failures occur.

2.1.2 Soft-State Resource Management

In wired networks, the resources are reserved at the beginning of a session and the reservation remains fixed during this session. However, this scheme is not flexible enough to be applied to MANETs, where the path could change frequently, requiring a corresponding reservation adjustment. Therefore, soft-state resource management is proposed in MANETs. In this scheme, reservations are updated based on the reception of in-session data. If a packet is not received in a certain pre-defined interval, resources are released.

2.2 OSI Reference Model and Cross-Layer Design

The OSI (Open Systems Interconnection) reference model [33], illustrated in Figure 2.1, includes seven layers. Each layer has a well-defined function, and the layer boundaries are chosen based on minimizing the information needed across layers. The seven layers are defined as follows:

- **Physical Layer** transmits bits over a communication channel.
- **Data Link Layer** breaks the input data into data frames and transmits them in a line appearing free of undetected transmission errors to the network layer.
- **Network Layer** determines how packets are routed from source to destination.
- **Transport Layer** accepts data from the session layer, breaks it into small pieces if necessary, and uses an efficient means of making sure the pieces can be collected correctly at the destination.
- **Session Layer** offers a way that sessions can be established among different machines.
- **Presentation Layer** provides a standard solution to users, and is concerned with the syntax and semantics of the information transmitted.
- **Application Layer** includes protocols such as FTP, HTTP, Real-time Audio, etc.

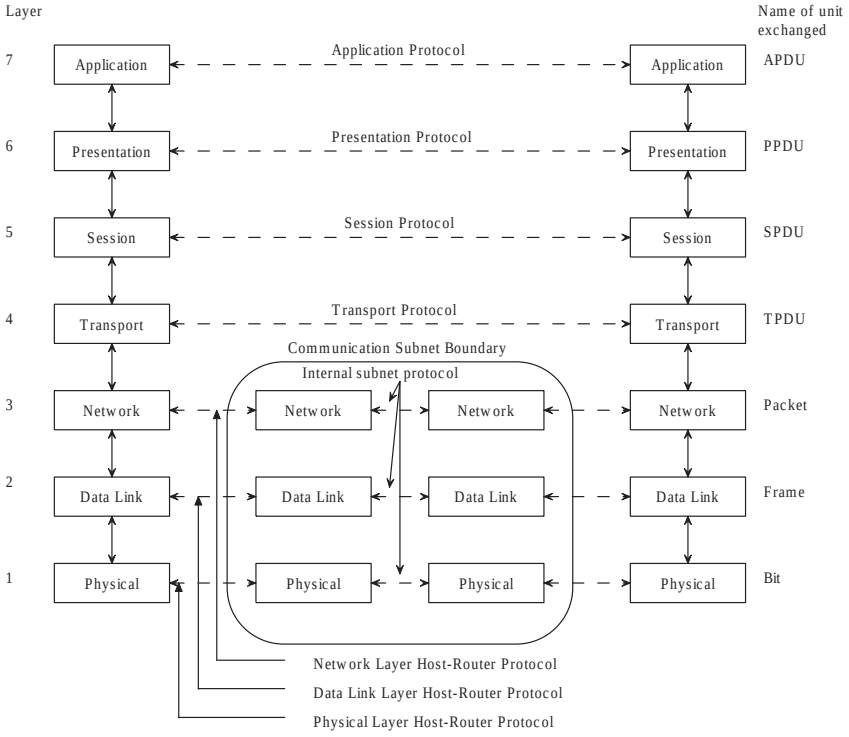

Figure 2.1: The OSI reference model [1].

This architecture allows each layer to be extracted independently, which simplifies the implementation of the architecture. However, the strictly layered architecture might not be the best model, because oftentimes it is not possible to optimize the network performance according to different situations without interaction among the different layers, as argued in [44]. Thus a cross-layer design has been adopted in various wireless networks such as sensor networks, cellular networks and ad hoc networks.

The cross layer design has proven efficient in the LEACH protocol, in which the application layer passes the requirements of the application to the lower layers [39]. The results shown in [39] demonstrate that even under harsh conditions, a high performance can still be achieved. Similarly, a cross-layer design for QoS content delivery in wireless cellular systems is proposed in [45], in which a cross-layer scheme spans among the application, MAC and physical layers, providing an adaptation feature for improving resource efficiency.

A cross-layer design framework is also proposed for MANETs. A cross-layer design for video steaming in MANETs is proposed in [46], in which an adaptive link technique is used to adjust the transmission rate and provide capacity information to the MAC layer from the physical layer; resource allocation is determined in the MAC layer according to multi-path routing information from the network layer; and packet scheduling and source coding are adopted in the transport and application layers based on the joint optimization between the MAC and network layers.

There are several other protocols that use the concept of cross-layer design, such as [41] [47] [48] [49] , which will not be detailed here.

2.3 Related Work

Research in offering QoS in MANETs has covered various aspects, such as architectures, transport protocols, routing protocols, queue management and MAC protocols. The work related to these aspects are presented here.

2.3.1 Architectures

Data communication is the result of each network layer's effort; thus, the cooperation of all network layers is needed to provide QoS support. A service differentiation in wireless ad hoc networks (SWAN model) has been presented in [37]. In the SWAN project, real-time data traffic is admitted by source-based admission control, and best-effort data traffic is regulated by a rate controller. Rate control and admission control are implemented between the IP layer and the MAC layer. In the SWAN project, bandwidth estimation is not directly done by evaluating the residual bandwidth, but it is estimated by the delay information fed back from the MAC layer. The SWAN project is mostly focused on the cross-layer design between the network layer and the MAC layer.

A flexible QoS Model for MANETs (FQMM) is proposed in [38]. The features and functions that should be implemented in the network layer to support QoS have been extensively discussed. However, these designs are not comprehensive enough to include all the networking layers.

2.3.2 Transport Layer Protocols

Most work done in the transport layer has focused on improving the performance of TCP in MANETs (e.g., [31] [50] [51] [52]), where TCP's performance is seriously degraded because it misinterprets link failure as congestion, analyzed in [53], and because of the out-of-order delivery caused by route changes, studied in [50].

TCP-F (TCP-Feedback) [31] relies on an explicit route failure notification (RFN) sent by an intermediate node when a route failure is detected. After receiving an RFN, the sender freezes the congestion window size and retransmission timer, and stops the transmission. ELFN (Explicit Link Failure Notification) [32] is another similar approach based on feedback. It differs from TCP-F in the way it informs the sender of routing failures. ELFN is based on the DSR routing protocol. The route failure message is modified to carry a payload of "host unreachable" message as the ELFN. The sender disables its retransmission timers and enters a "stand-by" state when it receives an ELFN. Therefore, these two approaches use explicit feedback to solve the link failure explained as a congestion problem. A-TCP (Ad hoc TCP) [51] keeps TCP untouched, but puts a thin layer between TCP and IP. This thin layer is designed for listening to the network state information. With the information from the network layer, TCP adjusts its performance by putting performance parameters in the corresponding state, such as persist state, congestion control state, or retransmit state. The fixed RTO (retransmit timer) heuristically assumes that consecutive timeouts signify a broken route. Thus the sender retransmits the unacknowledged packets but keeps the RTO fixed. OOO (Out-of-Order Delivery) is often caused by a route change in the network, so temporarily disabling congestion control and instant recovery during congestion avoidance are adopted in [50] to improve the throughput when multiple out-of-order packets are detected.

However, the performance degradation caused by the effects of congestion collapse in UDP, detailed in section 4.2, has not obtained as much attention as TCP.

2.3.3 Network Layer Protocols

Much research has been done in each network layer to support real-time data transmission. Various routing protocols have been proposed (e.g., [23] [24] [25] [26] [27]

[28] [29] [30] [54] [55]) that either provide admission control or find a path with large enough bandwidth to support a given request. The details of these protocols are presented in chapter 5.

2.3.4 Queue Management

Packet queue management in the link layer has been presented in [56] and [57], where [56] compares two different queue management algorithms and [57] evaluates the performance of different queue scheduling algorithms with the DSR and GPSR routing protocols. Priority scheduling in the MAC layer has been studied in [58] [59] [60].

2.3.5 MAC Protocols

IEEE 802.11, detailed in section 2.3.5.1, is the standard MAC protocol used in WLANs, and also widely used in MANETs. IEEE 802.11 is not designed for MANETs, so it faces several unique problems in MANETs, such as exposed nodes, hidden nodes and unfairness.

In this section, background on the IEEE 802.11 MAC protocol, QoS priority MAC approaches, and the fairness issue of the QoS problem will be detailed.

2.3.5.1 IEEE 802.11 MAC Background

The IEEE 802.11 MAC [3] uses a distributed coordination function (DCF), which is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), as the fundamental mechanism for channel access. In the DCF model, the medium must be sensed free for a time interval greater than the Distributed Inter Frame Space (DIFS) before a node is allowed to transmit. If the medium is not sensed free for DIFS time, the transmission is deferred a random time interval between 0 and Contention Window (CW) times the timeslot. IEEE 802.11 exponentially doubles its contention window when a collision occurs, and it returns to the minimum contention window upon a successful transmission. After the contention window is backed off to a maximum value, the MAC layer reports a transmission failure to the higher layer protocols and drops the packet.

DCF uses a four-way handshake, known as request-to-send (RTS)/clear-to-send (CTS)/Data/ACK. Before transmitting a packet, the source node sends an RTS message, and the destination sends a CTS back to the source node. The nodes that can hear either the RTS or CTS set their network allocation vector (NAV) according to the amount of time that will be used for exchanging the remaining data, as indicated in the RTS and CTS packets. This is designed to solve the "hidden terminal" problem. Providing support for real-time data transmission is an important yet challenging goal for MANETs, and some work has been done to address QoS at the MAC layer.

2.3.5.2 QoS support in the MAC layer

IEEE 802.11e [61] is a standard based on IEEE 802.11 for support of QoS in WLANs. QoS is provided by setting up different Traffic Categories (TCs). Each TC has different contention parameters such as Arbitration Interframe Space (AIFS) values, maximum and minimum backoff window size, and multiplication factors. Using these different contention parameters, virtual competition among traffic categories is created for accessing the channel.

Kanodia et al. [12] [59] proposed a distributed priority scheduling in which hosts put the relative priority overheard from packet exchanges within their broadcast region into a scheduling table. The backoff window is modified based on the information in the scheduling table. Distributed weighted fair queue (DWFQ) [62][63] is another approach based on modifying the backoff window size. The backoff window size is determined by the difference between the actual and expected throughput. Distributed Fair Scheduling (DFS) [13] [63] differentiates the backoff interval (BI) based on the packet length and traffic class. Distributed Deficit Round Robin (DDRR) [64] varies the interframe space (IFS) based on the values of accumulated traffic and assigned throughput.

2.3.5.3 Fairness support in the MAC layer

Fairness and probability of collision are two parameters used to measure the efficiency of a MAC protocol. IEEE 802.11 has been shown to be a low collision probability protocol, which is achieved by using *binary exponential backoff* (BEB). However, it

suffers from unfairness[1].

IEEE 802.11 based on CSMA/CA is an unfair MAC protocol. However, its counterpart, the wired Ethernet protocol based on CSMA/CD, is known to be a fair MAC protocol. The root reason for this difference is the media. Hosts in wired networks can sense whether other hosts are using the channel. In a wireless channel, the hosts' sensing range is strictly limited by the transmission power. In addition, BEB in IEEE 802.11 is used to solve the collision problem. However, it exacerbates the unfairness, because it always favor the host that captured the channel most recently, which results in extreme unfairness under a heavy traffic load.

MAC unfairness results in starvation of certain flows, which significantly affects real-time applications where delay and bandwidth are extremely sensitive. In addition, multi-hop relay data transmission could be degraded significantly, since the early hosts' data transmissions could starve later hosts.

The first fairness study for wireless LANs was pioneered by Barghavan et al. in the MACAW project [10], where a modified backoff algorithm, per-stream fairness, Data-sending packets and Request-for-Request-to-Send (RRTS) packets were proposed to improve fairness.

- *Modified BEB backoff algorithm:* A packet header includes the backoff value. Hosts copy this backoff value upon hearing a packet. Therefore, all hosts share the same backoff value in a single cell scenario, which eases the unfairness caused by the relatively high backoff value of the host that fails to capture the channel during the last packet transmission period. In a multiple cell scenario, a per-destination copying algorithm is used to avoid different base-stations using the same backoff value, which might result in a relatively large backoff value, thus degrading performance.

- *Per-stream fairness:* Per-steam fairness is achieved by separating queues for each flow, and using an independent backoff algorithm, which is the basis for the traffic categories used in 802.11e.

- *Data sending packet:* To avoid sending unnecessary RTS packets during a data

[1]Unfairness is defined as obvious performance differences without explicit differences on data traffic conditions.

packet transmission period, a 30-byte Data-Sending packet is sent. Another alternative way is using carrier sensing before sending an RTS packet, which is adopted by IEEE 802.11. Therefore, the backoff counter will not exponentially increase its value, which helps improve fairness.

- *RRTS:* RRTS is a packet sent by a station that receives an RTS but cannot respond due to deferral for another station's transmission. When a host receives an RRTS, it responds immediately with an RTS packet. Through this way, the fairness can be partially improved under an asynchronous scenario [10].

In [11][65][66], a fairness index is introduced to measure the degree of fairness, which is defined as follows:

$$FI = max\{\forall i, j, min_{i,j}(\frac{W_i}{\phi_i}, \frac{W_j}{\phi_j})/max_{i,j}(\frac{W_i}{\phi_i}, \frac{W_j}{\phi_j})\} \tag{2.1}$$

where ϕ_i is a pre-defined fair share and W_i is the actual achieved throughput. Thus, solving the fairness problem becomes maximizing the fairness index locally. After calculating the fairness index, hosts halve the contention window if they cannot get the share they are supposed to get, and they double the contention window if they obtain much more than their expected share.

A *Distributed Wireless Ordering Protocol* (DWOP) is presented in [12]. This approach explores the overheard information from others to build a scheduling table. An estimation of contention time is made based on the scheduling table. Each host is allowed to contend for the channel when it has packets with the lowest arrival time among its transmission range. Therefore, the starvation of a particular flow is avoided, which improves fairness.

A *Distributed Fair Scheduling* (DFS) protocol is proposed by Vaidya et al. [13][67], in which the idea of transmitting the packet with the smallest finish time is borrowed from the *Self-Clocked Fair Queueing* (SCFQ) protocol [68]. In DFS, the backoff interval is calculated as follows:

$$B_i = \lfloor scalingfactor \times \frac{L_i^k}{\phi_i} \rfloor \tag{2.2}$$

where L_i^k denotes the packet length of the k^{th} packet of flow i and ϕ_i is the i^{th} flow's fairness share. The backoff interval is further randomized by multiplying with a random

variable (scaling factor) within $[0.9, 1.1]$. Through modifying the backoff interval in IEEE 802.11 DCF, the backoff interval is inversely proportional to the weight of a flow. Thus, DCF's fairness is greatly improved.

An *Asynchronous Multi-Channel Protocol* [69] (AMCP) takes advantage of spare channels in the IEEE 802.11 standard, in which only one channel is used for transmission among the reserved eight channels. Hence, the data and the control packets can transmit in separate channels. Therefore, collisions between data packets and control packets can be significantly reduced, which results in starvation mitigation and throughput improvement.

Chapter 3

Network Architecture to Support QoS in Mobile Ad Hoc Networks

In this chapter, we present an information-sharing cross-layer QoS architecture for supporting real-time data transmission in MANETs. The QoS architecture includes a QoS transport layer, QoS routing, queue management and a priority MAC protocol. Through simulations, we find that the QoS architecture reduces packet delay and greatly improves the quality of real-time video streams in MANETs.

3.1 QoS Architecture

In Figure 3.1, we show our proposed QoS architecture, which includes all networking layers from the application layer to the MAC layer. The bold lines indicate the flow of data packets and the narrow lines indicate the flow of control packets. Each layer's features are detailed below.

3.1.1 Application Layer

Applications can be categorized into real-time and non-real-time applications based on their sensitivity to packet delay. Real-time applications have strict requirements on the packet delay. Therefore, packet retransmission is not allowed. The applications that fit into this category are on-line live movies and video conferencing. Many video compression technologies, such as MPEG-4, H.263, and multiple-description coding,

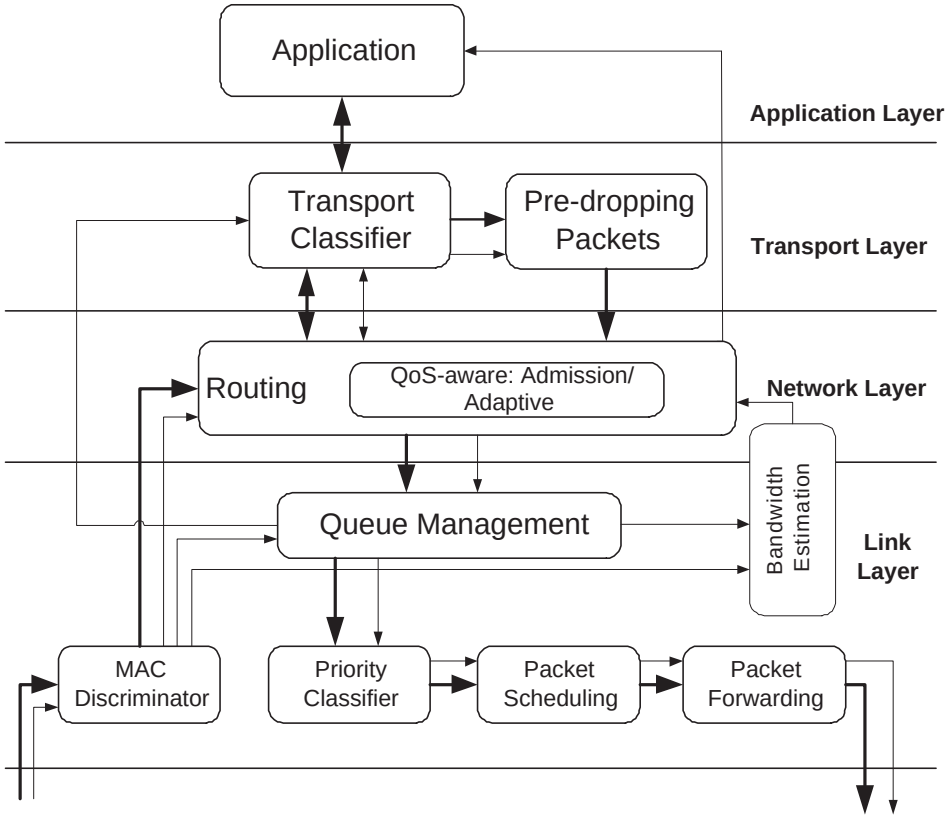

Figure 3.1: QoS architecture.

can compress video with different coding rates to meet different channel conditions. In addition, most of these compression schemes have error resilience features to recover the video frame, if some packets are lost. Thus, choosing the right coding rate to compress the video is important, and some reasonable packet loss is acceptable. On the other hand, for non-real-time applications such as Email and FTP, packet delay is not a big issue, and packet delivery is guaranteed by explicit acknowledgements in the transport layer. The network should be designed to meet the different packet delay requirements of these two types of applications.

3.1.2 Transport Layer

UDP and TCP are two transport layer protocols widely used in wired networks. UDP has no congestion control scheme to react to network congestion. Applications that

use UDP as the underlying transport protocol to transmit packets can easily overwhelm the network with data, which results in a considerable amount of wasted energy and bandwidth in transmitting packets that will be dropped due to congestion. Therefore, some pre-dropping of UDP packets should be investigated to react to congestion [70]. TCP has an inherent congestion control scheme, so congestion control is not a problem. However, TCP's performance should be optimized to adjust the TCP window, which requires feedback information from the lower network layers [31]. Therefore, some information from the packet queue and the routing layer should be sent to the transport layer for performance optimization.

3.1.3 Network Layer

To support QoS, the routing protocol should have an embedded scheme such as call admission or adaptive feedback that is designed to support QoS. At the same time, non-QoS-aware routing that is targeted at finding a feasible path should be offered as well. For QoS-aware routing, information about the current network status is provided to the application for performance optimization. Also, the routing layer should get enough channel information from the lower layers so that the admission/adaptive scheme can be performed based on the network status. Therefore, two cross-layer designs should be implemented in QoS-aware routing. One is to obtain the network resource information from lower layers, and the other is to send the network status to the application. To offer QoS to the application, resource reservation should be incorporated. An RSVP-type signaling scheme [43] is not desirable in MANETs due to its high overhead. Therefore, in-band and soft resource reservation (i.e., best effort rather than guaranteed reservations) should be done during the route discovery phase and during route maintenance. The transmissions that occur between the break down of old routes and the set up of new routes will severely affect the QoS provided by the network. Therefore, some prediction of route breaks should be incorporated. Overall, QoS-aware routing should have the following features that traditional routing does not support:

- obtain resource information from lower layers;
- offer bandwidth information to applications;
- incorporate resource reservation schemes; and

- predict route breaks.

3.1.4 Link Layer

The link layer needs to discriminate the different priority packets and schedule packet delivery according to priority levels. The service differentiation should be completed in the packet queue through queue management and in the MAC layer through a MAC discriminator and priority classifier.

Queue Management: The aim of queue management is to schedule the different priority packets. Real-time data should have higher priority to be sent to the channel compared with packets such as FTP and Email. Therefore, real-time data will be put in front of the non-real-time data in the packet queue. When the network is congested, the last packet in the packet queue will be dropped. Therefore, incorporating queue management will reduce the possibility that real-time packets are dropped in the packet queue when the network is congested. Thus, the delay of real-time application data packets can be reduced and the packet delivery ratio can be improved. Also, the packets whose delay has already exceeded the applications' requirement should be eliminated from the packet queue before transmission to save the transmission of packets that will be useless to the receiver. If different flows go through the same host, it is easier to do the priority regulation in the packet queue than in the MAC layer.

MAC Discriminator: The main function of the MAC discriminator is to differentiate data packets and control packets that arrive from the wireless channel. Data packets are sent to the network layer; ARP (address resolution protocol) packets go to the queue directly; MAC packets, such as the RTS, CTS, and ACK packets used in IEEE 802.11, stay in the MAC layer; and the bandwidth estimation control packets are sent to the bandwidth estimation module for use in the routing layer's admission/adaptive scheme.

Priority Classifier and Packet Scheduler: To offer service differentiation in a distributed ad hoc network, real-time packets should be granted higher priority to capture the channel. The priority classifier differentiates the different data packets that arrive from the packet queue and directs the packet scheduler to schedule the packet delivery based on the priority level of the current packet.

3.1.5 Bandwidth Estimation

In a distributed ad hoc network, a host's available bandwidth is not only decided by the raw channel bandwidth, but also by its neighbor's bandwidth usage and interference caused by other sources, each of which reduces a host's available bandwidth for transmitting data. Therefore, applications cannot properly optimize their coding rate without knowledge of the status of the entire network. Thus, bandwidth estimation is a fundamental function that is needed to provide QoS in MANETs. However, bandwidth estimation is extremely difficult, because each host has imprecise knowledge of the network status and links change dynamically. Therefore, an effective bandwidth estimation scheme is highly desirable. Bandwidth estimation can be done using various methods; for example, in [26] bandwidth estimation is a cross-layer design of the routing and MAC layers, and in [28], the available bandwidth is estimated in the MAC layer and is sent to the routing layer for admission control. Therefore, bandwidth estimation can be performed in several different network layers, as shown in Figure 3.1.

3.2 Simulations

We built a simplified network model to support real-time data transmission. In the simplified network model, the following designs are incorporated.

- Priority packet scheduling scheme: The packets with high priority (e.g., real-time data packets, routing packets and ARP packets) are always put in front of the non-real-time data packets. The packets in the tail of the queue are dropped when congestion occurs.

- Bandwidth estimation-based QoS-aware routing: Each host periodically estimates its own bandwidth use with MAC layer bandwidth estimation, and this information is disseminated to the host's two-hop neighbors through "Hello" packets. Each host's available bandwidth is estimated based on the bandwidth used by itself as well as each of its one-hop and two-hop neighbors. Either an admission control scheme is used during route discovery, or adaptive feedback is embedded in the route reply packets. This procedure, which is an extension to AODV, is detailed in [71].

Table 3.1: Architectures used in the simulations.

Architecture	**Routing**	**Queue**	**MAC**
No QoS	AODV	FIFO	IEEE 802.11
QoS1	QoS-aware	Priority Queue	IEEE 802.11
QoS2	QoS-aware	Priority Queue	Enhanced IEEE 802.11

- Enhanced IEEE 802.11: AIFS (Arbitration Interframe Space) and different window sizes [58] are applied to help high priority packets capture the channel. The MAC layer assigns shorter AIFS and smaller window sizes to real-time data packets.

To test the performance of our simple QoS architecture, we simulate an ad hoc network using the ns-2 simulator. The H.263 TMN simulator is used to show the video quality improvement using a QoS architecture in ad hoc networks. In our simulations, 50 static nodes are randomly placed within a 1000 m by 1000 m area, and we simulate twenty different random scenarios. The packet size used in our simulations is 500 bytes and the raw channel bandwidth is 2 Mbps. Three source-destination pairs are randomly chosen to simultaneously transmit non-real-time data packets, and one randomly chosen real-time data flow joins in after 10 seconds. The total simulation time is 200 seconds. UDP is used as the underlying transport layer protocol for both the real-time and the non-real-time streams.

The three different network architectures shown in Table 3.1 are tested. No QoS uses the regular AODV routing, a FIFO queue, and the IEEE 802.11 MAC. QoS1 uses QoS-aware routing with adaptive feedback, priority packet scheduling, and the IEEE 802.11 MAC. QoS2 uses QoS-aware routing with adaptive feedback, priority packet scheduling, and the IEEE 802.11 MAC with AIFS and smaller window sizes for high priority data packets.

All flows initially attempt to send at the same rate. The real-time application can adjust its sending rate according to the network status if QoS-aware routing is used and the available bandwidth information is sent to the application by the cross-layer design. Figure 3.2 shows the real-time data flow's delay. We can see that using QoS-

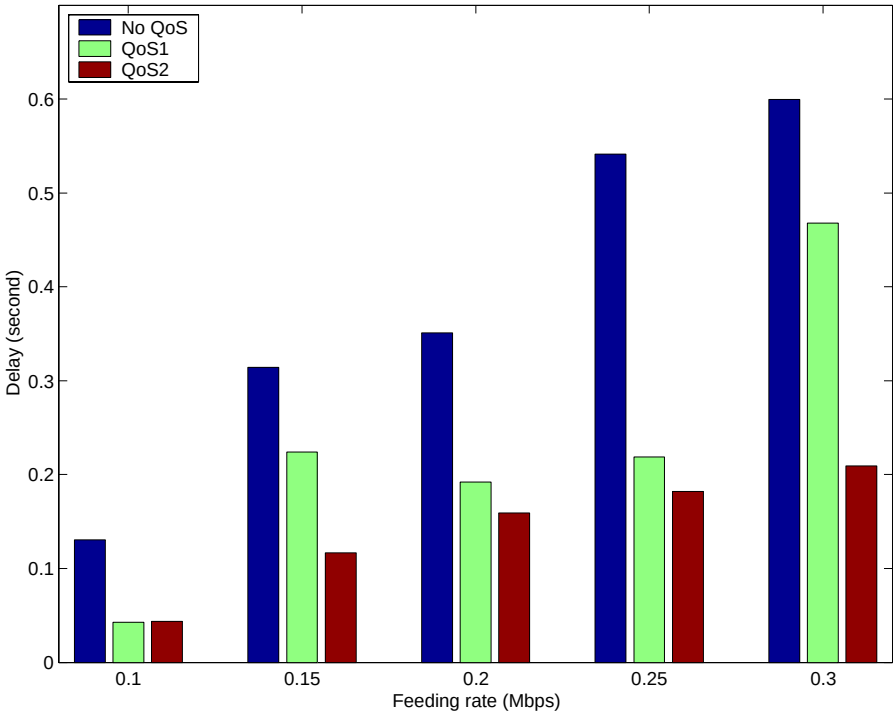

Figure 3.2: Real-time data flow's average packet delay.

aware routing and priority queue management reduces the average packet delay, and incorporating differentiated MAC further decreases the delay.

We randomly choose one of the twenty scenarios to test one real-time data flow and three non-real-time data flows. The three non-real-time data flows transmit data at 80 kpbs, and the real-time application begins to send a video stream at 300 kbps ten seconds later. Using the QoS2 architecture results in the video frame quality shown in Figure 3.3; using the No QoS architecture results in the video frame quality shown in Figure 3.4. Using adaptive feedback, the application compresses the video according to the network status, which avoids overwhelming the network, so congestion can be avoided. Therefore, the packet delivery rate is increased, which helps to improve the video quality.

Figure 3.3: Video frame using the QoS2 architecture.

Figure 3.4: Video frame using the No QoS architecture.

3.3 Conclusions and Future Work

In this chapter, we presented a network architecture that supports QoS in MANETs, and we simulated a simplified version of this QoS architecture. Our simulation results show that video quality can be greatly improved and average packet delay can be significantly decreased using our QoS architecture. However, the performance of multiple different priority streams has not been presented. This will be a topic of future research.

Chapter 4

End-to-End Congestion Control for Best-effort Transmission

UDP is the standard best-effort transport layer protocol. A well-known problem with UDP is that, unlike TCP, it does not include congestion control. Thus UDP connections have no way to detect or react to network congestion. This is particularly a problem in wireless networks, where a significant loss in achievable capacity occurs when using UDP transport over a congested multi-hop network running the 802.11 MAC and energy is wasted on transmitting packets that will be dropped before reaching their destination. Mobile ad hoc networks lack energy and bandwidth. Therefore, it is necessary to use congestion control to improve the network performance while using UDP as the transport layer protocol. In this chapter, we explore the use of UDP with congestion control, and our simulations show that adding congestion control to best-effort traffic can improve the network capacity greatly and increase energy efficiency dramatically.

4.1 Background

A mobile ad hoc network (MANET) consists of many individual hosts that operate not only as data sources and sinks, but also as routers to forward packets for other hosts in the network. As there are many advantages to using such an infrastructure-less network, much research has been done on protocols for MANETs. The challenge in wireless ad hoc networks is that neighboring hosts must share the bandwidth without centralized

control. Furthermore, intermediate hosts take part in forwarding packets. Therefore total effective capacity achievable is not only limited by the raw channel capacity, but it is also limited by how the MAC protocol schedules packet forwarding. Thus, increasing the channel spatial reuse and reducing packet loss are highly desirable.

Early research on capacity issues shows that the maximum capacity of a long chain of nodes in isolation is 1/4 of the channel bandwidth, but simulation results using the IEEE 802.11 MAC show the achievable capacity is only 1/7 of the channel bandwidth because the early nodes in the chain starve the later nodes from channel access [72]. Thus the MAC protocol has a significant impact on achievable throughput.

Similarly, routing may have a negative impact on achievable capacity. If a node cannot access the channel for a long time or data collides during transmission, this will be reported to the routing protocol as a routing failure. The problem is that the symptoms of routing failure and congestion are the same—packets cannot reach their destination node. The routing protocol has no way to determine whether the cause of a failed packet delivery attempt was due to congestion or a broken link caused by node mobility. Existing routing protocols may assume that packet delivery failure is due to a broken route. This in turn affects the end-to-end throughput, since route discovery is initiated even if the cause of the packet delivery failure was congestion, and all pending data transmissions are held until a new route is found. Therefore, having congestion control can reduce the mis-report of routing failure, and further assist the packet transmission.

Most ad hoc mobile devices operate on batteries, so the power consumption is another important issue. Various ways can be adopted to reduce the power consumption, such as adjusting the receiver power to optimize the power saving and putting hosts into the sleeping mode. However, the most efficient use of power is to not transmit the packets that will not eventually reach the destination under the highly congested situation. So under the constraints that the sender and receiver need to use a fixed power to communicate with each other, the power saving problem can be simplified to avoid wasting unnecessary packet transmissions when the packets will be dropped during transmission due to congestion. Thus, estimating the channel traffic and pre-dropping the traffic are effective ways to improve energy efficiency.

As the energy and bandwidth usage efficiencies are the main design factors for

wireless networks, therefore incorporating a congestion control scheme into best-effort traffic in ad hoc network is highly desirable. Even in wired networks that have no consideration for energy and bandwidth, incorporating congestion control into best-effort traffic has obtained some attention [73] [74]. Researchers are arguing that it is necessary to use end-to-end congestion control in wired networks in order to improve the overall networking performance. Three approaches are presented: using routers to schedule packet delivery for each flow [75]; having routers support end-to-end congestion control [73]; or using pricing mechanisms. The first two methods use routers to control packet delivery. In ad hoc networks, every host performs the router functions, and no additional routers are provided. Therefore, we need another way to deal with the congestion control problem for best-effort traffic in MANETs. TCP incorporates congestion control in the protocol. To implement congestion control for best-effort traffic in ad hoc networks, we propose incorporating congestion control into the UDP transport-layer protocol as well, not to ensure reliable delivery as in TCP but to ensure that the UDP source does not overwhelm the limited bandwidth of the wireless channel and dissipate energy for transmitting packets that will be dropped ultimately under the congestion situation. This will improve overall performance for all hosts in the network.

We begin our study on the problem of congestion in ad hoc networks using ideal routing[1] to study MAC scheduling and the effects of congestion control in static ad-hoc networks using best-effort traffic (UDP transport layer).

We then look at the performance in congested networks and find that end-to-end throughput is greatly decreased beyond the maximum achievable when congestion occurs. We build off this initial motivation by studying the problem of congestion for both static and mobile networks. We add congestion control to UDP traffic and compare this to UDP without congestion control for AODV networks using various topologies, from static to mobile and from single chain to complex random scenarios. Our results show the need to incorporate some form of congestion control with best-effort traffic in MANETs to improve effective network capacity. We are not arguing that our congestion control scheme is the best, but we are arguing the necessity of using congestion

[1]Ideal routing is one where the route is never broken during the transmission of data, so the route discovery procedure is never initiated.

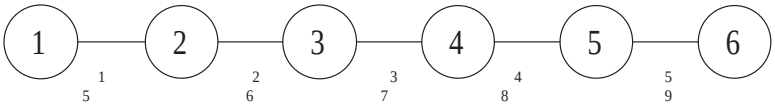

Figure 4.1: 6-node chain topology with optimum channel assignment schedule.

control for best-effort traffic to improve the ad hoc networking performance.

4.2 Motivation

MAC protocols play an important role in determining overall capacity in mobile ad hoc networks. We begin our research by exploring the relationship between the 802.11 MAC protocol and end-to-end throughput in a chain topology shown in Figure 4.1 assuming ideal routing (i.e., no route maintenance functions are employed). The transmission range is one hop, and the interference range is two hops. Figure 4.2 shows that for a six node chain topology using a 2 Mbps channel, end-to-end throughput increases linearly with offered load up to an offered load of 0.4 Mbps, when the end-to-end throughput drops dramatically. This “cliff” is called congestion collapse in wired networks. While congestion collapse does not occur in wired networks when the topology is a single chain, it does occur in MANETs for a single chain topology and is caused by end nodes in the chain topology facing less channel contention than intermediate nodes. Thus the source node captures the channel and starves intermediate nodes of channel access. Bandwidth is wasted by the source node transmitting packets that will be dropped before arriving at the final destination. Therefore the congestion of best-effort traffic in ad hoc networking is much more severe than in wired networking. Thus we need to incorporate congestion control in best-effort traffic.

We want to determine theoretically where the “cliff” occurs to find the optimal sending rate that maximizes end-to-end throughput. To do this, we introduce a time slot concept, where the time used to transmit one packet between two neighbors is one time slot. The best sending schedule for a six node chain topology is shown in Figure 4.1. The numbers marked between two neighbor nodes are the time slots assigned for data transmission over that link. The best forwarding scheme without collision occurs when node 1 feeds packets to node 2 every 4 time slot intervals. In time slot 1, node 1

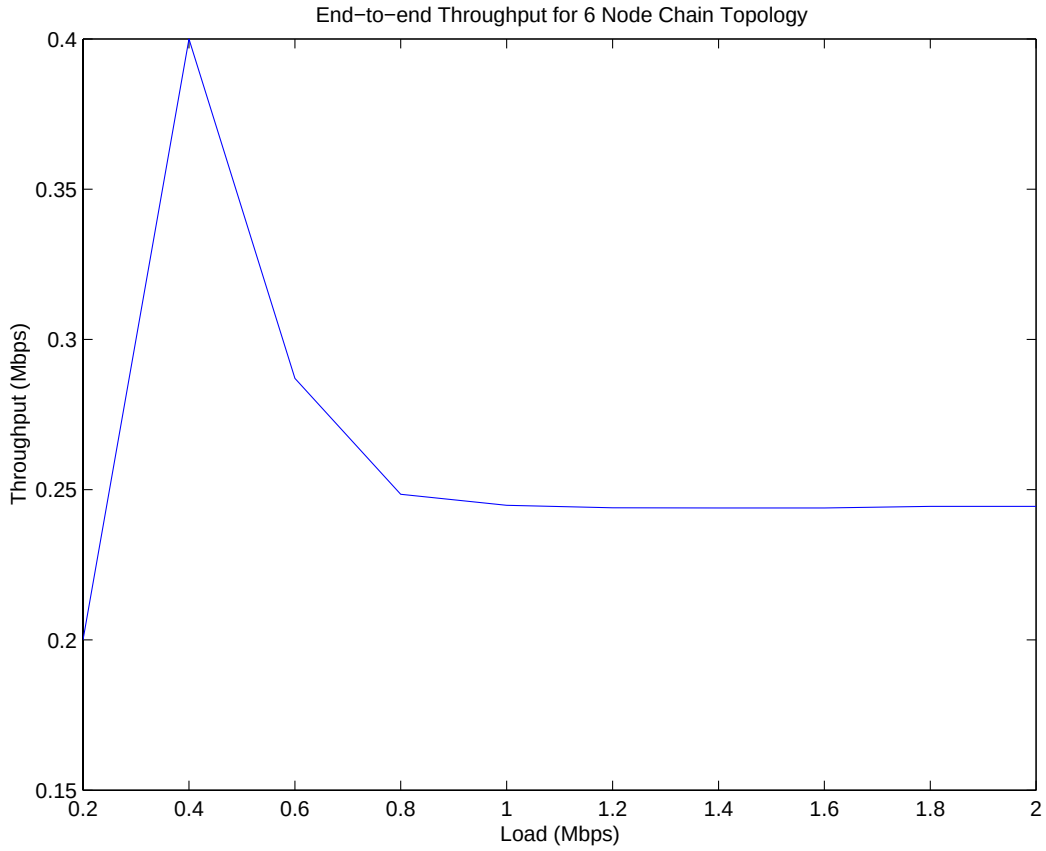

Figure 4.2: End-to-end throughput for the 6-node chain topology using ideal routing.

forwards the packet to node 2; in time slot 2, node 2 forwards the packet to node 3, so node 1 is in the interference range of node 2; in time slot 3, node 3 forwards the packet to node 4, and nodes 1 and 2 are in the interference range of node 3; in time slot 4, node 4 forwards the packet to node 5, and nodes 2 and 3 are in the interference range of node 4, but node 1 still cannot transmit another packet to node 2 because node 2 is in the interference range. Therefore, the earliest time slot available for node 1's subsequent transmission is time slot 5. If we do not consider DIFS, SIFS, backoff time and propagation time, the rough time for one slot should be the time for transmitting the RTS, CTS, data packet and ACK. An RTS packet is 44 bytes, CTS and ACK packets are 38 bytes each, the MAC header is 52 bytes, the IP header is 20 bytes, and the data

packet is 1500 bytes. Therefore, one time slot's length is

$$\frac{RTS + CTS + (Data + MACHdr + IPHdr) + ACK}{Capacity} \tag{4.1}$$

$$= \frac{(44 + 38 + (1500 + 52 + 20) + 38) \times 8}{2 \times 10^6} = 6.77\ ms$$

According to this, we can get the analytical result that the optimal end-to-end throughput can be achieved when the load is 0.44 Mbps.

$$\frac{PacketSize}{SendingInterval} = \frac{1500 \times 8}{0.00677 \times 4} = 0.44\ Mbps \tag{4.2}$$

If we consider DIFS, SIFS, backoff time and propagation time, the optimum load should be slightly lower than this value. The maximum end-to-end throughput from our simulation results shown in Figure 4.2 is very close to the theoretical value above.

If the source can feed the network at the optimum load, the throughput should be improved. Therefore, the problem becomes how to let the source node know the best feeding rate. In a simple network such as the chain network, the source could know a-priori the optimum sending rate. However, in real networks, the flow from source to destination will not be the only flow in the network, and the flow will experience contention not only from its own packets but also from those of other flows in the network. Thus we need some mechanism to dynamically inform the source node of the appropriate sending rate as traffic in the network changes.

As stated previously, packet loss occurs in MANETs due to both congestion and broken links due to node mobility. Using ideal routing, all lost packets are assumed to result from congestion. This holds true in a static network, but in a mobile network, packet losses may be due to broken routes as well as congestion. On the other hand, ad hoc routing protocols assume that all lost packets are due to route failure and thus re-initiate route discovery to find a new route to the destination. In congested networks, performing route discovery may not only be unnecessary, it may also further increase congestion. For example, Figure 4.3 shows how the end-to-end throughput changes with different offered loads using UDP in a static 6-node chain topology. Due to the interaction between the routing protocol and the link layer protocol, the peak throughput achievable using real routing (AODV) is less than that achievable using the ideal routing protocol (Figure 4.3), but the trend that after a certain load the end-to-end

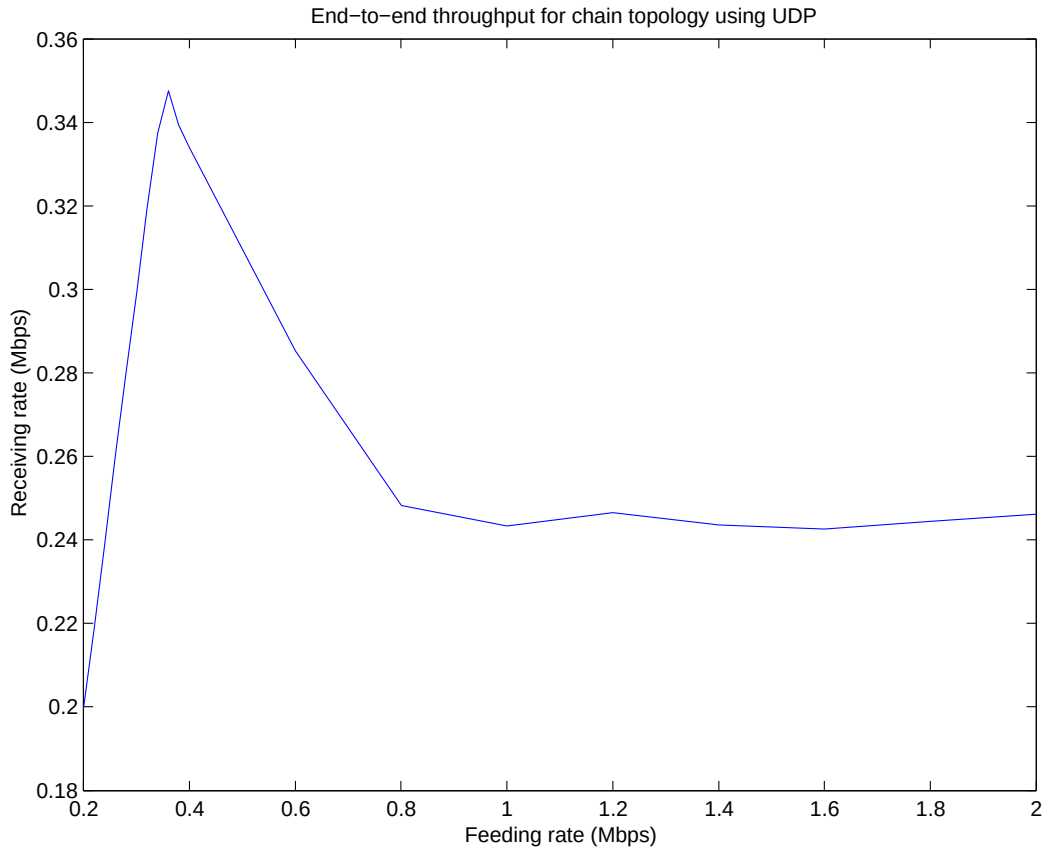

Figure 4.3: End-to-end throughput for the 6-node chain forwarding topology using AODV.

throughput decreases sharply and finally converges to a certain value is the same for both routing protocols. The peak throughput obtained using ideal routing is higher than using real routing because ideal routing does not consider routing failure (and since this is a static network, there are no routing failures). Thus there is no wasted bandwidth and transmission time searching for new routes in a static topology. On the other hand, the route maintenance scheme tries to detect transmission failures and repairs broken routes. Thus, sometimes congestion will cause a routing failure to be reported incorrectly.

From Figure 4.3, we can see that throughput decreases rapidly after the end-to-end throughput reaches its peak value. In wireless networks, the total available bandwidth is limited and spatial sharing of this limited bandwidth among neighboring nodes further

deceases the bandwidth per traffic flow. Therefore, in order to obtain the best possible performance from the network, the offered load should be kept to a rate at or near the optimal offered load, which delivers the optimal end-to-end throughput. This motivates us to use a congestion control scheme to deal with this problem for mobile ad hoc networks.

Our goal is to find an approach whereby network resources are not wasted with the transmission of packets that will be dropped before reaching their destination. We accomplish this goal by using UDP with congestion control, and we show that improvements in overall end-to-end throughput are possible for both static and mobile networks.

4.3 Congestion Control

Traditionally, the best-effort service model treats every packet equally without any discrimination or explicit delivery guarantees. Users do not need to request permission before transmission, therefore the performance of every flow is affected not only by itself but also other flows. The network suffers due to this uncoordinated behavior, which is inherent in the best-effort datagram in wireless ad hoc networking, because of wireless channel sharing and the lack of router functions.

TCP congestion control is the transport layer protocol used for reliable packet transmission in wired networks. TCP uses the well-proven Additive Increase and Multiplicative Decrease (AIMD) to efficiently respond to the resource availability. With the unique nature of wireless channels, it has been shown that rate-based congestion control is more desirable than window-based congestion control. However, AIMD is still widely accepted as the best congestion control scheme used in a reliable transport protocol for wireless wide-area networks [23].

As our concern is using congestion control in the UDP protocol, we do not want the overhead that TCP's acknowledgements will bring and also guaranteed transmission is not necessary for best-effort traffic. Thus congestion control using AIMD that is adopted by TCP cannot directly apply for UDP. Therefore, we implement both AIMD and Multiplicative Increase and Multiplicative Decrease (MIMD) rate-based congestion control for UDP.

UDP with Additive Increase and Multiplicative Decrease (UDP-AIMD) congestion control, and UDP with Multiplicative Increase and Multiplicative Decrease (UDP-MIMD) congestion control are two end-to-end congestion control protocols for best-effort traffic that use explicit feedback from the destination. They are built on UDP and add two features to the UDP protocol: 1. receiver feedback (ACK packets) that inform the sender of the amount of data received at the destination and the time interval in which this data was received, and 2. an adaptive sending rate algorithm at the sender to adjust the sending rate using the feedback information. We chose a rate-based scheme instead of a window-based scheme as TCP uses because window-based protocols introduce burstiness, which is not desirable in wireless mobile ad hoc packet delivery. It has been proven that rate-based congestion control algorithms provide better performance than window-based algorithms in wired networks as well [76].

UDP-MIMD and UDP-AIMD can be divided into two modules: sender module and receiver module. In the sender module, there is one timer, one buffer and five registers. Of the five registers, two are time registers, two are received data registers, and one is a received data rate register. The two time registers, $OldAckTime$ and $NewAckTime$, record the time when ACK packets are created at the receiver, as specified in the ACK packets. The two received data registers, $OldAckBytes$ and $NewAckBytes$, record the total amount of data received at the receiver, as specified in the ACK packets. The receiving data rate register is used to calculate the receiver's actual receiving data rate

$$ReceiveRate = \frac{NewAckBytes - OldAckBytes}{NewAckTime - OldAckTime} \tag{4.3}$$

The algorithm to update the registers is shown in Figure 4.4.

As the sending rate is the main factor in UDP-AIMD and UDP-MIMD, in AIMD, we use an additive increase and multiplicative decrease scheme, which is the same as in TCP. The sending rate adjustment works as follows:

If ($\frac{SendRate}{ReceiveRate} \geq gap$)

$$SendRate = \frac{SendRate}{2}$$

Else if ($SendRate \geq SSThresh$)

$$SendRate = SendRate + IncRate$$

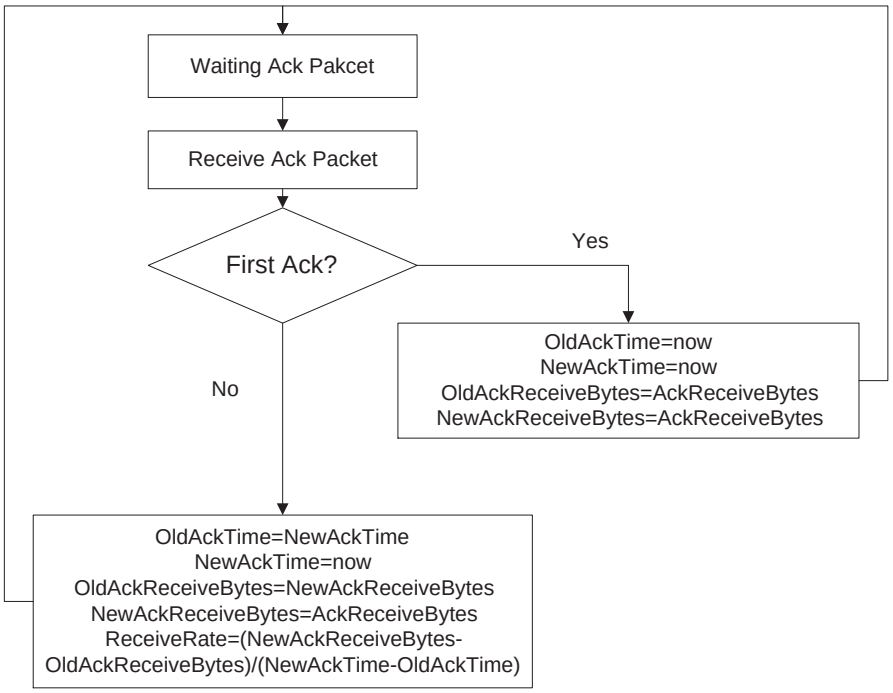

Figure 4.4: Algorithm for updating the sender's registers.

Else if ($ReceiveRate \times 2 \geq SSThresh$)

$$SendRate = SSThresh$$

Else

$$SendRate = ReceiveRate \times 2$$

UDP-AIMD congestion control's idea is that if the receiving rate differs greatly from the sending rate, it is assumed that there is congestion. Thus the source will multiplicatively decrease the sending rate to meet the channel condition. If the source finds that congestion does not occur after the multiplicative decrease of the sending rate, it multiplicatively increases its sending rate, until it reaches the slow start threshold. While the sending rate is greater than the slow start threshold, the sending rate increases additively. There are three parameters that need to be selected: feedback interval, additive increase rate and gap. The "gap" is used to detect congestion. Intuitively, if a large number of packets are dropped, we consider that there is congestion. Therefore if we

choose the value 1.1 as the value, it will be reasonable. In TCP, the additive increase rate is chosen as a conservative value. The purpose of using TCP is to offer reliable data transmission instead of best utilizing capacity. Also, TCP acknowledges every data packet, so the additive increase will not be too slow even using the conservative value. However, in UDP, we do not want to know whether every packet is received, so the feedback packet should be sent in a relative long interval compared with the acknowledgement of every packet. It is quite reasonable if we choose sending feedback every second. Therefore the additive increase rate affects the end-to-end throughput greatly. We will discuss this more in section 4.4.

In MIMD we use an adaptive multiplicative control scheme to set the sending rate. A slow multiplicative increase and decrease is used when the sending rate is close to the receiving rate, and a large multiplicative decrease is used when the sending rate is far away from the receiving rate. The sending rate adjustment works as follows:

If ($\frac{SendRate}{ReceiveRate} \geq \beta$)
$$SendRate = \frac{SendRate}{2}$$

Else if ($\gamma \leq \frac{SendRate}{ReceiveRate} \leq \beta$)
$$SendRate = SendRate \times \epsilon$$

Else if ($\zeta \leq \frac{SendRate}{ReceiveRate} \leq \gamma$)
$$SendRate = SendRate \times \eta$$

Else ($\frac{SendRate}{ReceiveRate} \leq \zeta$)
$$SendRate = ReceiveRate \times \theta$$

where $\gamma, \beta, \eta > 1$, $\zeta, \epsilon < 1$, $\zeta < \gamma < \beta$, and θ is a value around 1. Thus we can get the packet sending interval to meet the desired sending rate as:

$$SendInterval = \frac{PacketSize}{SendRate} \tag{4.4}$$

When a new packet arrives at the transport layer for transmission, the interval between

the last transmitted packet and this packet is calculated as:

$$ArrivalInterval = CurrentTime - LastSendTime \tag{4.5}$$

In UDP-MIMD and UDP-AIMD, we use a timer combined with a buffer to precisely schedule packet transmission according to the sending rate set by the sender module. The timer is needed to trigger the packet delivery when $SendInterval$ time has passed since the last packet was transmitted. The buffer is used to store packets that arrive before $SendInterval$ time has passed and are thus not allowed to be sent immediately. When there are packets in the buffer, the sending packet timer is set according to the $SendInterval$ value. When the buffer is empty, if $ArrivalInterval$ is greater than $SendInterval$ the packet is sent; otherwise the packet is put in the buffer. This scheme is shown in Figure 4.5.

The receiver module is simple. It keeps a timer and a data arrival counter. The counter and timer are initiated when the first packet arrives. After α seconds, the receiver sends back an acknowledgment (ACK) packet that informs the sender of how many total packets have been received so far and the time between this ACK packet and the last ACK packet. The receiver module then resets the timer. We decide not to send back an ACK for every packet that arrives because this would add extra traffic, further increasing congestion and collisions. Since the traffic is best-effort, it does not matter which packets have been received, only the receiving rate.

The UDP-MIMD and UDP-AIMD protocols drop packets in the transport layer. However, this is not the only choice; we use this method only to verify that congestion control is essential to mobile ad hoc networks. UDP-MIMD and UDP-AIMD can be easily modified to offer an API to the application layer so that the application can figure out how to scale back its sending rate. For example, UDP-MIMD or UDP-AIMD could send the $SendRate$ information directly to the application.

4.4 Simulations

To determine the effects of congestion in MANETs, we ran several simulations using the ns-2 simulator [77] to experiment with UDP-AIMD and UDP-MIMD in static and mobile networks. We use the IEEE 802.11 MAC protocol in RTS/CTS/Data/ACK mode

with a channel data rate of 2 Mbps. The packet size used in our simulations is 1,500 bytes. All of the simulations are run for 100 seconds, and each plot represents the average of 5 independent runs. To the best of our knowledge, there is no proposed congestion control protocol used for UDP. Thus we compare UDP-AIMD and UDP-MIMD with conventional UDP. Here we are not arguing that our protocol is the best, but arguing the importance of using congestion control in UDP, especially in MANETs.

4.4.1 UDP-AIMD

First we examine UDP-AIMD using a simple six node chain topology, shown in Figure 4.1. Each node is in the transmission range of its neighbors, and the interference range is two-hop distance. We vary the source feeding rate from 0.2 Mbps to 2 Mbps and measure the received throughput and energy used for transmitting every packet. The simulation results are shown in Figures 4.6 and 4.7.

From Figure 4.6 we can see that no matter how the parameters are set, using congestion control can achieve much better performance than without congestion control in both end-to-end throughput and energy saving. Also, it seems that there is an "optimum" additive increase rate: 12Kbps or 24Kbps. Apparently, the optimum additive increase rate in the chain cannot apply for all the scenarios, since the gap between the receiving rate when congestion happens and the highest congestion-free receiving rate are different. Therefore, the difference between the optimum congestion-free sending rate and the congested sending rate determines the optimum additive increase rate. However in most situations this difference is unknown. Thus it is hard to do performance optimization. Table 4.1 shows the total capacity that can be achieved for a topology composed of 50 nodes and 5 source-destination pairs. We can see the optimum additive increase rate is 9Kbps.

Table 4.1: The capacity difference with varying the increase rate.

$IncRate(Kbps)$	1	3	5	7	9	12	18	24
$Capacity(Mbps)$	1.52	2.01	1.90	2.06	2.12	1.87	2.07	1.85

4.4.2 UDP-MIMD

To determine the effect of the particular congestion control algorithm we choose, we ran tests varying the UDP-MIMD parameters and measured the performance for the six node chain topology. We fix five parameters used in the protocol and vary the final parameter to see the effects on the end-to-end throughput. In Figures 4.8, 4.12, and 4.13 we can see that choosing β, θ, and ζ in certain reasonable ranges gives almost identical performances. As the $ReceiveRate$ will almost never be greater than the $SendRate$[2], we would not expect that parameters ζ and θ have much effect on the performance.

Choosing different γ (Figure 4.9), ϵ (Figure 4.10), and η (Figure 4.11) values gives some performance differences. γ value indicates when it is regarded as congestion. This value should be close to 1. However, it will be very reasonable to choose 1.3 as the upper bound value. η value decides the increase ratio, since it is used to fine adjustment. Therefore, intuitively we choose 1.3 as the upper bound value. As ϵ, the large scale decrease chooses 0.5, therefore the small scale decrease should be bigger than 0.5. So we choose 0.7 as the lower bound. From Figures 4.9, 4.10 and 4.11, we can find that the differences are not significant. We believe that the different scenarios and the traffic status in the network will require different "optimal" parameters. Therefore, since the performance of UDP-MIMD is not greatly affected by the choice of parameters, it is not necessary to find the "optimal" parameters for different scenarios and traffic patterns, as long as the parameters are chosen in a reasonable range.

Due to the fact that UDP-MIMD does not heavily depend on the choice of parameters comparing with UDP-AIMD, we use UDP-MIMD for remaining simulations.

4.4.3 Large Scale Random Static Topology

The previous results show that UDP-MIMD can greatly improve the end-to-end throughput of ad hoc networks in simple static scenarios. Here, we investigate UDP-MIMD performance in more realistic random scenarios. For these simulations, there are 50 nodes located randomly in a 1000 meters $\times$ 1000 meters square. Five nodes are ran-

[2] $ReceiveRate$ will only be greater than $SendRate$ due to the delay between the measurement of these two parameters and packets being buffered at intermediate nodes.

domly chosen as the sources, and five nodes are randomly chosen as the destinations. The metrics used in measuring the improvement are capacity and per packet energy consumption. The reason for using capacity instead of end-to-end throughput is that under several flow interaction conditions, there is a max/min flow problem, so how the algorithm works to avoid congestion is not clear if based on the end-to-end flow throughput calculation. Figure 4.14 shows the capacity and energy results obtained by averaging 20 different scenarios. The average capacity improvement is 25% and the average energy improvement is 47%.

4.4.4 Mobile Topology

Our protocol is a feedback based congstion control transport layer protocol. Therefore, if the network changes too fast, the feedback cannot reflect the current network status. Thus we do not expect that UDPC will work well in high speed conditions. Normally an ad hoc network is used under pedestrian speeds, thus we choose low mobility senario to test our protocol. In the scenario we choose, each node moves towards a random destination using a speed randomly chosen between 0-3 m/s. Five source-destination pairs send packets using a rate between 0.2 Mbps and 2 Mbps. Figure 4.15 shows the result obtained by averaging 10 different scenarios. Average capacity improvement is 13%, and the energy per packet improvement is 21%.

4.5 Conclusions and Future Work

This chapter shows that adding congestion control to best-effort traffic delivery greatly improves overall network capacity and reduces average energy/packet/hop in mobile ad hoc networks. We explored the use of UDP-AIMD and UDP-MIMD, two congestion control protocols built on top of UDP to reduce congestion in the network. UDP with congestion control improves performance in different types of topologies.

UDP-AIMD and UDP-MIMD only reduce the unnecessary bandwidth cost on a per-flow basis. Furthermore, they do not explicitly take fairness into consideration. Therefore UDP-AIMD and UDP-MIMD do not incorporate any mechanisms for network-wide optimization, nor do they incorporate any way to tradeoff total network throughput for per-flow fairness. Network-wide design using a cost function that combines

both overall throughput and per-flow fairness will be a future research topic.

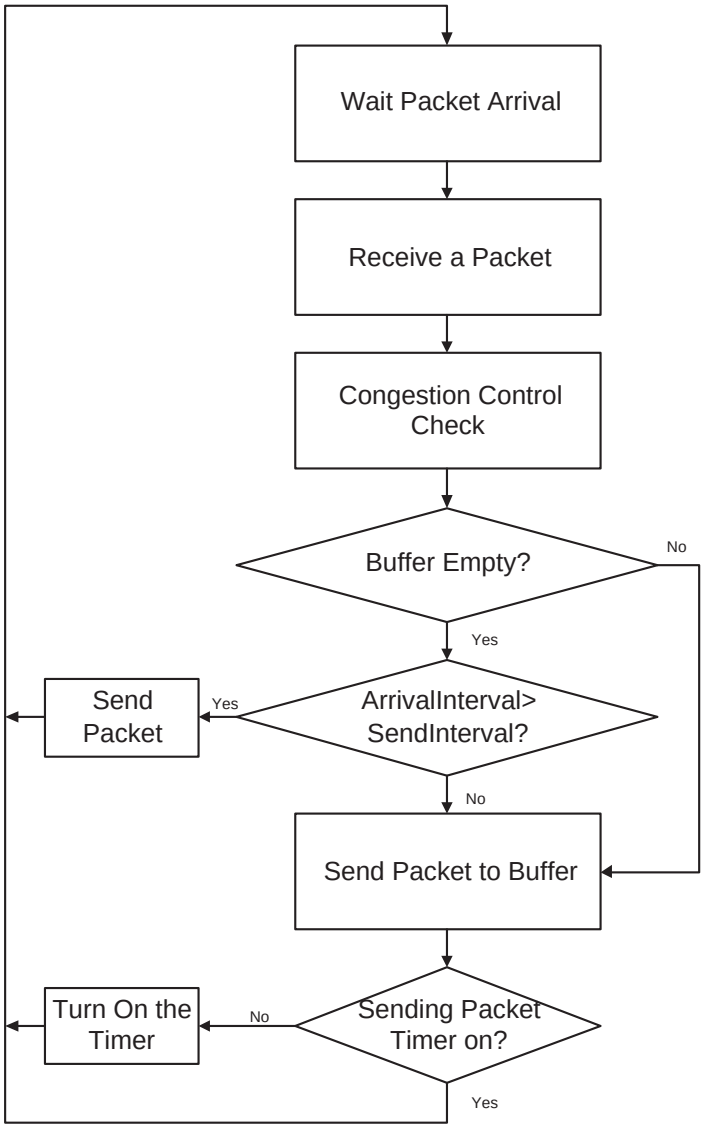

Figure 4.5: Procedure for determining when to send a packet at the source.

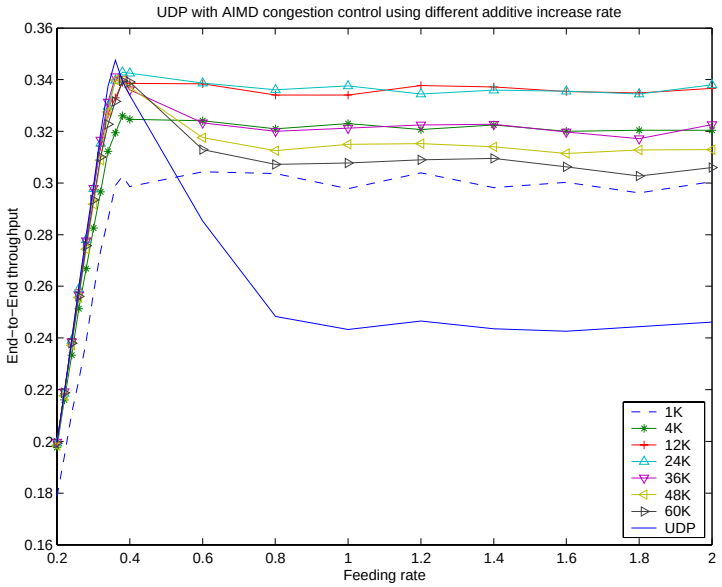

Figure 4.6: The end-to-end throughput of the six node chain using AIMD congestion control with different additive increase rates.

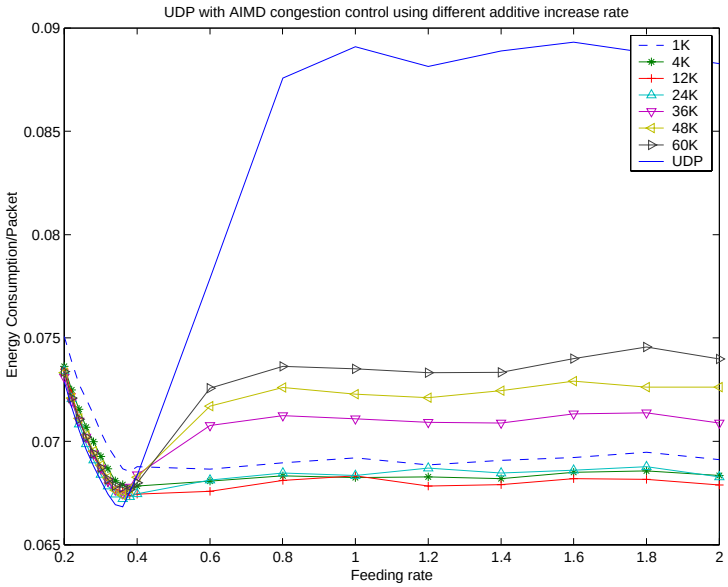

Figure 4.7: Energy consumption of transmitting each packet of the six node chain using AIMD congestion control with different additive increase rates.

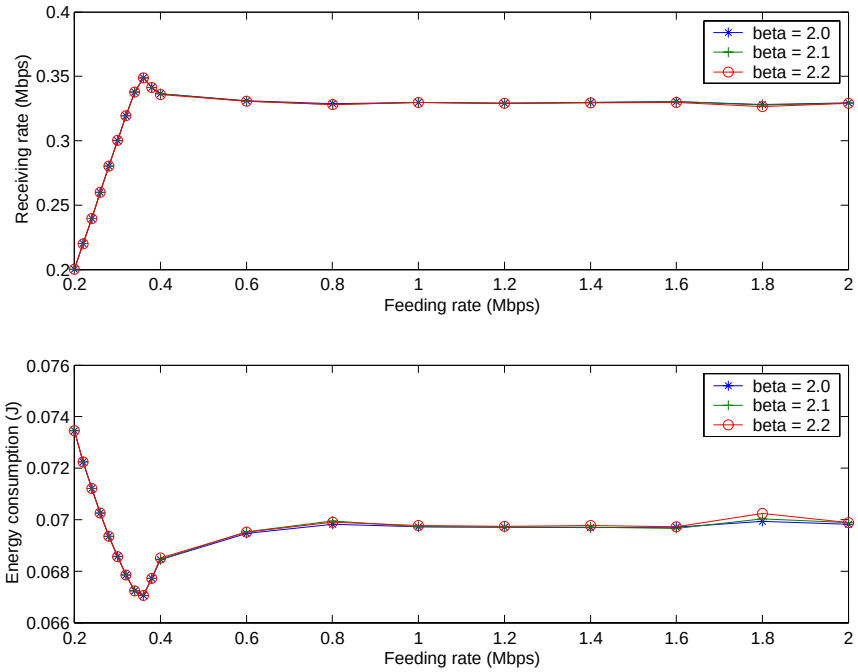

Figure 4.8: Performance as β is varied.

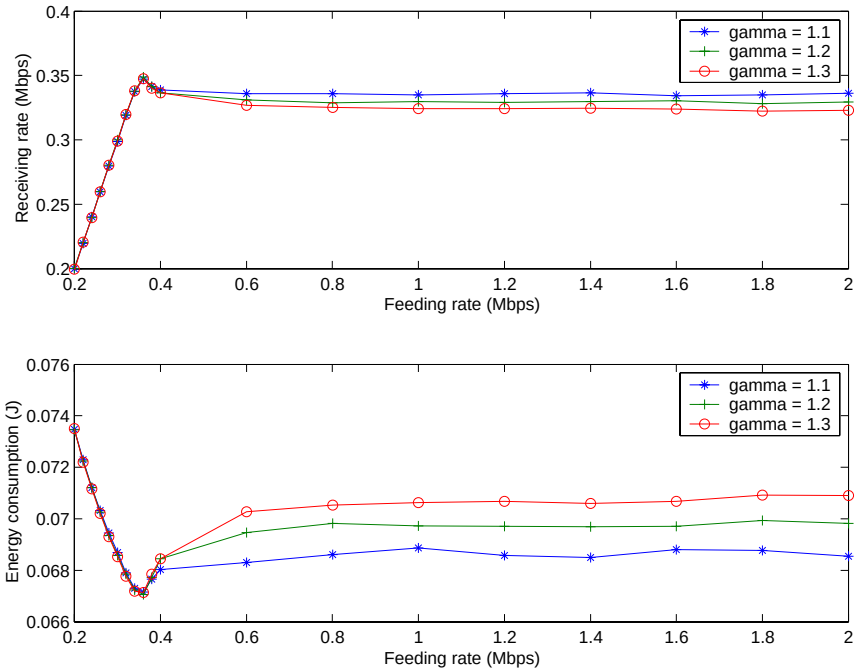

Figure 4.9: Performance as γ is varied.

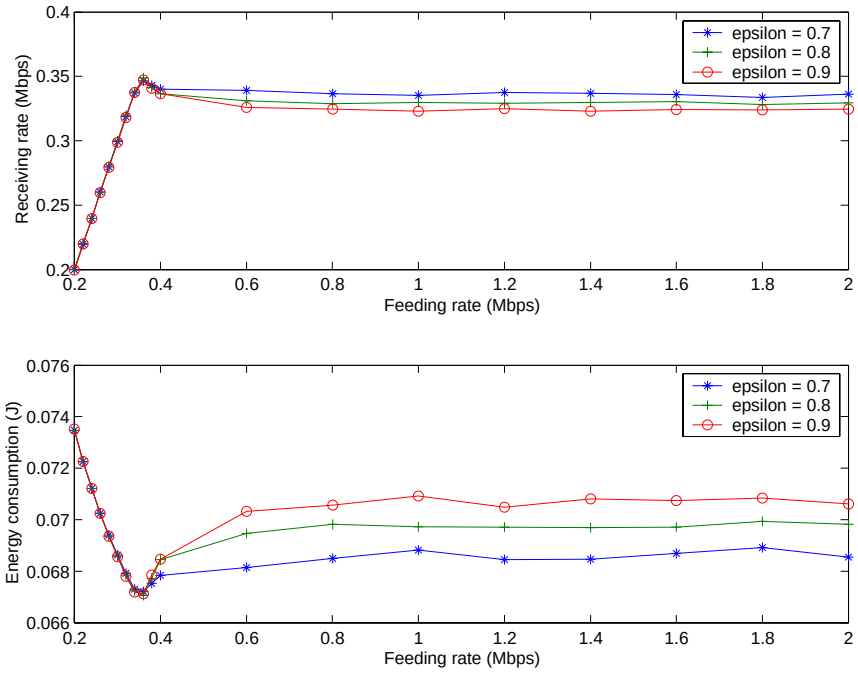

Figure 4.10: Performance as ϵ is varied.

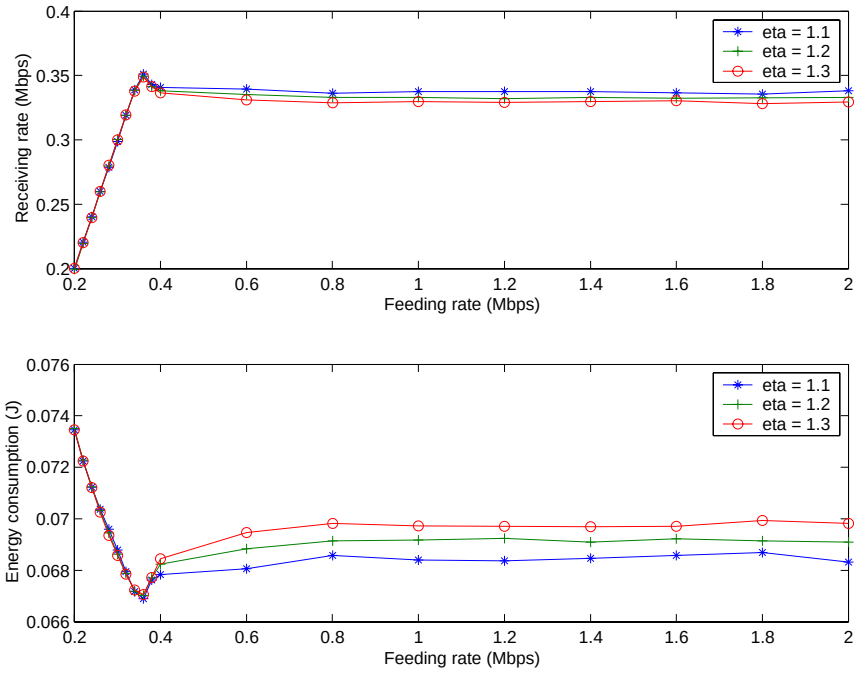

Figure 4.11: Performance as η is varied.

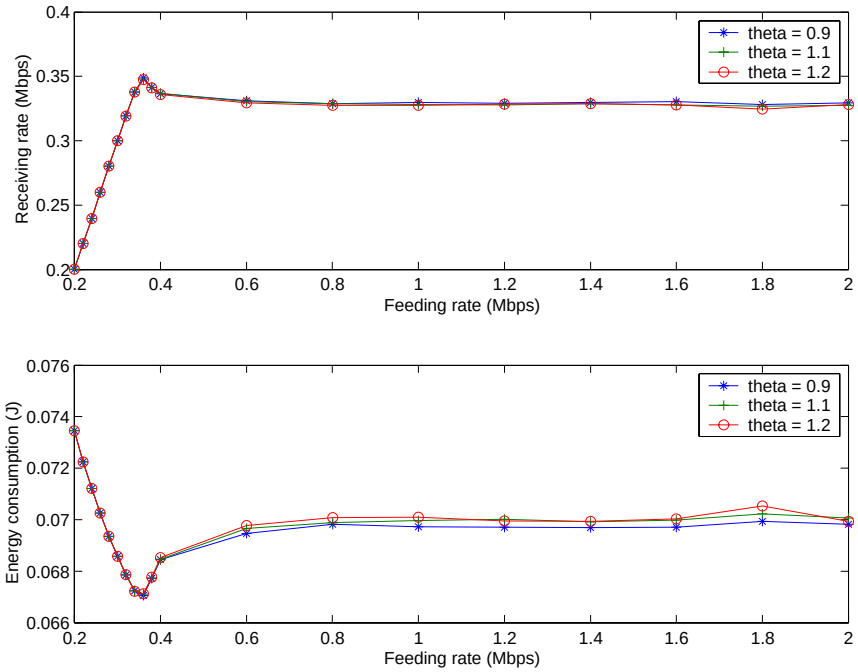

Figure 4.12: Performance as θ is varied.

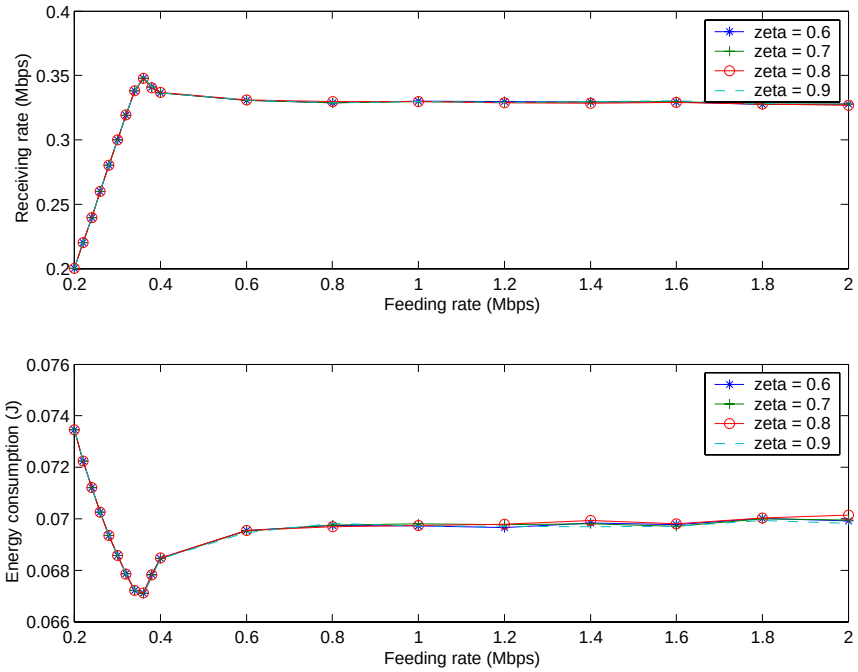

Figure 4.13: Performance as ζ is varied.

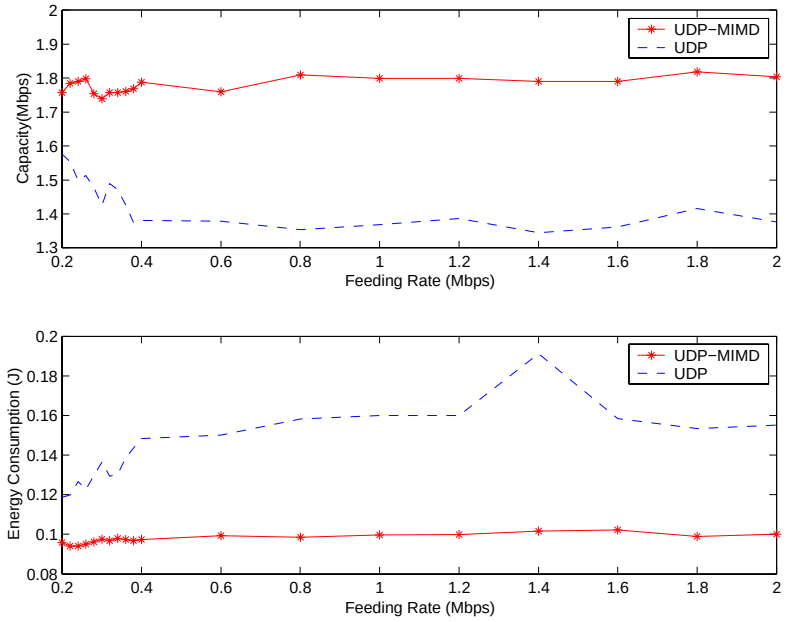

Figure 4.14: The capacity and energy improvement averaged over 20 random static scenarios.

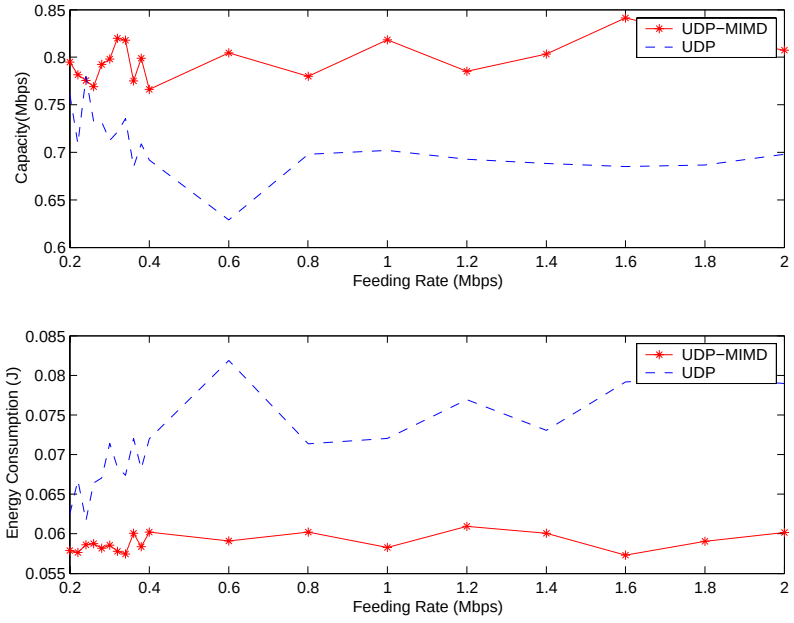

Figure 4.15: The capacity and energy improvement averaged over 10 random mobile scenarios.

Chapter 5

QoS-aware Routing Protocol Survey

In this chapter, we discuss several different approaches to providing QoS at the routing layer, and we provide a comparison of these techniques and a look at some open issues to supporting QoS at the routing layer.

Due to the characteristics of MANETs discussed in chapter 2, QoS guarantees are not possible in MANETs, and soft QoS and QoS adaptation are proposed instead. Soft QoS implies that failure to meet QoS is allowed, for example when paths break or the network becomes partitioned [24]. However, if a network changes too fast to propagate the topology status information, it is impossible to offer even soft QoS. Therefore, combinatorial stability[1] must be met in order to provide QoS.

Most real-time applications can optimize their performance based on feedback about network resource availability. For example, layered coding allows enhanced layers of different quality levels to be transmitted, provided a minimum bandwidth is guaranteed for transmitting the base layer. Therefore, these types of applications can benefit from QoS adaptation. By providing feedback to the application about available resources, the application can alter its coding strategy to provide the best quality for the current resource limitations.

Routing is used to set up and maintain paths between nodes to support data transmission. Early MANET routing protocols, such as AODV [79], DSR [80], TORA [81], and DSDV [82] focused on finding a feasible route from a source to a destination,

[1]Combinatorial stability means that, given a specific time window, topology changes occur sufficiently slowly to allow successful propagation of all topology updates as necessary [78].

without considering any optimization for utilizing the network resources or supporting specific application requirements. To support QoS, the essential problem is to find a route with sufficient available resources to meet the QoS constraints, and possibly add some additional optimizations such as finding the lowest cost or most stable of the routes that meet the QoS constraints. Given these goals, the following are the basic design considerations for a QoS-aware routing protocol.

- Bandwidth Estimation: To offer a bandwidth-guaranteed route, the key idea is to obtain information about the available bandwidth from lower layers. This bandwidth information helps in performing call admission and QoS adaptation. In MANETs, hosts share the bandwidth with their neighbor hosts, and thus the bandwidth available to a node is a dynamic value that is affected by its neighbors' traffic. Therefore, the two key problems in bandwidth estimation are how exactly to estimate the available bandwidth and how frequently to do the estimations. Also, the trade-off between the benefit from using bandwidth estimation and the cost in terms of packet overhead and computing resources used for bandwidth estimation is another key issue.

- Route discovery: There are two main approaches to routing in MANETS: reactive routing and proactive routing. Reactive routing reduces overhead at the expense of delay in finding a suitable route, whereas the reverse is true for proactive routing. For QoS-aware routing, another issue is determining what combination of reduced latency and reduced overhead is best for supporting QoS.

- Resource reservation: The bandwidth resources are shared by neighboring hosts in MANETs. Therefore, another challenging issue is how to allocate this shared resource and what type of resource reservation scheme should be used for setting up and maintaining the QoS-aware route.

- Route maintenance: The mobility of nodes in MANETs causes frequent topology changes in the network, making it difficult to meet the QoS constraints. Incorporating a fast route maintenance scheme into QoS-aware routing is the fourth design consideration. The typical approach to route maintenance, which entails waiting for the host to discover a route break, significantly affects the routing per-

formance. Therefore, some prediction scheme or redundant routing is necessary to assist in route maintenance.

- Route selection: QoS-aware routing has more stringent requirements on route stability, since frequent route failures will adversely affect the end-to-end QoS. Thus, in some sense the path with the largest available bandwidth is not the only consideration-path reliability should also be considered when selecting a suitable path for a QoS-aware routing protocol.

Several routing protocols [23] [24] [25] [26] [27] [28] [29] [30] [54] [55] [71] have been developed that support QoS by choosing routes with the largest available bandwidth, providing a call admission feature to deny route requests if there is not enough bandwidth available to support the request, or providing feedback to the application about available bandwidth resources. These protocols address all of the issues described above.

5.1 QoS-aware Routing Protocols

5.1.1 Core-Extraction Distributed Ad Hoc Routing (CEDAR)

CEDAR [23] is a routing protocol that dynamically establishes a core set for route set-up, QoS provisioning, routing data, and route maintenance. A core is an approximation of a minimum dominating set, whereby all hosts in the network are either members of the core or one-hop neighbors of core hosts. CEDAR assumes that the MAC/link layer can estimate the available link bandwidth of each core host, and every core's available bandwidth information is disseminated to all other cores. CEDAR uses this core structure to reduce routing overhead, as only core nodes must keep track of bandwidth information. CEDAR employs increase waves and decrease waves to propagate the QoS state information, and it uses the state information to determine appropriate routes to support QoS.

- **Core Extraction**

  The core structure is used to limit the number of nodes that must participate in the exchange of topology and available bandwidth information. The goal of setting

up the core is to proactively create a core set such that every node is either a core node or a neighbor of a core node.

To generate the core, a greedy algorithm is used to select core nodes to obtain a good approximation of a minimum dominating set. Each core node maintains local topology information and performs route discovery, route maintenance and call admission on behalf of these nodes.

- **Link State Propagation**

  To propagate state information (available bandwidth) among the core nodes, *increase waves* and *decrease waves* are used. These waves are generated when a core node's available bandwidth has changed by a certain amount. Therefore, information about small changes in available bandwidth is kept locally, and only relatively stable bandwidth information is propagated among the core hosts.

  Increase waves, which provide information about an increase in a core node's available bandwidth, are propagated periodically, whereas decrease waves, which provide information about a decrease in a core node's available bandwidth, are propagated immediately so that core nodes never overestimate another core node's available bandwidth.

- **Route Computation**

  Route computation includes establishment of the core path from the source to the destination via the core nodes, QoS route computation using local information cached by the core hosts along the core path for call admission, and dynamic re-routing for ongoing connections.

  To establish a route, a source node sends a request to its dominator, the node's selected core host, and the dominator initiates a core broadcast. The core hosts who relay this broadcast attach their ID in the packet. The dominator of the destination will send a *core_path_ack* message to the dominator of the source. The *core_path_ack* indicates a path from the dominator of the source to the dominator of the destination and thus sets up a valid core route from the source to the destination via the core nodes.

Otherwise, if the source dominator has cached a core path to the destination dominator, the source dominator tries to find paths to the furthest core host (say host *T*) in the cached core path that guarantees the required bandwidth, using cached local information. The shortest-widest path is chosen among all the admissible paths using a two-phase Dijikstra's algorithm [83]. Then, host *T* performs the QoS route computation, just as the source dominator would do if it did not have any cached routes. Finally, the concatenation of the partial paths provides a QoS core route from the source dominator to the destination dominator.

Setting up a core path requires one round trip, and the QoS route computation requires another round trip. Therefore, the time required to discover a route is two round-trip times if the core path is not cached in the source dominator. Otherwise, one round trip is required.

Source-initiated route maintenance and dynamic route maintenance initiated in the intermediate nodes are used to deal with route breaks. The former works effectively when a link failure occurs near the source, whereas the latter works effectively when a link failure occurs near the destination.

5.1.2 Ticket-based QoS Routing

Chen and Nahrstedt propose a distributed, ticket-based QoS routing protocol [24] that uses tickets to find delay-constrained or bandwidth-constrained routes. Tickets are distributed during route discovery to provide a means to measure bandwidth/delay and limit the flooding for route request packets.

Two types of tickets are used during route discovery: yellow and green tickets. Yellow tickets are used for finding a feasible route with certain delay/bandwidth constraints. Green tickets are used for determining low cost routes. The number of tickets indicates the number of probes made to find a feasible path. Therefore, when a source node wants to find a QoS-aware path, it first decides the number of tickets it should issue according to the QoS constraint. More tickets are issued by the source host to increase the chance of finding a feasible path if the constraints are strict.

In order to find a delay-constrained path, intermediate hosts forward more yellow tickets to their neighbors that have lower delay links and more green tickets to their

neighbors that have lower cost links. If the delay in a certain intermediate host exceeds the maximum delay allowed, this intermediate host sets the ticket as invalid. The destination chooses the path with the lowest cost among the paths that have valid tickets.

In order to find a bandwidth-constrained path, the intermediate hosts relay the yellow tickets to their neighbors according to their neighbors' residual bandwidth, and they forward the green tickets to their neighbor according to their neighbors' link cost. Thus, neighbors whose bandwidth exceeds the request more get more yellow tickets and neighbors whose cost is lower get more green tickets. If none of the neighbors has sufficient bandwidth, the yellow tickets are marked as invalid. Similar to the delay-constrained path, the destination chooses the lowest-cost feasible path.

This approach incorporates the imprecision of each node's estimate of their neighbors' available resources for delay-aware and bandwidth-aware routing by using an imprecision model. The imprecision model uses a weight function with the variables of old bandwidth/delay state and new bandwidth/delay state to estimate the current bandwidth/delay within some precision tolerance. Furthermore, tickets are forwarded so as to provide multi-path searching for paths that satisfy the QoS constraints, thereby adding redundancy for fault tolerance.

5.1.2.1 OLSR-based QoS Routing

Ge et al. [25] integrated QoS features into the Optimized Link State Routing (OLSR) protocol [55] to find a path with larger bandwidth. This approach does not modify the routing scheme of OLSR, but it chooses the different criteria to set the multipoint relays (MPR) set so as to find a larger bandwidth path.

OLSR is an optimization of the classical link state flooding algorithm. In OLSR, a set of nodes is chosen to form an MPR set such that broadcast packets are forwarded only among the MPR set. In this way, overhead is reduced significantly compared with classical flooding where every node needs to forward broadcast packets. Therefore, how to choose the MPR set is the key point of the OLSR algorithm. In the OLSR IETF draft [55], the one-hop neighbors that cover more two-hop neighbors are elected to the MPR set, in order to minimize the number of MPRs. Using this scheme for MPR election, it is quite possible that the low available bandwidth nodes will be chosen for the MPR set, which causes the routes to go through nodes with low available bandwidth.

Thus, in the OLSR-based QoS routing protocol [25], the MPR selection criteria are modified.

Three approaches have been explored:

1. If there is more than one one-hop neighbor that can reach the same number of two-hop neighbors, the one with the largest bandwidth link to the current node is selected for the MPR set.

2. Select the largest bandwidth one-hop neighbors as the MPR set, until all the two-hop neighbors are covered.

3. Select the MPR set to satisfy the requirement that two-hop neighbors have the optimal bandwidth[2] to the current node. Besides the modification of the MPR selection scheme, the routing table computation changes from finding the shortest path route to finding the maximum bandwidth spanning tree.

Simulations have shown that the modified schemes (2) and (3) work best.

Therefore, the OLSR-based QoS protocol works as follows. When the network is initially created, the MPR set is selected based on the bandwidth resource availability among the hosts. Routing requests are broadcast to the nodes in this MPR set, which guarantees the largest bandwidth path will be chosen.

Another contribution of this algorithm is that bandwidth estimation is incorporated. The available bandwidth is obtained by taking advantage of the carrier-sense capability in the IEEE 802.11 MAC protocol and measuring the percentage of busy time.

Thus, OLSR-based QoS routing incorporates QoS features in MPR selection and route selection, and it also uses the carrier-sense capability of the MAC protocol to estimate available bandwidth. However, this work is based on static scenarios and does not explore the impact of node mobility and bandwidth changes, so route maintenance is not studied.

5.1.3 Ad Hoc QoS On-demand Routing (AQOR)

AQOR [26]is a QoS-aware routing protocol with the following features: (1) available bandwidth estimation and end-to-end delay measurement, (2) bandwidth reservation,

[2]The optimal bandwidth path is the path with the largest bottleneck bandwidth among all possible paths

and (3) adaptive route recovery.

AQOR is an on-demand QoS-aware routing protocol. When a route is needed, the source host initiates a route request, in which the bandwidth and delay requirements are specified. The intermediate hosts check their available bandwidth and perform bandwidth admission hop-by-hop. If the bandwidth at the intermediate host is sufficient to support the request, an entry will be created in the routing table with an expiration time. If the reply packet does not arrive in the allotted time, the entry will be deleted. Using this approach, a reply packet whose delay exceeds the requirement will be deleted immediately in order to reduce overhead.

To estimate available bandwidth for assisting in call admission, each node puts its reserved bandwidth in periodic Hello messages that are sent to their neighbors. AQOR uses the sum of a node's neighbors' traffic as the estimated total traffic affecting the node. Note that this estimated traffic can be larger than the real overall traffic (detailed in [26]). This overestimation imposes a stringent bandwidth admission control threshold. The available bandwidth is thus a lower bound on the real available bandwidth. End-to-end one way downstream delay is approximated by using half the round trip delay. With the knowledge of available bandwidth and end-to-end delay, the smallest delay path with sufficient bandwidth is chosen as the QoS route.

Temporary reservation is used to free the reserved resources efficiently at each node when the existing routes are broken. If a node does not receive data packets in a certain interval, the node immediately invalidates the reservation. This avoids using explicit resource release control packets upon route changes.

The adaptive route recovery procedure includes detection of broken links and triggered route recovery at the destination, which occurs when the destination node detects a QoS violation or a time-out of the destination's resource reservation.

5.1.4 Adaptive QoS Routing Algorithm (ADQR)

Hwang and Varshney proposed an adaptive QoS routing algorithm (ADQR) to find multiple disjoint paths with long lifetimes [27]. ADQR differs from other QoS routing protocols by using signal strength to predict the route breaks and initiate a fast reroute of data.

Three levels of signal strength, Th_1 , Th_2, and S_r ($Th_1 > Th_2 > S_r$), are defined.

S_r is the minimal signal strength to receive a data packet. Three different classes are also defined for nodes, links and routes. If the received signal strength from a neighbor node is higher than Th_1, that neighbor node is in the first node class. If the received signal strength from the neighbor is between Th_1 and Th_2, that neighbor node is in the second node class. If the signal strength is between Th_2 and S_r, that neighbor node is in the third node class. Links between the first node class nodes are in the first link class; links between the second node class nodes are in the second link class; and links between the third node class nodes are in the third link class. Also, three route classes are defined, where the bottleneck link determines the path class.

Each node keeps a neighbor table, which records the node's neighbors and their corresponding cumulative signal strength, defined as:

$$SS_{new-cummulative} = \delta \times SS_{old-cummulative} + (1 - \delta) \times SS_{new-measured} \qquad (5.1)$$

where δ is adjusted according to network conditions and is the current received signal strength. Also, two symbols are used to indicate the relative motion of the two nodes: "+" indicates that the two nodes are moving away from each other; while "-" indicates that the distance between the two nodes is shrinking. Each node also keeps a routing table, of the form $\langle$*source, destination, next_hop, hop_count, available_bw, reserved_bw, active, route_class, first_class_link, second_class_link, third_class_link*$\rangle$.

The source node sends a $Route\_Request$ packet, which carries the information $\langle$*source, destination, request_id, hop_cnt, QoS_metric, route_class, int_nodes, first_class_link, second_class_link, third_class_link*$\rangle$. Intermediate nodes append their own address in the int_nodes field, update the parameters QoS_metric, $route\_class$, and hop_cnt, and forward the $Route\_Request$ to their neighbors. The destination node checks whether this path is disjoint from other paths already found and whether $route\_class$ is anything but "+3" . If the first condition is true and the second is false, the destination node does the same procedure as an intermediate node, creates a $Route\_Reply$ packet, and inserts the route information into its routing table. When an intermediate node receives a $Route\_Reply$ packet, the node inserts the route into its local routing table, if there is no corresponding route entry; or the node updates its routing table, if the route already exists.

When the source node receives multiple routes, the choice of which route to use is based on the $route\_class$ information. The $first\_route\_class$ routes obtain higher

priority than the $second\_route\_class$ and the $third\_route\_class$ routes. Similarly, the $second\_route\_class$ routes obtain higher priority than the $third\_route\_class$ routes. After selecting the desired route(s), bandwidth is reserved by sending a $QoS\_Reserve$ packet from the source to the destination along the selected route(s).

ADQR uses a fast route maintenance scheme, called two-phase monitored rerouting, which is composed of $Pre\_Rerouting$ and Rerouting. The $Pre\_Rerouting$ phase occurs when the route changes from $first\_route\_class$ to $second\_route\_class$, and the Rerouting phase is invoked when the route changes from $second\_route\_class$ to $third\_route\_class$. In $Pre\_Rerouting$, the source node finds alternate paths in advance, before the current path becomes unavailable, and in Rerouting, the source node switches to one of these alternate paths in advance of the current path becoming unavailable.

5.1.5 Trigger-based Distributed QoS Routing (TDR)

TDR is a location-based routing protocol proposed by Ge et al. [28]. This protocol distinguishes itself from other location-based protocols by using a local neighborhood database, an activity-based database, call admission during route discovery, soft reservations, and route break prediction to support QoS.

Every host keeps two databases: a local neighbor database and an activity-based database. Hosts are required to periodically broadcast beacons that carry their location and mobility information. The neighbors that receive these beacons record the power level of the received beacon and the location and mobility information in their local neighbor database. Besides the neighborhood database, every node that participates in a data transmission session keeps an activity-based database. In the activity-based database, *session ID, source ID, destination ID, source location, maximum bandwidth demand, maximum acceptable delay, destination location, next node ID, previous node ID, distance from source and activity flag* are recorded for every session. The activity-based database is refreshed by in-session data packets, which makes this a soft-state [84] database.

When the source node wants to initiates a route discovery, it floods route discovery packets to its neighbors, but to ensure stable routes, only neighbors who receive the packet with power greater than a certain threshold will be considered as possible links

in the route. When the destination location is available in the source cache, selective forwarding based route discovery is used. During the process of forwarding the route discovery packet, intermediate hosts check whether their residual bandwidth is sufficient to meet the request. If not, the intermediate hosts do not forward the route discovery packet. Thus admission control is performed according to the resources available in the network.

The destination node sends back a route acknowledgement when it receives the first discovery packet. Upon receiving this acknowledgement packet, the reserved bandwidth in the databases of all intermediate nodes is updated. The destination also sends its location update via the route acknowledgement packet when there has been an appreciable change in its location (based on the destination's own GPS information).

To predict route breaks, three different receive power levels are defined: $P_{th1} > P_{th2} > P_{cr}$. When the receive power level at a particular link is lower than P_{cr}, the upstream active node initiates a rerouting process, which is called link degradation triggered rerouting. When the power level is between P_{th2} and P_{cr}, the intermediate node sends a rerouting request to the source node. Upon receiving the request, the source initiates a rerouting procedure. When the power level is between P_{th1} and P_{th2}, the intermediate node initiates the rerouting.

5.1.6 TDMA Scheduling Supported QoS Routing

Lin et al. propose a QoS routing protocol derived from DSDV with time division multiple access (TDMA) in [85] (here we call this protocol DSDV/TDMA) and an on-demand QoS routing protocol in [30] (here we call it Reactive/TDMA). These two routing protocols assume that code division multiple access (CDMA) is used in conjunction with TDMA to avoid the hidden terminal problem.

In the DSDV/TDMA routing protocol, the source host sends a reservation packet to the destination. The intermediate hosts, who are chosen to participate in the data forwarding for this flow, are asked to calculate their available bandwidth before forwarding the reservation packet. If the intermediate node's available bandwidth is sufficient to support the request, the corresponding resources are reserved using a slot scheduling scheme . Otherwise, a RESET message is sent back to the source to free the reserved time slots hop-by-hop. Once the reservation packet reaches the destination and passes

the bandwidth check, the destination sends back a REPLY packet along the reserved path set up by the reservation packet. If the REPLY does not go through the hosts that reserved bandwidth for this flow within a certain expiration time, the time slots are freed. After the source host receives the REPLY packet, the path is set up. To enable fast route rerouting in the event of route failure, a standby path is always found in addition to the main path.

The Reactive/TDMA protocol uses the same techniques as the DSDV/TDMA protocol, namely bandwidth calculation and slot assignment. To initiate a route discovery, the source broadcasts a RREQ with fields $\langle$ *packet_type, source_addr, dest_addr, sequence number, route_list, slot_array_list, data, TTL* $\rangle$. The hosts that receive the RREQ append themselves in the $route\_list$, calculate their available bandwidth, and record their available time slots in the $slot\_array\_list$. Once the destination receives a RREQ, it returns a route reply (RREP) to the source. Resources are reserved on a hop-by-hop basis as the RREP packet is sent from the destination to the source. If resource reservation cannot be accomplished due to time slots reserved by other flows, a RESERVE_FAIL packet is sent back to the destination. The destination will restart the reservation by choosing another path. If all the trials fail, a NO_ROUTE packet will be sent to the source. If a route is broken, a ROUTE_BROKEN packet will be sent to both the source and the destination to release the reserved bandwidth.

5.2 Comparison and Open Issues

The previous section highlighted the many and diverse protocols available for providing QoS support at the routing level. The fundamental problems in QoS-aware routing are bandwidth/delay estimation, route discovery, resource reservation and rerouting. We compare all the QoS routing protocols in terms of these functions. A summary of our discussion is shown in Table 5.1.

5.2.1 Bandwidth Estimation

The challenge in wireless ad hoc networks is that neighboring hosts must share the bandwidth, and there is no centralized control for allocating bandwidth among the nodes. Furthermore, intermediate hosts take part in forwarding packets. Therefore,

Table 5.1: Comparison of QoS-aware routing protocols.

Routing Protocol	QoS Metric	Bandwidth Estimation	Route Discovery	Resource Reservation	Route Break Prediction	Redundant Paths
CEDAR	BW	No	Proactive	Yes	No	Yes
Ticket-based	BW, Delay	No	Proactive	No	No	Yes
OLSR-based	BW	No	Proactive	No	No	NO
AQOR	BW, Delay	Yes	Reactive	Yes	No	No
ADQR	BW	No	Reactive	Yes	Yes	Yes
TDR	BW	No	Reactive	Yes	Yes	No
DSDV/TDMA	BW	Yes	Proactive	Yes	No	Yes
Reactive/TDMA	BW	Yes	Reactive	Yes	No	No

the total effective capacity achievable is not only limited by the raw channel capacity, but it is also limited by the interaction and interference among neighboring hosts. Thus, in order to offer bandwidth-guaranteed routing, bandwidth estimation is needed, yet accurately estimating available bandwidth at each host is a challenging problem.

Most QoS-aware routing protocols, such as CEDAR, Ticket-based QoS Routing, ADQR and TDR, assume that the available bandwidth is known. However, some routing protocols try to propose an appropriate way to estimate the available bandwidth, such as OLSR-based QoS routing, AQDR, DSDV/TDMA and Reactive/TDMA. Five different methods have been proposed in these protocols for estimating available bandwidth at the nodes.

1. Exploit the carrier-sense capability of IEEE 802.11 and measure the idle and busy time ratio (used in OLSR-based QoS routing protocol).

2. Add bandwidth consumption information to AODV routing packets (or Hello messages) and exchange this information with neighbor hosts (used in AQOR).

3. Monitor and schedule free time slots using a TDMA scheme (used in DSDV/TDMA and Reactive/TDMA).

4. Broadcast queries with limited hop count to actively contact all neighbors in the carrier-sensing range (used in CACP[3] [60]-Multihop).

5. Take advantage of power control and send queries to cover the carrier-sensing range (used in CACP-Power).

6. Approximate the available bandwidth by using a moving average (used in CACP-CS).

A drawback of AQOR's bandwidth estimation method is that it assumes that the interference range is same as the transmission range, which is not true in general. Thus AQOR's bandwidth estimation method will not correctly incorporate the bandwidth being used by neighbors in the interference range of the node.

[3]CACP stands for Contention-aware Admission Control Protocol, which is a protocol for admission control rather than routing and hence is not discussed in detail here. The reader is referred to [60] for details about this protocol.

The estimated available bandwidth is different than the rates of the supported flows, due to intra-flow contention. To address the intra-flow contention, CACP introduces the contention count to exactly calculate the bandwidth sharing among the hosts in the path. Pre-Reply Probe (PRP) and Route Request Tail (RRT) have been proposed to calculate the contention count in more efficient ways [86].

This brings us to the first open issue in QoS-aware routing: **what is the best way to estimate available bandwidth to maximize accuracy and minimize overhead for bandwidth estimation?** The available bandwidth depends on the MAC scheduling, and several of the bandwidth estimation techniques currently proposed are associated with the underlying MAC protocols. Therefore, bandwidth estimation should be done with the assistance of the MAC protocol. A cross-layer design between the MAC and routing layers is the key to solve this problem.

5.2.2 Delay Estimation

Only two routing protocols incorporate delay estimation: Ticket-based QoS aware routing and AQOR. Ticket based QoS aware routing does not support a specified delay; it only determines the shortest delay route during route discovery. AQOR uses half the round-trip time of the route discovery process as the estimated path delay. These two schemes do not consider that changes in contention levels will impact the end-to-end delay significantly after the flow is started. Also, the effect of intra-flow contention on delay has not been sufficiently studied. Therefore, the second open issue in QoS-aware routing is: **how should end-to-end delay be estimated to support delay-constrained real-time data transmission?**

5.2.3 Route Discovery

Route discovery can be categorized as proactive, reactive and/or location-based. CEDAR, OLSR-based QoS routing and DSDV/TDMA are proactive approaches, while ticket-based QoS routing, AQOR, ADQR, TDR and reactive/TDMA are reactive approaches. TDR also is a location-based routing protocol.

Generally, reactive routing protocols perform better in term of overhead, while proactive routing protocols require less time for route discovery. To provide QoS,

timely information about the networks status and fast rerouting in the event of path breaks are desired. The proactive routing protocols show advantages in minimizing delay for route set-up and maintenance. However, the overhead that proactive routing protocols bring is a problem for bandwidth-constrained MANETs. Therefore, the third open issue is: which class of routing protocols, reactive or proactive, is better for supporting QoS routing to balance overhead and delay? The traditional proactive approach may not properly meet the requirements of a QoS-aware routing protocol due to the large amount of overhead to proactively maintain routes, but protocols such as CEDAR, which provides a core to minimize overhead, might be a good solution. Alternatively, some combination of reactivity and proactivity in the protocol may provide the optimal solution.

Finally, using flooding for route discovery requires a large amount of overhead. Therefore, incorporating limited flooding for route discovery, such as the scheme used in ticket-based QoS routing and OLSR-based QoS routing, can improve the routing performance.

5.2.4 Resource Reservation

One difference between regular routing protocols and QoS-aware routing protocols is that QoS-aware routing requires some form of resource reservation. TDR uses temporary reservation of bandwidth during route discovery and updates the reservation upon receiving a route deactivation packet. Also, the reserved bandwidth is updated in a fixed "soft state" interval. AQOR also uses a temporary reservation mechanism to eliminate the connection tear-down process along the old path when the route is adjusted. One unique feature in signaling of AQOR is that QoS violations are detected at the destination, prompting destination-initiated path recovery. ADQR uses a $QoS\_Reserve$ packet to reserve bandwidth from the source to the destination. The reservation algorithm in DSDV/TDMA and reactive/TDMA utilizes a timeout scheme for each reserved slot to release bandwidth when a route is broken. CEDAR does not explicitly describe the signaling approach used, but it assumes the existence of some instantaneous reservation mechanism. Ticket-based QoS Routing and OLSR-based QoS routing have not incorporated any reservation schemes.

RSVP [43] type signaling, used extensively in wired networks, requires a large

amount of overhead, and thus is not directly suitable for MANETs. An in-band signaling technique for MANETs has been proposed in [34] and shown to work well in MANETs. Thus, the fourth open issue in QoS-aware routing is: **how should in-band signaling be coupled with the routing protocol?** The following ideas have been proposed to minimize overhead exchange for resource reservation:

- "soft state" using the active data transmission to reserve the corresponding bandwidth,
- "temporary reservation" where the bandwidth is only reserved for a certain interval and if no data packet is received for a while, the reservation is automatically released, and
- "destination-initiated recovery" where the destination initiates a routing request procedure when a violation is found.

5.2.5 Rerouting

In MANETs, routes change frequently when topology and traffic patterns change. Therefore, rerouting data when paths break or can no longer support the requested QoS is another challenging problem. Predicting route breaks allows better utilization of bandwidth, since packets are not sent via a route that will be disconnected soon. Another useful technique is to provide redundant paths, which offer instant alternative paths when the primary route is broken. Furthermore, some rerouting optimizations have been studied to provide QoS support during rerouting.

Signal Strength Triggered Reroute To the best of our knowledge, using received signal strength to predict link breaks (and hence route breaks) [87] is the only method proposed to predict route breaks. Both TDR and ADQR define three signal strength threshold levels. These protocols prepare to reroute data when the received signal strength is lower than level one and higher than level two; and the data is rerouted through a new path when the signal strength is lower than level two and higher than level three and the signal strength is showing a decreasing tendency. Therefore, the data is rerouted through a new path that can support the

QoS requirements before the route breaks, reducing the transmission break time and avoiding sending packets to a route that will be broken soon.

Path Redundancy DSDV/TDMA maintains secondary paths to use when the primary path fails. This technique is also employed in ticket-based QoS Routing, where this kind of path redundancy is named as second and third level redundancy. In addition, ticket-based QoS Routing has first level redundancy, in which multiple disjoint routes are used simultaneously.

However, there is a tradeoff between path redundancy and overhead. Therefore, the fifth open issue for QoS-aware routing is: **how should a QoS-aware routing protocol balance path redundancy with overhead?**

Other Proposed Schemes for Rerouting Data CEDAR uses route re-computation at the failure point when a link failure occurs near the destination and route re-computation at the source when a link failure occurs near the source. AQOR uses destination triggered rerouting and neighbor loss detection that can trigger the source to perform rerouting of the data. We can conclude from the discussion above that the sixth open issue in QoS-aware routing is: **how should the prediction of route breaks, path redundancy and rerouting optimization be incorporated into a rerouting scheme?**

5.3 Conclusions

This chapter presented a survey of several unicasting QoS-aware routing protocols for MANETs, including CEDAR, ticket-based QoS routing, OLSR-based QoS Routing, AQOR, ADQR, TDR, DSDV/TDMA and Reactive/TDMA. We compared these routing protocols in terms of their different approaches to bandwidth/delay estimation, route discovery, signaling and rerouting, and we pointed out the open issues that need to be addressed in QoS-aware routing protocols. However, as end-to-end communication is the result of the cooperation of all network layers [88][89][90][91], we believe that a cross-layer design will be the key to providing QoS to applications in MANETs.

Chapter 6

QoS-aware Routing Based on Bandwidth Estimation for Mobile Ad Hoc Networks

Routing protocols for mobile ad hoc networks (MANETs) have been explored extensively in recent years. Much of this work is targeted at finding a feasible route from a source to a destination without considering current network traffic or application requirements. Therefore, the network may easily become overloaded with too much traffic, and the application has no way to improve its performance under a given network traffic condition. While this may be acceptable for data transfer, many real-time applications require QoS support from the network. We believe that such QoS support can be achieved by either finding a route to satisfy the application requirements or offering network feedback to the application when the requirements cannot be met.

In this chapter, we propose a QoS-aware routing based on bandwidth estimation to provide information about the current network status to the application layer. Our proposed QoS-aware routing protocol incorporates an admission control scheme and a feedback scheme to meet the QoS requirements of real-time applications. The novel part of this QoS-aware routing protocol is the use of the approximate bandwidth estimation to react to network traffic. Our approach implements this scheme by using two bandwidth estimation methods to find the residual bandwidth available at each node to support new streams. We simulate our QoS-aware routing protocol for nodes running

the IEEE 802.11 MAC. Results of our experiments show that the packet delivery ratio increases greatly, and packet delay and energy dissipation decrease significantly, while the overall end-to-end throughput is not impacted, compared with routing protocols that do not provide QoS support.

6.1 Introduction

The attractive infrastructure-less nature of mobile ad hoc networks (MANETs) has gained a lot of attention in the research community. With the success of solving the most basic but important problems in all network layers, people realize there is commercial value in MANETs. Most applications that attract interest for use in current wired networks (e.g., video conferencing, on-line live movies, and instant messenger with camera enabled) would attract interest for MANETs as well. However, ad hoc networks present unique advanced challenges, including the design of protocols for mobility management, effective routing, data transport, security, power management, and QoS provisioning. Once these problems are solved, the practical use of MANETs will be realizable. The overall design of a solution for all of these problems is currently too complex. In this chapter, we focus on supporting quality of service (QoS) in the network (routing) layer.

In order to design good routing protocols for supporting QoS in MANETs, it is important to understand the fundamental properties of these networks.

- Dynamicity: Every host can randomly change position. The topology is generally unpredictable, and the network status is imprecise.
- Non-Centralization: There is no centralized control in the network, and thus network resources cannot be assigned in a pre-determined manner.
- Radio properties: The channel is wireless, so it will suffer fading, multi-path effects, time variation, etc.

With these constraints, Hard QoS (e.g., guaranteed constant bit rate and delay) is difficult to achieve. The reasons are as follows.

- To support QoS, in principle, the end host should have precise knowledge of the global status of the network. The dynamic nature of MANETs makes it difficult

for hosts to determine information about their local neighborhood, much less the global status of the network.

- It is hard to establish cooperation between neighboring hosts to determine a transmit schedule for guaranteed packet delivery without centralized control. In MANETs, all hosts share the same physical channel, and each host's transmissions will interfere with neighboring hosts' transmissions. This unpredictability makes it hard to guarantee successful transmissions.

- The wireless channel's main deficiency is its unreliability caused by various reasons such as fading and interference.

Thus, our aim is to develop a routing protocol that provides Soft QoS [24] or better than best-effort service, rather than guaranteed Hard QoS. However, if the topology changes too frequently, the source host cannot detect the network status changes and cannot make the corresponding adjustment to meet the specific QoS requirements, rendering the QoS meaningless. Therefore, combinatorial stability[1] must first be met before we can consider providing QoS to real-time applications. There are many networks that satisfy this requirement. For example, consider a network made up of students in a class; students may join the lecture late, some may leave the classroom, but most stay in the stationary position.

Providing QoS is desirable for many applications, as this allows them to alter what data they transmit. For example, several image compression techniques, such as MPEG-4 [92], H. 263 [93], and multiple description coding [94], are designed to meet various channel conditions. Our QoS-aware routing protocol can provide feedback to the application about the current network state to allow the application to appropriately adjust the amount of compression applied to the video. Without this information, the video may not be compressed enough, causing congestion in the network and a large number of dropped packets, which is much worse than transmitting video using low data rate coding. Some applications require minimum bandwidth support. If the minimum bandwidth cannot be met, all data will be useless. Thus, it is better not to transmit data in this case, because it will just waste network bandwidth and energy. Therefore,

[1] Combinatorial stability means that, given a specific time window, the topology changes occur sufficiently slowly to allow successful propagation of all topology updates as necessary [78].

an admission control scheme is also embedded into our QoS-aware routing protocol to address this issue.

Another challenge of QoS is MAC layer design. We argue that the IEEE 802.11 MAC is not the best MAC for supporting QoS. However, it is widely adopted in the WLAN community, and many devices have been commercialized with IEEE 802.11. Therefore, in our design we choose the IEEE 802.11 standard as the underlying MAC layer. IEEE 802.11 has no support for constant bit rate streams, guaranteed delay, etc. Thus, our intention here is to develop a QoS-aware routing protocol using IEEE 802.11 that provides better than best-effort service for real-time video and audio applications. We will explore the issues involved with providing QoS at the MAC layer in Chapter 7.

6.2 Motivation

Routing protocols have attracted a great deal of attention from the beginning of MANET research until the present time. Early work focused on finding feasible routes without considering energy costs or QoS.

Ad-hoc On-Demand Distance Vector Routing (AODV) is one of the most widely used table-based and reactive routing protocols [18] [79]. In AODV, a source host broadcasts a Route Request (RREQ) packet when it needs a route to a specific host. Each host that receives the RREQ packet checks whether it is the destination; if it is, it sends a Route Reply (RREP) packet, otherwise it rebroadcasts the RREQ packet. Intermediate hosts between the source and the destination create an entry in their routing tables and record the neighbor ID of the host from which the RREQ packet was received. The destination host responds to the first RREQ packet it receives by unicasting a RREP to the neighbor from which it received the RREQ packet. The intermediate hosts forward the RREP packet to the source according to their own routing tables. One unique feature in AODV is that hosts use “Hello” messages to probe their neighbors in order to validate routes. Hosts broadcast “Hello” messages in a reasonable interval. If a host does not receive a “Hello” message from a particular neighbor for a certain period, it will delete this neighbor from its neighbor cache and mark the corresponding routes as invalid.

From the description of AODV, we can see that AODV is designed to find a feasible

route only. Therefore, the established route has no knowledge about the network status. Other standardized routing protocols, such as DSR [80], DSDV [82] and TORA [81], also do not incorporate schemes to detect the network status. Therefore, the established routes using these routing protocols cannot inform the application about the network condition, so the application must send its data using a default feeding rate and cannot take advantage of the adaptation feature in various coding technologies. In addition, without knowing the bottleneck throughput, the source may send much more data than the bottleneck host on the route can accommodate. The overwhelmed host must drop data, which wastes a considerable amount of energy and needlessly consumes bandwidth. Also, much time is used in transmitting these data that will eventually be dropped. Therefore, the data that finally reach the destination have to wait in packet queues for a considerably long time, which results in a significantly increased delay.

Therefore, we propose a QoS-aware routing protocol, which is based on residual bandwidth estimation during route set up. Our QoS-aware routing protocol is built off AODV, in which the routing table is used to forward packets, "Hello" messages are used to detect broken routes and "Error" messages are used to inform upstream hosts about a broken route. We explore two ways to perform bandwidth estimation, and we incorporate both an adaptive feedback-based scheme and an admission control scheme.

6.3 QoS-aware Routing

QoS is an agreement to provide guaranteed services, such as bandwidth, delay, delay jitter and packet delivery rate, to users. Supporting more than one QoS constraint makes the QoS routing problem NP-complete [36]. Therefore, we only consider the bandwidth constraint when studying QoS-aware routing for supporting real-time video or audio transmission. We propose a QoS-aware routing protocol that either provides feedback about the available bandwidth to the application (feedback scheme), or admits a flow with the requested bandwidth (admission scheme). Both the feedback scheme and the admission scheme require knowledge of the end-to-end bandwidth available along the route from the source to the destination. Thus, bandwidth estimation is the key to supporting QoS.

Our work focuses on exploring different ways to estimate the available bandwidth,

incorporating a QoS-aware scheme into the route discovery procedure and providing feedback to the application through a cross-layer design.

6.3.1 Bandwidth Estimation

To offer bandwidth-guaranteed QoS, the available end-to-end bandwidth along a route from the source to the destination must be known. The end-to-end throughput is a concave parameter [88], which is determined by the bottleneck bandwidth of the intermediate hosts in the route. Therefore, estimating the end-to-end throughput can be simplified into finding the minimal residual bandwidth available among the hosts in that route. However, how to calculate the residual bandwidth using the IEEE 802.11 MAC is still a challenging problem, because the bandwidth is shared among neighboring hosts, and an individual host has no knowledge about other neighboring hosts' traffic status. We use two methods for estimating bandwidth. One is for hosts to listen to the channel and estimate the available bandwidth based on the ratio of free and busy times ("Listen" bandwidth estimation). The other is for every host to disseminate information about the bandwidth it is currently using in the "Hello" messages, and for a host to estimate its available bandwidth based on the bandwidth consumption indicated in the "Hello" messages from its two-hop neighbors ("Hello" bandwidth estimation).

6.3.1.1 "Listen" Bandwidth Estimation

To estimate the available bandwidth, intuitively, each host can listen to the channel to track the traffic state and determine how much free bandwidth it has available every second. The IEEE 802.11 MAC utilizes both a physical carrier sense and a virtual carrier sense (via the network allocation vector, NAV), which can be used to determine the free and busy times. The MAC detects that the channel is free when the following three requirements are met:

- NAV's value is less than the current time,
- Receive state is idle, and
- Send state is idle.

The MAC claims that the channel is busy when one of following occurs:

- NAV sets a new value,
- Receive state changes from idle to any other state, or
- Send state changes from idle to any other state.

A host estimates its available bandwidth for new data transmissions as the channel bandwidth times the ratio of free time to overall time, divided by a weight factor. The weight factor is introduced due to the nature of IEEE 802.11. The DIFS, SIFS and backoff scheme represent overhead, which must be accounted for in each data transmission. This overhead makes it impossible in a distributed MAC competition scheme to fully use the available bandwidth for data transmission.

Using the "Listen" method to estimate residual bandwidth is straightforward. However, using this approach, the host cannot release the bandwidth immediately when a route breaks, because it does not know how much bandwidth each node in the broken route consumes. "Listen" only counts the used bandwidth, but does not distinguish the corresponding bandwidth cost for each flow. This greatly affects the accuracy of bandwidth estimation when a route is broken. Therefore, we introduce another approach — "Hello" bandwidth estimation — that is better able to reallocate available bandwidth when routes break.

6.3.1.2 "Hello" Bandwidth Estimation

In the "Hello" bandwidth estimation method, the sender's current bandwidth usage as well as the sender's one-hop neighbors' current bandwidth usage is piggybacked onto the standard "Hello" message. Each host estimates its available bandwidth based on the information provided in the "Hello" messages and knowledge of the frequency reuse pattern. This approach avoids creating extra control messages by using the "Hello" messages to disseminate the bandwidth information.

To know the frequency reuse pattern, we first study the underlying IEEE 802.11 MAC. As defined in the IEEE 802.11 MAC, hosts are allowed to access the wireless channel when the media is free. The media can be free if no hosts are transmitting packets within the interference range. Normally, the interference range is twice the transmission range, based on the settings of the 914MHz Lucent WaveLAN card. Therefore, the frequency can be reused outside of the second neighboring hosts' range. The actual

ID	**Consumed Bandwidth**	**Timestamp**
Neighbor ID 1	Consumed Bandwidth	Timestamp
. . . .	. . . .	. . . .
Neighbor ID n	Consumed Bandwidth	Timestamp

Figure 6.1: Hello Structure. The bold item of the first row is the host's own information. The following rows are the host's neighbors' information.

upper bound of bandwidth in the two-hop circle varies with the topology and the traffic status, but the raw channel bandwidth is the soft upper bound of total bandwidth. We use this soft upper bound bandwidth in the estimation to approximate the bandwidth usage. With the above frequency reuse pattern, we can simplify the bandwidth calculation problem to determining the residual bandwidth within the two-hop neighborhood range. Therefore, each host can approximate its residual bandwidth information based on information from hosts within two-hops (the interference range).

The first neighboring hosts' information can be obtained directly, but there is no way to get the second neighboring hosts'[2] bandwidth information directly. There are several ways to get the second neighboring hosts' information, such as disseminating the host bandwidth information using higher transmission power to reach the two-hop neighborhood, and setting up a separate signaling channel to broadcast the bandwidth information. However, using higher power to disseminate information not only consumes much more power, it also destroys the frequency reuse pattern and causes much more interference. Using a separate channel to disseminate the bandwidth information requires additional control that is a heavy burden for the ad hoc network in terms of bandwidth consumption and hardware support. Therefore, we propose using hop re-

[2]If the hosts are located beyond the transmission range but within the interference range, we call them second neighboring hosts or second neighbors.

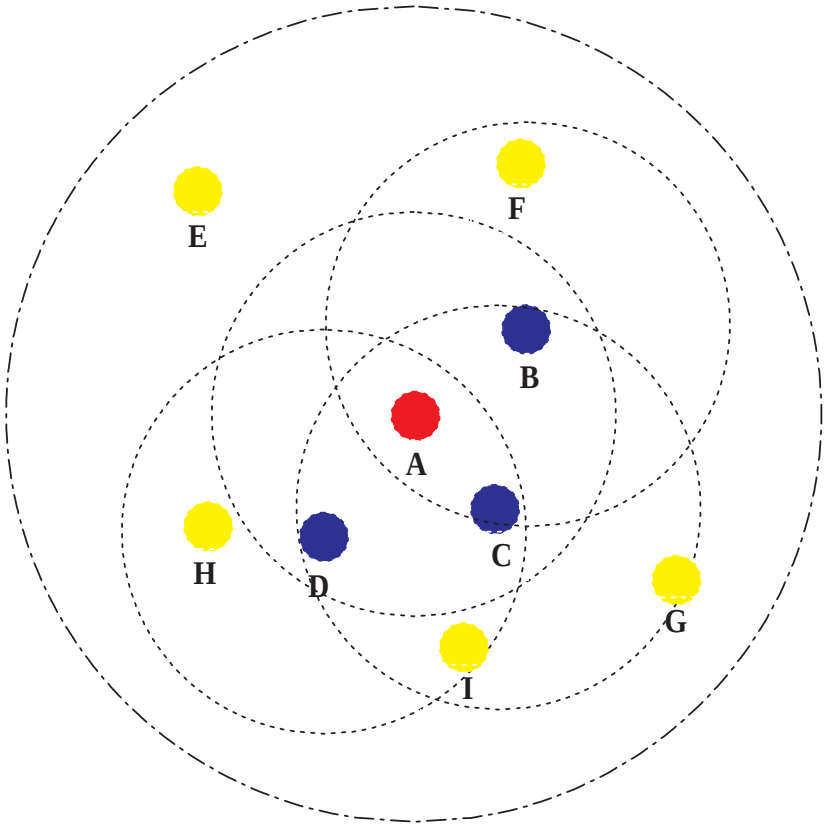

Figure 6.2: Hidden Node Scenario. The big circle indicates host A's interference range. The small circles indicate host A and its first neighboring hosts' transmission range. Hosts B, C and D are A's first neighbors, and hosts F, G, H and I are host A's second neighbors. Host E is in host A's interference range, but it is hidden to A.

lay to disseminate the second neighboring hosts' information. AODV uses the "Hello" messages to update the neighbor caches. The "Hello" message used in AODV only keeps the address of the host who initiates this message. We modify the "Hello" message to include two fields. The first field includes ⟨host address, consumed bandwidth, timestamp⟩, and the second field includes ⟨neighbors' addresses, consumed bandwidth, timestamp⟩, as shown in Figure 6.1. Each host determines its consumed bandwidth by monitoring the packets it feeds into the network. This value is recorded in a bandwidth-consumption register at the host and is updated periodically.

Using this approach to gather the first and second neighboring hosts' information

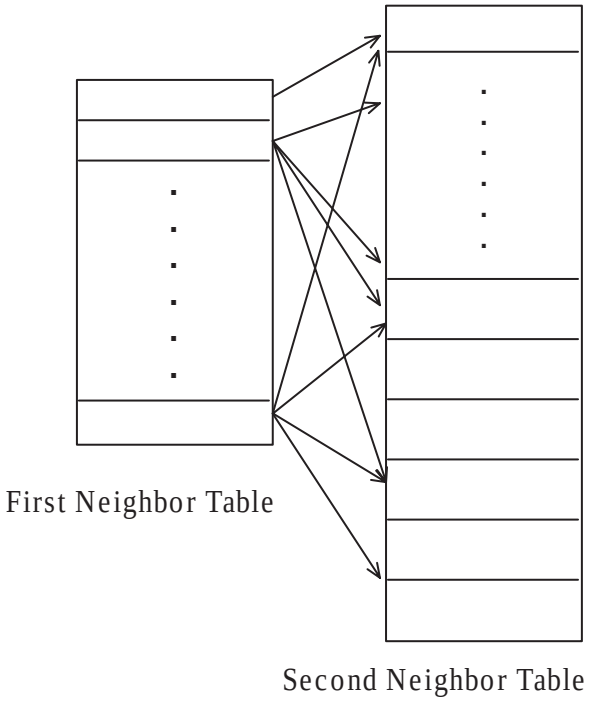

Figure 6.3: Neighbor cache structure.

is imprecise. Figure 6.2 shows an example topology that will result in imprecise information. The outside big circle indicates host A's interference range, and the other small-size dotted circles indicate host A and its neighbors' transmission ranges. Host E is not in A's transmission range, but it is in A's interference range. In addition, E does not fall into any of A's neighbors' transmission range. In this situation, A will never know E's status. If E transmits data, A's knowledge of available bandwidth is imprecise. However, this "Hidden Node" problem does not happen frequently since it has to meet strict requirements to "hide" the host. We argue that this kind of inaccuracy is tolerable because we use a wireless channel, our ultimate aim is better than best-effort, and the possibility of "Hidden Nodes" is low in a well connected network. Even if this situation occurs, it can be overcome by using a conservative bandwidth estimate that leaves some extra bandwidth to conceal this "Hidden Node" effect.

Once a host receives a "Hello" message from its neighbors, it determines whether this "Hello" is an updated one by examining the message's timestamp. We use the cache structure shown in Figure 6.3, which includes a first neighbor table and a second neigh-

bor table. The second neighbors are linked with their corresponding first neighbors in the cache.

Once a host knows the bandwidth consumption of its first neighbors and its second neighbors, the available bandwidth estimation becomes simple. The residual bandwidth is simply the raw channel bandwidth minus the overall consumed bandwidth, divided by a weight factor. We need to divide the residual bandwidth by a weight factor due to the IEEE 802.11 MAC's nature and some overhead required by the routing protocol. In the MAC layer, RTS, CTS, and ACK packets consume bandwidth, the backoff scheme cannot fully use the entire bandwidth, and packets can collide, resulting in packet retransmissions. Furthermore, the routing protocol needs some overhead to maintain or discover the routes.

6.3.2 Incorporating QoS in Route Discovery

As we stated previously, our QoS-aware routing protocol utilizes a cross-layer design. Therefore, the routing features depend on the application requirements. Our design supports two kinds of applications. One is where the application indicates in the request message the minimal bandwidth that must be guaranteed. The other is where the application can adjust its coding rate according to feedback received from the network.

To initiate QoS-aware routing discovery, the source host sends a RREQ packet whose header is changed to ⟨model-flag, bandwidth request, min-bandwidth, AODV RREQ header⟩. The model-flag indicates whether the source is using the admission scheme or the adaptive feedback scheme. When an intermediate host receives the RREQ packet, it first calculates its residual bandwidth. If the model-flag is the admission scheme, the host compares its residual bandwidth with the requested bandwidth. If its residual bandwidth is greater than the requested bandwidth, it forwards this RREQ. Otherwise, it discards this RREQ. If the model-flag is adaptive, the host compares its residual bandwidth with the min-bandwidth field in the RREQ. If its residual bandwidth is greater than the min-bandwidth, it forwards the RREQ. Otherwise, it updates the min-bandwidth value using its residual bandwidth. The whole procedure is shown in Figure 6.4.

When the destination host receives the RREQ packet, it also needs to do the checking procedure as described above. However, after completing this checking procedure,

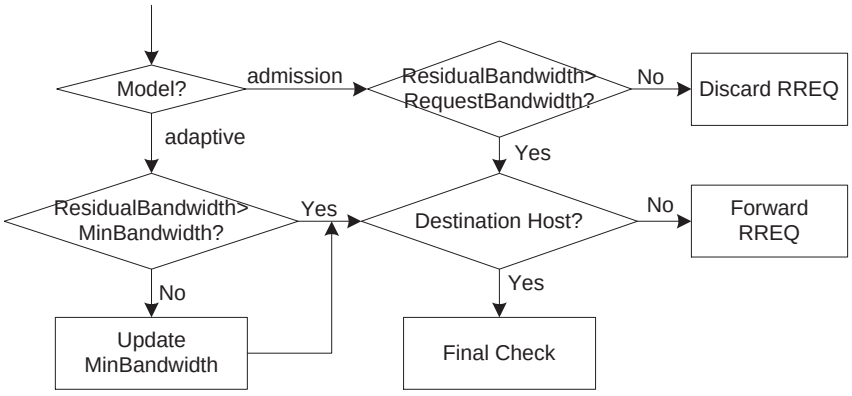

Figure 6.4: Hosts' working procedure after receiving a RREQ.

it is not sufficient to say that the current network can offer the min-bandwidth indicated in the RREQ packet. The reason is that if the route is chosen, the chosen hosts will bring mutual interference into the network during transmission. We cannot put this kind of potential interference into consideration while estimating the residual bandwidth during the route discovery procedure. Therefore, one final check procedure is required before sending the RREP packet back to the source host. We directly use the relation of the end-to-end throughput with the number of hops (*HopNumber*) and the bottleneck bandwidth (*MinBandwidth*) in the route as follows (the details can be found in [72]).

If ($HopNumber$=1)

$$MinBandwidth = MinBandwidth$$

Else if ($HopNumber$=2)

$$MinBandwidth = \frac{MinBandwidth}{2}$$

Else if ($HopNumber$=3)

$$MinBandwidth = \frac{MinBandwidth}{3}$$

Else

$$MinBandwidth = \frac{MinBandwidth}{4}$$

This equation offers the upper bound of the available bandwidth. A more accurate estimation is studied in [60][86], where the inter flow contention is accounted for by using the contention counter.

Finally, the destination host sends the RREP with a modified header ⟨min-bandwidth, AODV RREP header⟩ to the source host. Once intermediate hosts receive the RREP, they enable the route and also record the min-bandwidth in their routing table, which is useful for route maintenance of QoS-aware routing with "Hello" bandwidth estimation.

6.3.3 Route Maintenance

AODV detects a broken route by monitoring the "Hello" messages. If a host does not receive a "Hello" message from a specific neighbor within a pre-defined interval, it marks the routes using that neighbor host as invalid and sends a corresponding "Error" message to the upstream hosts. Only the source host re-initiates a routing discovery procedure, once receiving the "Error" message. Thus, using caches to respond to a route break in the intermediate host is not utilized.

When using QoS-aware routing with "Listen" bandwidth estimation, AODV's route maintenance scheme is used, because releasing bandwidth from the bandwidth consumption registers is impossible without knowing how much bandwidth is consumed by each host in the route. Therefore, no change in AODV's route maintenance scheme is needed to address the bandwidth releasing issue.

However, we cannot directly use AODV's route maintenance scheme in the QoS-aware routing protocol with "Hello" bandwidth estimation. We use the simple topology shown in Figure 6.5 to illustrate what will happen if we adopt AODV's route maintenance scheme without any modification. The topology is a single chain and is composed of 5 hosts. Every host is in its neighbor's transmission range and its second neighbor's interference range. The source host sends packets with a 0.5 Mbps feeding rate[3]. The first table shows the host's first neighbors and the linked tables show the host's second neighbors. If the link between C and D is broken, an "Error" message is initiated in C and A receives it through B's propagation. Once A gets the error message, A sends a new RREQ. The time interval between claiming a broken route and initiating a route discovery is only several milliseconds. Therefore, the host neighbors' caches have not yet updated their bandwidth consumption when the new RREQ arrives. If we do not consider the weight factor, when the new RREQ passes by, host C reports that it has no available bandwidth, since it has not released the bandwidth used by the

[3]Note that for a 2 Mbps channel, 0.5 Mbps is the maximum data rate that can be supported, see [72].

broken route. In fact, all bandwidth is offered to this single chain transmission and the available end-to-end bandwidth is actually 0.5 Mbps. This problem is caused by the fact that the neighbor cache was not updated in a timely fashion. Therefore, we should incorporate a forced cache update in the route maintenance scheme.

The QoS-aware routing with "Hello" bandwidth estimation uses the first neighbors' relay to get the second neighbors' information. Therefore, once the neighbors get the forced updates, they should disseminate the update information immediately to their neighbors. We use an "Immediate Hello" message to address this concern. This special message's content is exactly the same as the "Hello" message, except the packet type is marked as "Immediate Hello" in order to differentiate with the regular "Hello" message. When a host receives an "Immediate Hello" message, it sends its regular "Hello" message immediately.

The "Error" message is also adopted to trigger an update of bandwidth consumption registers and the dissemination of "Immediate Hello" messages. Once a host receives an "Error" message, it will deduct the amount of bandwidth that the broken route consumes from its bandwidth consumption register to reflect the bandwidth allocation changes. We decide to use two separate packets ("Immediate Hello" and "Error"), because the bandwidth should be released among all the neighboring hosts, which a broadcast packet can do, but the "Error" message is a unicast packet. The procedure by which hosts update their neighbor cache is shown step by step in Figures 6.5 – 6.13, in which host A sends data with 0.4 Mbps to host E.

Once host C detects the broken route between C and D, it first brings down the route that is recorded in the routing table, and at the same time it updates its bandwidth consumption register. Then it sends an "Immediate Hello" to its neighbors to inform them of the host's update, as shown in Figure 6.6. B updates its neighbor cache after receiving C's "Immediate Hello", and C's consumed bandwidth changes from 0.4 to 0. Right after sending the "Immediate Hello" message, C creates an "Error" message to inform its upstream hosts that the route between C and D is broken.

Host B sends a "Hello" message, which was triggered by the "Immediate Hello" received from C, to its neighbors A and C. A updates its neighbor cache record about C (from 0.4 to 0), as shown in Figure 6.7. Host B also receives the "Error" message from C; therefore, B marks the corresponding route as invalid, updates its bandwidth

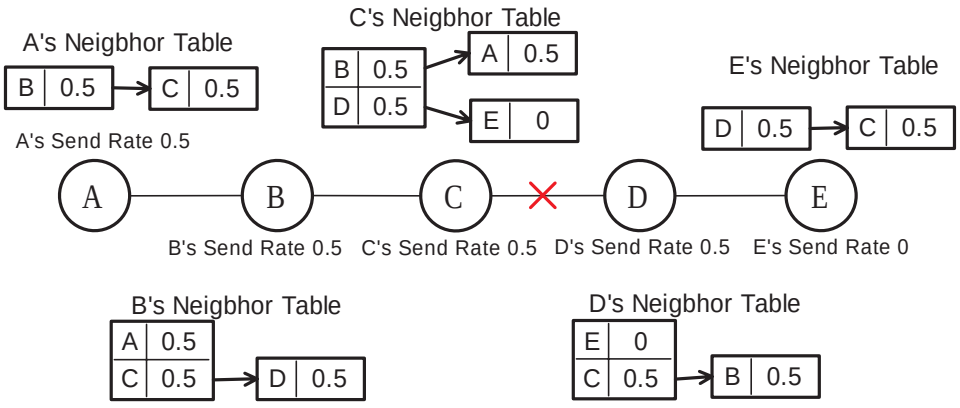

Figure 6.5: Route maintenance failure example.

consumption register (from 0.4 to 0) as shown in Figure 6.8, and sends an “Immediate Hello”. Both A and C change their neighbor caches regarding B’s update, after receiving the “Immediate Hello”. Of course, B sends an “Error” message to A right after the “Immediate Hello”, as shown in Figure 6.9.

Once A gets the “Error” message from B, A tears down the corresponding route in its routing table, updates its record about its own consumed bandwidth, and sends an “Immediate Hello” to B as shown in Figure 6.10. B updates its record about A’s consumed bandwidth in its neighbor cache, then sends the triggered “Hello” as shown in Figure 6.11. C updates it’s neighbor cache item about host A after receiving the “Hello” message from B. Therefore, the bandwidth used by the broken route is released correctly in hosts A, B and C.

The bandwidth releasing in D and E is done during the route discovery procedure. Once C receives the RREQ, it sends an “Immediate Hello” first, then broadcasts the RREQ, as shown in Figure 6.12. Therefore D can update its neighbor cache before receiving the RREQ, and so can host E, as shown in Figure 6.13.

6.4 Simulations and Discussions

To test the performance of our QoS-aware routing protocol, we ran simulations using ns-2. We use the IEEE 802.11 MAC protocol in RTS/CTS/Data/ACK mode with a channel data rate of 2 Mbps. The packet size used in our simulations is 1,500 bytes.

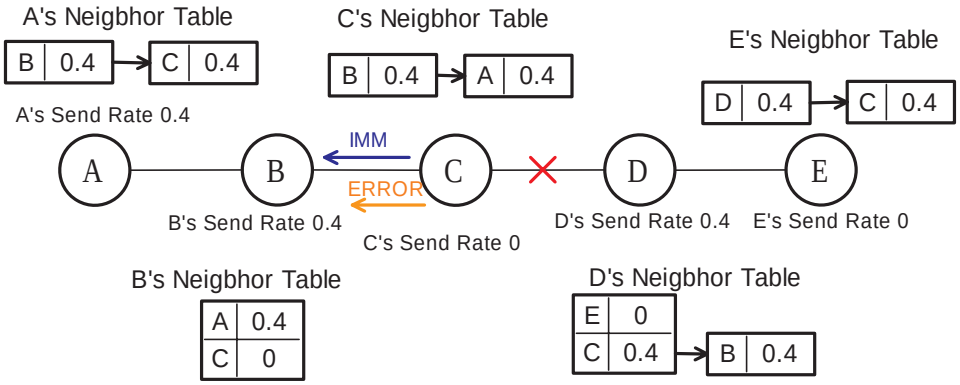

Figure 6.6: QoS-aware routing with “Hello” bandwidth estimation route maintenance procedure 1.

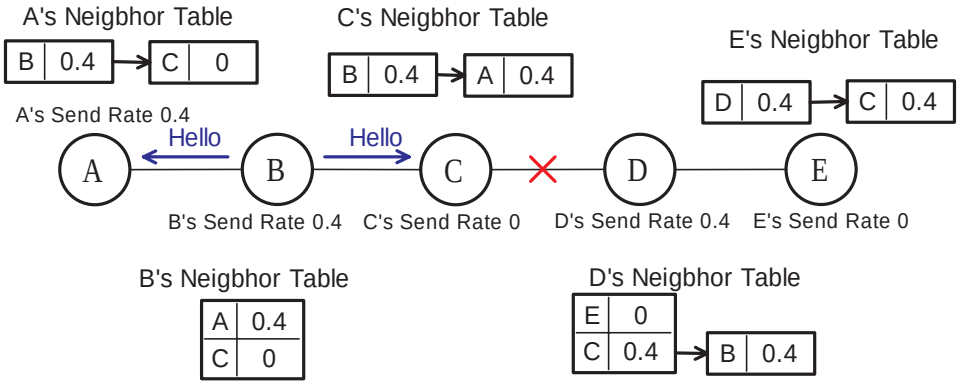

Figure 6.7: QoS-aware routing with “Hello” bandwidth estimation route maintenance procedure 2.

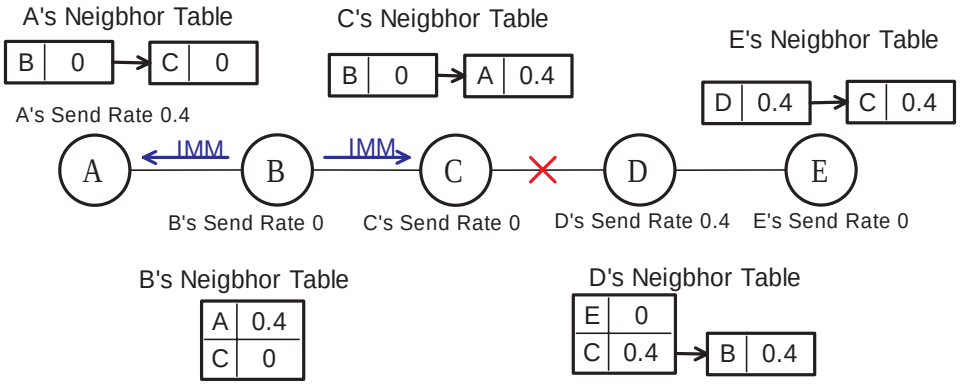

Figure 6.8: QoS-aware routing with “Hello” bandwidth estimation route maintenance procedure 3.

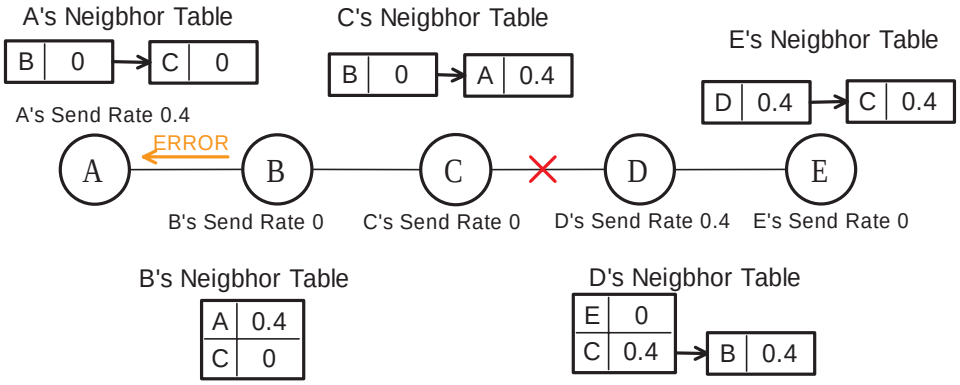

Figure 6.9: QoS-aware routing with “Hello” bandwidth estimation route maintenance procedure 4.

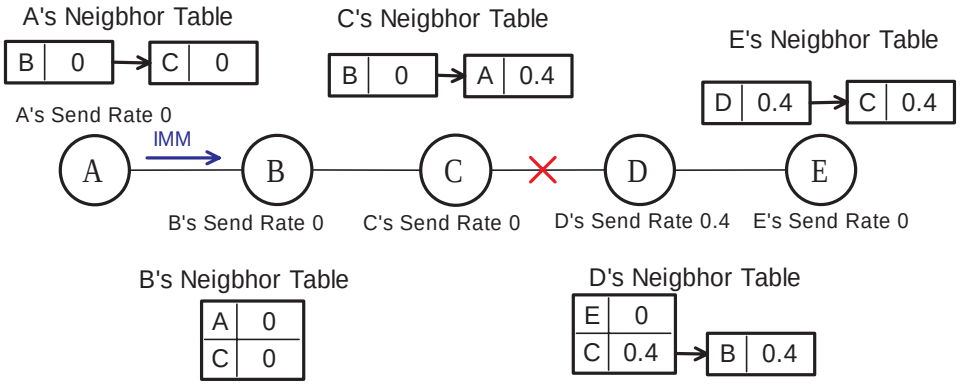

Figure 6.10: QoS-aware routing with “Hello” bandwidth estimation route maintenance procedure 5.

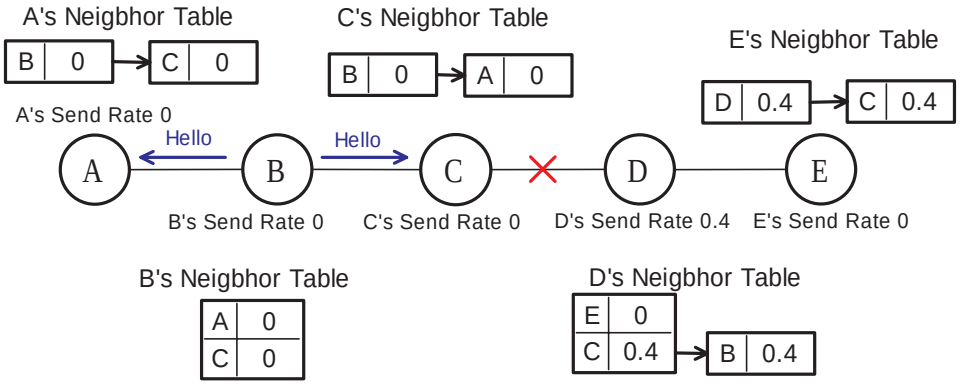

Figure 6.11: QoS-aware routing with “Hello” bandwidth estimation route maintenance procedure 6.

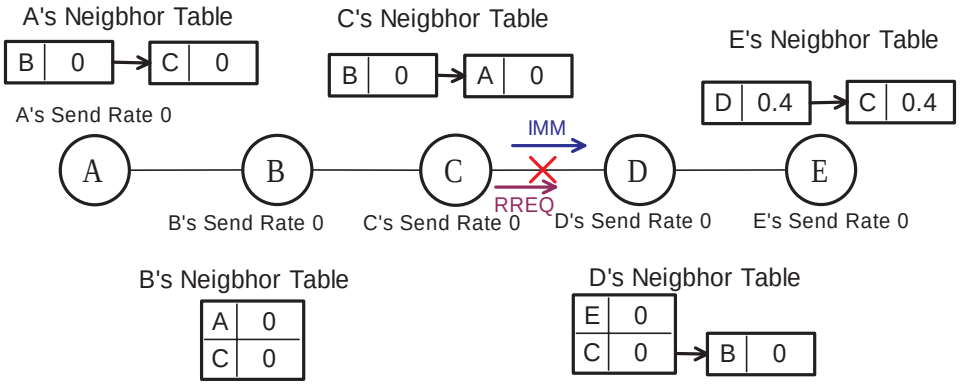

Figure 6.12: QoS-aware routing with “Hello” bandwidth estimation route maintenance procedure 7.

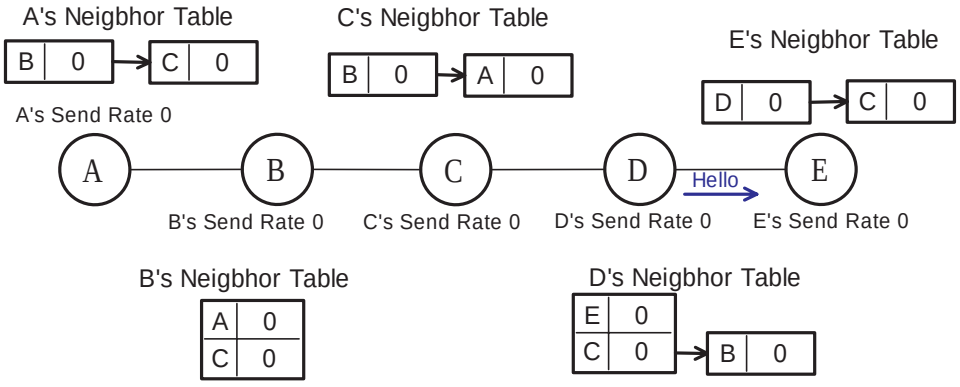

Figure 6.13: QoS-aware routing with “Hello” bandwidth estimation route maintenance procedure 8.

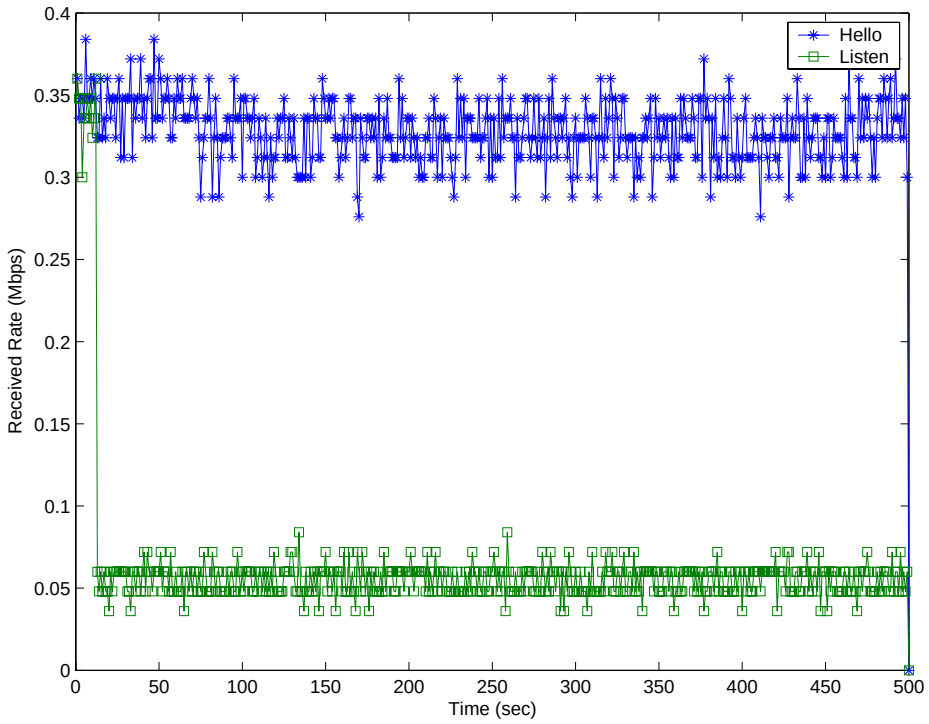

Figure 6.14: The received packet rate using a six-node chain topology with “Listen” bandwidth estimation and “Hello” bandwidth estimation.

The topologies vary according to the different simulation purposes.

6.4.1 “Hello” vs. “Listen” Bandwidth Estimation When Routes Break

A broken route can be caused by two reasons: (1) the hello messages collide several times (in which case the route is not really broken), and (2) a host in that route moves out of its neighbor’s transmission range. We study these two different cases separately.

6.4.1.1 Route break caused by losing “Hello” messages

One flow in a network can be viewed as a single static chain. In order to simplify our analysis, we do the simulations in a chain topology to explain the effects brought by a broken route that is caused by losing broadcasted “Hello” messages. The simulated chain topology is composed of six hosts, where the header host is the source host and

the tail host is the destination host. The source host sends data packets to the destination host using a 0.35 Mbps feeding rate. By studying the trace files, we find that a supposed route break occurs at 13 seconds using the QoS-aware routing protocol with "Listen" bandwidth estimation. Supposed route breaks occur at 27 seconds, 73 seconds, 236 seconds, and 468 seconds using the QoS-aware routing protocol with "Hello" bandwidth estimation. Figure 6.14 shows that using the route maintenance procedure described in section 6.3.3, "Hello" bandwidth estimation can correctly estimate the residual bandwidth after the reported route breaks; however, using "Listen" bandwidth estimation cannot, so the source host is forced to transmit below the channel capacity.

In this case, "Hello" packets are dropped often when traffic becomes heavy. After 3 consecutive "Hello" packets are dropped, a broken route is claimed. However, this route is not physically broken, because these 3 "Hello" messages are dropped by coincidentally colliding with other packets. Therefore, the packets are still successfully transmitted to the destination host during the time between the first "Hello" message being dropped and the third "Hello" message being dropped. The route discovery procedure is initiated right after the source host receives the "Error" message. The time interval between claiming a route break and setting up the route is only several milliseconds. In such a small time interval, it is almost impossible for the hosts to automatically and correctly update their bandwidth registers in the "Listen" bandwidth estimation method, since the consumed bandwidth estimation is based on averaging bandwidth consumption every one second interval and the hosts in the broken route were transmitting data in the previous second. Therefore, the "Listen"-based bandwidth estimation approach has difficulty correctly estimating the residual bandwidth. Even if some forced update schemes can be adopted, the hosts still cannot release the bandwidth correctly, since the hosts do not know how much bandwidth each node in the broken route consumes. In contrast, the "Hello"-based bandwidth estimation approach can easily solve this problem by using the forced update scheme.

6.4.1.2 Route break caused by moving out of a neighbor's transmission range

To simplify the explanation, we use the topology shown in Figure 6.15 to mimic the topology that will cause a route break because of a moving node. The topology is composed of 30 hosts. Host 18 is the destination host, and host 13 is the source host.

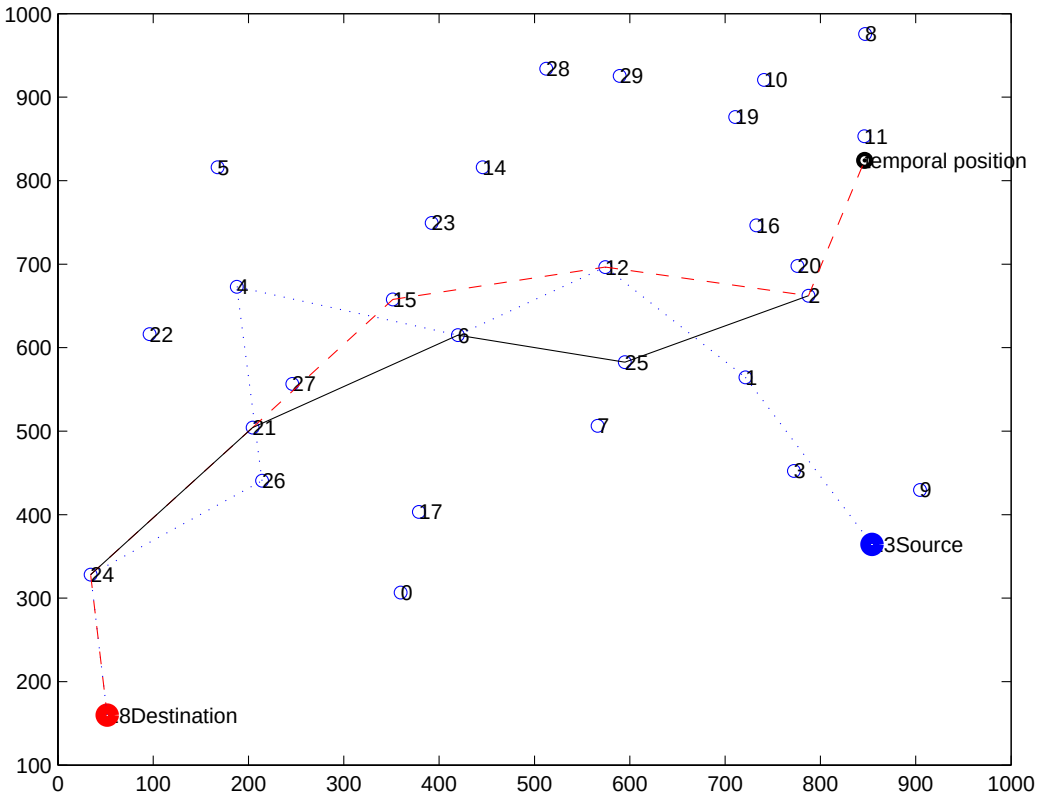

Figure 6.15: The scenario used to simulate a route break caused by a moving node.

Host 13 is moving towards host 11 with a speed of 10 m/s. The source host sends data packets to the destination host using a 0.25 Mbps sending rate. We ran simulations using the QoS-aware routing protocol with "Listen" bandwidth estimation and the QoS-aware routing protocol with "Hello" bandwidth estimation. In the beginning of the simulation, the chosen route goes through hosts 13, 1, 12, 6, 4, 26, 24 and 18 (the dotted line in Figure 6.15). At the simulation time of 43 seconds, host 13 moves to a position (shown in Figure 6.15) that is out of host 1's transmission range. This causes a route break and host 13 must initiate a new discovery procedure. Using the routing protocol based on using "Listen" to estimate residual bandwidth, the new route goes through hosts 13, 2, 12, 15, 21, 24 and 18 (the dashed line in Figure 6.15). Using the routing protocol based on using "Hello" to estimate residual bandwidth, the new route goes through host 13, 2, 25, 6, 21, 24 and 18 (the solid line in Figure 6.15). The simulation results are shown in Figure 6.16. We can see the end to end throughput using "Listen"

to estimate bandwidth is much less than by using “Hello” to estimate bandwidth after the route changes. Studying the trace file, we find the reason for this difference is that there are approximately 3 seconds between host 13 moving out of host 1’s transmission range and the route break being claimed. During these 3 seconds, all hosts correctly update their bandwidth consumption registers except host 2 who is next to the source. This is caused by the fact that the source host keeps on sending RTS packets, so host 2 can hear all these RTS packets and sets its NAV vector according to the packet length that the RTS indicates. Therefore, its estimated free time is significantly less than the real free time. Thus, host 2 cannot offer the correct bandwidth estimation after receiving a “RREQ” packet. However, using “Hello” to estimate residual bandwidth will not be affected by the above reason.

These results show that the “Listen” technique cannot react well to a broken route due to the fact that the MAC’s NAV cannot truly reflect the traffic status, and the bandwidth consumption registers cannot be updated in time. Thus, when routes break, “Hello” bandwidth estimation performs better than “Listen” bandwidth estimation.

6.4.2 Weight Factor Comparison

We cannot compare the performance of “Hello” bandwidth estimation and “Listen” bandwidth estimation using the same weight factor, because these two methods define the consumed bandwidth differently.

- “Listen” mode – accounts for RTS, CTS, ACK, retransmission, routing packets, and transmitted packets.
- “Hello” mode – counts the transmitted packets only.

Therefore, the “Hello” weight factor should be larger than the “Listen” weight factor if we want to get the same performance. In addition, if congestion occurs, the listen mode cannot release the bandwidth immediately, so we should choose a large weight factor to avoid congestion when we compare these two different estimation methods.

If we do not consider the bandwidth used in the retransmission of data, RTS and CTS packets, and the bandwidth used for transmitting the routing packets, the approximate ratio between the weight factor used in “Listen” mode and the weight factor used

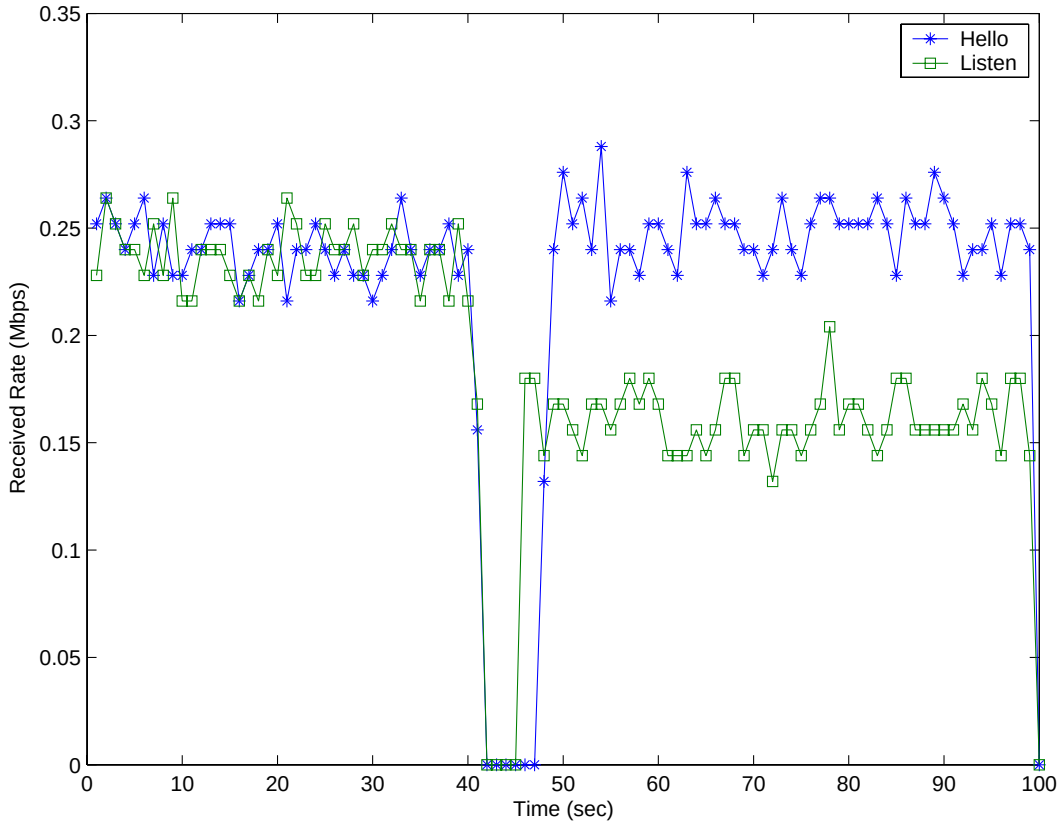

Figure 6.16: The received rate using the source moving topology shown in Figure 6.15 for the "Hello" bandwidth estimation method and the "Listen" bandwidth estimation method.

in "Hello" mode should be as follows:

$$\frac{RTS + CTS + (Data + MACHdr + IPHdr) + ACK}{Data} \tag{6.1}$$

$$= \frac{44 + 38 + (1500 + 52 + 20) + 38}{1500} = 1.128$$

Therefore, if we randomly choose the weight factor of "Listen" mode as 1.25, which is large enough to avoid the route breaks caused by losing "Hello" messages, the weight factor used in "Hello" mode should be larger than $1.25 \times 1.128 = 1.41$.

We investigate the performance of the "Hello" scheme and the "Listen" scheme using topologies where 50 static nodes are located randomly in 1000 meters by 1000 meters. Five nodes are randomly chosen as sources and five nodes are randomly choose

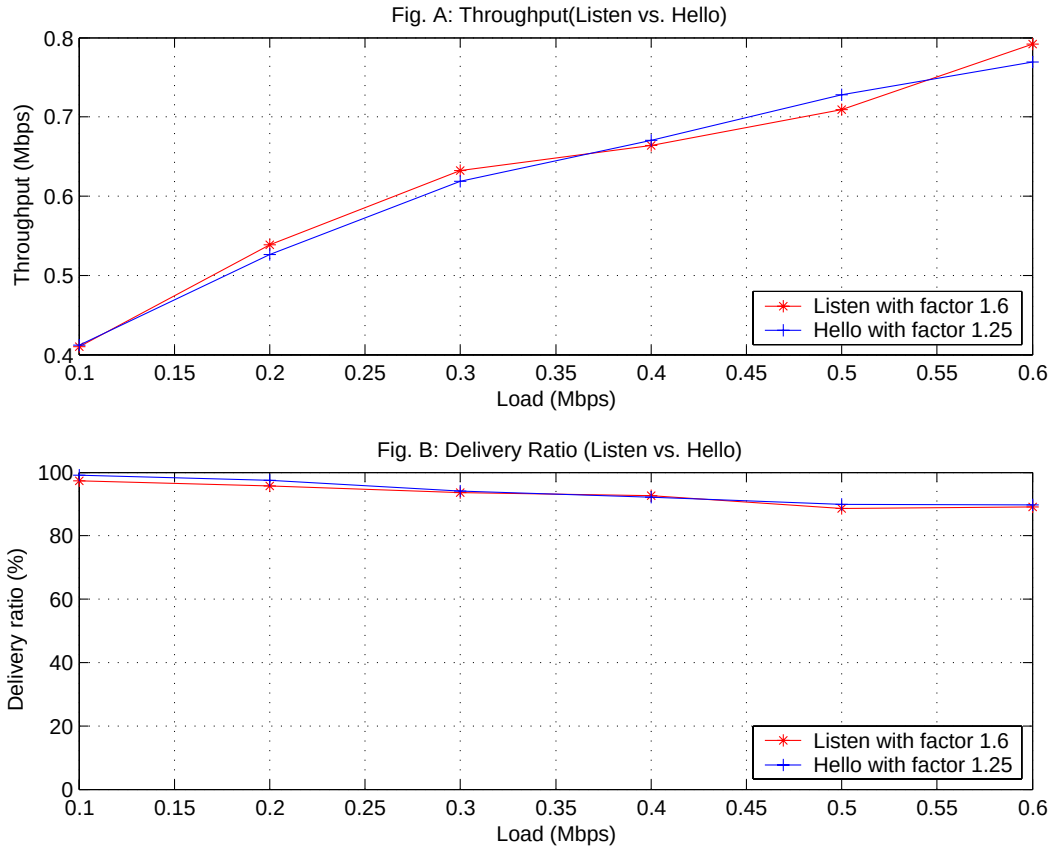

Figure 6.17: Throughput and packet delivery ratio comparison (“Listen” vs. “Hello”).

as destinations. All sources feed the same data rate to their destinations, and the feeding rate varies from 0.1 Mbps to 0.6 Mbps. After every 10 second interval, one source will begin to send data into the network. We randomly choose 20 different scenarios and run the simulation for 500 seconds. The average simulation results are as shown in Figure 6.17. We find that the performance of choosing weight factor 1.6 in “Hello” mode matches well with the performance of choosing weight factor 1.25 in “Listen” mode. Therefore, we deduce that the QoS-aware routing protocols based on “Listen” bandwidth estimation and “Hello” bandwidth estimation work equally well as long as their weight factors are chosen appropriately.

The RTS, CTS and ACK overheads affect differently small size packets and large size packets. Therefore, different weight factors should be used for different packet sizes. In addition, different physical phenomena can bring different fading errors. The

fading errors can cause necessary retransmission of RTS and data packets. Thus, these overheads may change the required weight factor's value. However, here we use the same physical channel for all the simulations.

6.4.3 Static Topology Using the Adaptive Feedback Scheme

For these simulations, we use the same topologies and simulation time as used in section 6.4.2, and we compare QoS-aware routing with "Hello" bandwidth estimation, QoS-aware routing with "Listen" bandwidth estimation, and conventional AODV, which has no QoS support. The metrics used in measuring the protocols' performance are delay, packet delivery ratio, energy consumption per packet per hop and overall end-to-end throughput.

As the number of flows and the number of hosts increases, the negative effects brought by using "Listen" bandwidth estimation under a broken route will not be very significant. In the case that the broken route is caused by losing "Hello" messages, the underestimated bandwidth will be consumed by other flows. Therefore, we expect that both "Listen" and "Hello" bandwidth estimation will work well. Figure 6.18 shows the performance using the QoS-aware routing protocol with "Hello" bandwidth estimation and AODV. Figure 6.18A shows that there is a great improvement in packet delivery ratio (up to 260%) using QoS-aware routing with "Hello" bandwidth estimation compared with AODV.

We also find that the packet delivery ratio increases with increasing weight factor. This is because the available bandwidth allowed to schedule packet transmissions is $\frac{ResidualBandwidth}{WeightFactor}$. The bigger the weight factor is, the more conservative the packet transmission scheduling is. Therefore, there is a tradeoff between bandwidth usage and the packet delivery ratio. However, as bandwidth usage is one of the most important metrics to measure the network performance, we do not want to completely sacrifice bandwidth to get an improvement in packet delivery ratio. Figure 6.18B shows that actually we can get almost equal overall end-to-end throughput for QoS-aware routing with "Hello" bandwidth estimation compared with AODV, and even some improvement in a highly congested network, when choosing a reasonable weight factor.

The packet delivery ratio improvement also brings side benefits such as decreased delay and energy consumption, due to congestion avoidance and the control nature

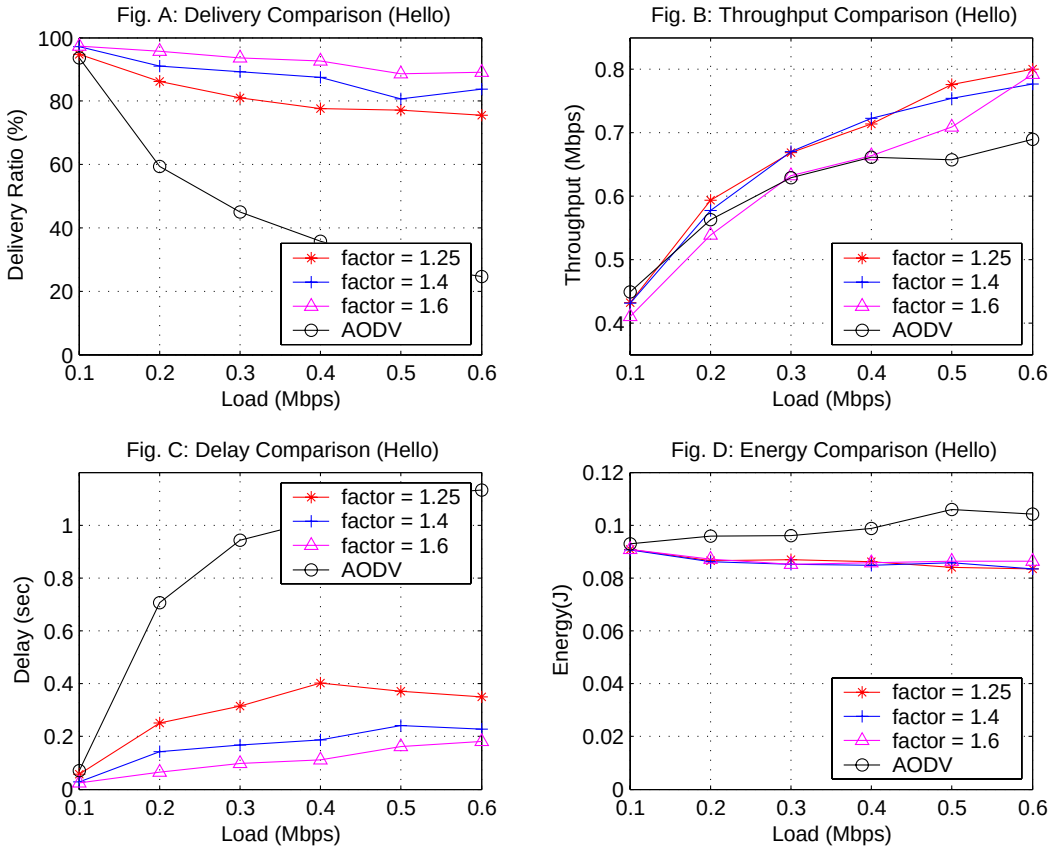

Figure 6.18: Results for QoS-aware routing with "Hello" bandwidth estimation with different weight factors and AODV. Fig. A: Packet delivery ratio. Fig. B: End-to-end throughput. Fig. C: Delay. Fig. D: Energy.

inherited in the QoS-aware routing protocol. The time used waiting in the packet queue and contending for the channel decreases, and the energy used on transmitting packets which will ultimately be dropped is saved. Therefore, delay is decreased up to 795% and energy/packet/hop is decreased up to 29%, as shown in Figures 6.18C and D.

Figure 6.19 shows the performance when the QoS-aware routing protocol with "Listen" bandwidth estimation is used compared with AODV. Figure 6.19A shows the great improvement in packet delivery ratio (up to 280%) using QoS-aware routing with "Listen" bandwidth estimation compared with AODV. However, the end-to-end throughput is decreased by 10% as shown in Figure 6.19B, when the feeding rate is low, even when the weight factor is quite small (e.g., 1.1). Our best guess is that the possibility

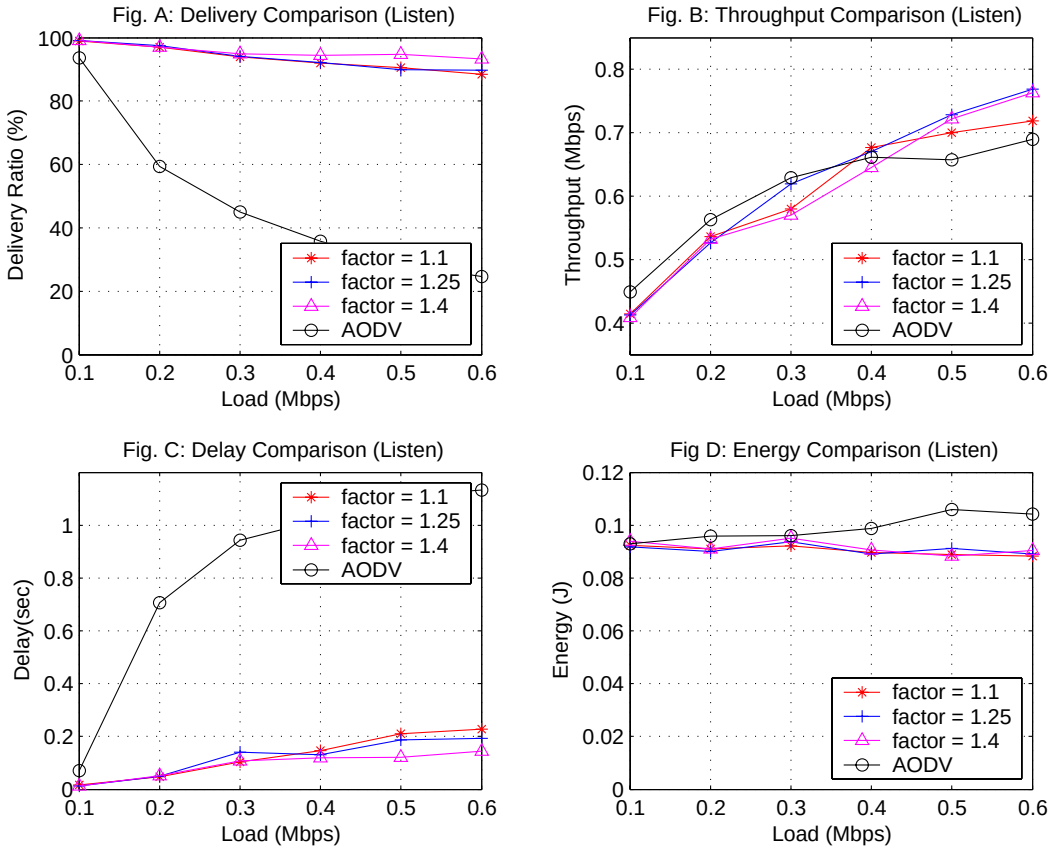

Figure 6.19: Results for QoS-aware routing with "Listen" bandwidth estimation with different weight factors and AODV. Fig. A: Packet delivery ratio. Fig. B: End-to-end throughput. Fig. C: Delay. Fig. D: Energy.

of route breaks caused by losing "Hello" messages is high in some scenarios, when a small weight factor is used. Using the QoS-aware routing protocol with "Listen" bandwidth estimation underestimates the bandwidth after a route break, and the residual bandwidth cannot fully be used by other flows. Therefore, the end-to-end throughput is lower than using AODV. However, the underestimated bandwidth can be used when the load is high; therefore, there is bandwidth improvement when the load is high. There are also some side benefits brought by the improvement of the packet delivery ratio. The delay is decreased up to 800% and energy/packet/hop is decreased up to 22%, as shown in Figures 6.19C and 6.19D.

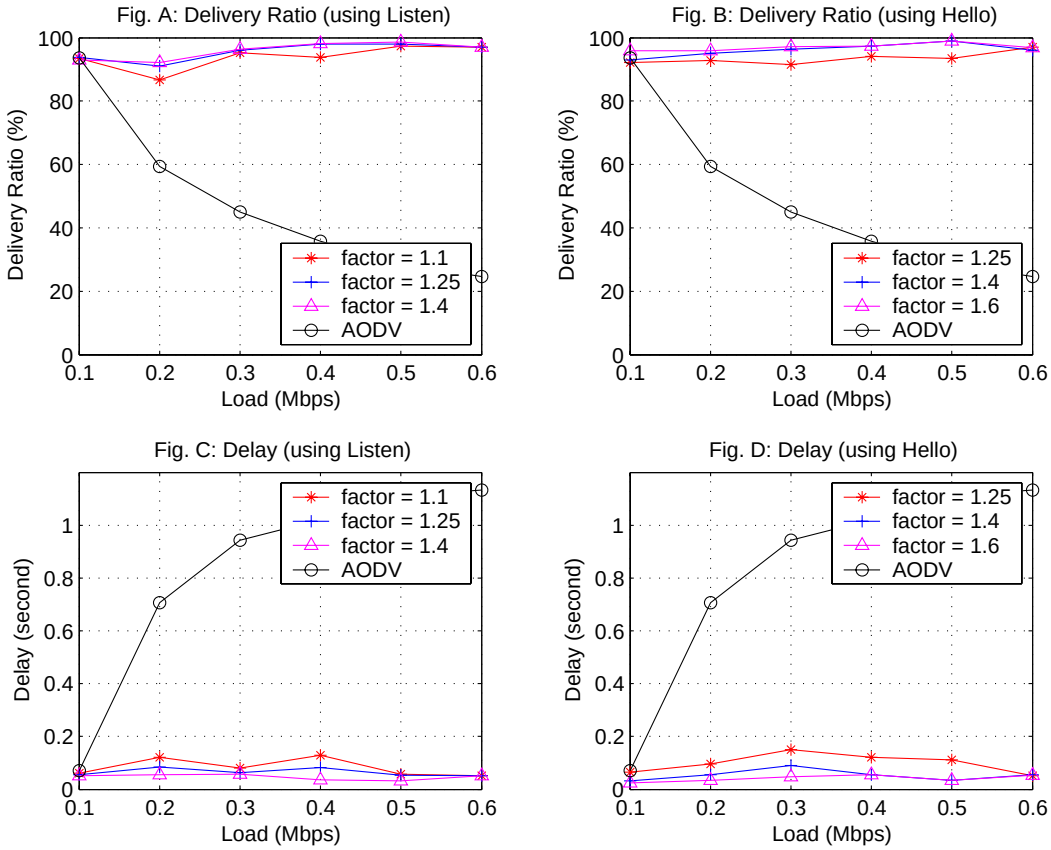

Figure 6.20: Results for QoS-aware routing using the admission scheme with different weight factors and AODV. Fig. A: Packet delivery ratio using “Listen”. Fig. B: Packet delivery ratio using “Hello”. Fig. C: Delay using “Listen”. Fig. D: Delay using “Hello”.

6.4.4 Static Topology using the Admission Scheme

The other scheme incorporated into our QoS-aware routing protocol is the admission scheme. In the admission scheme, flows are denied if there is not enough bandwidth available to support their request. This results in the total capacity of the admitted flows being less than that of the feedback scheme, so packet collisions occur less frequently. Thus, we expect that the packet delivery ratio using the admission scheme should be larger than that of using the feedback scheme. Correspondingly, the packet delay should be decreased significantly due to fewer collisions. We use the same topologies as in section 6.4.4, and we obtain the simulation results shown in Figure 6.20. Using QoS-

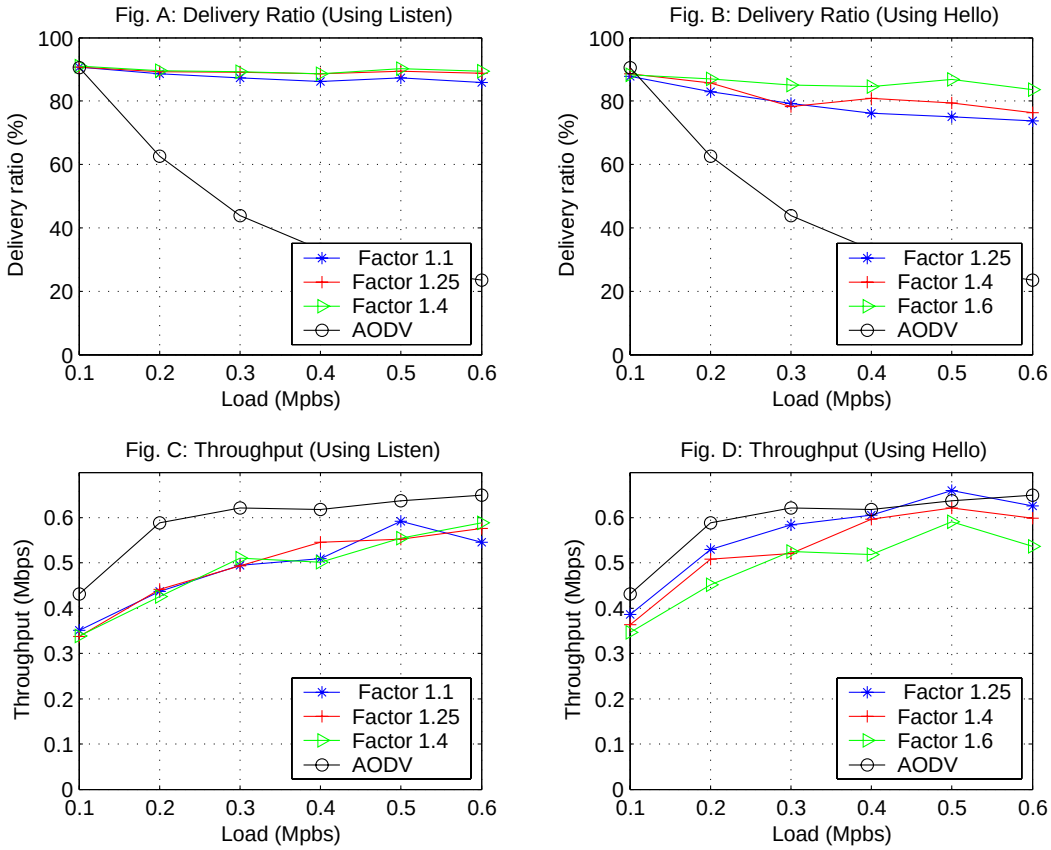

Figure 6.21: Results for QoS-aware routing using mobile topologies (maximum speed of 3 m/s) with different weight factors and AODV. Fig. A: Packet delivery ratio using "Listen". Fig. B: Packet delivery ratio using "Hello". Fig. C: End-to-end throughput using "Listen". Fig. D: End-to-end throughput using "Hello".

aware routing, the packet delivery ratio remains constantly above 90%, and the delay remains lower than 0.17 seconds.

6.4.5 Mobile Topology

Our routing protocol is designed with the restriction of combinatorial stability. Therefore, if the network changes too fast, we do not expect the QoS-aware routing protocol to perform well. Thus, we choose low mobility scenarios that mimic pedestrian speeds to test our protocol. In the scenarios we choose, each node moves towards a random

destination using a speed randomly chosen between 0-3 m/s. Five random source-destination pairs send packets using a requested rate between 0.1 Mbps and 0.6 Mbps. The simulation time is 500 seconds. Figure 6.21 shows the results obtained by averaging 10 different scenarios. The packet delivery ratio is between 85% and 90% using the QoS-aware routing protocol with "Listen" bandwidth estimation, and the packet delivery ratio is between 75% and 90% using the QoS-aware routing protocol with "Hello" bandwidth estimation. QoS-aware routing shows great improvement over using AODV, which achieves very low packet delivery ratio for high requested loads. As there is a trade-off between packet delivery ratio and throughput that we discussed previously, the higher the packet delivery ratio, the lower the achievable throughput. Therefore, using the "Listen" scheme, the end-to-end throughput is slightly decreased compared with using the "Hello" scheme, as shown in Figure 6.21.

We would expect that the QoS-aware routing protocol's performance will degrade as the moving speed increases, because we designed the QoS-aware routing protocol with a model of low mobility. Therefore, we did not incorporate any predictive scheme to find a new route before the old route is broken. This results in very long transient time when the required QoS is not guaranteed, due to a route break or network partition, which significantly decreases the packet delivery ratio. However, our QoS-aware routing protocol still gets relatively higher packet delivery ratio compared with AODV, as shown in Figure 6.22. The "Hello" scheme's performance is better than the "Listen" scheme's performance in term of end-to-end throughput, while the "Listen" scheme's performance is better than the "Hello" scheme's performance in term of packet delivery ratio.

6.5 Conclusions and Future Work

This chapter proposes incorporating QoS into routing, and introduces bandwidth estimation by disseminating bandwidth information through "Hello" messages. A cross-layer approach, including an adaptive feedback scheme and an admission scheme to provide information to the application about the network status, are implemented. Simulations show that our QoS-aware routing protocol can improve packet delivery ratio greatly without impacting the overall end-to-end throughput, while also decreasing the

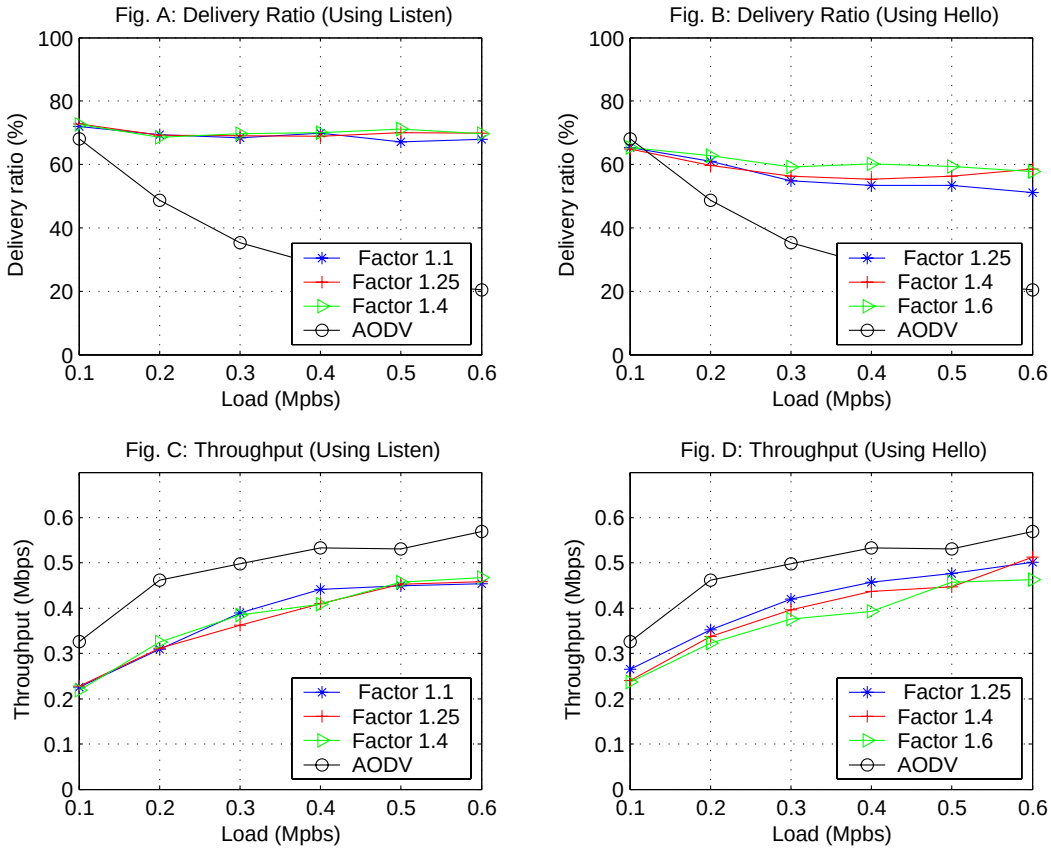

Figure 6.22: Results for QoS-aware routing using mobile topologies (maximum speed of 20 m/s) with different weight factors and AODV. Fig. A: Packet delivery ratio using "Listen". Fig. B: Packet delivery ratio using "Hello". Fig. C: End-to-end throughput using "Listen". Fig. D: End-to-end throughput using "Hello".

packet delay and the energy consumption significantly.

We have compared two different methods of estimating bandwidth. The "Hello" bandwidth estimation method performs better than the "Listen" bandwidth estimation method when releasing bandwidth immediately is important. The "Hello" and "Listen" schemes work equally well in static topologies by using large weight factors to reduce the congestion and minimize the chance of lost "Hello" messages incorrectly signaling a broken route. In a mobile topology, "Hello" performs better in term of end-to-end throughput, and "Listen" performs better in term of packet delivery ratio. From the perspective of overhead, "Listen" does not add extra overhead, but "Hello" does add

overhead by attaching neighbors' bandwidth consumption information in the "Hello" messages.

In our protocol, we have not incorporated any predictive way to foresee a route break, which causes a performance degradation in mobile topologies. Therefore, some methods such as preemptive maintenance routing [87] and route maintenance based on signal strength [95] might help to reduce the transient time when the required QoS is not guaranteed due to a route break or network partition, so that the routing protocol can react much better to mobile topologies.

The accurate measurement of the capacity of a multi-hop mobile network is an open issue right now. Further study of the 802.11 MAC layer's behavior could be helpful to understand this capacity issue. Also, in a real scenario, shadowing will cause a node's transmission range to vary, and it will not be the ideal circle that is assumed here. How to incorporate these non-idealities into our protocol is the subject of our future research.

Furthermore, incorporating different transmission ranges among all the hosts and analyzing fairness among the hosts will be explored in our future work. Our ultimate goal is to provide a model from the application layer to the MAC layer for supporting service differentiation.

Chapter 7

Dual Channel MAC for Improving Fairness in Ad Hoc Networks

Designing good Medium Access Control (MAC) protocols is still an open research field in the area of wireless ad hoc networks. Scheduled medium access protocols, such as those that require global time synchronization (e.g., TDMA) or those that require pre-distribution of codes (e.g., CDMA), require considerable infrastructure support or global knowledge of the entire network status, which is not feasible in ad hoc wireless networks. Thus, CSMA-type MAC protocols, like IEEE 802.11, are better suited for MANETs. To support QoS, it is desirable to reduce collisions between control packets and data packets, which is an inherent problem for IEEE 802.11 because the control and data packets are transmitted using the same channel and thus compete for the transmission resources. In addition, studies show that severe unfairness occurs using IEEE 802.11 as the underlying MAC protocol in MANETs. The MAC unfairness impacts the performance of real-time applications, such as audio or video streaming, which are sensitive to delay and jitter. We propose a dual channel MAC protocol, which separates the control channel and the data channel in order to avoid collisions between data and control packets and to improve fairness. In the remainder of this chapter, we discuss the design issues, the performance and the trade-offs for supporting QoS using a dual channel MAC.

7.1 Motivation

The IEEE 802.11 MAC protocol is the standard for wireless LANs. It is widely used in wireless ad hoc networks as well. However, as IEEE 802.11 is designed for wireless LANs, it has inherent limitations when used in wireless ad hoc networks, especially for supporting QoS.

A study on the reasons why the IEEE 802.11 MAC protocol does not work well in multi-hop wireless ad hoc networks is conducted by Xu et al. [96]. Xu et al. evaluated the performance of TCP in MANETs using IEEE 802.11 as the underlying MAC protocol first, and then determined that the fundamental reasons for TCP's instability are IEEE 802.11's "hidden node", "exposed node" and unfairness problems. "Hidden node" refers to a node within interference range of the destination node, but out of the sender's sensing range. "Exposed node" is a node within the sender's sensing range, but out of the receiver's interference range. The RTS/CTS scheme greatly lessens the "hidden node" problem, under the condition that any possible station interfering with a transmission between the sender and the receiver is within the sender's interfering range. However, this is not true for all scenarios. For the "exposed node" problem, as IEEE 802.11 does not include mechanisms to address this, the exposed nodes prevent the intermediate nodes from passing packets to their neighbors. Therefore, a severe instability problem happens. Besides these two reasons, the unfairness caused by the exponential backoff scheme, which always favors the nodes that last successfully captured the channel, brings another problem.

MANETs are distributed networks, so CSMA type MAC protocols best meet their medium access needs. One technique that has been successfully employed by cellular systems for avoiding some of the problems caused by collisions of data and control packets is to use separate channels for control packets and data packets. Therefore, this motivates us to examine what will be gained and what will be lost by adopting the same idea—i.e., separating the control channel and the data channel for MANETs.

Information unfairness [69] and information incompleteness are two reasons causing unfairness. Figure 7.1 is an example topology showing an information unfairness case, where S2 is within the transmission range of D1. If both S1 and S2 back off, S2 always knows when to contend for the channel. On the other hand, once S1 backs off, it may face multiple further exponential backoffs, because S1's RTS might further

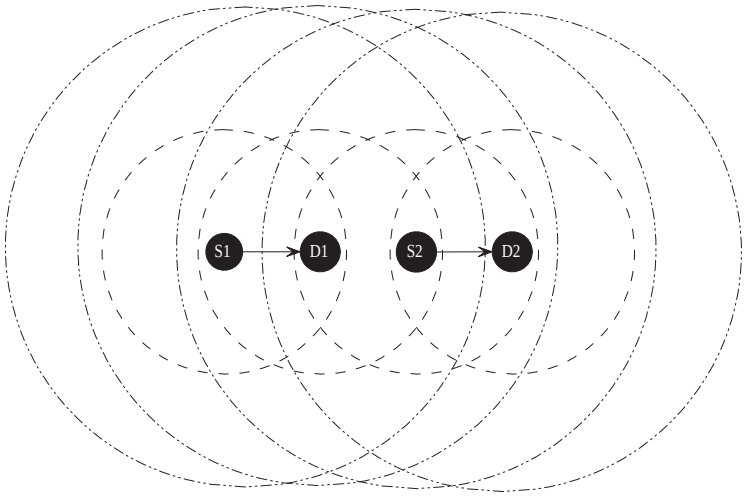

Figure 7.1: Example topology for illustrating information unfairness: S1 transmits to D1 and S2 transmits to D2. The dotted lines indicate the transmission ranges and the dash-dot lines indicate the interference ranges.

continuously collide with the long data packets being sent by S2. This results in an unfair allocation of the channel resources.

The information incompleteness scenario is shown in Figure 7.2, where S1 and S2 are not in each other's interference range. When S1 sends an RTS to D1, if at the same time S2 sends an RTS to D2, these two RTSs collide, resulting in both S1 and S2 backing off. If S2's backoff time is shorter than S1's, S2 captures the channel in the next trial. However, S1 cannot hear the on-going transmission between S2 and D2. Thus, it continues trying to send an RTS. The following RTS collides with S2's data packets. Thus, S1 exponentially backs off its contention window, resulting in unfairness to S1 under high traffic conditions. The flow activities of this case are illustrated in Figure 7.3.

To reduce the unfairness caused by information unfairness, we need to ensure that there are no (or minimal) collisions between control packets and data packets. To reduce the unfairness caused by information incompleteness, a possible solution is that the sender pauses its attempts to capture the channel when the channel is busy. These two issues could be addressed using a dual channel system and an out-of-band busy tone, which are incorporated in our proposed dual channel MAC and will be detailed in Section 7.2.2.

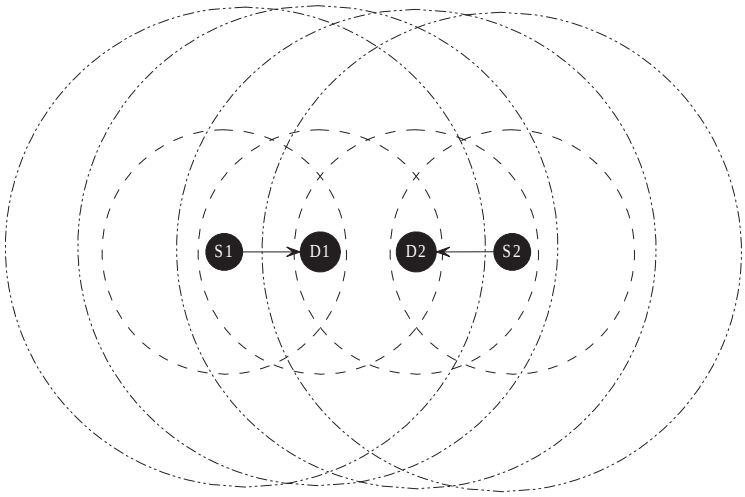

Figure 7.2: Two symmetric flows for illustrating information incompleteness: S1 transmits to D1 and S2 transmits to D2. The dotted lines indicate the transmission ranges and the dash-dot lines indicate the interference ranges.

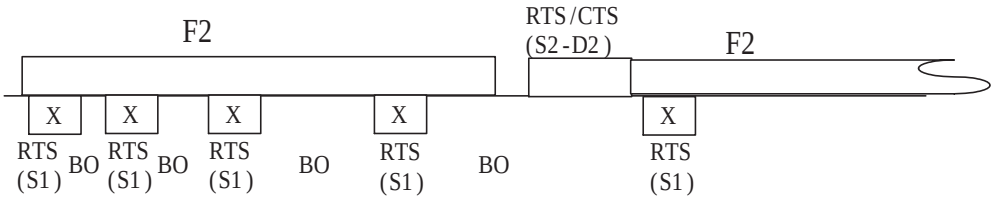

Figure 7.3: Flow activities illustration.

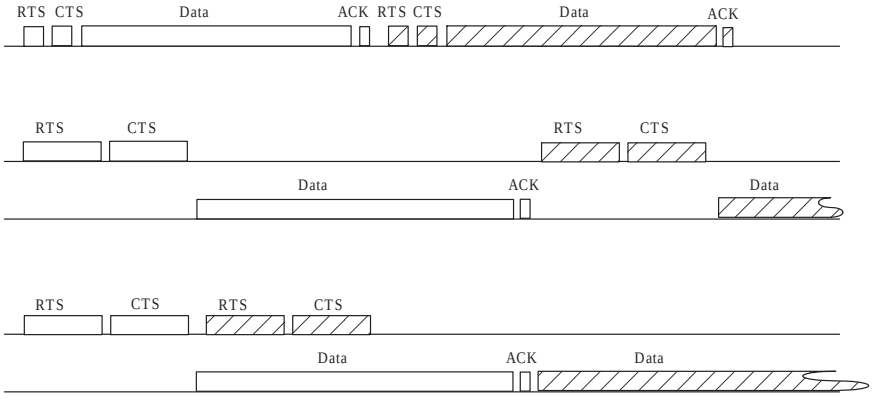

Figure 7.4: Top: Exchange of IEEE 802.11 control packets and data packets. Middle: Exchange of dual channel control packets and data packets. Bottom: Exchange of dual channel control packets and data packets with the pre-scheduling scheme.

7.2 Dual Channel MAC Protocol with Pre-schedule (DMAC)

We first present the reasons for using pre-scheduling of data packets and the basic principles of DMAC, and then we present a detailed implementation of DMAC.

7.2.1 Channel Partitioning and Pre-scheduling Scheme

When using a dual-channel system, the total bandwidth is divided into two channels. One is dedicated for control packets and the other is used exclusively for data packets. Therefore, channel partitioning is the first problem that a dual channel MAC protocol must address. If the control channel consumes too much bandwidth, the overall throughput for data transfers is decreased. On the other hand, if the control channel's bandwidth is not sufficient, it means that the data channel cannot be fully utilized as data transfers must wait a long time for the signaling handshake to complete.

At the same time, another problem brings negative effects on the performance of a dual channel MAC. In IEEE 802.11, the RTS/CTS packets are relatively small compared with the data packets, so the handshake time can be neglected. If we use a dual channel system, the control channel occupies a relatively small amount of bandwidth. Although the RTS/CTS packet is still small, the handshake time could be very long due to the control channel's limited bandwidth. Under this situation, if every data packet

waits until the completion of an RTS/CTS handshake, the data channel is free during that handshake time slot. Thus, the system capacity could be significantly decreased.

For reasonable partitions of the bandwidth into control and data channels, the RTS/CTS handshake time will be less than the time to transmit a data packet. Under this scenario, it is desirable to pre-schedule additional packets in the control channel during the transmission of a data packet. A simple example is given in Fig. 7.4, which shows one sender and one receiver. The top figure shows the exchange of control and data packets in IEEE 802.11. The middle figure presents the exchange of data packets in a dual channel system with the same sequence of RTS-CTS-Data-ACK. The bottom figure shows the exchange of control and data packets using pre-scheduling. If we compare the middle figure with the bottom figure, an intuitive conclusion is that using pre-scheduling improves channel utilization. A detailed discussion on how channel partitioning and pre-scheduling affect the performance of a dual channel MAC will be presented in Section 7.3.

7.2.2 DMAC Protocol Overview

As with the IEEE 802.11 MAC protocol, we adopt an RTS/CTS/Data/ACK exchange in our dual channel MAC (DMAC) protocol, and we adopt the use of a network allocation vector (NAV) in the control channel. In addition, DMAC uses two schedule vectors (SCH^1 and SCH^2), and one neighbor reservation vector RSV. SCH^1 is used to record the negotiated start and end transmission time of the first packet in the host's packet queue. SCH^2 is used to store the next packet's transmission time in the host's packet queue. The RSV vector is similar to the NAV vector in IEEE 802.11 and the NAV vector in the control channel. The only difference is that this vector is exclusively used for recording information about a neighbor's data transmission negotiation to avoid data collision while neighbors are transmitting data packets in the data channel.

Transmissions begin with RTS and CTS exchanges in the control channel. When a sender creates an RTS, it puts the earliest data channel available time in the RTS packet. The neighbors who overhear the RTS packet update their NAV vector to reserve the control channel for the remainder of the RTS/CTS exchange. The receiver checks its own earliest data channel available time and puts the negotiation value in the CTS packet header. The neighbors that overhear the CTS packet update their SCH

vector. Once the RTS/CTS exchange is finished, the sender updates its SCH vector and schedules the data transmission according to the data channel negotiation result. When the RTS/CTS exchange is finished, the control channel is prepared for the next packet channel access, as the next data packet transmission can be scheduled when the previous data packet is still being transmitted.

7.2.3 Implementation Details

7.2.3.1 Sending RTS

When the link layer sends down a data packet, the MAC layer creates an RTS. The RTS packet carries information about the earliest available time A_i^k of node N_i to transmit data packet P_k and the duration D_i^k required for transmitting the corresponding data packet. If the control channel is sensed free, the RTS packet is scheduled to be transmitted after DIFS time; otherwise, the contention window size is doubled.

The earliest available time is decided by the scheduling vector's value SCH_i^1 and the reservation time for the neighbors (reservation vector RSV_i) at the time when the RTS is sent to the channel. SCH_i^2 does not affect the earliest available time's calculation, since it is assumed that at least the host is able to schedule the next packet transmission. If there are no data packets scheduled for transmission in the data channel and no reserved channel time for neighbors' transmissions, the earliest time for data transmission is after SIFS + CTS + SIFS time. Otherwise, the earliest available time is decided by the time difference among the schedule vector's ending time, reservation vector value and current time. How to decide the earliest available time is shown as follows:

If ($RSV_i \geq SCH_i^1$)
    If ($RSV_i - CT > SIFS + CTS + SIFS$)
        $A_i^k = RSV_i$
    Else
        $A_i^k = CT + SIFS + CTS + SIFS$
Else
    If ($SCH_i^1 - CT > SIFS + CTS + SIFS$)

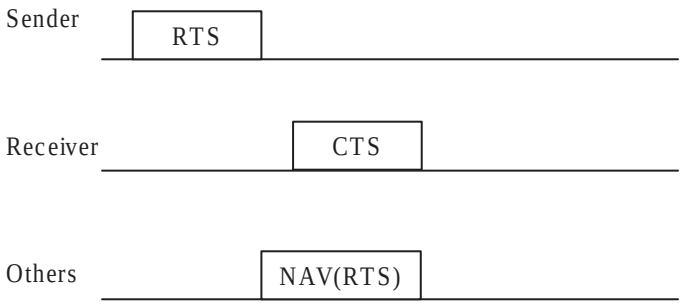

Figure 7.5: NAV vector value updating.

$$A_i^k = SCH_i^1$$

Else

$$A_i^k = CT + SIFS + CTS + SIFS$$

where CT is the current time.

7.2.3.2 Sending CTS

A host receiving an RTS packet first reads the sender's earliest available time from the RTS packet. Then, it compares its available time with the sender's earliest available time. Finally, it puts the agreed data transmission scheduling time, which is the later earliest time between the sender and receiver, in the CTS packet. A host replies back with a CTS after SIFS spacing.

7.2.3.3 Setting the NAV

The hosts overhearing the neighbors' RTS set their NAV with a value of SIFS+CTS to reserve the control channel for the corresponding CTS. As the dual channel MAC uses a separate data channel to transmit data, the NAV setting is different from the IEEE 802.11 standard, which considers data and ACK packets as well. In addition, the NAV is updated when the host hears other nodes' CTS in IEEE 802.11, but in the dual channel case it is not necessary. Figure 7.5 shows the setting of the NAV vector.

7.2.3.4 Pre-scheduling control packet exchanges

When a DATA or ACK packet is being transmitted, the control channel is free, if we use the same RTS-CTS-DATA-ACK four-way handshake as used in IEEE 802.11. Therefore, a pre-scheduling scheme is adopted to efficiently use the control channel. Pre-scheduling is triggered when the data channel is transmitting a data packet and there are packets in the nodes' queues waiting to be transmitted, but the control channel is free. We propose a pre-scheduling scheme as follows:

- After finishing the scheduling of the first packet, the sender checks whether there is a packet waiting in the queue. If there is, it releases one packet (the second packet) to the MAC layer.
- The MAC layer waits DIFS time to do the RTS-CTS exchange procedure in the control channel for scheduling the second data packet transmission.
- The pre-scheduling does not start again for the third packet, if the second packet signaling is finished but the data channel is still transmitting the first packet.
- If the first packet is not transmitted successfully, an RTS/CTS signaling for the first packet retransmission is initiated.

The pre-scheduling diagram is shown in Figure 7.6.

7.2.3.5 Setting the neighbor reservation vector

In the control channel, NAV is used for reducing collisions among signaling packets. In the data channel, neighbors should also avoid transmitting data packets simultaneously. Therefore, a reservation vector is used to inform hosts of their neighbors' data transmission schedules. Reservation information can be obtained from RTS and CTS packets in the control channel. Once a host receives an RTS or a CTS packet from its neighbors, it updates its reservation vector correspondingly.

7.2.3.6 Out-of-band busy tone for the control channel

In IEEE 802.11, RTS or CTS packet collisions are not a major problem due to their relatively short packet lengths. However, if the channel is divided into a control and a

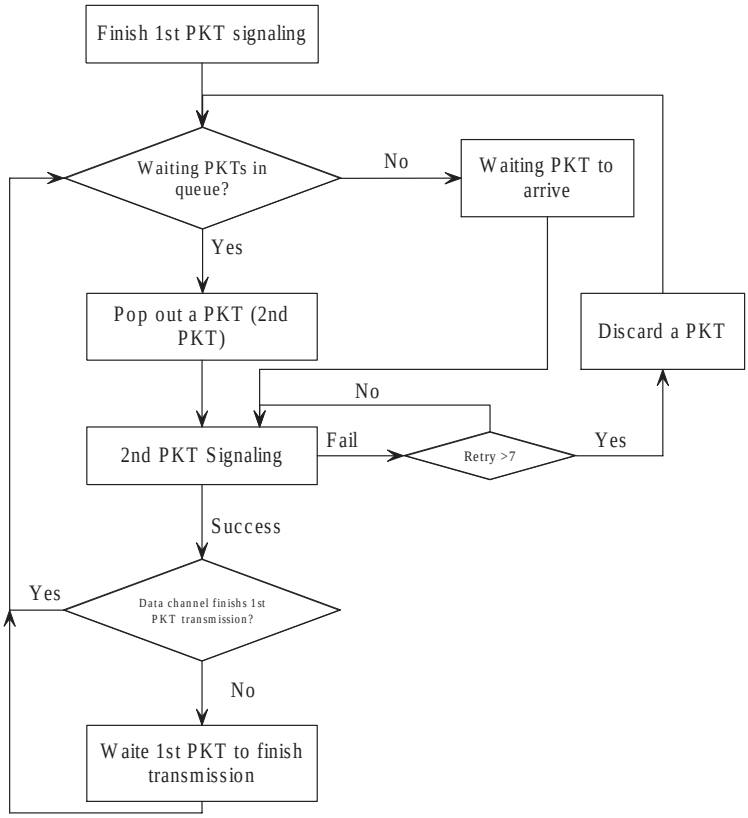

Figure 7.6: Pre-scheduling procedure diagram.

data channel, the control channel only occupies a small portion of the total bandwidth, which results in the transmission time of RTS/CTS being relatively long. Thus, the probability of RTS or CTS packet collisions increases dramatically, which results in significant adverse impact on the MAC performance. Therefore, we propose using an out-of-band busy tone to reduce collisions in the control channel.

An out-of-band busy tone is a simple tone signal transmitted out of the control and data channels. An out-of-band busy tone is sent when a host begins receiving an RTS packet destined to itself. The neighbors that hear the busy tone freeze their backoff timers for an RTS duration.

7.3 Bandwidth Partitioning

Bandwidth partitioning is an important issue in the DMAC protocol design. More specifically, how much bandwidth is allocated for the control channel and how much is allocated for the data channel are important factors affecting the final performance of DMAC. The control packet and data packet lengths are the important factors in deciding the best partitioning of the bandwidth.

7.3.1 Parameters for Analyzing Bandwidth Partitioning

We use the default packet length settings in NS-2 for IEEE 802.11. DMAC's control packets need more bytes to carry extra information about the reservation information than those for IEEE 802.11. Detailed packet size information is shown in Table 7.1.

Table 7.1: Packet length (bytes) for IEEE 802.11 and DMAC control packets.

...	DMAC	IEEE 802.11
RTS	48	44
CTS	56	38
ACK	38	38

7.3.2 Theoretical Analysis

We first define some parameters to assist our analytical study. We define the time used for transmitting an RTS and a CTS as control-time-slot (CTslot), and the time used for transmitting a data packet and an ACK as data-time-slot (DTslot). First, as shown in Figure 7.4, if we do not consider the backoff window, SIFS and DIFS, the bandwidth partition should allow that one CTslot equals one DTslot. Otherwise, too much bandwidth is wasted in the data channel waiting for the completion of an RTS/CTS exchange in the control channel. However, we do not want the control channel to occupy too much bandwidth, which would result in the available bandwidth for the data channel being too small. Thus, at least four CTslots equal one DTslot. With this observation and with the assumption of data packet size being 1500 bytes with an IP header of 82

bytes, we conclude that a reasonable partition for the control channel's bandwidth is as follows:

$$\begin{gathered}\frac{RTS+CTS}{RTS+CTS+Data+ACK}<CP<\frac{4\times(RTS+CTS)}{4\times(RTS+CTS)+Data+ACK} \\ \frac{48+56}{48+56+(1500+82)+38}<CP<\frac{4\times(48+56)}{4\times(48+56)+(1500+82)+38} \\ 6.03\%<CP<20.43\%\end{gathered} \tag{7.1}$$

where CP refers to the control channel bandwidth percentage.

Hence, when the total bandwidth is 2 Mbps, the control channel should be at least 0.12 Mbps and at most 0.42 Mbps. Thus, if we assume that RTSs arrive according to a Poisson distribution, the average rate of λ should be

$$\begin{gathered}\frac{MinControlBandwidth}{RTS+CTS}<\lambda<\frac{MaxControlBandwidth}{RTS} \\ 1200<\lambda<8000\end{gathered} \tag{7.2}$$

The probability of a successful RTS transmission $P_{success}$ equals no arrival RTS during the propagation or medium sense (μ) time [97]. The following is the success probability function:

$$P_{success}=e^{-\lambda\mu} \tag{7.3}$$

Thus the failure probability is $P_{failure}=1-e^{-\lambda\mu}$. The equation to calculate the number of retries needed, on average, for a successful, collision-free RTS with error rate $\epsilon=10^{-5}$ is as follows:

$$(1-e^{-\lambda\mu})^k<\epsilon \tag{7.4}$$

$$k>\frac{log(\epsilon)}{log(1-e^{-\lambda\mu})} \tag{7.5}$$

k must be an integer, so if $\lambda=1200$, $k=2$; and if $\lambda=8000$, $k=3$. A reasonable bandwidth partition should allow the RTS/CTS exchange to be completed before the completion of a transmission on the data channel. Therefore, the time used for the

control channel to finish k attempts should equal the data transmission time. During the k times contention, the control channel transmits at least k RTS and one CTS packets and at most k RTS and k CTS packets, if we ignore the SIFS, DIFS and backoff time. So the mean value is k RTS and $(k/2 + 1/2)$ CTS packets. Hence, the following equation is obtained:

$$\frac{k \times RTS + \frac{k+1}{2} \times CTS}{Control\_BW} = \frac{Data + ACK}{Data\_BW} \tag{7.6}$$

$$CP = \frac{k \times RTS + \frac{k+1}{2} \times CTS}{k \times RTS + \frac{k+1}{2} \times CTS + Data + ACK} \tag{7.7}$$

According to the above function, when $k = 2$, $CP = 9.9\%$; and when $k = 3$, $CP = 15.6\%$. Thus, theoretically, without considering SIFS, DIFS or backoff time and without considering the hidden node problem, the control channel should occupy 9.9% to 15.6% of the total bandwidth, under the condition that the control and data channels be fully utilized.

7.4 Simulations and Discussion

We have done extensive simulations to evaluate the performance of the proposed DMAC protocol, in terms of throughput, bandwidth partition rate and fairness. All the simulations are conducted over ten independent runs using NS-2. The transmission range of each node is set according to a 914 MHz Lucent WaveLAN DSSS radio model. The packet size is 1500 bytes. Other parameters use the default settings in NS-2.

7.4.1 Bandwidth Partitioning

In the theoretical analysis, we did not consider the hidden node problem, backoff time, SIFS and DIFS. Thus, in a real scenario, we would expect the lower bound of the control channel's bandwidth should be fairly tight but the upper bound might vary somewhat. However, we emphasize that we do not seek an optimal partition, as the traffic pattern will affect the optimal channel partition. However, the statistically based analysis offers a reasonable partition that can be used as a general guide.

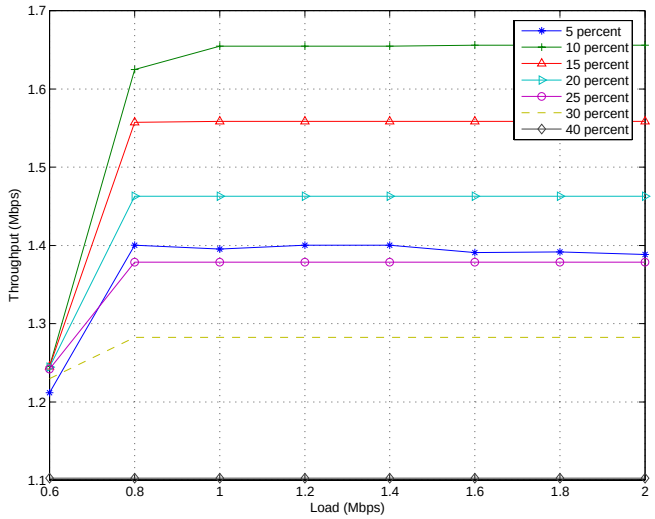

Figure 7.7: Bandwidth partition comparison study for the scenario in Figure 7.2.

In our first simulations, we use the scenario in Figure 7.2 and vary the control channel bandwidth percentage from 5% to 40%. The achieved average total throughput of ten independent runs is shown in Figure 7.7. The results show that using 10% control channel allocation achieves the best data channel usage. The best control channel partition percentage is very close to the lower bound of the theoretical analysis. This is because the simple scenario only contains two flows and the traffic is regulated compared with multiple flows. Therefore, the RTS success probability $P_{success}$ is high, which results in low RTS retries.

Next we use the scenario in Figure 7.8 and vary the control bandwidth from 5% to 40%. The simulation results are shown in Figure 7.9. This four flows scenario is a typical information incompleteness case, but the out-of-band busy tone signaling scheme helps to avoid the blind contention, which will be further explained in section 7.4.2.3. Therefore, the reasonable bandwidth partition still matches the theoretical analysis. In this simulation, the control channel percentage 10% once again obtains the best data channel utilization.

Finally, we choose scenarios with fifty nodes within a 1000 m by 1000 m square area. Five neighbor pairs are randomly chosen to transmit the same amount of data. We have done extensive simulations, and among these simulations we pick two simulations with scenarios that have hidden node problems and ensured some amount of interfer-

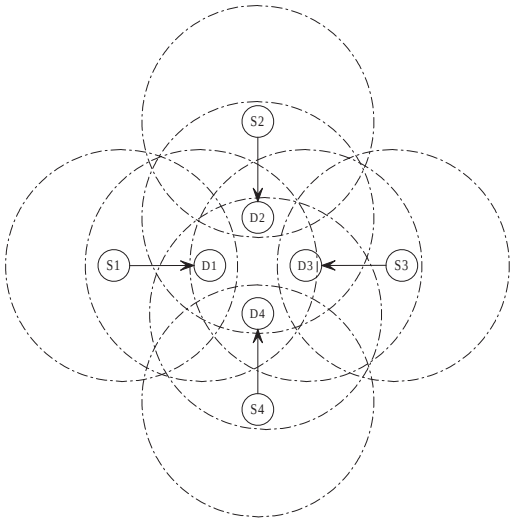

Figure 7.8: Four symmetric flows: S1 transmits to D1, S2 transmits to D2, S3 transmits to D3 and S4 transmits to D4. The dotted lines indicate the transmission ranges.

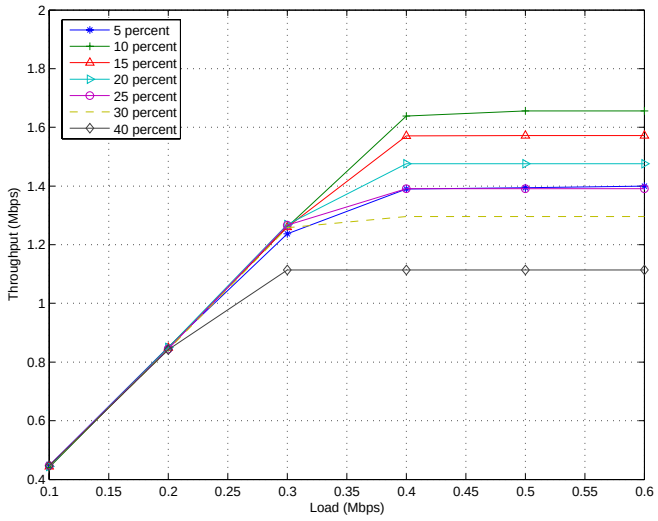

Figure 7.9: Bandwidth partition comparison study for the scenario in Figure 7.8.

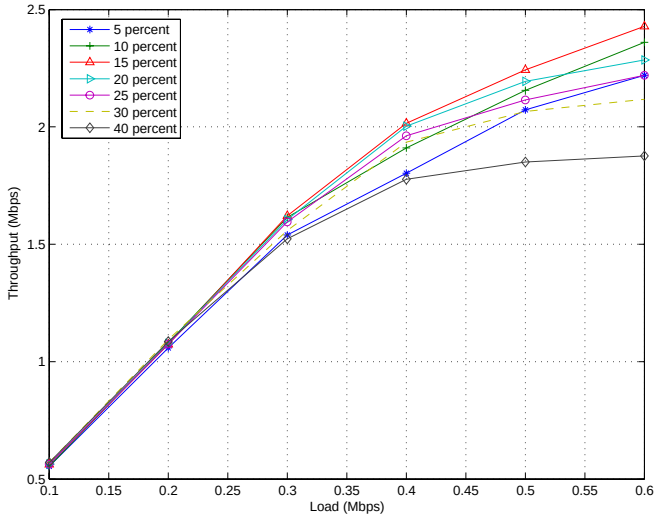

Figure 7.10: Bandwidth partition comparison study for random scenario 1.

ence. The simulation results are shown in Figure 7.10 and Figure 7.11. In Figure 7.10, due to the hidden node problems and complicated flow infererence/competitions, the best control bandwidth percentage is not 10% as in the previous two cases, but 15%. However, this value still falls within the analytical range. In Figure 7.11, the best performance is obtained when the control bandwidth percentage is still 15%, but the 20% control channel percentage's performance is very close to that for 15% control bandwidth percentage. The reason could be that the SIFS, DIFS and backoff time effects become critical. In addition, the analytical result is based on the assumption of no hidden nodes. However, 15% control bandwidth percentage partition still reaches the best performance. Thus the simulation and the analysis match.

7.4.2 Fairness

In this section, we present the metric used for evaluating fairness, analyze the reason why DMAC improves fairness, and then discuss the simulation data.

7.4.2.1 Jain's Fairness Index

To measure the fairness of equally shared flows, one good metric is Jain's fairness index, which is defined as follows:

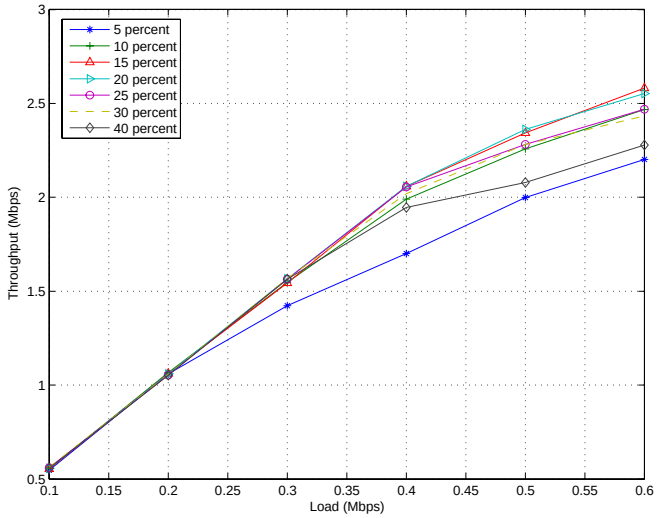

Figure 7.11: Bandwidth partition comparison study for random scenario 2.

$$F(x_1, x_2, x_3, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \tag{7.8}$$

where F is between 0 and 1, x_i is the traffic load for each flow i, and n is the number of total flows. If the F value is closer to 1, this indicates a more fair sharing of the bandwidth among the flows.

7.4.2.2 Information Unfairness

In section 7.1, the unfairness caused by information unfairness is presented and a possible solution is proposed. In DMAC, the separate control and data channel automatically address this issue. For the example topology in Figure 7.1, once S1's RTS collides, S1 backs off and re-sends an RTS. The RTS will not continuously collide with a data packet. Hence, the continuous backoff problem is reduced. Figure 7.12 illustrates the flow activities using DMAC.

In the fairness discussion, we focus on analyzing the short-term fairness. Unfairness over a short time results in significant performance degradation for applications like real-time voice or video applications and for transport protocols like TCP. The fairness index comparison is shown in Figure 7.13. Jain's fairness index is between 0.9984 and 0.9999 for DMAC, while the fairness index is between 0.9954 and 0.9997 for IEEE

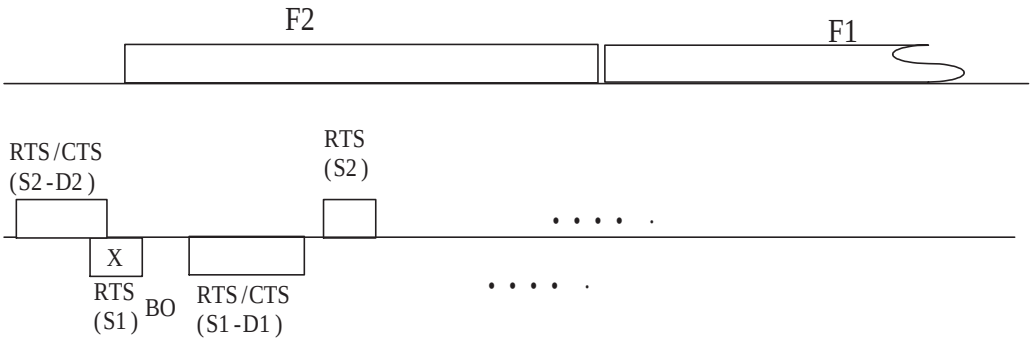

Figure 7.12: Flow activities illustration for information unfairness.

802.11.

7.4.2.3 Information Incompleteness

The out-of-band busy signal design helps reduce the unfairness caused by information incompleteness. In the scenario shown in Figure7.2, if S1 and S2 send an RTS simultaneously and thus the RTSs collide, S1 and S2 both back off. If S1 first captures the channel, S2 does not continuously exponentially back off because when it hears the out-of-band busy-tone, it pauses its backoff timer and does not attempt to transmit an RTS. Therefore, theoretically, the unfairness is greatly lessened. The flow activities using DMAC for the information incompleteness case are shown in Figure 7.14.

We first consider the simplest scenario in Figure 7.2 in which two symmetric flows are transmitted simultaneously. Simulation results are shown in Figure 7.15. Using DMAC, the two flows equally share the channel and the Jain's fairness value is 1. However, using IEEE 802.11, the Jain's fairness value is between 0.8056 and 0.9542.

Next we simulate a more complicated scenario using symmetric flows in which four flows compete for data packet transmission, as shown in Figure 7.8. Ten independent simulations are conducted, and the mean fairness value is shown in Figure 7.16. Using DMAC, the four flows almost equally share the channel and the Jain's fairness value is between 0.9995 and 0.9998 as the load varies. However, using IEEE 802.11, the unfairness is obvious, and the Jain's fairness value is between 0.6590 and 0.9801 as the load varies.

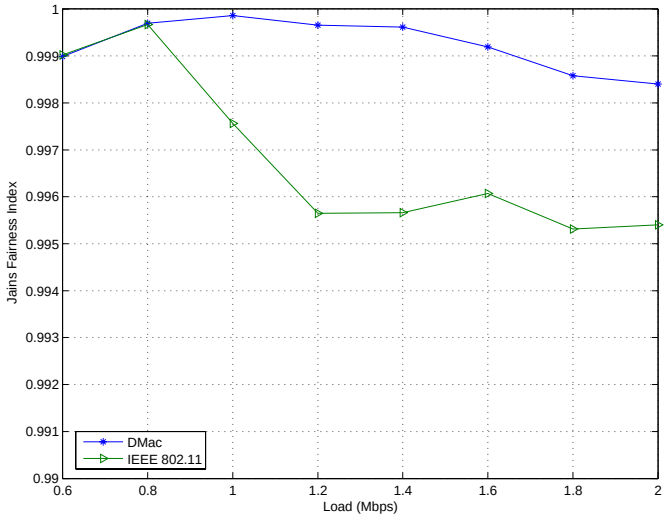

Figure 7.13: Jain's fairness index comparison for the scenario shown in Figure7.1.

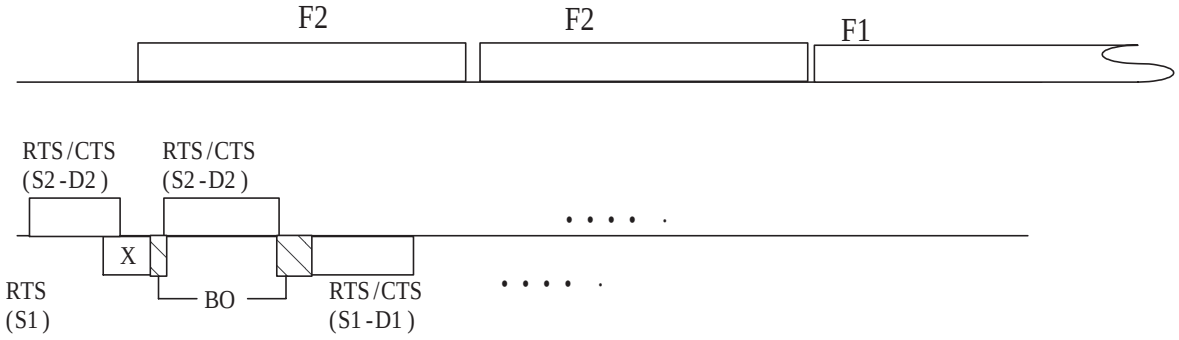

Figure 7.14: Flow activities illustration for information incompleteness.

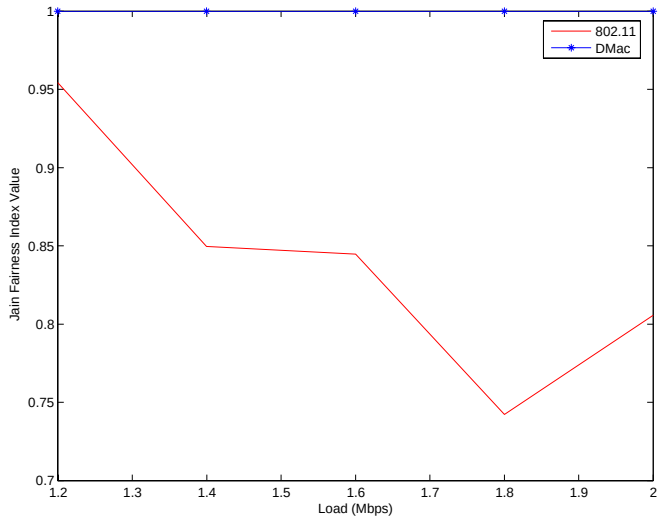

Figure 7.15: Jain's fairness index comparison for DMAC and IEEE 802.11 using two symmetric flows shown in Figure 7.2.

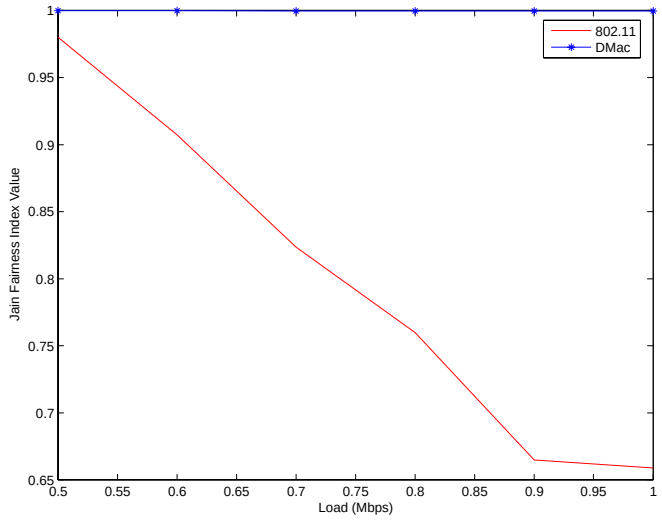

Figure 7.16: Jain's fairness index comparison for DMAC and IEEE 802.11 using four symmetric flows shown in Figure 7.8.

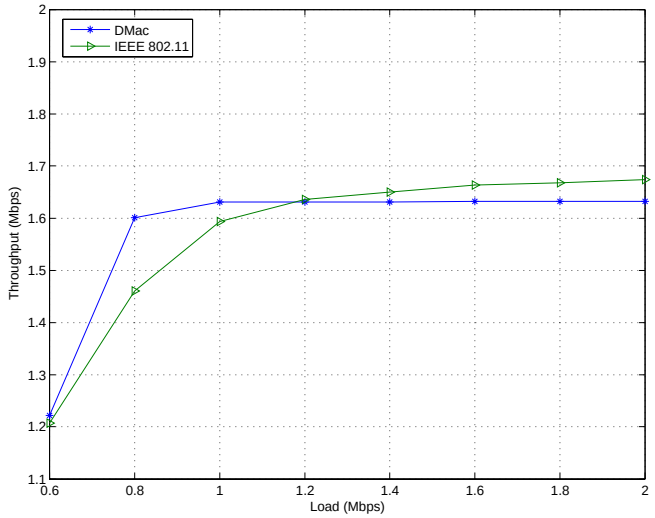

Figure 7.17: Total throughput for the two flow scenario in Figure 7.2.

7.4.2.4 Discussion

The fairness improvement in the above cases seems to occur when we impose strict constraints. The first constraint is that one flow's sender is within the other flow's receiver's transmission range. The second constraint is that the sender must hear the out-of-band busy-tone signal. Actually, these are the most common cases in real transmission scenarios, such as the chain topology. The starvation due to inner flow contention in multi-hop transmission scenarios is the reason for poor throughput usage in MANETs. Therefore, we would expect that the DMAC protocol may bring positive effects on reducing the flow starvation problem.

7.4.3 Total Throughput

To improve IEEE 802.11's throughput, researchers put their efforts in solving the exposed node problem [14], reducing collisions due to hidden nodes [14], or avoiding long backoff times [98]. In our design, we do not incorporate any scheme to address the above issues, so a gain in overall throughput improvement is not expected. However, we still want to obtain comparable total throughput similar to that of IEEE 802.11.

We perform simulations using the scenarios shown in Figure 7.2, Figure 7.8 and

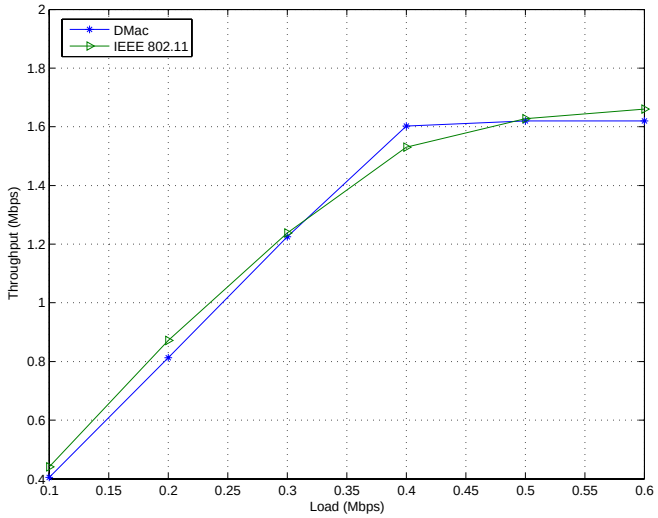

Figure 7.18: Total throughput for the four flow scenario in Figure 7.8.

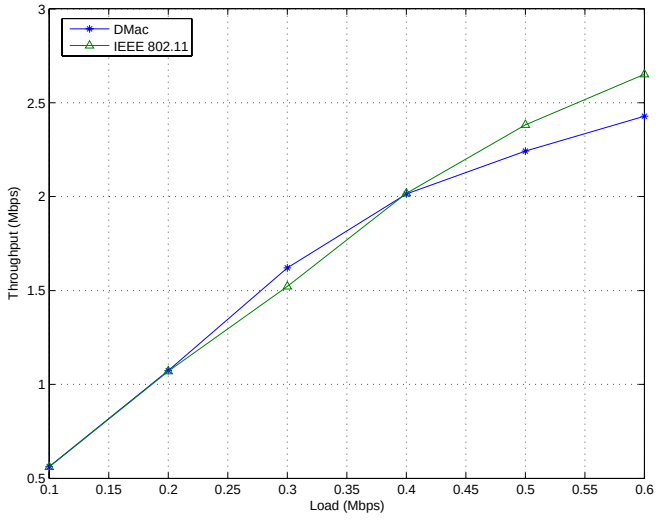

Figure 7.19: Total throughput for random scenario 1.

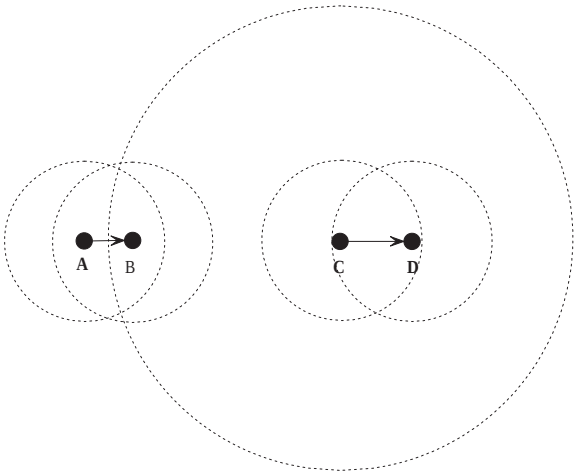

Figure 7.20: Scenario for illustrating incomplete data channel reservation information.

the random fifty nodes scenarios, and the best bandwidth partition is used. The results of throughput vs. load are shown in Figure 7.17, Figure 7.18 and Figure 7.19. The results show that the total throughput of using DMAC is comparable to that using IEEE 802.11, especially for the two cases that do not have frequency reuse concern, shown in Figure 7.17 and Figure 7.18.

However, the total throughput using DMAC is slightly lower than that using IEEE 802.11 when the traffic load is high. This is due to the extra bandwidth cost used for the control packets, and incomplete reservation information from the data channel.

Figure 7.20 provides one scenario that shows the data channel incomplete reservation information problem. The data channel incomplete reservation information occurs when one sender is within the other receiver's interference range, but the sender cannot overhear the CTS sent by that receiver or the RTS sent by that receiver's partner (the sender). In the example shown in Figure 7.20, C cannot overhear the information sent by either A or B. Thus, C has no idea about the correct data channel reservation. However, the data sent by the sender (C) interferes with the other receiver (B) if they transmit simultaneously, which results in data collisions especially in high traffic situations.

7.5 Discussion

In this section, we focus on the design issues of DMAC, and we discuss the possibility of supporting more control functions.

Design issues

To design a dual channel MAC protocol with a control channel and a data channel, the control channel should be relatively small, which results in a long duration for exchanging RTS and CTS packets. This brings a high probability of collision. Hence, extra efforts are needed to prevent the high collision probability for control packets. We incorporate the use of an out-of-band busy tone sent by the receiver to address this problem. The out-of-band busy tone brings extra benefit in improving fairness with the constraint that the busy tone can reach the sender that might suffer unfairness due to the information incompleteness problem. However, the out-of-band busy signal scheme requires extra hardware support, although the out-of-band signal is only a simple tone signal.

To improve the channel usage, pre-scheduling must be incorporated into the design. Pre-scheduling for the data channel requires that the control packets carry extra information for the negotiation procedure. This brings further burden for the control channel, since the control channel is relatively small.

Feasibility of supporting more control functions in the control channel

Our initial motivation to design DMAC was to have more flexibility to control the signaling channel. One idea was to add a cancelation scheduling scheme by taking advantage of pre-scheduling. Using the pre-scheduling scheme, a high priority packet that arrived later can be scheduled immediately by canceling the waiting schedule. Therefore, the important data can get higher priority. However, with the comprehensive study of the channel partitioning and control channel collisions, we found it is too expensive to complete this. A cancelation scheme at least requires one pair of RTS/CTS for the high priority packet, one cancelation control packet for the scheduled sender and one cancelation control packet for the scheduler receiver to inform their neighbors to release the previous reservation. Hence, we would expect at least four packets whose length is larger than regular RTS/CTS packets to achieve this prioritization of packets. This makes the control channel extremely congested, or it requires a large percentage of control channel bandwidth. Either way, the result is a poor throughput achievable in

the data channel.

The gain from using pre-scheduling with a cancelation scheme to support QoS and the loss of data performance cannot balance in this case. Therefore, we decided not to pursue this idea.

7.6 Conclusions and Future Work

In this chapter, we described DMAC, a dual channel MAC protocol that separates the control and data channels. From this study, we found that:

- DMAC can improve fairness under the situations of information incompleteness and information unfairness.
- There is no optimal channel partition for supporting all traffic conditions. We have provided general guidlines based on a statistical model to offer a reasonable partition.
- The achievable throughput heavily relies on the channel partition. A total throughput comparable to the single channel case could be achieved when using a best channel partitioning for a particular data traffic pattern.

Our future work will focus on providing further QoS support in the MAC layer for MANETs. Instead of partitioning a channel into two channels to obtain an exclusive channel for data packets, we will investigate the feasibility of taking advantage of spare channels in the IEEE 802.11 reserved channels [69].

Chapter 8

Conclusions and Future Work

With over a decade of research efforts aimed at improving their performance, mobile ad hoc networks have developed from an initial concept to a mature field with numerous supporting protocols. Although many fundamental issues have been studied, MANETs are still in the development phase due to their design complexity. Therefore, a large research space remains open for further exploration. Among the many challenges for MANETs, offering QoS is of particular interest due to the popularity of real-time applications. While there has been some research on protocols to support QoS in MANETs, there are still many unsolved problems in this domain. The work described in this dissertation has demonstrated the advantages of sharing information among the layers of the traditional OSI protocol stack, providing a cross-layer design to support QoS in MANETs.

8.1 Conclusions

In chapter 3, QoS in MANETs is discussed, and a cross-layer network architecture for supporting QoS in MANETs is presented. This cross-layer architecture uses information sharing rather than layer fusion. Each layer's functions and features are defined, and the necessary cross-layer interactions are presented. A simple QoS network model is implemented, and simulation results show that incorporating QoS support into MANETs is feasible and provides a large improvement for video frame delivery.

In chapter 4, the congestion behavior in MANETs is investigated. Analysis of the

cause of congestion is studied. UDPC, a feedback based congestion control transport layer protocol, is proposed correspondingly for congestion avoidance. The transport layer protocol acknowledges the network status to the application layer, so that the application can best optimize its performance. The simulation results of using two different schemes, UDP-MIMD and UDP-AIMD, show that overall network capacity is improved and average energy/packet/hop is reduced using congestion control in UDP. UDPC shows the advantage of providing information sharing between the transport and application layers.

In chapter 5, a survey of QoS-aware routing protocols in MANETs is presented and then a novel QoS routing protocol based on bandwidth estimation is proposed in chapter 6. This cross-layer approach includes an adaptive feedback scheme and an admission scheme to provide information about the current network status to the application. At the same time, the routing layer obtains the necessary traffic information from the MAC layer to assist in bandwidth estimation. Two different methods of bandwidth estimation –"Listen" and "Hello"– have been compared in detail using different topologies and different weight factors.

In chapter 7, a study of a dual channel MAC protocol is performed. By separating the control and data channels and using an out-of-band busy tone, the unfairness, which is an important metric for supporting QoS, is improved for both the information unfairness and information incompleteness cases. A theoretical analysis is conducted to investigate a reasonable partitioning of the bandwidth between the control and data channels. This study shows that separating the control channel and the data channel brings added complexity to support even the basic MAC functions. Therefore, adding more control features by using more handshaking in the control channel might not be a good idea. The gain for supporting more QoS features and the loss for poor total throughput and design complexity may not balance appropriately.

8.2 Future Work

There is much work to be done to finally realize stable support for QoS in MANETs. Future work in this area can be divided into two parts – further detailed work on remaining issues to be solved layer by layer and combining the layers into a complete

system. The possible work to be done is as follows:

Transport Layer As real-time applications generally use UDP for the underlying transport protocol, in this dissertation only UDP's performance is investigated. However, in order to not lose generality for supporting different types of traffic, it is important to also consider TCP's performance for providing QoS. In addition, TCP's overall performance is poor in MANETs, especially its stability. We believe that obtaining network status information from the lower layers and actively adjusting TCP's window instead of only passively adjusting its window according to the acknowledgements from the destination could help improve TCP's overall performance in MANETs.

Routing Layer The QoS-aware routing based on bandwidth estimation does not incorporate any route break prediction. Therefore, there is a performance degradation as the mobility of the nodes increases. A route break predict scheme could aid in the quick response of the protocol to route breaks.

MAC Layer MAC protocol design is a very challenging task in MANETs. DMAC only partially solves the QoS problem, which is far from sufficient. DMAC does not currently offer scheduling based on different priority level. Therefore, designing a MAC protocol that fully supports QoS is the subject of future work.

Architecture The QoS architecture proposed in this dissertation is a general model. The specified details for implementing all layers, the possible interactions and the overall performance are needed for providing QoS in MANETs. Furthermore, the implementation of all these protocols into a hardware solution is the final goal for enabling a real MANET to support QoS.

Bibliography

[1] A. Tanenbaum, *Computer Networks*. Prentice Hall, 1996, ch. 1.

[2] T. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 2002, ch. 10.

[3] IEEE Computer Society LAN MAN Standards Committee, "WirelessLAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE std. 802.11-1997, 1997.

[4] J. Haartsen and S. Mattisson, "BLUETOOTH—A New Low-Power Radio Interface Providing Short-Range Connectivity," in *Proceedings of the IEEE Special Issue on Low-Power RF Systems*, 2000.

[5] S. Roy, J. R. Foerster, V. S. Somyazulu, and D. G. Leeper, "Ultrawideband Radio Design: the Promise of High-speed, Short-range Wireless Connectivity," in *Proceedings of the IEEE*, vol. 92, no. 2, Feburary 2004, pp. 295–311.

[6] C. S. R. Murthy and B. S. Manoj, *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall, 2004.

[7] *IEEE Standard for Local and Metropolitan Area Networks*, IEEE Computer Society and the IEEE Microwave Theory and Techniques Socitety Std. 802.16, 2004.

[8] T. Sikora and L. Chiariglione, "The MPEG-4 Video standard and Its Potential for Future Multimedia Applications," in *Proc. IEEE ISCAS Conf.*, 1997.

[9] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.164/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems*, July 2003.

[10] V. Bharghavan, A. J. Demers, S. Shenker, and L. Zhang, “MACAW: A Media Access Protocol for Wireless LAN’s,” in *SIGCOMM*, 1994, pp. 212–225.

[11] Z. Fang, B. Bensaou, and Y. Wang, “Performance Evaluation of a Fair Backoff Algorithm for IEEE DFWMAC,” *Mobile Computing and Networking*, pp. 52–58, 2003.

[12] V. Kanodia, C. Li, A. Sabharwal, B. Sandeghi, and E. Knightly, “Ordered Packet Scheduling in Wireless Ad Hoc Networks: Mechanisms and Peformance Analysis,” in *Proceedings of ACM MOBIHOC*, September 2002.

[13] N. H. Vaidya, P. Bahl, and S. Gupta, “Distributed Fair Scheduling in a Wireless LAN,” in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing Table of Contents*, 2000, pp. 167–178.

[14] H. Zhai, J. Wang, Y. Fang, and D. Wu, “A Dual-Channel MAC Protocol for Mobile Ad Hoc Networks,” in *Proceedings of Global Telecommunications Conference Workshops*, November 2004.

[15] P. Karn, “MACA-A New Channel Access Method for Packet Radio,” in *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, 1990.

[16] C. L. Fullmer and J. J. Garcia-Luna-Aceves, “Solutions to Hidden Terminal Problems in Wireless Networks,” in *Pro. ACM SIGCOMM’97*, September 1997.

[17] Z. J. Haas and J. Deng, “Dual Busy Tone Multiple Access (DBTMA) - A Multiple Access Control for Ad Hoc Networks,” *IEEE Transactions on Communications*, vol. 50, no. 6, pp. 975–985, June 2002.

[18] “Ad Hoc on Demand Distance Vector (AODV) Routing,” http://www.ietf.org/internet-drafts/draft-ietf-manet-adov.03.txt June 1999. IETF Internet Draft (work in progress), 1999.

[19] L. Qian, D. L. Jones, K. Ramchandran, and S. Appadwedula, “A General Joint Source-channel Matching Method for Error Resilient Wireless Video Transmission,” in *Data Compression Conference*, 1999, pp. 414–423.

[20] *Draft Supplement to Standard for Telecommunications and Information Exchange Between Systems – LAN/MAN Specific Requirements – Part II: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS)*, IEEE 802.11 WG Std. IEEE 802.11e/D2.0, November 2001.

[21] J. Sobrihho and A. S. Krishna, "Quality-of-service in Ad Hoc Carrier Sense Multiple Access Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1353–1368, August 1999.

[22] C. R. Lin and M. Gerla, "Asynchronous Multimedia Multi-hop Wireless Networks," in *IEEE INFOCOM*, 1997.

[23] P. Sinha, R. Sivakumar, and V. Bharghavan, "CEDAR: a Core-Extraction Distributed Ad hoc Routing Algorithm," in *IEEE Infocom*, New York, NY, March 1999.

[24] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in Ad-Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, 1999.

[25] Y. Ge, T. Kunz, and L. Lamont, "Quality of Service Routing in Ad-Hoc Networks Using OLSR," in *Proceeding of the 36th Hawii International Conference on System Science*, 2003.

[26] Q. Xue and A. Ganz, "Ad hoc QoS On-demand Routing (AQOR) in Mobile Ad hoc Networks," *Journal of Parallel and Distributed Computing*, Feburary 2003.

[27] Y. Hwang, H. Lee, and P. Varshney, "An Adaptive QoS Routing Protocol with Dispersity for Ad-hoc Networks," in *Proceedings of the 36th Hawaii International Conference on System Sciences*, 2003.

[28] S. Ge, S. K. Das, H. Wu, and C. Qiao, "Trigger-Based Distributed QoS Routing in Mobile Ad Hoc Networks," in *ACM SIGMOBILE Mobile Computing and Communications Review*, no. 3, 2002.

[29] C. R. Lin, “On-Demand QoS Routing in Multihop Mobile Networks,” in *INFOCOM*, 2001, pp. 1735–1744.

[30] C. R. Lin and J. S. Liu, “QoS Routing in Ad Hoc Wireless Networks,” *IEEE Journal on Selected Areas in Communications*, pp. 1426–1438, August 1999.

[31] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Praksah, “A Feedback-based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks,” *IEEE Personal Communications Magazine*, vol. 8, no. 1, pp. 34–39, Feburary 2001.

[32] G. Holland and N. Vaidya, “Analysis of TCP Performance over Mobile Ad Hoc Networks,” *Wireless Networks*, vol. 8, no. 2/3, pp. 275–288, 2002.

[33] H. Zimmermann, “OSI Reference Model-The ISO Model of Architecture for Open Systems Interconnection,” *IEEE Transaction on Communications*, vol. 28, no. 4, April 1980.

[34] S. Lee, G. Ahn, X. Zhang, and A. T. Campbell, “Insignia: An ip-based quality of service framework for mobile ah hoc networks,” *Journal of Parallel and Distributed Computing*, vol. 60, pp. 374–406, 2002.

[35] K. Chen, S. H. Shah, and K. Nahrstedt, “Cross Layer Design for Data Accessibility in Mobile Ad Hoc Networks,” *Journal of Wireless Communications*, vol. 21, pp. 49–75, 2002.

[36] S. Chen, “Routing support for providing guaranteed end-to-end quality-of-service,” Ph.D. dissertation, Univ. of IL at Urbana-Champaign, 1999.

[37] G. Ahn, A. Campbell, A. Veres, and L.Sun, “SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks,” in *Proceedings of IEEE INFOCOM*, 2002.

[38] H. Xiao, K. Chua, and W. Seah, “A Quality of Service Model for Ad Hoc Wireless Networks,” in *Handbook of Ad Hoc Wireless Networks*. FL USA: CRC, 2002.

[39] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-Efficient Communication Protocol for Wireless Microsensor Networks,” in *Proceedings of the 33rd International Conference on System Sciences (HICSS ’00)*, January 2002.

[40] W. Heinzelman, J. Kulik, and H. Balakrishman, “Adaptive Protocols for Information Dissemination in Wireless Sensor Network,” in *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom ’99)*, August 1999, pp. 174–185.

[41] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks,” in *Mobile Computing and Networking*, 2000, pp. 56–67.

[42] K. Kulkarni and G. Minden, “Composing Protocol Frameworks for Active Wireless Networks,” *IEEE Communications Magazine*, March 2000.

[43] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, “RSVP: A New Resource ReSerVation Protocol,” in *IEEE Network*, September 1993.

[44] W. Heinzelman, “An application-specific protocol architecture for wireless microsensor networks,” dissertation, MIT, Cambridge, 2000.

[45] J. Chen, T. Lv, and H. Zheng, “Cross-layer Design for QoS Wireless Communications,” in *Proceedings of the 2004 International Symposium on Circuits and Systems*, vol. 2, May 2004, pp. 23–26.

[46] T. Yoo, E. Setton, X. Zhu, A. Goldsmith, and B. Girod, “Cross-layer design for video streaming over wireless ad hoc networks,” in *2004 IEEE 6th Workshop on Multimedia Signal Processing*, October 2004, pp. 99–102.

[47] H. Balakrishnan, V. Padmanabhan, S. Sesha, and R. Katz, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Link,” *IEEE/ACM Transactions on Networking*, December 1997.

[48] J. Inouye, S. Cen, C. Pu, and J. Walpole, “System Support for Mobile Multimedia Applications,” in *Proceedings of IEEE 7th Intenational Workshop on Network and Operating System Support for Digital Audio and Video*, May 1997, pp. 135–146.

[49] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, “The Design and Implementation of an Intentional Naming System,” in *Proceedings of 17th ACM SOSP*, December 1999.

[50] F. Wang and Y. Zhang, “Improving TCP Peformance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response,” in *Proceedings of the 2002 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc’02)*, June 2002.

[51] J. Liu and S. Singh, “ATCP: TCP for mobile ad hoc networks,” *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300–1310, July 2001.

[52] T. D. Dyer and R. V. Boppana, “A comparison of TCP performance over three routing protocols for mobile ad hoc networks,” in *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking Computing (MobiHoc’01)*, October 2001.

[53] G. Holland and N. Vaidya, “Analysis of TCP performance over mobile ad hoc networks,” in *Proceedings of ACM MobiCom’99*, August 1999.

[54] C. Zhu and M. Corson, “QoS Routing for Mobile Ad Hoc Networks,” in *IEEE Inforcom*, 2002.

[55] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot, and T. Clauseen, “Optimized Link State Routing Protocol draft-ietf-manet-olsr-05.txt,” INTERNET-DRAFT, IETF MANET Working Group.

[56] H. Xiao, K. C. Chua, and W. S. A. Lo, “On Service Prioritization in Mobile Ad-hoc Networks,” in *Proceeding of ICC*, 2001.

[57] B.-G. Chun and M. Baker, “Evaluation of Packet Scheduling Algorithm in Mobile Ad Hoc Networks,” in *Mobile Computing and Communications Review*, 2002.

[58] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, and L. Stibor, “IEEE 802.11e Wireless LAN for Quality of Service,” in *European Wireless*, 2002.

[59] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, “Distributed Priority Scheduling and Medium Access in Ad Hoc Networks,” in *Wireless Networks*, vol. 8, 2002, pp. 455–466.

[60] Y. Yang and R. Kravets, “Contention-Aware Admission Control for Ad Hoc Networks,” Technical Report 2003-2337, University of Illinois at Urbana-Champaign, 2003.

[61] Y. Xiao and H. Li, “Evaluation of Distributed Admission Control for the IEEE 802.11e EDCA,” *IEEE Radio Communications*, September 2004.

[62] A. Banchs and X. Perez, “Distributed Weighted Fair Queuing in 802.11 Wireless LAN,” in *IEEE ICC02*, 2002.

[63] A. Bachs and X. Perez, “Providing Throughput Guarantee in IEEE 802.11 Wireless LAN,” in *IEEE WCNC’02*, 2002.

[64] W. Pattara-Atikom, S. Banerjee, and P. Krishnamurthy, “Starvation Prevention and Quality of Service in WIreless LANs,” in *Proc. 5th Int. Symp. Wireless Pers. Multimedia Commun*, 2002.

[65] B. Bensaou, Y. Wang, and C. C. Ko, “Fair Media Access in 802.11 based Wireless Ad-Hoc Networks,” in *IEEE/ACM MobiHOC*, August 2000.

[66] Y. Wang and B. Bensaou, “Achieving Fairness in IEEE 802.11 DFWMAC with Variable Packet Lengths,” in *IEEE Globlecom*, November 2001.

[67] N. H. Vaidya, “Fair Scheduling in Broadcast Environments,” Microsoft Research, Tech. Rep MSR-TR-99-61, Dec 1999.

[68] S. J. Golestani, “A Self-Clocked Fair Queueing Scheme for Broadband Applications,” in *IEEE INFOCOM’94*, 1994.

[69] J. Shi, T. Salonidis, and E. W. Knightly, “Starvation Mitigation through Multi-channel Coordination in CSMA Multi-hop Wireless Networks,” in *Proceedings of the Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2006.

[70] L. Chen and W. Heinzelman, “End-to-End Congestion Control for Best-effort Transmission,” in *Proceedings of Wireless Networking Symposium*, October 2003.

[71] L. Chen and W. Heinzelman, "QoS-aware Routing Based on Bandwidth Estimation in Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications, Special Issue on Wireless Ad Hoc Networks*, vol. 23, no. 3, 2005.

[72] J. Li, C. Blake, D. D. Couto, H. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks," in *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking (MobiCom '01)*, 2001.

[73] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.

[74] P. Gevros, J. Crowcroft, P. Kirstein, and S. Bhatti, "Congestion control mechanisms and the best effort service model," *IEEE Network*, vol. 15, no. 3, pp. 16–25, 2001.

[75] S. Shenker, "Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines," in *SIGCOMM Symposium on Communications Architectures and Protocols*, August 1994, pp. 47–57.

[76] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An Integrated Congestion Management Architecture for Internet Host," in *Proceedings of ACM SIGCOMM*, 1999.

[77] U. Berkeley/LNBL/ISI, "The ns-2 network simulator with the cmu mobility extensions," http://www.isi.edu/nsnam/ns/, 2002.

[78] S. Chakrabarti, "QoS Issues in Ad Hoc Wireless Networks," *IEEE Communications Magazine*, pp. 142–148, February 2001.

[79] C. Perkins and E. Royer, "Ad hoc on-demand distance vector routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999, pp. 90–100.

[80] D. Johnson and D. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic Publishers, 1996, vol. 353.

[81] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proceeding of INFOCOM*, April 1997.

[82] C. Perkins and P. Bhagwat, "Highly dynamic destination sequenced distance vector routing (dsdv) for mobile computers," in *Proc. ACM SIGCOMM'94*, October 1994.

[83] Q. Ma and P. Steenkiste, "On Path Selection for Traffic with Bandwidth Guarantees," in *Proceedings of Fifth IEEE International Conference on Network Protocols*, 1997.

[84] D. D. Clark, "The Design Philosophy of the DARPA Internet Protocols," in *Proc. ACM SIGCOMM'88*, August 1988.

[85] C. Lin and J. Liu, "QoS Routing in Ad Hoc Wireless," in *IEEE Journal on Selected Areas in Communications*, August 1999, pp. 1426–1438.

[86] K. Sanzgiri, I. Chakeres, and E. Belding-Royer, "Determining Intra-Flow Contention along Multihop Paths in Wireless Networks," in *Proceedings of Broadnets 2004 Wireless Networking Symposium*, San Jose, CA, October 2004.

[87] T. Goff, N. B. Abu-Ghazaleh, D. S. Phatak, and R. Kahvecioglu, "Preemptive Maintenance Routing in Ad Hoc Networks," in *Mobicom*, 2001.

[88] P. Mohapatra, J. Li, and C. Gui, "Qos in mobile ad hoc networks," *Special Issue on QoS in Next-Generation Wireless Multimedia Communications Systems in IEEE Wireless Communications Magazine*, June 2003.

[89] H. Zhu, M. Li, I. Chlamtac, and B. Prabhakaran., "A survey of quality of service in ieee 802.11 networks," *IEEE Wireless Communications*, August 2004.

[90] K. Wu and J. Harms, "Qos support in mobile ad hoc networks," *Crossing Boundaries - an interdisciplinary journal*, vol. 1, no. 1, 2001.

[91] L. Chen and W. Heinzelman, "Network Architecture to Support QoS in Mobile Ad Hoc Networks," in *Proceedings of International Conference on Multimedia and Expo*, 2004.

[92] Y. Zhang, "Very low bitrate video coding standards," in *Proc. Vis. Comm. Image Proc.*, vol. 2501. SPIE, May 1995, pp. 1016–1023.

[93] G. Coete, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video coding at low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, November 1998.

[94] S. Servetto, K. Ramchandran, V. Vaishampayan, and K. Nahrstedt, "Multiple-Description wavelet based image coding," in *Proceedings of the IEEE International Conference on Image Processing*, 1998, pp. 659–663.

[95] Y. Hwang and P. Varshney, "An Adaptive Routing Protocol for Ad-hoc Netowrks using Multiple Disjoint Path," in *VTC*, May, 2003.

[96] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks," *IEEE Communications Magazine*, pp. 130–137, June 2001.

[97] J. Garcia-Luna-Aceves and C. L. Fullmer, "Floor Acquisition Multiple Access (FAMA) in Single-channel Wireless Networks," *ACM Mobile Networks and Applications Journal (MONET)*, vol. 4, no. 3, pp. 157–174, 1999.

[98] Y. Li, H. Wu, N. Tzeng, D. Perkins, and M. Bayoumi, "MAC-SCC: a Medium Access Control Protocol with Separate Control Channel for Reconfigurable Multihop Wireless Networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 7, pp. 1805–1817, 2006.