2004

Congestion-aware cross-layer design for wireless ad hoc networks

Ning, Yang
University of South Florida

Congestion-Aware Cross-Layer Design for Wireless Ad Hoc Networks

by

Ning Yang

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Ravi Sankar, Ph.D.
Miguel Labrador, Ph.D.
Arthur David Snider, Ph.D.

Date of Approval:
July 8, 2004

Keywords: interlayer, ns-2 simulation, rate adaptation, congestion estimation

Table of Contents

List of Tables

List of Figures

Congestion-Aware Cross-Layer Design for Wireless Ad Hoc Networks

Ning Yang

ABSTRACT

Ad hoc networks have emerged recently as an important trend of future wireless systems. The evolving wireless networks are seriously challenging the traditional OSI layered design. In order to provide high capacity wireless access and support new multimedia network, the various OSI layers and network functions should be considered together while designing the network.

In this thesis, we briefly discuss performance optimization challenges of ad hoc networks and cross-layer design. Ad hoc wireless networks were implemented by using Network Simulator *NS-2* and the wireless physical, data link, dynamic source routing (DSR) routing protocol models have been included in the simulation. Simulations show that the performance begins to drop at the moderate offered load due to congestion. In addition, the mobility and fading cause the route failures and packet loss in wireless environment.

To improve the performance for wireless networks, we implemented a congestion-aware cross-layer design in *NS-2*. The MAC layer adaptively selects a transmission data rate based on the channel signal strength information from physical layer and congestion information from network layer. The MAC layer utilization gathered at MAC layer is sent to DSR as a congestion aware routing metric for optimal route discovery. We modified the source codes of 802.11 MAC layer and DSR protocol. The simulations show that rate adaptation in MAC layer improves the network performance in terms of throughput, delivery ratio, and end-to-end delay; using congestion information from MAC layer in routing discovery improves the performance of the network benefited from overall network load balance.

Chapter One

Introduction

1.1   Ad Hoc Network

An ad hoc network is a collection of wireless mobile nodes dynamically forming a temporary network without the aid of any predetermined network infrastructure or centralized administration. In the past, ad hoc networks [1] have been primary used for the communications at battlefields and disaster arena, where a centralized infrastructure is expensive, or inconvenient, or even impossible. For instance, the sensor networks based on ad hoc multi-hop networking are applied in environmental monitoring, surveillance, and security. Now, as the novel radio technologies are growing, e.g., Bluetooth, the applications of ad hoc networking extend to commercial sectors through the interaction among various portable devices such as cellular phones, laptops, and PDAs. The trends of next generation wireless systems, characterized by the convergence of fixed and mobile networks and realization of the seamless and ubiquitous communications, are moving to ad hoc networking.

The nature of temporary links and the mobility of the nodes together with wireless transmission effect on attenuation, interference, and multipath propagation, bring some inherent issues of ad hoc networks. The links of the ad hoc networks are dynamic in a sense that they are likely to break or change with the movement of the nodes. As the topology changes, the route must be updated immediately by sending a control message. Those unique features to ad hoc networks results in lots of control overhead for route discovery and maintenance. This is highly unacceptable in bandwidth-constrained ad hoc networks. Usually these devices have limited computing resources and severe energy constraints. Routing strategies along with mobility management and resource allocation become big challenges to network designers and service providers. The network protocols of ad hoc networks must consider the routing efficiency as well

as security, minimizing various link latency, and power efficiency. In addition, the balance between network coverage, capacity, delay, and power consumption is needed.

There are a lot of research investigations being carried out in the performance optimization of ad hoc networks concerning these issues. Such as efficient flooding [24], load aware routing [25], and Selective Intermediate Nodes (SIN) algorithm [26], which improve the scalability and throughout of the on-demand routing protocols by avoiding unnecessary routing overhead. However, most of the research works are based on optimization at individual layer. Optimizing a particular layer might improve the performance of that layer locally but might produce non-intuitive side effects that will degrade the overall system performance [4]. Thus, the idea of cross-layer or interlayer networking is proposed.

1.2   Motivations for Cross-Layer Design

Cross-layer or interlayer networking can be considered as one in which different layers of the network protocol stack inter-communicate the useful information so as to collectively achieve the desired vertical optimization goal.

For the sake of QoS (quality of service) requirements varying with applications, the network or higher layers function should directly base on the information from the lower physical and MAC layers [1]. It blurs the line between layers and leads the optimization of cross layer functionalities. Currently ad hoc routing protocols work mainly on the network layer. It guarantees the independency of the network layer. However each layer needs to do redundant processing and unnecessary packet exchange to get information that is easily available to other layers. This increases control signals resulting in wastage of bandwidth, packet collision, etc. By using interlayer interaction, different layers can share locally available information. This will result in substantial amount of performance improvement.

A working group of the Internet Engineering Task Force (IETF) has been studying the interlayer interactions and performance in Mobile Ad Hoc Network (MANET). In their working Internet draft document [5], they summarized the interlayer interaction metrics and the benefits of such information exchange between lower layers, network layer, and transport layer. For example, the signal strength or SNR (signal to noise ratio) information from physical layer, and interference or

congestion awareness from link layer, may be used for routing selection at network layer and TCP window size adjustment or flow control at transport layer.

The paper [8] illustrates how the inter-layer methods – channel-state dependent techniques can improve the network throughput and survey some current related works.

1.3   Research Objectives

Rate adaptation is the process of dynamically changing data transmission rate based on the channel quality estimation. Rate selection is sensitive to accuracy and delay incurring in the feedback of the channel quality estimates. If the channel is changing faster than it can be estimated and fed back to the transmitter, adaptive techniques will perform poorly. The paper [16] observes that channel quality information is more efficiently and timely acquired at the receiver than at the sender. A new cross-layer design [6] shows the significant performance enhancement by exploiting the lower layer channel SNR information for rate adaptation at MAC layer and interference estimation of MAC layer for optimal routing selection at network layer.

Another rate control mechanism [23] controls the rate of route through traffic depending on traffic load on the channel. To meter its own rate, each node needs global knowledge of the total number of nodes in the entire network, which is impractical in the spontaneous ad hoc network. Woo and Culler [23] propose an implicit adaptive scheme to approximate it. In our approach, by taking the advantage of sharing interlayer information, the node in MAC layer can easily using the queue length from the network layer as the congestion metric for rate control. An adaptive strategy in MAC layer multicast [27] uses the queue length at the sender as one of the metrics for transmission decision. However, for the purpose of congestion control, considering the queue length only at the sender is not enough. At the sender side, when congestion occurs (the queue length increased above a threshold), it tries to quickly send out the packets and so it prefers to select high data rate. But at the receiver side, it is prone to reduce the data rate to avoid more incoming packets waiting in its already congested queue. We must balance the data rate between the two sides to avoid high level congestion occurring at any of them. So that the overall network performance will not degrade. So in our congestion aware based rate adaptation approach, we select the data rate based on the queue lengths of both sender and receiver.

3
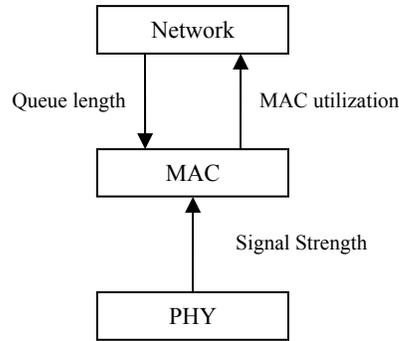
Figure 1.   Illustration of Cross Layer Communications in the Network Stack

In ad hoc networks, some areas of the network have higher data forwarding loads than other areas. Specially, the nodes near the center of the network carry high loads when the routing protocol uses shortest path route strategy. It can make certain areas prone to congestion, thus decreasing the throughput of the overall network. The paper [7] demonstrates the performance improvement resulting from the use of congestion information from MAC layer in network and transport layer. In network layer, the congestion information can be used to allow the routing protocols avoid selecting route through the congestion area. Within transport layer, the congestion level measurement may be applicable for several modifications of TCP, based on setting the Explicit Congestion Notification (ECN) [19] bits in a packet's IP header or using Wireless ECN (W-ECN) [20] or Explicit Transport Error Notification (ETEN) [21].

Motivated by these research ideas, we implemented a congestion-aware cross-layer interaction system that consists of rate adaptation and congestion aware optimizations to improve the overall performance in terms of throughput, packet delivery, and latency. In rate adaptation scheme, the signal strength information from physical layer is used in MAC layer; or the queue length information from network layer is used as the adaptive metric. By congestion aware scheme, the congestion information – MAC layer utilization from MAC layer is used in network layer. We illustrate our cross-layer system in the flow diagram in Figure 1.

1.4   Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we introduce the channel model of the system and some background of protocols, such as IEEE 802.11 MAC, DSR which we modified for interlayer implementation and DSDV. Chapter 3 provides the modification of those protocols

in the cross-layer system. Chapter 4 presents the simulations and the results in *NS-2* [13]. Finally, in Chapter 5 we give our conclusion and discuss future work.

Chapter Two

Background

In our cross-layer scheme, the physical and MAC layers knowledge of the wireless medium around the node is shared with the network layer. In this chapter, we briefly describe the channel model that is used in our simulations, the MAC layer, and network layer routing protocols.

2.1 Channel Model

In mobile radio channel, fading is the result of interference between two or more paths of a transmitted signal arriving at a receiver. When the time dispersion is small compared to the transmitted symbol period, the received signal undergoes flat fading. Flat fading is often modeled as a Rayleigh or Rician distributed random processes. It's well known that the sum of quadrature Gaussian noise signals obeys a Rayleigh distribution.

Our channel model is based on an efficient simulation of a small scale Rayleigh fading [14], which is incorporated in our simulator *NS-2*. The physical radio parameters of the channel, such as the antenna gain, transmit power, and receiver sensitivity, were chosen to approach the 914 MHz Lucent WaveLAN direct sequence spread spectrum (DSSS) radio.

2.2 Overview of the IEEE 802.11 DCF MAC Protocol

As the lower sublayer of the Data Link layer, the MAC (Medium Access Control) layer provides access control methods of the channel medium for data transfer and establish the network for efficiently sharing the resources. Current wireless MAC protocols can be classified based on random or scheduled access, centralized or distributed control, single or multiple channels, and using TDMA or FDMA or CDMA or hybrid combination. In our research we use the IEEE 802.11 Distributed Coordination Function (DCF) MAC protocol [9], since it has been adopted as

a wireless LAN standard and is widely used in both traditional wireless systems and multi-hop wireless ad hoc networking research.

The architecture of 802.11 MAC layer includes two separate coexisting coordination functions that provide support for asynchronous data transfer and optionally distributed time-bounded services. Asynchronous data transfer is insensitive to time delays and is supported by the *Distributed Coordination Function* (DCF). Delay constrained traffic is implemented by the *Point Coordination Function* (PCF), an access method that is similar to a polling system and uses a point coordinator to determine which station has the right to transmit.

The basic medium access method is DCF, known as *carrier sense multiple access with collision avoidance* (CSMA/CA), is designed to reduce the collision probability among multiple stations sharing the medium. The idea behind this paradigm is to reserve transmission channel at the originator by carrier sensing. CSMA/CA works by a "listen before talk scheme". This means that a station wishing to transmit must first sense the radio channel to determine if another station is transmitting. If the medium is not busy, the transmission may proceed. Otherwise, the station shall defer until the end of the current transmission. After deferral, or prior to attempting to transmit again immediately after a successful transmission, the station shall select a random backoff interval and shall decrement the backoff interval counter while the medium is idle. The period of time immediately following a busy medium is the highest probability of collisions occurring, especially under high utilization.

DCF describes two schemes used to reserve the medium and distribute the information. The default one is two-way handshaking Data/Acknowledge and optional one is four-way handshaking RTS/CTS (request-to-send/clear-to-send) prior to actual Data transmission/ACK.

The RTS and CTS frames contain a Duration/ID field that defines the period of time that the medium is to be reserved to transmit the actual data frame and the returning ACK frame. All nodes within the reception range of either the source, which transmits the RTS or the destination, which transmits the CTS shall learn of the medium reservation and may save it in their network allocate vector (NAV). Thus a node can be unable to receive from the source, yet still know from the destination about the impending use of the medium to transmit a data frame. The NAV maintains a prediction of future traffic on the medium based on duration information that is
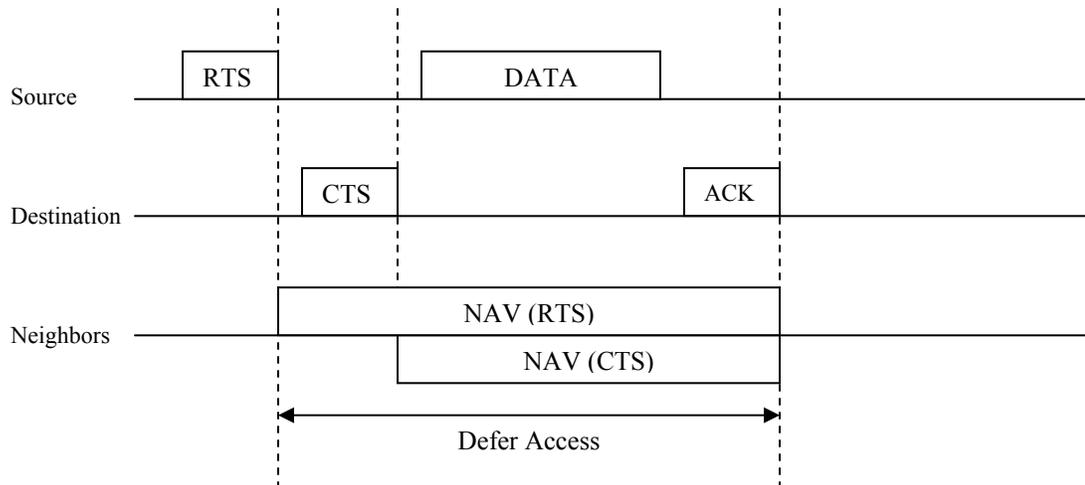
Figure 2.   Illustration of the 802.11 RTS/CTS/Data/ACK Access Mechanism

announced in RTS/CTS frames. A station receiving a valid frame shall update their NAV with the information received in the Duration/ID field, but only when the new NAV value is greater than the current NAV value and only when the frame is not addressed to this station. Figure 2 illustrates the 802.11 RTS/CTS/Data/ACK and NAV access mechanism.

The RTS/CTS exchange also performs fast collision inference and transmission path check. If the originating node does not receive CTS responding its RTS, it may repeat the process, after observing the other medium-use rules, more quickly than if the long data frame had been transmitted and a return ACK frame had not been detected.

The RTS/CTS mechanism cannot be used in broadcast and multicast packets because there are multiple destinations for the RTS, and thus potentially multiple concurrent senders of the CTS in response. The additional RTS and CTS frames add overhead inefficiency, so this mechanism is not always justified, especially for short data frames.

2.3 Overview of the Routing Protocol

Many different protocols have been proposed to solve the multi-hop ad hoc routing problem. Those routing protocols can be classified into two types: *Proactive* (or *Table-Driven*) routing protocols and *Reactive* (or *On-Demand*) routing protocols. In proactive routing protocols, nodes exchange routing information with each other before they are needed. So each node maintains an

8

up-to-date route table to all possible destinations. The On-Demand routing scheme is aimed to reduce the routing overhead by maintaining route tables based on the necessity. The price of the reduced control overhead is the route setup latency. In most on-demand routing protocols, a node establishes a route only when data packets need a new path. Many performance comparisons [2][3] done have shown that the on-demand algorithms perform better than the proactive algorithms.

2.3.1 Dynamic Source Routing – DSR

The well-known on-demand routing protocol Dynamic Source Routing (DSR) [10][11] [12], is introduced in this section. We use DSR in our study since previous simulation study [2] has shown that the protocol performs well among a number of MANET routing protocols.

DSR is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. It allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration.

The operation of DSR is based on *source routing*, with each packet to be routed carrying in its header the complete sequence of hops through which the packet past. The important advantage of source routing is that the intermediate nodes do not need to maintain up-to-date routing information for forwarding packets, since the routing information has been cached in the packets for their own future use. Unlike other protocols, DSR requires no periodic packets of any kind at any layer within the network. All aspects of the protocol operate entirely on-demand and lack periodic activity allows the number of overhead packets caused by DSR to scale automatically to only that needed to react to changes in the currently using routes. So, DSR has very low overhead and is able to react very quickly to changes in the network.

DSR protocol consists of two mechanisms:

2.3.1.1 Route Discovery
When a source node originates a new packet addressed to a destination node, it will search its *Route Cache* for a previous learned *source route*, a sequence of hops that the packet is following to pass on its way to that destination. If no route is found in the cache, the sender initializes Route

9

Discovery by broadcasting a *Route Request* (RREQ) packet, containing destination node, unique *request identification*, and a list (initial empty) of intermediate nodes traveled by this packet.

A node receiving the RREQ checks whether it has recently seen another RREQ from the same source node with the same request identification and target address, or whether its own address has already presented in the traveled node list of this RREQ. If either check is true, the node silently drops this packet. Otherwise, the node appends its address to the node list and forwards the packet.

When the RREQ packet reaches the target node or a node with a route to the target in its Route Cache, this node returns a *Route Reply* (RREP) to the source node with a copy of the node list from the RREQ. The RREP can be delivered to the initiator by simply reversing the node list, by using a route to the initiator in its own cache, or "piggybacking" the packet on a new Route Request to the original initiator. For MAC protocols such as IEEE 802.11 that require a bidirectional frame exchange, the reversed route of RREQ is the only way for returning RREP since this route is tested to be bidirectional before it can be used by Route Discovery initiator; this route reversal also avoids the overhead of a possible second Route Discovery.

When the initiator receives the RREP, it adds the source route in its route cache for use in sending subsequent packets to the target.

2.3.1.2   Route Maintenance

While using a source route to send a packet to the destination, each node transmitting the packet is responsible for confirming that it successfully reaches the next hop in the route. The node can provide the confirmation by the link-layer acknowledgement such as in IEEE 802.11, by a "passive acknowledgement" (in which, for example, a node A confirms receipt at the next hop B by overhearing B forwarding it to B's next hop C), or by additional network-layer acknowledgement. A acknowledge request may be retransmitted up to a maximum number of times. If no acknowledgement is received after maximum times request retransmission, the node assumes the link to the next hop is broken, then remove this route from its route cache, and return a Route Error packet to the source node. A node receive Route Error will remove the broken link from its cache.

An additional feature of Route Maintenance is *packet salvaging*. When an intermediate node forwarding a packet determines the next hop is unreachable over the link, in addition to sending back Route Error, it attempts to replace the original route with an alternate route to the destination if it can find any from its route cache. Then it forwards the packet to the next hop along the new route. To prevent an endless routing loop, the salvaged packet maintains a count to record the number of times it has been salvaged. It can no longer been salvaged if the count reaches a limit.

2.3.2   Destination-Sequenced Distance Vector – DSDV

Destination-Sequenced Distance Vector (DSDV) [17] is a typical proactive routing protocol. The key advantage of DSDV to other distance vector routing protocols is that it solves the looping problem by attaching sequence numbers to routing entries. Each node maintains a routing table listing the distance, in terms of number of hops, to all possible destinations. Each route entry is tagged a sequence number which is generated by the destination node. To keep up with dynamical varying topology, each node periodically broadcasts its full routing table and transmits updates immediately when significant changes happen. Each node increases its current even sequence number and advertises it along with the distance information. A node invalids or modifies its routing entry if it receives a routing entry to the same destination with greater sequence number or the same sequence number and better metric. When a node **A** determines its route to a destination **B** is broken, it advertises the route to **B** with an infinite metric number and an odd sequence number one greater than the current sequence number of this route. Other nodes receive this infinite metric immediately update their routing tables until they hear a new route to **B** with a higher sequence number.

DSDV is shown to have very high routing overhead compared with other on-demand routing protocols [2]. The Adaptive Distance Vector (ADV) [18] optimizes DSDV by reducing the unnecessary triggered updates based on well-selected *Trigger meter* and *Trigger threshold*.

Chapter Three

Congestion-Aware Cross-layer System

In this chapter, we describe our congestion-aware cross-layer system. It consists of two schemes: (i) rate adaptation (RA) adapts data rate in the MAC layer based on the channel estimation information from physical layer and congestion information from network layer; (ii) congestion aware (CA) exploits congestion information in network layer from MAC layer.

3.1   Rate Adaptation Scheme

3.1.1   Using Channel Information in Rate Adaptation Scheme

The rate adaptation scheme is implemented with a minor modification to the IEEE 802.11 MAC protocol. In this scheme, a source node sends a RTS packet before it transmits any data. When the destination node receives this RTS, it estimates the *signal strength* instead of SNR [6] of the transmission channel since signal strength is a more practical parameter we can get than SNR in *NS-2* simulation. The transmission power of each packet is known and the receive power (signal strength) can be estimated by using the channel model simulation [14]. Then the signal strength is mapped to a transmission data rate based on an efficient rate adaptation algorithm [15]. This transmission rate information is contained in CTS packet and sent back to the source node. The source node will then send data at this rate. Other neighbor nodes that hear the CTS will update the information of the channel busy period in their network allocation vector (NAV) and hold their transmission until current transmission is completed. To avoid the additional overhead to obtain rate information, we modified the IEEE 802.11 RTS/CTS frame formats, which are illustrated in Figure 3. The original *Duration* field is changed to *Rate* (high 4 bits) and *Length* (low 12 bits) field.

Our modification of the source code MAC 802.11 model in *NS-2* can be described as following.

Bits: 4   12

Rate	Length

Octets: 2   2   6   6   4

Frame Control	Duration	Dest Address	Source Address	FCS

(a) RTS control packet frames

Bits: 4   12

Rate	Length

Octets: 2   2   6   4

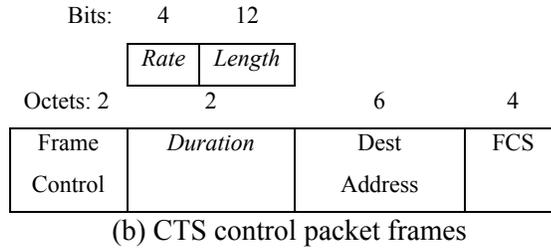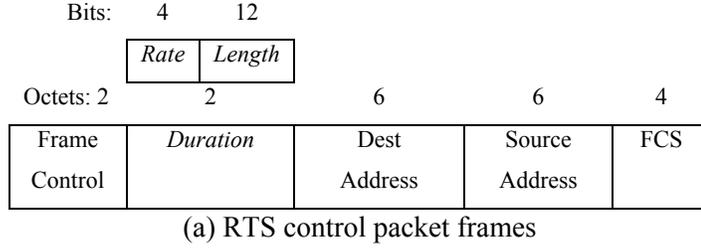| Frame Control	Duration	Dest Address	FCS

(b) CTS control packet frames

Figure 3.   RTS/CTS Frame Formats Used in RA Scheme

Source node sends RTS which includes the basic data rate and packet length information instead of transmission duration. The destination node selects an optimal data rate based on signal strength of the received RTS packet. The new selected data rate and packet length (got from RTS packet) replacing duration field in CTS is sent back to source. Then source will use the new data rate to transmit the following data packets. The neighbors that hear either the RTS or CTS calculate the duration from the data rate and packet length and set their NAV.

For the rate adaptation algorithm, we used a simple threshold based technique. In this scheme, the rate is chosen by comparing the channel quality estimation against a series of thresholds related to the available M-QAM modulation schemes [15]. Power-received indication (Pr), which is the

Table 1.   The Thresholds Used in Rate Adaptation

Rate b_i (Mbps)	Power-received indication Pr
0 (discard)	<1
1	$1 \leq Pr < 40$
2	$40 \leq Pr < 150$
2.5	$150 \leq Pr < 1e4$
3	$1e4 \leq Pr < 2e5$
3.5	$2e5 \leq Pr < 1e8$
4	$1e8 \leq Pr < 1e9$
6	$\geq 1e9$

receive signal strength over received threshold, presents the channel quality estimation in the simulation. The rate is given by $b_i = \log_2(M_i)$, where M_i is the constellation size of M-QAM. We set supported rate in the units of 500 kbit/s by adapting the IEEE 802.11 MAC standard. The thresholds we used in the simulation are listed in Table 1.

Accurate estimation of transmission rate usually reduces the probability of collision. Instead of using one instantaneous sample of the received signal strength of the current RTS packet [16], we estimated the signal strength by a simple moving average window (window length is 5 in our simulation) of the previous and current samples of RTS to smooth deep fading. This results in better channel usage efficiency and higher total throughput especially when the communicating nodes are moving.

3.1.2   Using Congestion Information in Rate Adaptation Scheme

For rate adaptation scheme based on congestion information, we use the queue length as the congestion metric. At the sender (who sends RTS) side, when congestion happens, MAC layer try to quickly send out the packets, so it prefers to select high data rate. But the receiver (who receives RTS) side is prone to reduce the data rate to avoid more incoming packets waiting in its already congested queue.

While the queue length *Qs* of the sender is greater than a preset *Threshold* (we use 40), the sender rate *Bs* is increased by a factor α ($0 < \alpha < 1$), otherwise it uses *defaultRate*. On the other hand if the receiver is congestion, the receiver rate *Bq* based on queue length *Qr* is decreased by a factor β ($0 < \beta < 1$). We also use the same power-received indication *Pr* defined in the last section as another rate selection metric at receiver to get rate *Bss*. Here we use $\log_e (Pr) - 1$ to approximate the rate selection table used in the last section. Adding *Bq* and *Bss* together is the final receiver selected rate. The sender and receiver select the transmission data rate using the following equations. Our support rates are all in the units of 500 kbit/s, which is set as *unitRate*. So our Bs and Br are all intergers and the *selectRate* is the final rate used in the RTS/CTS frame.
At the sender:

$$Bs = \mathrm{int}\left[\frac{(1 + \alpha * Qs / Threshold) * defaultRate}{unitRate}\right] \tag{1}$$

$$selectRate = Bs * unitRate \tag{2}$$

14

At the receiver:

$$Bq = Bs * (1 - \beta * Qr / Threshold) \qquad (3)$$

$$Bss = \log_e (Pr) - 1 \qquad (4)$$

$$Br = \text{int } [Bq + Bss] \qquad (5)$$

$$selectRate = Br * unitRate \qquad (6)$$

Note: int[x] means get the integer part of variable x.

Qs, Qr: queue length of the sender, receiver; defaultRate = 1 Mbit/s.

The RTS/CTS mechanism used here is the same with the signal strength based only rate adaptation scheme as we described in the last section.

3.2   Congestion Aware Scheme

3.2.1   Congestion Metrics

We measure the congestion information at a node by two congestion aware metrics [7]. One is average MAC layer utilization. The instantaneous MAC layer utilization is considered as 0 only when the medium around the node is available for beginning a transmission. We consider instantaneous MAC layer utilization at a node is 1 when the node is not idle, such as detecting physical carrier or deferring or backoff due to virtual carrier sensing. Since the instantaneous MAC layer utilization is either 1 (use) or 0 (idle), we average the value within a period (1 second in our simulation) to indicate the use percentage of the wireless medium around the node. We sample the value per 0.01 second and get the average MAC layer utilization over 100 latest samples.

The other metric is instantaneous queue length at the network layer. If a node has many packets waiting in the queue, it may result in long packet latency or even packet drop. So the queue length is another important congestion indication.

3.2.2   Using Congestion Information in Network Layer

To use congestion information in routing protocol DSR, we modified the *Routing Discovery* mechanism. When an intermediate node receives a *Route Request* packet that is not directed to it, it then checks the congestion metrics. If the congestion metrics are higher than some thresholds,

that indicates high level congestion around the node. Forwarding RREQ packet to other nodes will aggravate the congestion in two factors. First, transmission of this RREQ packet increases the use of the medium around the congested area. Second, it leads a new route across the busy area founded resulting in following additional transmission burden. Instead of handling and forwarding this RREQ packet, the busy node just silently drops the packet. So the discovered routes bypass the congestive nodes. The flowchart in Figure 4 illustrates our procedure of handling the received RREQ at a node.

The less congested routes rather than the shortest routes are more likely to be used in this scheme. It increases the fairness of the overall ad hoc network, which is especially important to the energy constrained sensor networks. It also reduces the total end-to-end delay of a packet, which includes transmission delay, processing delay, and queuing delay. When a packet travels through the longer but less congested routes, the transmission delay may slightly increase but it reduces more waiting time due to shorter queues of intermediate nodes. As we can see in our simulation results discussed in the next chapter, the total delay is reduced using the congestion aware scheme.

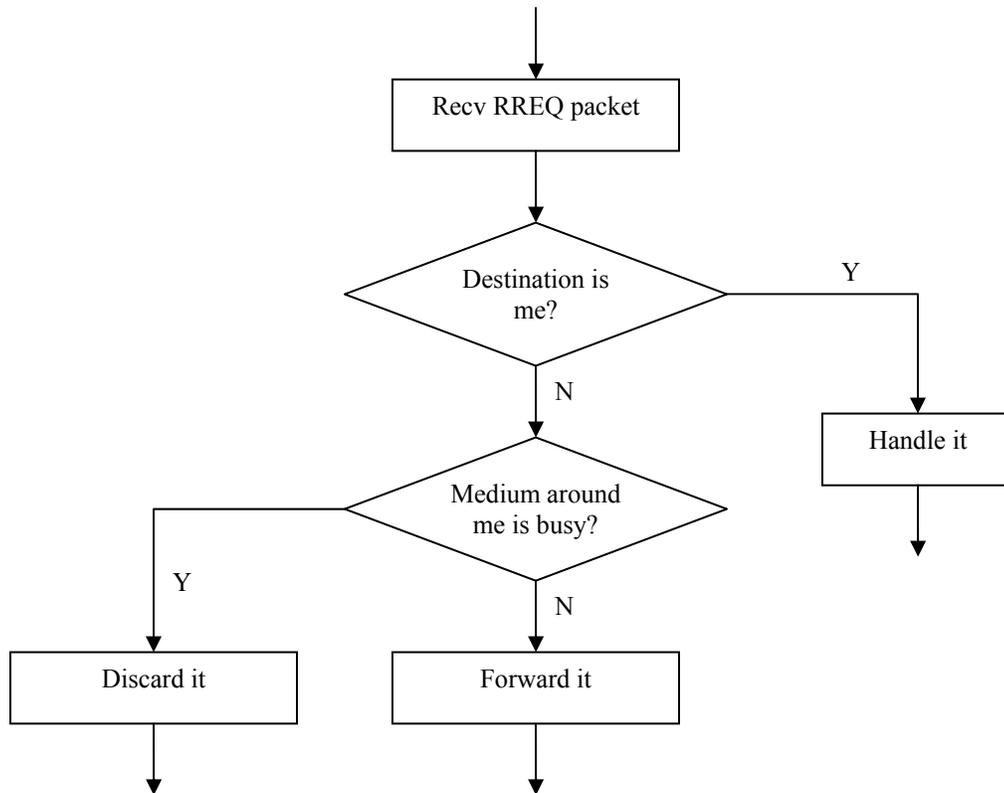Figure 4.   The Flowchart of Handling Received RREQ Packet at a Node

Suppressing RREQ forwarding at busy nodes has two tradeoffs. It may unable to discover the route to destination when a route actually exists but through the busy area. Forcing the route to avoid the congested area may cause a node to find a longer route than the shortest one. This leads to additional traffic overhead to other areas of the network.

Chapter Four

Simulation Results

4.1   The Network Simulator 2

The simulations are implemented using Network Simulator *NS-2* [13], which is an open-source simulation tool that runs on Unix/Linux/Windows. *NS-2* is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl (Tcl script language with Object-oriented extensions developed at MIT). It implements network protocols such as TCP and UPD, traffic source behavior such as FTP, Telnet, Web, CBR (continuous bit rate), route queue algorithms include fair queuing, deficit round-robin and FIFO, routing algorithms such as Dijkstra, and more. *NS-2* has provisions for implementing multicasting and some of the MAC layer protocols and LAN simulations. Current *NS-2* has already included wireless ad hoc networking protocols such as DSR, DSDV and AODV from CMU Monarch projects [22].

Simple scenarios should run on any reasonable machine; however, very large scenarios benefit from large amounts of memory. Besides ns-2.26, it requires the following packages to run: Tcl release 8.3.2, Tk release 8.3.2, OTcl release 1.0a8 and TclCL release 1.0b13.

4.1.1   Installing NS-2

Installation of *NS-2* could be a bit lengthy and a time-consuming process. It involves downloading and setting up a 250 MB package. We downloaded the version 2.26 all-in-one package from the *NS-2* home page http://www.isi.edu/nsnam/ns/ns-build.html#allinone into local directory, says /usr/src and extracted it as follows:

```
cd /usr/src
tar xzvf ns-allinone-2.26.tar.gz
cd ns-allinone-2.26
```

The package contains an "install" script to automatically configure, compile and install these components. Then we executed the script to complete the installation of binaries:

```
./install
```

After the installation process succeeded, we updated environment variables to include *NS-2* path and library. The last step of installation is to run the validation suite:

```
cd ./ns-2.26
./validate
```

4.1.2 Generating Traffic and Mobility Models

To run the simulation, we generated traffic and mobility models, which are called by tcl simulation code at *NS-2*.

4.1.2.1 Traffic Models

Random traffic connections of TCP and CBR can be setup between mobile nodes using CMU's traffic-scenario generator scripts. This traffic generator script is available under ~ns/indep-utils/cmu-scen-gen and is called cbrgen.tcl. It can be used to create CBR and TCP traffics connections between wireless mobile nodes. The command line looks like the following:

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections] [-rate rate]
```

We created a CBR connection file *cbr20_s1_r10*, having maximum of 10 connections among 20 nodes, with a seed value of 1 and a rate of 10.0 by the prompt:

```
ns cbrgen.tcl -type cbr -nn 20 -seed 1 -mc 10 -rate 10.0 > cbr20_s1_r10
```

4.1.2.2 Mobility Models

The node-movement generator script for generating the *Random Waypoint* mobility model is available under ~ns/indep-utils/cmu-scen-gen/setdest directory. Run setdest with arguments as shown below:

./setdest [-n num_of_nodes] [-p pausetime] [-s maxspeed] [-t simtime] \

    [-x maxx] [-y maxy]

We created a pedestrian scenario consisting of 20 nodes moving with maximum speed of 1.0 m/s with an average pause between movements being 1 second. The simulation will stop after 200 s and the topology boundary is defined as 500 meters X 500 meters. So our command line looks like:

./setdest -n 20 -p 1.0 -s 1.0 -t 200 -x 500 -y 500 > scen20ped_s1

4.1.3   Creating Simulation Code

We wrote a tcl code to set up the wireless simulation components: network components types, parameters such as the type of Interface Queue (IfQ), the type of antenna, the radio-propagation model, the type of ad-hoc routing protocol, traffic model and mobility model generated in last section etc. For details about these network components see chapter 15 section 1 of *NS-2* Manual [13]. Our tcl code is available in the Appendix 1.

4.1.4   Parsing the Simulation Trace Files

The trace files such as *test_ped_s1_r10.tr* recorded the traffic and node movements are generated after the simulation. These files need to be parsed in order to extract the information needed to measure the performance metrics. The new CMU trace format was used for parsing.

The new trace format of cbr at AGT level (Network layer) in our simulations looks like Table 2.

Table 2.   Trace Format of cbr in NS-2

Event	Time	Node ID	Trace level	Reason	Event ID (Seq)	Packet Type	Packet Size	MAC Info	Flag	IP Info	cbr Seq Num	Forw Num	Opt Forw Num

Event: s – send, r – receive, d – drop, f – forward

MAC Info: [Duration Destination_Ether_address   Source_Ether_address Ether_type]

IP Info: [Source_addr**:** port Destination_addr**:** port   TTL Next_hop]

The following is an example trace record:

r 0.513563811 _2_ AGT --- 4 cbr 512 [13a 2 1 800] ------- [1:0 2:0 32 2] [2] 1 0

It shows that a packet was received (r) at time 0. 513563811 second at destination node 2. The trace level is AGT, the reason is not specified (blank) and the event ID (receive sequence number) is 4, the packet type is cbr, and the packet size is 512 bytes. The MAC level information was given by duration 0x13a, destination Ethernet address 2, the source Ethernet address is 1 and Ethernet type is 0x800. The IP packet level information listed that source address:source port is 1:0, the destination address:destination port is 2:0 while TTL is 32 seconds and next hop is 2. The last part of this record was for cbr: sequence number is 2, packet had been forwarded (including itself) once and the optional forwarding number is zero.

4.2   The Simulation on NS-2

The simulations are implemented using *NS-2*, CMU Monarch extension. We created node-movement and traffic-pattern files for our simulations as described in the last section. To simulate pedestrian and stationary scenarios, we set the *max speed* of each node at 1.0 meter/sec and 0 respectively, the *pause time* at 1.0 second and packet size at 512 bytes. The application traffic pattern consists of 20 CBR nodes running on UDP within 300 m×1000 m or 500 m×500 m area and 20 or 10 CBR source-destination connections were generated randomly. The traffic of each of CBR connections begins at randomly generated start time within 1 second from the beginning of the simulation and the simulation stops after 200 seconds. For each pedestrian or stationary type, we randomly generated 10 node-movement and traffic-pattern scenarios. We vary offered load by changing the CBR packet rate from 10 packets/second to 60 packets/second at a step of 10 packets/second. So the total number of simulation scenarios for each scheme is 10*6*2 = 120.

Physical/MAC/Network layers Specifications

Channel type	Wireless Channel
Antenna type	Omni Antenna
Transmission range	250 meters
Radio Propagation model	Ricean/Rayleign
MAC protocol	Mac 802_11
Queue type	CMUPriQueue
Max queue length	50

Ad hoc Routing protocol   DSR

The tcl code in Appendix 1 lists more details of our simulation parameters and environment.

Evaluation Metrics

We compared the performance of different schemes by the following three important metrics at Agent layer, which is in the network layer in *NS-2*.

Throughput — The average bit rate per flow at the destination nodes that is measured at MAC layer.

Packet delivery ratio — The total ratio of the data packets received at the destinations to those generated by the CBR sources.

Delay — The average end-to-end delay per flow (due to transmission, processing, collision, and queuing) of packets traveling from the source to the destination node.

4.3   Results and Discussion

The simulation results of stationary scenario and the pedestrian scenario under various offered load of different rate adaptation and congestion aware schemes are shown in Figures 5-6.

### 4.3.1   Comparison of Two Rate Adaptation Schemes

To demonstrate the performance of our MAC layer rate adaptation scheme based on both congestion information and signal strength, we compare two metrics, signal strength only and congestion aware, used in rate adaptation with the original DSR. The dimension of the network boundary in this simulation is 300 meters × 1000 meters, the number of cbr connections is 20.

4.3.1.1   Effect of Rate Adaptation

In Figure 5, we observe that congestion aware plus signal strength based rate adaptation (RAcs) and signal strength based rate adaptation (RAss) both improve the performance of the original DSR protocol (DSR). RAcs performs better than RAss.

22

Both RA schemes perform better than the original DSR protocol. It increases the throughput about 5% at stationary scenario and 10% at pedestrian scenario. The packet delay of RA is only 85% of DSR protocol.

According to Figure 5 (a) and (d), RAcs increases throughput of DSR about 5% and 8% more than RAss does at stationary and pedestrian scenario, respectively. RAcs also increases delivery ratio about 2% to 6% more than RAss from Fig.5 (b) and (e). It shows that RAcs drops fewer packets due to reduced congestion by adaptive transmission rate, thus packets can transfer faster and the overall throughput of network is increased by load balance. This demonstrates our congestion aware based rate adaptation scheme improve the overall performance.
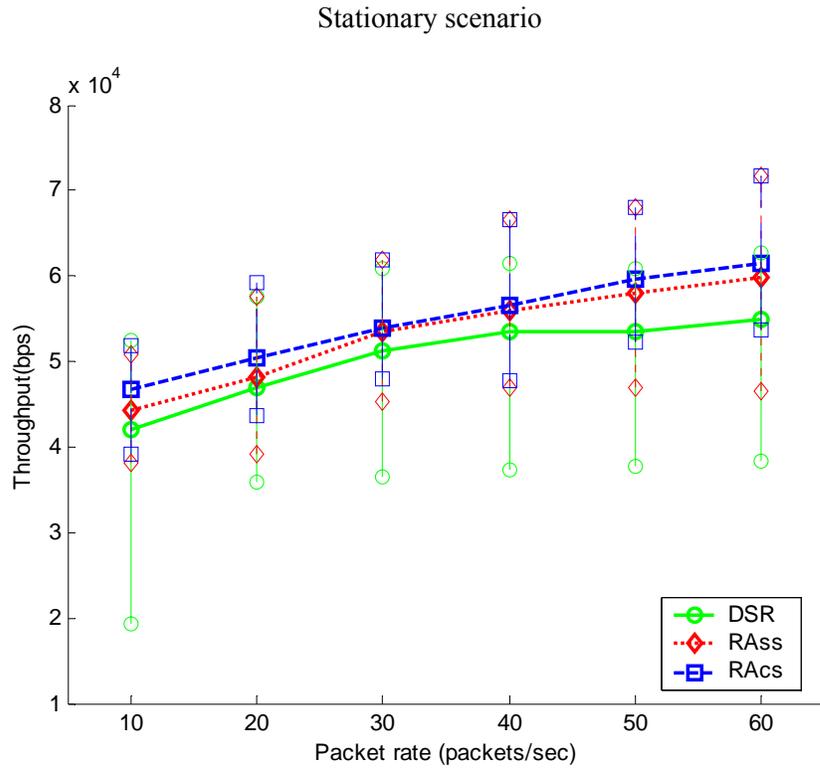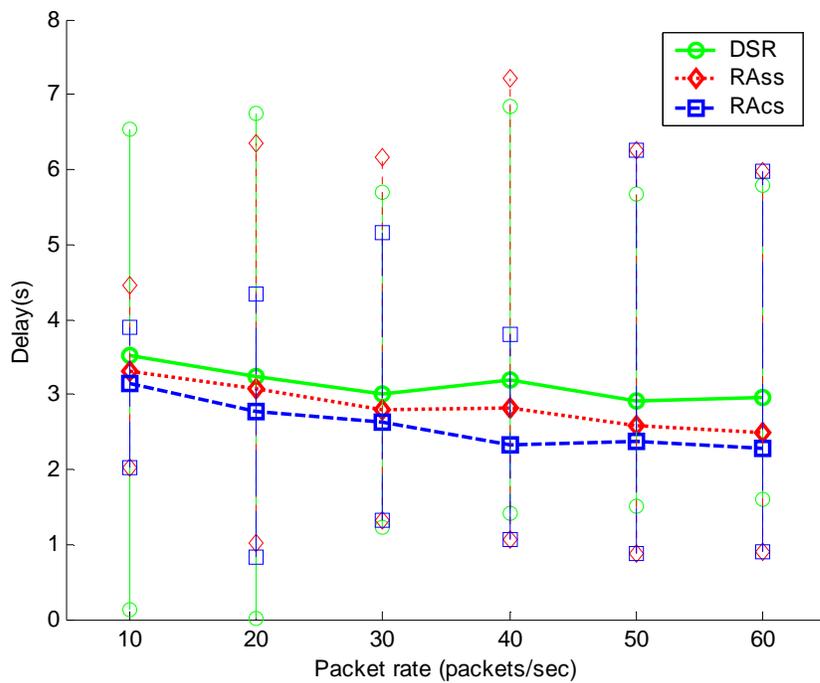

(a) Throughput vs. Packet Rate, max speed = 0

Figure 5. Two Rate Adaptation Metrics Simulations

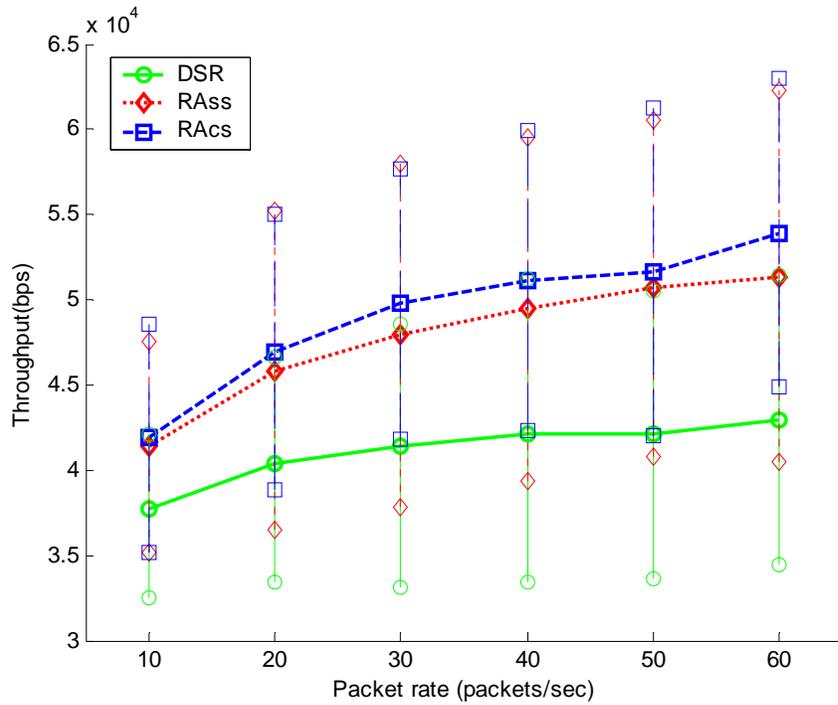Dimension 300 m $\times$ 1000 m
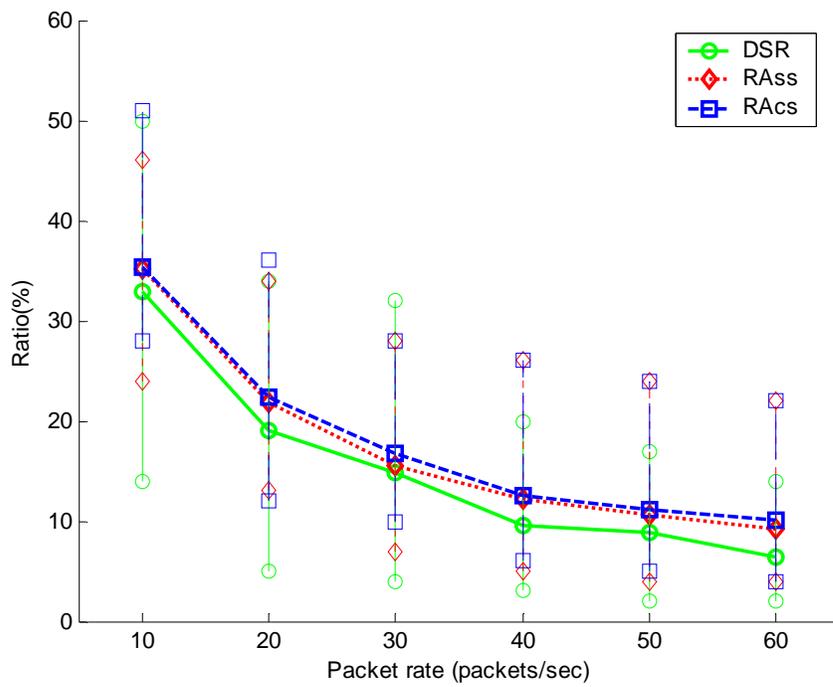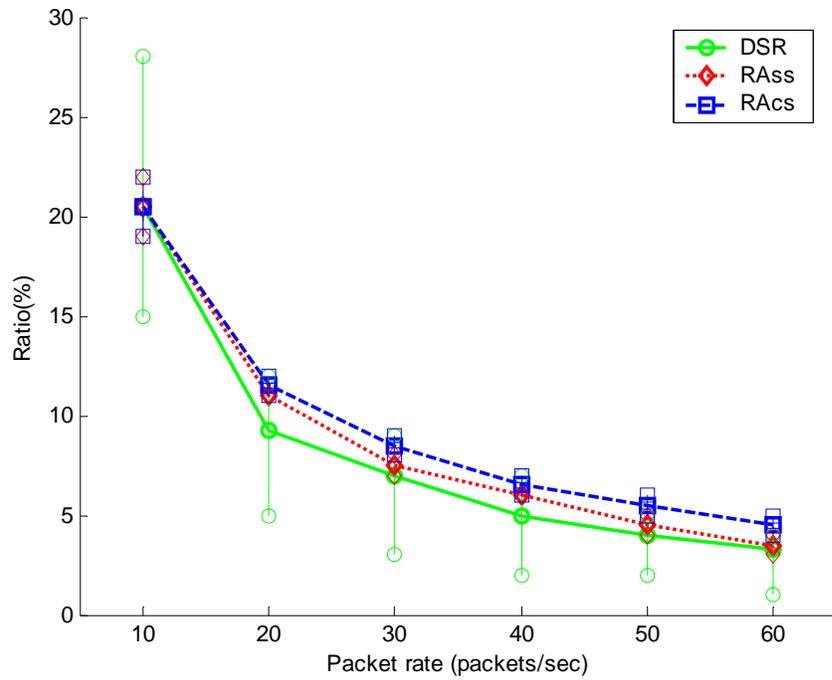
(b) Delivery Ratio vs. Packet Rate, max speed = 0

(c) End-to-end Delay vs. Packet Rate, max speed = 0

Figure 5. Continued

Pedestrian scenario, max speed = 1.0 m/s

(d) Throughput vs. Packet Rate, max speed = 1.0 m/s

(e) Delivery Ratio vs. Packet Rate, max speed = 1.0 m/s

Figure 5. Continued

25

(f) End-to-end Delay vs. Packet Rate, max speed = 1.0 m/s

Pedestrian scenario, max speed = 5.0 m/s

(g) Throughput vs. Packet Rate, max speed = 5.0 m/s

Figure 5. Continued

26

(h) Delivery Ratio vs. Packet Rate, max speed = 5.0 m/s
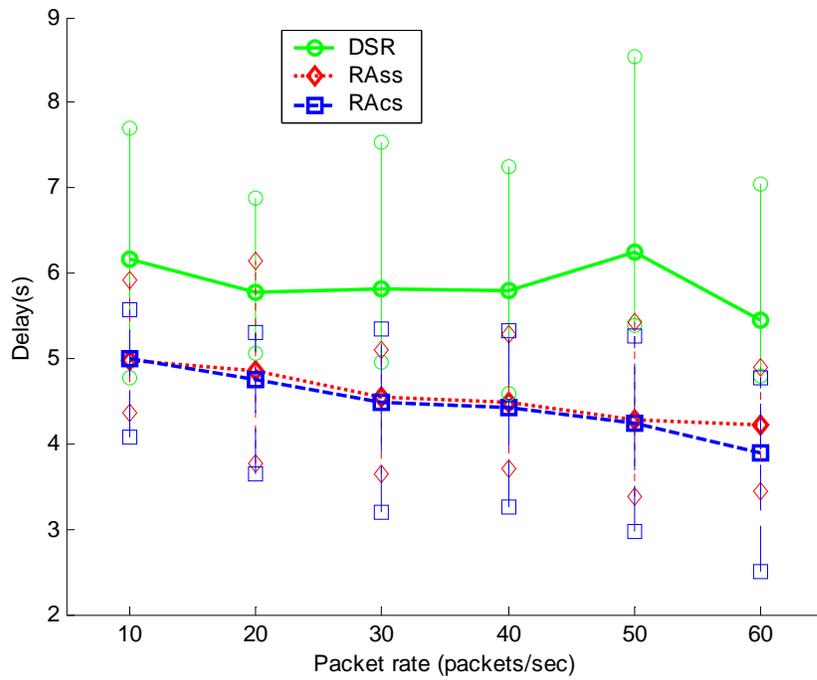
(i) End-to-end Delay vs. Packet Rate, max speed = 5.0 m/s

Figure 5. Continued

Table 3. Summary of Two RA Schemes with DSR in Different Mobility Scenarios

	Zero mobility			Max speed 1.0 m/s			Max speed 5.0 m/s		
	Throughput (kbps)	Ratio (%)	Delay (s)	Throughput (kbps)	Ratio (%)	Delay (s)	Throughput (kbps)	Ratio (%)	Delay (s)
DSR	43~50	34~6	3.5~3.0	37~42	20~4	4.2~5.0	32~34	18~2	5.5 ~ 6.3
RAss	+5%	+4%	-1~-3%	+9 ~ +20%	+5%	-1~-21%	+9 ~ +29%	+5~+40%	-19~-23%
RAcs	+10%	+6%	-4~-10%	+11~+25%	+8~+12%	-2~-30%	+18~+41%	+5~+50%	-19~-39%

4.3.1.2   Performances of Different Mobility

Figure 5 shows three different mobility scenarios: stationary, pedestrian with max speed of 1.0 m/s, and pedestrian with max speed of 5.0 m/s. The performances of DSR, RAss, and RAcs in these three scenarios are given in Table 3. According to Table 3, RAss and RAcs both improve the performance when the mobility increase while RAcs outperforms RAss.

4.3.2   Comparison of Rate Adaptation and Congestion Aware Schemes

We simulated the traditional DSR routing protocol, our Rate Adaptation (RA) scheme, Rate Adaptation scheme with moving averaging window length 5 (RAwin) and Congestion Aware routing plus Rate Adaptation (CARA) scheme and compared their performances. Figure 6 shows the simulation results of all the schemes we used in our cross layer system. The dimension of the network boundary in this simulation is 500 meters × 500 meters, where the nodes are more densely distributed than the simulation described in the last section in order to make more congestion. The number of the cbr connections is decrease from 20 in the last simulation to 10. So there is less traffic in this smaller network area. We observe that at all scenarios the average throughput decrease as the offered load increase due to congestion. The packet delivery ratio decreases and delay increases with the increasing offered load. The packets are dropped at higher traffic due to increase in control overhead and route setup delay.

In pedestrian scenario, the throughput and packet delivery ratio of all schemes are smaller than the stationary scenario by about 20% ~ 30% since the mobility attributes to route failure and packet loss.

The effects of the different schemes are discussed below.

(a) Throughput vs. Packet Rate, max speed = 0
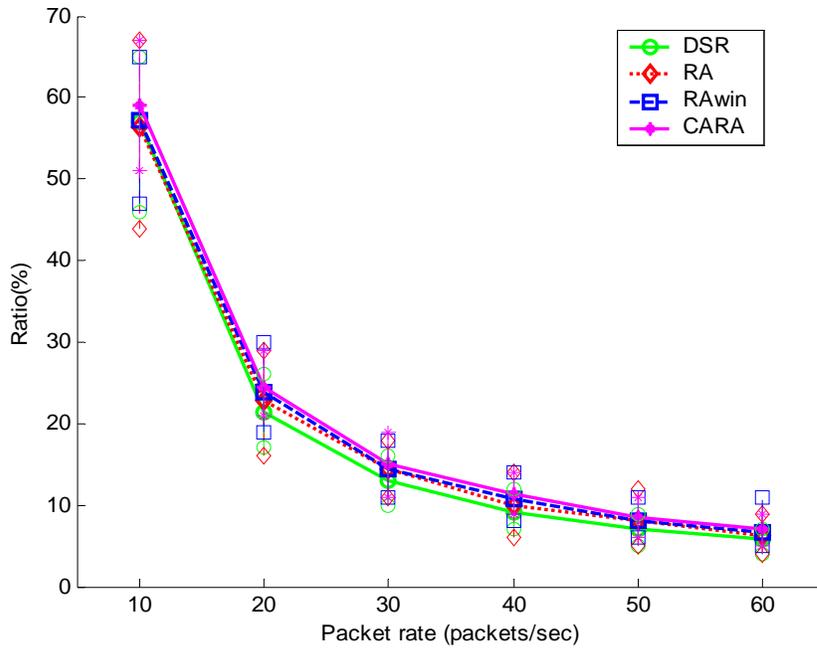
(b) Delivery Ratio vs. Packet Rate, max speed = 0

Figure 6.   DSR, RA, RAwin, and CARA Offered Load Simulations

Dimension 500 m × 500 m
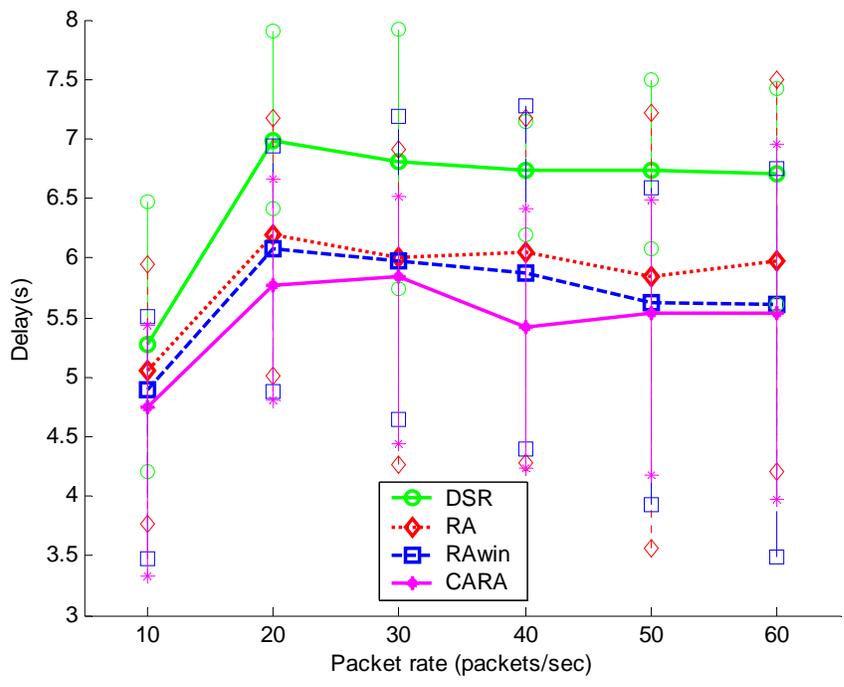
(c) End-to-end Delay vs. Packet Rate, max speed = 0
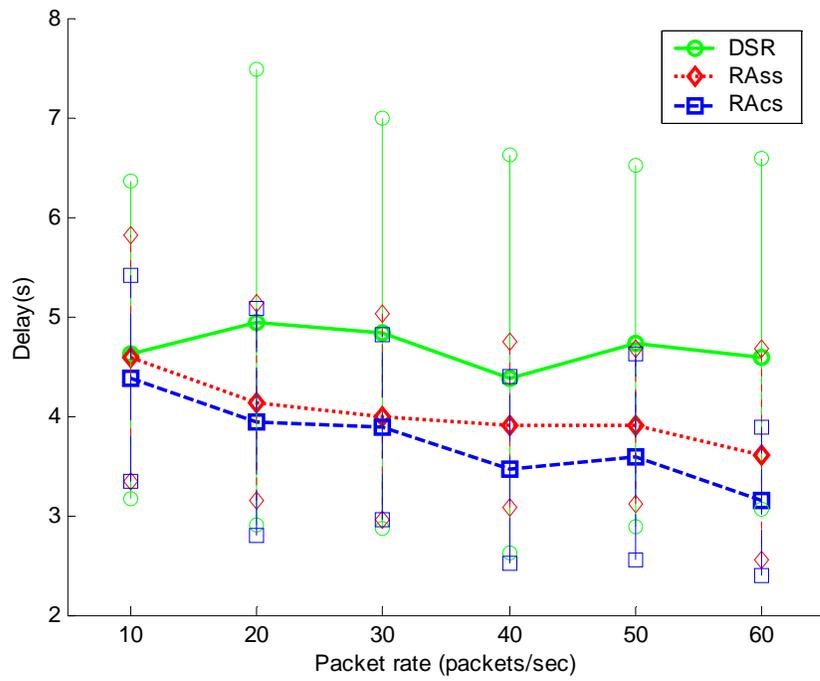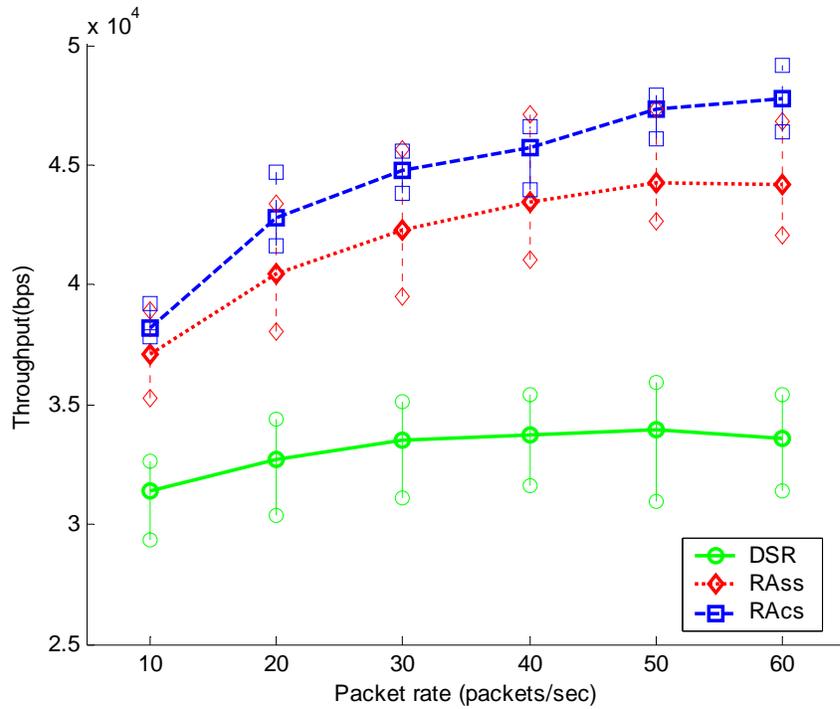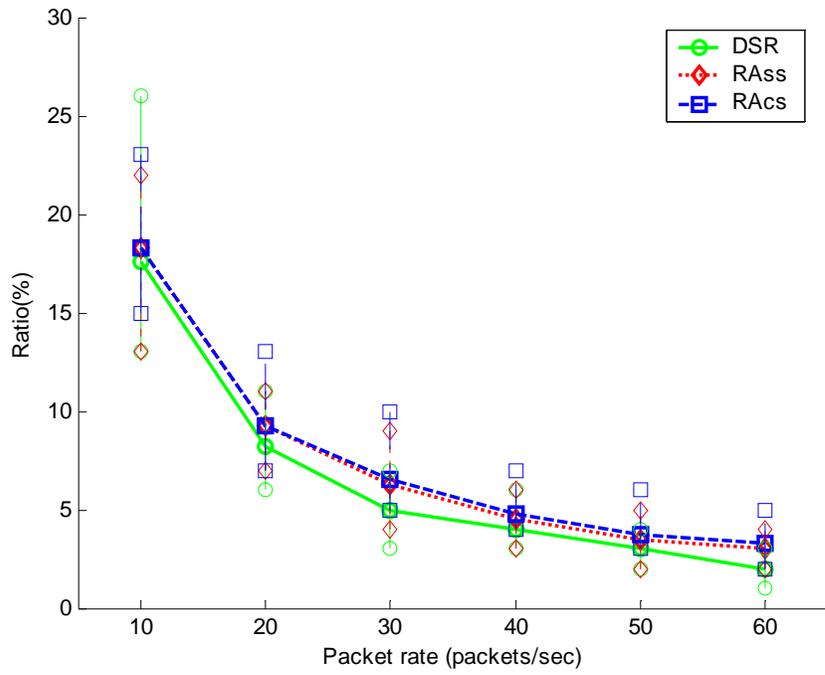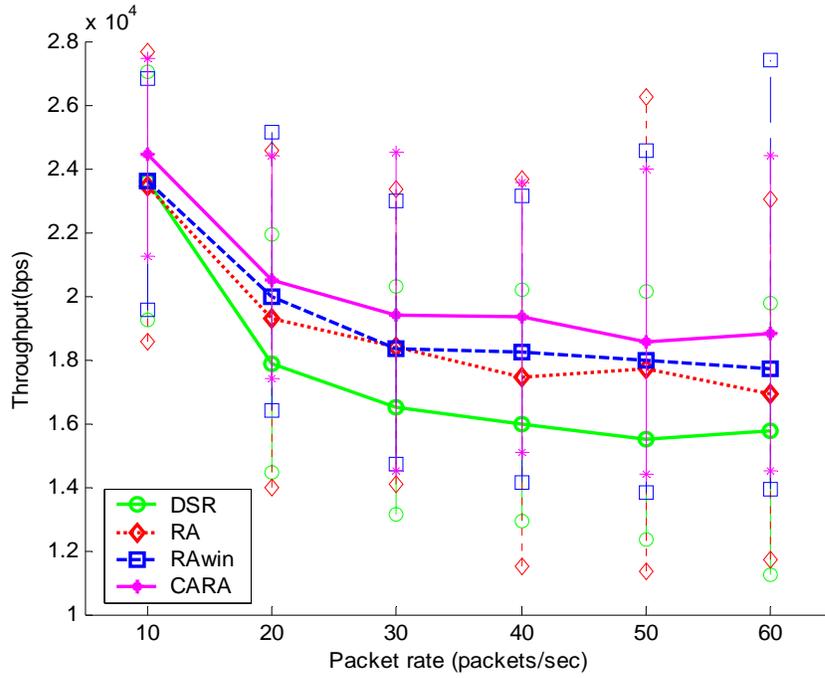
Pedestrian scenario, max speed = 1.0 m/s
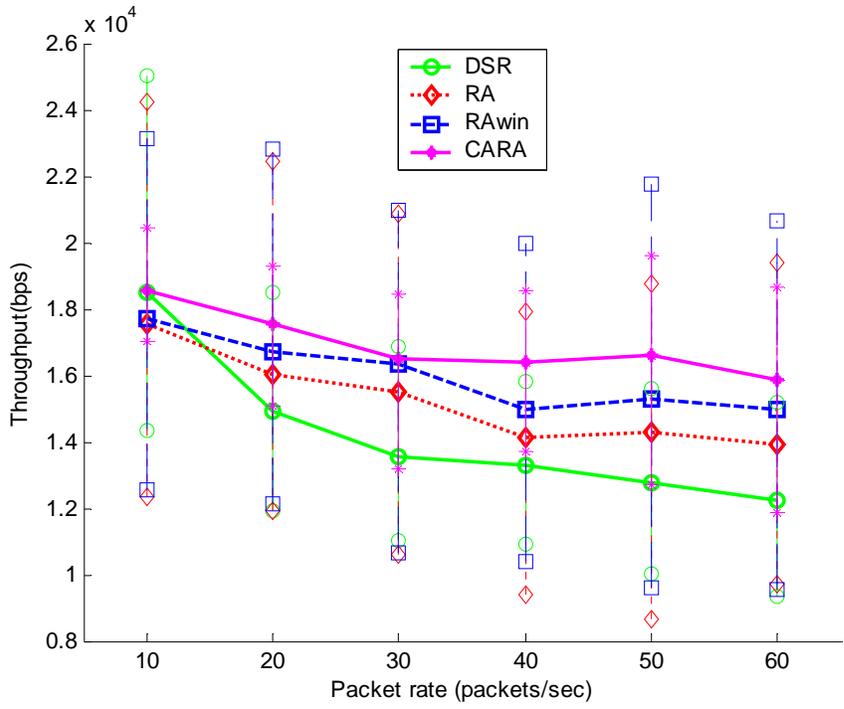
(d) Throughput vs. Packet Rate, max speed = 1.0 m/s

Figure 6. Continued

(e) Delivery Ratio vs. Packet Rate, max speed = 1.0 m/s

(f) End-to-end Delay vs. Packet Rate, max speed = 1.0 m/s

Figure 6. Continued

4.3.2.1 Effect of Averaging Window

The RAwin scheme enhances the throughput and delay performance more than RA especially at high offered load, which proves the moving average window is more accurate estimation method than only based on current signal strength sample.
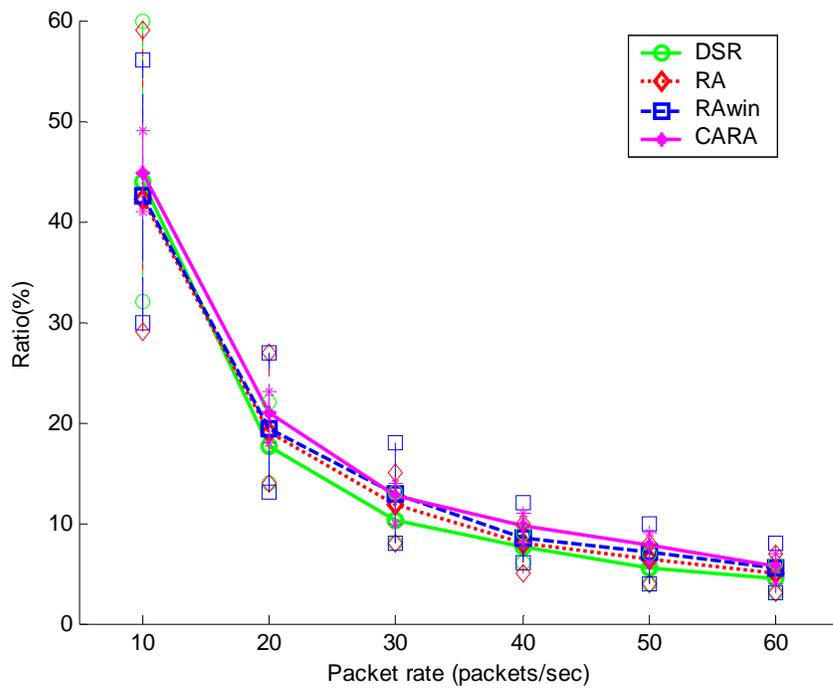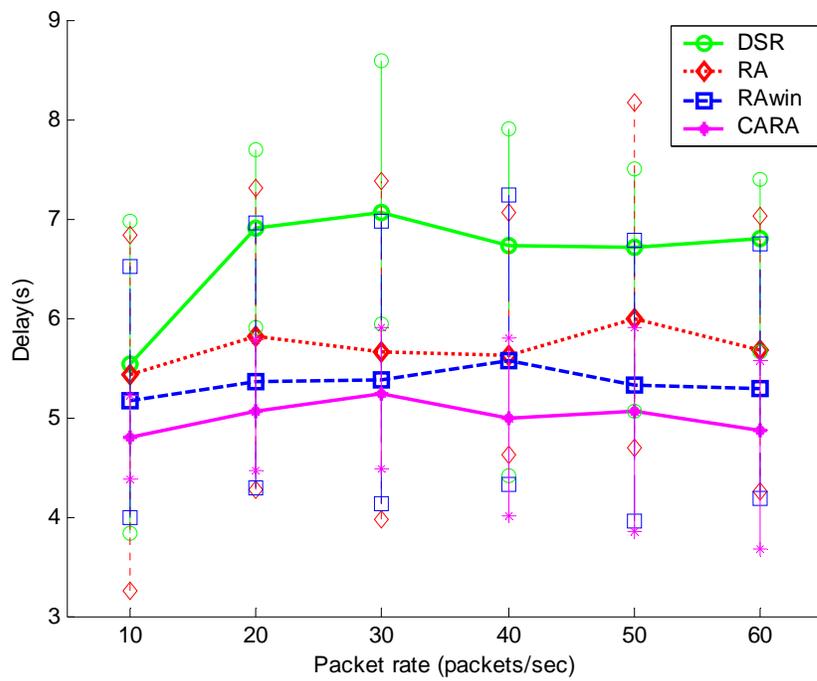
4.3.2.2 Effect of Congestion Aware

The CARA scheme performs best among the four schemes. It increases 30% of throughput and 20% of delivery ratio than the DSR. The delay of CARA is only 85% of RA and 70% of the DSR. This shows that congestion aware scheme reduces transmission through congestion area thus benefits from overall network load balance.

4.3.3 Discussion

Another interesting question that is raised for DSR routing protocol when the network nodes are stationary is whether it can find a new alternate route once the initial routes are established. In other words, in our congestion-aware routing scheme, whether nodes will send out any new Route Request (RREQ) packet to find a new route to avoid the congested areas of the network.

In this research, we tried to verify whether our congestion-aware scheme works in the stationary scenario, we ran several tests. From the trace files which record most important DSR events during the entire running time, we find that the DSR does find the link break and sends the Route Error packets, and then it sends out new RREQ packets. Compared to the pedestrian scenario, it does send out fewer RREQ packets in the stationary scenario, about 1000 and 1300 respectively. The tests prove that our congestion-aware scheme also works in stationary scenario.

So in the wireless outdoor environment, the channel conditions change and links break even in the stationary scenario, thus causing DSR to find route error and send new RREQ. Another possible guess is that due to the packets dropped at the congested nodes, the sender may not receive the ACK before timeout and even after maximum number of retransmissions, the sender may assume the next hop is unreachable and then send out Route Error packet.

Comparing Figures 5 (a) (d) (g) with Figures 6 (a) (d), we notice that when the offered load increases, the throughputs of all the schemes increase in the larger area 300 m × 1000 m (Fig.5),

but decrease in the smaller area 500 m × 500 m (Fig.6). The average throughput of Figure 5 is also higher than that of Figure 6. The possible reason for such difference is that the transmission range of 250 m for the nodes is comparable with the dimension of the network area. Within the smaller area, the nodes are distributed closer to each other, so they are more likely in the collision area of others. When the nodes try to transmit, especially in the high traffic load, collisions cause congestion and reduce the throughput of the whole network.

Chapter Five

Conclusion

In this thesis, we briefly discuss performance optimization design challenges of ad hoc networks and how network performance can be improved by using congestion-aware cross-layer protocol design. The experiments were conducted to assess the performance improvements by simulating different cross-layer implementations for an ad hoc network.

The results show that the rate adaptation in MAC layer improves the network performance. Average window of signal strength can estimate the channel quality more accurately than only one instantaneous sample. Using congestion information in both sender and receiver sides along with signal strength helps maximizing network capacity at MAC layer. Using congestion information from MAC layer in routing discovery improves the performance of the overall network.

Future work involves using the congestion information in other routing protocols and transport layer. We may test exploiting congestion information in proactive routing protocol such as DSDV. For example, a node using DSDV may increase the advertisement period time. For TCP protocol, besides setting ECN bits to improve the performance, the congestion information may help TCP react better by distinguishing the packet loss due to congestion from other reasons such as link failure.

References

[1]   T. S. Rappaport, A. Annamali, R. M. Buehrer, and W. H. Tranter, "Wireless Communications: Past events and a future perspective," *IEEE Communications Magazine,* pp. 148-161, May 2002.

[2]   J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," In *the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking* (MobiCom'98), pages 85~97, October 1998.

[3]   S.-J. Lee, M. Gerla, and C. -K. Toh, "A Simulation Study of Table-driven and On-Demand Routing Protocols for Mobile Ad Hoc Networks," *IEEE Network Magazine*, pp. 48-54, August 1999.

[4]   E. M. Royer, S. J. Lee, and C. E. Perkins, "The effects of MAC protocols on ad hoc networks communication," *Proc. of IEEE WCNC'00*, Vol. 2, pp. 543-548, September 2000.

[5]   J.-H. Lee, S. Singh, and Y.-S. Roh, "Interlayer Interactions and Performance in Wireless Ad Hoc Network," IRTF ANS Working Group, Internet-Draft, *draft-irtf-ans-interlayer-performance-00.txt*, September 2003.

[6]   W. H. Yuen, H.-no Lee and T. D. Andersen, "A Simple and Effective Cross Layer Networking System for Mobile Ad Hoc Networks," *IEEE PIMRC,* Vol.4, pp.1952–1956, September 2002.

[7]   Y.-C. Hu and D. B. Johnson, "Exploiting Congestion Information in Network and Higher Layer Protocols in Multihop Wireless Ad Hoc Networks," In *the 24th International conference on Distributed computing Systems* (ICDCS 2004), pp.301-310, IEEE, Japan, March 2004.

[8]   S. Shakkottai, T. S. Rappaport and P. C. Karlsson, "Cross-layer Design for Wireless Networks," *IEEE Communications Magazine*, pp.74-80, October 2003.

[9]   ISO/IEC 8802-11; ANSI/IEEE Std 802.11, 1999 edition. Information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements. Part 11: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. The Institute of Electrical and Electronics Engineers, 1999.

[10] D. B. Johnson and D. A. Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks," In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pp. 153-181. Kluwer Academic Publishers, 1996.

[11] D. B. Johnson, D. A. Maltz, and J. Broch. "The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks," In *Ad Hoc Networking*, edited by Charles E. Perkins, chapter 5, pp. 139-172, Addison-Wesley, 2001.

[12] D. B. Johnson, D. A. Maltz, and Y.-C. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," IETF MANET Working Group, Internet-Draft, *draft-ieft-manet-dsr-09.txt*, April 2003. Available at http://www.ietf.org/html.charters/manet-charter.html.

[13] Network Simulator Notes and Documentation, UCB/LBNL, http://www.isi.edu/nsnam/ns/

[14] R. J. Punnoose, P. V. Nikitin, and D. D. Stancil, "Efficient Simulation of Ricean Fading within a Packet Simulator," *IEEE Vehicular Technology Conference*, pp.764-767, September 2000.

[15] A. J. Goldsmith, S.-G. Chua, "Variable-rate variable-power M-QAM for fading channels", *IEEE Trans. on Communications*, Vol. 45, pp. 1218-1230, October 1997.

[16] G. Holland and N.Vaidya, P. Bahl, "A Rate-Adaptive MAC Protocol For Multi-Hop Wireless Networks," *Mobicom*, pp.236-251, Rome, Italy, July 2001.

[17] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pp. 234-244, August 1994.

[18] R. Boppana and S. P. Konduru, "An Adaptive Distance Vector Routing Algorithm for Mobile, Ad Hoc Networks," In *Proceedings of IEEE INFOCOM 2001*, pp.1753-1762, March 2001.

[19] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, Vol.24, No.5, pp.10-23, October 1994.

[20] F. Peng and J. Ma, "An Effective way for Enhancement of TCP Performance in Wireless and Mobile Networks," Internet Draft, draft-fpeng-wecn-04.txt, June 2001.

[21] R. Krishnan, M. Allman, C. Partridge, and J. P.G. Sterbenz, "Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks," *BBN Technical Report* No. 8333, BBN Technologies, March 2002.

[22] The Monarch Project of Carnegie Mellon University, http://monarch.cs.cmu.edu/

[23] A. Woo and D. E. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," *Proc. ACM/IEEE Conf. On Mobile Computing and Networks* (MobiCOM 2001), pp. 221 – 235, Rome, Italy, July 2001.

[24] S. -Y. Ni et al., "The broadcast storm problem in a mobile ad hoc network, *IEEE MobiCom'99*, pp. 151-162, August 1999.

[25] M. R. Pearlman et al., "On the impact of alternate path outing for load balancing in mobile ad hoc networks", *IEEE MobiHOC*, pp. 3-10, August 2000.

[26] Y. Yi, M. Gerla and T. J. Kwon, "The Selective Intermediate Nodes Scheme for Ad Hoc On-Demand Routing Protocols", *ICC 2002*, Vol. 5, pp. 3191-3196, April-May 2002.

[27] P. Chaporkar, A. Bhat, and S. Sarkar, "An Adaptive Strategy for Maximizing Throughput in MAC layer Wireless Multicast," *MobiHoc'04,* pp.256-267 , Roppongi, Japan, May 2004.

Appendices

Appendix A: tcl Code Used in NS-2 Simulations

```
# newtest.tcl
# tcl code for wireless simulation on NS-2, reference to wireless1.tcl,
# cbrgen.tcl and example4.tcl

# ===================================================================

proc getopt {argc argv} {
        global argv0 seed rate cp sc
        if {$argc < 4} {
            puts "\nusage: $argv0 seed rate cbr-connection_file node-movement_file\n"
            exit
        }

        set seed [lindex $argv 0]
        set rate [lindex $argv 1]
        set cp [lindex $argv 2]
        set sc [lindex $argv 3]
        #puts "\n $seed $rate $cp $sc \n"
}


# ===================================================================
# Define options
# ===================================================================

set val(chan)           Channel/WirelessChannel
set val(prop)           Propagation/Ricean        ;# radio-propagation model
set val(netif)          Phy/WirelessPhy
set val(mac)            Mac/802_11
set val(ifq)            CMUPriQueue  ;# for dsr
set val(ll)             LL
set val(ant)            Antenna/OmniAntenna
set val(x)              500   ;# X dimension of the topography
set val(y)              500   ;# Y dimension of the topography
set val(ifqlen)         50          ;# max packet in ifq
set val(adhocRouting)   DSR
set val(nn)             20          ;# how many nodes are simulated
set val(cn)             10          ;# how many connections
set val(stop)           200         ;# simulation time

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0
```

```
# Initialize the SharedMedia interface with parameters to make
# It works like the 914MHz Lucent WaveLAN DSSS radio interface
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set Rb_ 2*1e6
Phy/WirelessPhy set Pt_ 0.2818
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0


####################################################################
#   Set Scheme type and Rate Adaptation threshold table
#          SCHEME_NONE         0
#          SCHEME_RASS         1         signal strength
#          SCHEME_RACS         2         congestion information plus signal strength
#   Set DSR exploiting congestion information Suppress & metric type
#          SUPPRESS_NONE              0
#          SUPPRESS_RREQ_FORWARD          1
#          SUPPRESS_SALVAGE        2
#          SUPPRESS_BOTH           3
#          METRIC_MAC_UTIL         1
#          METRIC_QUEUE            2
#          METRIC_BOTH             3
#   Set DSR free air time threshold (atimeThreshold_) in percentage
####################################################################
Mac/802_11 set schemeType_ 1
Mac/802_11 set winLength_ 5
Mac/802_11 set metricType_ 1
Mac/802_11 set alpha_ 50
Mac/802_11 set beta_ 25
Agent/DSRAgent set suppressType_ 1
Agent/DSRAgent set metricType_ 1
Agent/DSRAgent set atimeThreshold_ 5
Agent/DSRAgent set qlenThreshold_ 20


# =======================================================================
# Main Program
# =======================================================================


#
# Initialize Global Variables
#
getopt $argc $argv

# create simulator instance

set ns_            [new Simulator]
```

Appendix A: (Continued)

```
# setup topography object

set topo [new Topography]

# create trace object for ns and nam
set tracefd        [open out${sc}_s${seed}_r${rate}.tr w]
#set namtrace     [open out.nam w]

$ns_ trace-all $tracefd
#$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# define topology
$topo load_flatgrid $val(x) $val(y)

#
# Create God
#
set god_ [create-god $val(nn)]

#
# define how node should be created
#
# New API to config node:
# 1. Create channel (or multiple-channels);
# 2. Specify channel in node-config (instead of channelType);
# 3. Create nodes for simulations.

# Create channel #1
set chan_1_ [new $val(chan)]

#global node setting
# Create node "attached" to channel #1
$ns_ node-config -adhocRouting $val(adhocRouting) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channel $chan_1_ \
            -topoInstance $topo \
            -agentTrace ON \
            -dsrTrace OFF \
            -routerTrace OFF \
            -macTrace ON \
            -movementTrace OFF
```

41

```
###################################################
#   Set Ricean and Rayleigh fading propagation parameter
## #############################################
set prop_inst [$ns_ set propInstance_]
$prop_inst MaxVelocity  2.5;
$prop_inst RiceanK      0;
$prop_inst LoadRiceFile  "rice_table.txt";


#
#  Create the specified number of nodes [$val(nn)] and "attach" them
#  to the channel.

for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0              ;# disable random motion
}


#
# Define traffic model
#
#puts "Loading connection pattern..."
source $cp


#
# Define node mobility model
#
#puts "Loading scenario file..."
if { $sc != 0} {
   source $sc
}

# Define node initial position in nam

for {set i 0} {$i < $val(nn)} {incr i} {

   # 20 defines the node size in nam, must adjust it according to your scenario
   # The function must be called after mobility model is defined

   $ns_ initial_node_pos $node_($i) 20
}


#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
   $ns_ at $val(stop).0 "$node_($i) reset";
}
```

```
#Start logging the received bandwidth
#$ns_ at 1.0 "record"

puts $tracefd "M 0.0 nn=$val(nn) x=$val(x) y=$val(y) rp=$val(adhocRouting)"
puts $tracefd "M 0.0 sc=$sc cp=$cp seed=$seed"
puts $tracefd "M 0.0 prop=$val(prop) ant=$val(ant)"
#Define a 'finish' procedure
proc finish {} {
        global ns_ tracefd
        $ns_ flush-trace
        #Close the output files
        close $tracefd
        #Call xgraph to display the results
        #exec xgraph out0.tr out1.tr out2.tr -geometry 800x400 &
}

#Call the finish procedure after simulation stop
$ns_ at $val(stop).0002 "finish"

#$ns_ at  $val(stop).0004 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at  $val(stop).0004 "$ns_ halt"

#puts "Starting Simulation..."
$ns_ run

# end of this tcl code
```