EFFECTS OF PHYSICAL LAYER MODELS ON WIRELESS NETWORK SIMULATIONS

BY UMUT AKYOL

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Roy D. Yates

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2005

ABSTRACT OF THE THESIS

Effects of Physical Layer Models on Wireless Network Simulations

by Umut Akyol

Thesis Director: Professor Roy D. Yates

In the study of wireless networks, simulation has become the most common tool for evaluation of devices and protocols due to its ease of use. However, the correct level of detail that should be implemented in simulations is not well known by the research community. For example the most common simulators are essentially packet level network protocol simulators which use simple channel models for computational efficiency. In this thesis, we explore the effects of physical layer details on wireless network simulations.

In this thesis, we focus on a specific network simulator, ns-2, due to its open source code base and a specific protocol 802.11b, due to the fact that it's already implemented in ns-2. The ns-2 simulator focuses on the higher layer protocols, while abstracting the details of models at other layers, particularly the interactions with physical layer models. In this thesis, we examine physical layer models for transmitter interference, signal transmission and reception for 802.11b that are relevant to the performance evaluation of higher layer protocols. Starting with an overview of 802.11b's physical layer and current physical layer

modeling of ns-2 with the Monarch extensions, we propose and implement enhancements to the physical layer models. We also describe how these changes can be used to model physical layers other than 802.11b. We then quantify the impact of these changes under typical scenarios used for the performance evaluation of wireless networks.

Acknowledgements

I would like to thank Prof. Roy Yates for his constant support, guidance and patience throughout this thesis, which helped me to have a great educational experience at WINLAB. I am deeply thankful for having the privilege of working with him as my advisor.

I would like to thank Prof. Dipankar Raychaudhuri, Prof. Marco Gruteser and Prof. Yanyong Zhang for being in my thesis committee. I would also like to thank all WINLAB colleagues for making a friendly and collaborative research environment.

I would like to thank all my friends, here in USA and especially in Turkey, The Crew of Destabilize61 (I wouldn't survive without you guys). Finally, my deepest thanks go to my family for the lifelong encouragement, support and love.

Dedication

To my father Mahir Akyol

to my mother Saadet Akyol

and to my brother Ilter Akyol

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

To date, the most common tool in wireless networking research has been simulation. There are two important motivating reasons to use simulation. The first is the difficulty of creating a real implementation which requires use of a system with many components. The second reason is that it gives maximum experimental control to the researcher. In a simulator, the code is contained within a single component which is clearly defined and accessible. Therefore, the researcher can control the whole system and design experiments without limitations.

However, it has become increasingly difficult to build accurate analytical models of the devices and protocols of modern wireless networks. Simulation of wireless networks differ considerably from simulation of wired networks in that they have to include modeling of physical channel properties (fading and interference) which has a profound effect on network performance. But it is not possible to exactly replicate physical channel properties inside a computer model, so when creating a simulation some factors must be statistically or otherwise approximated. However, adjusting the level of detail is a difficult problem. The failure to properly capture the behavior of first-order factors can lead to incorrect results. On the other hand, excessive detail may increase simulation run-time without even affecting the results.

OPNET Modeler [1], ns-2 [2, 3] and GloMoSim [4] are among the most popular simulators. Each provides an advanced simulation environment to test and debug wireless networking protocols. Among these, ns-2 is widely used in the research community due to its open source code base. In ns-2, two MAC layer protocols, 802.11 and TDMA are implemented for mobile networks. Due to its popularity in wireless networking research, we focus on the 802.11 MAC protocol in this thesis.

The ns-2 simulator focuses on the higher layer protocols, while abstracting the details of models at other layers, particularly the interactions with physical layer models. This thesis examines physical layer models for transmitter interference and signal transmission and reception that are relevant to the performance evaluation of higher layer protocols such as AODV and DSR. The plan of the thesis is as follows:

- In Chapter 1, motivation and objectives of this thesis will be explained.
- In Chapter 2, the details of 802.11b PHY Layer which may affect the results of a simulation will be presented.
- In Chapter 3, the fundamentals of ns-2 and its wireless physical layer model will be described.
- In Chapter 4, the details of the modification of ns-2's wireless physical layer model will be given.
- In Chapter 5, simulation results of ad hoc routing protocols with the modified ns-2 will be presented and the reasons for divergence from the results of the original ns-2 will be explained.
- In Chapter 6, a brief summary of the results and the conclusion will be given.

1.2 Motivation and Objectives

The ns-2 simulator has highly developed models for the network and transport layers, and has proven invaluable as a community research tool. Ns-2 has become a de facto standard for characterizing performance differences in networking and transport protocols for its particular model of the wireless physical layer. Not surprisingly, most recent studies on the interactions on the MAC and network layer protocols, [5, 6] for example, employ the ns-2 simulator.

In these works, it is understood that end-to-end network performance evaluation is essential. Thus a simulator must support appropriate models at all layers. However, with the increasing complexity of radio physical systems, fine grain physical layer simulations implement bit detection and packet decoding an order of magnitude slower than real time *for a single communication link.* [7]. This implies that detailed physical layer simulations are orders of magnitude too slow for the simulation of wireless network protocols. Thus, the objective of physical layer modeling is to capture the representative characteristics of interfering transmissions in a model of sufficient simplicity to support practical network simulation. In fact, the Monarch extensions have achieved this balance for CSMA wireless networks in which the MAC protocol prevents large numbers of interfering transmissions.

In this thesis, we propose modifications to the Monarch extensions intended both to enhance the accuracy of simulations in low-power wireless networks as well to advance the development of ns-2 models of other wireless communication technologies. Specifically, we modify the Monarch extensions to account for multiple simultaneous interferers by tracking the signal to interference plus noise ratio (SINR) during packet reception. We use SINR tracking to determine whether a packet is received successfully. In simulation experiments,

we observe that performance variations between the SINR tracking model and the standard ns-2 pairwise collision model emerge when the transmit power is low. In this case, the transmission range is small and the network can support a significant number of simultaneous transmissions. We quantify such differences under typical scenarios used for the performance evaluation of wireless ad hoc routing protocols. In particular, in comparing DSR and AODV under the modified physical layer, we observe that their performance can be very sensitive to the variations in the SINR tracking model in various scenarios.

There has been similar studies comparing the performance of ad hoc routing protocols with ns-2 [8–15]. Broch, Maltz, Johnson, Hu and Jetcheva, the original authors of the simulation model, evaluated four ad hoc routing protocols including AODV and DSR [13]. DSR demonstrated vastly superior routing load performance, and somewhat superior packet delivery and route length performance. This is contradictory to some of our results.

A more recent work, by Johansson, Larssson, Hedman and Mielczarek [14], extended the above work by using new mobility models. To characterize these models, a new mobility metric is introduced that measures mobility in terms of relative speeds of the nodes. In low loads DSR was more effective, while AODV was more effective at higher loads. The packetwise routing load of DSR was almost always significantly lower than AODV. The authors attributed the comparative poor performance of DSR to the source routing overheads in data packets. They used small data packets (64 bytes), thus making things somewhat unfavorable for DSR.

In these works, the routing protocols were compared only under varying traffic load or the mobility patterns. The performance of the routing protocols weren't tested under different transmit power levels, which we found to be a very significant performance factor which affects the routing protocols in very different ways.

Other than comparisons, there are also several recent papers that have dealt with DSRs caching performance, an important performance determinant in our experience as presented in this thesis. In [6], the authors concluded that even though many cache replies carried stale routes, route maintenance in DSR is able to adapt and deliver good performance which conflicts most of our results. However, Holland et al. [16] have shown that the stale caches in DSR have a harmful effect on TCP performance in a mobile environment, and observed that performance could be improved by switching off replies from caches. More recently, the effects of cache structure, cache capacity, cache timeouts, and mobility patterns on the performance of DSR were studied [17]. It was observed that, in general, expiration of cached routes improved performance. We observed that in stationery networks, caching is actually the main factor that makes DSR overperform AODV.

Another study has described details of physical layer modeling in OPNET Modeler [1], ns-2 [2] and GloMoSim [4] [18]. They showed the impacts of their differences on the overall network performance for scenarios typically used for the evaluation of ad hoc routing protocols. However, this difference cannot be interpreted as stemming only from the difference of PHY layer models in these simulators. These three simulators have many other differences than their PHY layer models.

Also in another work [19], they concluded that simulations which lack necessary details can result in misleading or incorrect answers, therefore, researchers must chose their level of simulation detail with care. They have offered several case studies in wireless network simulation to offer guidance for when detail is or is not required. However, they didn't focus on details in packet reception modeling.

The most similar study to ours is given in [20] and [21]. In these works, they have implemented an extensive indoor radio propagation model and included an implementation

of SINR tracking model in the physical layer model of ns-2. However, no details are given about the SINR tracking model, and there are no comprehensive ad hoc routing simulations with this model. The authors also noted that their simulations required a runtime of up to 100 times longer than standard ns-2. By comparison, we didn't observe such a degradation. They concluded that, as the power of processors increase drastically, heavy computation will become less of an issue and their modeling simulator may be used to re-run previous, well-known validations such as [13]. Our work, in fact, does simulate a number of the large-scale scenarios given in [5] and [13]. Therefore, to our knowledge, this is the first study to have made extensive ad hoc routing protocols evaluation with a modified PHY layer model in ns-2.

Chapter 2

802.11b PHY Layer

In this chapter we present the details of 802.11b PHY layer which may affect the results of a simulation. In the next chapters we also explain the level of detail of ns-2's physical layer model.

The IEEE 802.11b standard actually provides three variations for PHY layer. These include Direct Sequence Spread Spectrum (DSSS), Frequency Hopping Spread Spectrum (FHSS), and Infrared. In practice, only DSSS has any significant presence in the market.

802.11b DSSS operates in the 2.4Ghz ISM band which is allocated from 2400 to 2483Mhz. The channel assignments in North America are channels 1 to 11, starting at 2412Mhz and spaced at 5Mhz intervals to 2462Mhz. Each channel is about 22Mhz wide so there is substantial overlap. Therefore, channels 1, 6, and 11 can be used as non-overlapping channels.

The DSSS system has different modulation modes for every transmission rate. These are:

- Differential Binary Phase Shift Keying (DBPSK) for 1Mbps,
- Differential Quaternary Phase Shift Keying (DQPSK) for 2Mbps,
- Complementary Code Keying (CCK) for 5.5Mbps and 11Mbps.

The spreading is performed by multiplying binary data by a pseudo random (PN) binary

waveform. At 1 and 2Mbps, the PN code is an 11-chip long Barker sequence.

For the CCK modulation, 8-chip long Walsh codes are used. The transmitter divides the data into 4-bits or 8-bits. At 5.5Mbps, 2 of the 4 bits are used to select one of 4 complex spread sequences from a table of CCK sequences and then DQPSK modulates this sequence with the other two bits. At 11Mbps, 6 bits are used to select one of 64 sequences and the remaining 2 bits are used for modulation. For 5.5Mbps data rate, 4 bits are encoded into the 8-chip long codeword. So the processing gain is only 2. For the 11Mbps data rate, there's no processing gain because 8 bits are encoded into 8-chips.

2.1 Details of Data Transmission and Reception

Like other 802.11 physical layers, 802.11b has Physical Layer Convergence Procedure (PLCP) and Physical Medium Dependent (PMD) sub-layers. The standard uses these terms to divide the major functions that occur within the physical layer. The PLCP maps the IEEE 802.11 MAC protocol data units (MPDU) into a framing format suitable for sending and receiving user data and management information between wireless nodes. Then the PLCP directs these frames to the PMD to actually transmit and receive signals.

The PLCP takes each 802.11 frame that a station wishes to transmit and forms what the 802.11 standard refers to as a PLCP protocol data unit (PPDU). The resulting PPDU includes the following fields:

- **Sync:** This field consists of alternating 0s and 1s, alerting the receiver that a receivable signal is present. The receiver begins synchronizing with the incoming signal after detecting the Sync.
- **Start Frame Delimiter (SFD):** This field is always 1111001110100000 and defines

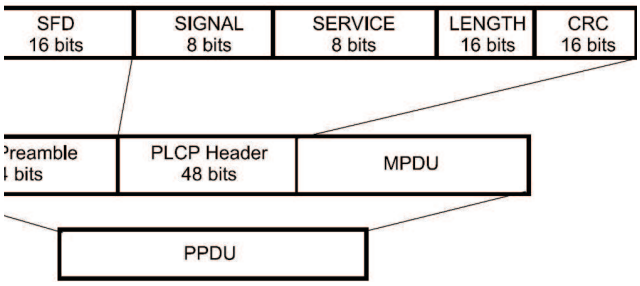


Figure 2.1: PPDU Frame Format

the beginning of a frame.

- **Signal:** This field identifies the data rate of the 802.11 frame. The preamble and header are always transmitted using the 1Mbps DBPSK modulation, while the MPDU can be transmitted using DBPSK, DQPSK or CCK. This ensures that the receiver is initially uses the correct demodulation mechanism, which changes with different data rates.

- **Service:** This field is always set to 00000000, and the 802.11 standard reserves it for future use.

- **Length:** This field represents the number of microseconds that it takes to transmit the contents of the PPDU, and the receiver uses this information to determine the end of the frame.

- **Cyclic Redundancy Check (CRC):** In order to detect possible errors in the Physical Layer header, the standard defines this field for containing 16-bit cyclic redundancy check (CRC) result. The MAC Layer also performs error detection functions on the PPDU contents as well.

- **MPDU:** This is the actual 802.11 frame that's being transmitted.

2.1.1 Data Transmission

In order to transmit data, the PHY layer should be in the transmit state. Based on the status of clear channel assessment (CCA), the MAC assesses that the channel is clear. If the channel is clear, transmission of the PPDU is initiated. This means defining the PLCP Header parameters. Then the PHY entity initiates data scrambling and transmission of the PLCP Preamble.

Once the PLCP Preamble transmission is complete, data is exchanged between the MAC and the PHY. The modulation rate change, if any, is initiated with the first data symbol of the MPDU. The PHY will proceed with MPDU transmission through a series of data octet transfers from the MAC. Termination will occur after the transmission of the final bit of the last MPDU octet according to the number supplied in the DSSS PHY preamble LENGTH field. The packet transmission will be completed and the PHY entity will enter the receive state.

2.1.2 Data Reception

In order to receive data, the PHY layer should be in the receive state. Further, the PHY is set to the appropriate channel and the Clear Channel Assessment (CCA) method is chosen. Upon receiving the transmitted energy, according to the selected CCA mode, an indication of the RSSI strength reaching the energy detection threshold (EDT) and/or presence of a DSSS signal (code locking) will be enabled. These conditions are used to indicate activity to the MAC. After this, the PHY entity shall begin searching for the SFD field. Once the SFD field is detected, PLCP header will be received. Then the CRC check will be processed. If it fails, the PHY receiver shall return to the RX Idle state. If the PLCP Header reception is successful (and the SIGNAL field is completely recognizable and supported), the rate

change indicated in the IEEE 802.11 SIGNAL field shall be initiated with the first symbol of the MPDU. After the reception of the final bit of the last MPDU octet indicated by the PLCP Preamble LENGTH field, the receiver will return to the RX Idle state.

In case of a decrease in RSSI or loosing code lock before the complete reception of the MPDU as indicated by the PLCP LENGTH field, the error condition will be reported to the MAC. However, the DSSS PHY will ensure that the CCA still indicates a busy medium for the intended duration of the transmitted packet.

2.1.3 Clear Channel Assessment (CCA)

There are three methods to perform CCA:

- *CCA Mode 1*: Energy above threshold. CCA shall report a busy medium upon detection of any energy above the ED threshold.
- *CCA Mode 2*: Carrier sense only. CCA shall report a busy medium only upon detection of a DSSS signal. This signal may be above or below the ED threshold.
- *CCA Mode 3*: Carrier sense with energy above threshold. CCA shall report a busy medium upon detection of a DSSS signal with energy above the ED threshold.

Among these modes, mode 3 is usually set as default. However, it can be changed through the driver [22].

Chapter 3

Fundamentals of Ns-2

3.1 Introduction to Ns-2

Ns-2 is a discrete event network simulator that began in 1989 as a variant of the REAL network simulator which was initially intended only for wired networks. It is widely accepted as an environment for studying TCP and other protocols over networks like the conventional internet. However the increasing popularity of Mobile Ad Hoc Networks (MANET) [23] prompted researchers to extend ns-2 to provide support for the simulation of wireless LANs. In the most successful wireless extensions, developed by the Monarch Group at CMU, the ns-2 radio model is based on the DSSS PHY reference configuration in the IEEE 802.11 standard [24], but with the parameters of the older 914 MHz WaveLAN card [25]. As the Monarch extensions are included in the standard ns-2 distribution [13], we use the name ns-2 to denote both the simulator and the Monarch extensions.

3.2 Ns-2 Abstraction

Ns-2 is an object oriented simulator, written in C++, with an OTcl (object oriented Tcl) interpreter as a frontend. The simulator supports a class hierarchy in C++ (also called the compiled hierarchy), and a similar class hierarchy within the OTcl interpreter (also called the interpreted hierarchy). The two hierarchies are closely related to each other; from the user's perspective, there is a one-to-one correspondence between a class in the

interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is the class TclObject. Users create new simulator objects through the interpreter; these objects are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy. The interpreted class hierarchy is automatically established through methods defined in the class TclClass. User instantiated objects are mirrored through methods defined in the class TclObject.

Ns-2 uses two languages because simulator has two different kinds of tasks it needs to do. On one hand, detailed simulations of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important.

On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios. In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important.

Ns-2 meets both of these needs with two languages, C++ and OTcl. C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration.

3.3 Mobile Networking in Ns-2

This section briefly describes the wireless model that was originally ported as CMU's Monarch group's mobility extension to Ns-2.

In this extension, mobile node is implemented as an object with functionalities such as

movement and the ability to transmit and receive on a channel that allows it to be used to create mobile, wireless simulation environments. The mobile node is designed to move on a flat terrain with height always equal to zero. Thus the mobile node has X, Y, Z co-ordinates that are continually adjusted as the node moves. When creating a mobility scenario, the starting position of the node and its future destinations may be set explicitly. These directives are normally included in a separate movement scenario file. Other than this, the topography for mobile nodes always needs to be defined. Normally a flat topology is created by specifying the length and width of the topography.

The network stack for a mobile node consists of a link layer, an ARP module connected to the link layer, an interface priority queue, a MAC layer and a network interface, all connected to the channel. Each component is briefly described here.

Link Layer: The link-layer object is responsible for simulating the data link protocols. Many protocols can be implemented within this layer such as packet fragmentation and reassembly, and reliable link protocol.

Another important function of the link layer is setting the MAC destination address in the MAC header of the packet. Normally for all outgoing (into the channel) packets, the packets are handed down to the link layer by the Routing Agent. Then the link layer hands down packets to the interface queue. For all incoming packets (out of the channel), the MAC layer hands up packets to the link layer.

ARP: The Address Resolution Protocol module receives queries from Link layer. If ARP has the hardware address for destination, it writes it into the MAC header of the packet. Otherwise it broadcasts an ARP query, and buffers the packet temporarily. Once the hardware address of a packet's next hop is known, the packet is inserted into the interface queue.

Interface Queue: For the purposes of ad hoc routing, the interface queue is implemented as a priority queue which gives priority to routing protocol packets by inserting them at the head of the queue.

MAC Layer: The IEEE 802.11 distributed coordination function (DCF) MAC protocol has been implemented by CMU. DCF is similar to MACA and MACAW and is designed to use both physical carrier sense and virtual carrier sense mechanisms to reduce the probability of collisions due to hidden terminals. The details of this implementation will be covered in the next section along with the network interface implementation which is used by mobile nodes to access the channel.

3.4 Packet Reception in Ns-2

In ns-2, each mobile node has one or more wireless network interfaces, linked together by a single physical channel. When a network interface transmits a packet, it passes the packet to the appropriate physical channel object. This object then computes the propagation delay from the sender to every receiver on the channel and schedules a packet reception event for each. This event notifies each receiving interface when the first bit of a new packet has arrived.

After this notification, a receiver at distance d computes the received power of the packet to be $P_r = G(d)P_t$ where P_t is the transmitter power and $G(d)$ is the link gain from the transmitter to the receiver. The link gain $G(d)$ is calculated either by the Friis free space model [26],

$$G^{(1)}(d) = \frac{G_t G_r \lambda^2}{(4\pi d)^2 L}, \tag{3.1}$$

or the two-ray ground model [26],

$$G^{(2)}(d) = \frac{G_t G_r (h_t^2 h_r^2)}{d^4 L}. \tag{3.2}$$

Note that G_t and G_r are the transmitter and receiver antenna gains which have default value 1, L is the system loss which has a default value 1, h_t and h_r are the heights of the transmit and receiver antennas which have default value 1.5 m. If d is less than the distance $d_0' = 4\pi h_t h_r / \lambda$ where $G^{(1)}(d_0') = G^{(2)}(d_0')$, the Friis equation is used. Otherwise the two-ray ground model is used to compute the received power of the packet. It follows that

$$G(d) = \min(G^{(1)}(d), G^{(2)}(d)).$$

The received power level of the arriving packet is then compared to two different values: the carrier sense threshold (CST) and the receive threshold (RXT). The CST has two functions.

- If the received power level is below CST, the packet is discarded as noise; the receiver interface operates as if that packet never existed.
- CST is also used for purposes of CSMA/CA. The transmitter cannot start transmission of a new packet if it senses another signal with a received power level higher than CST.

This use of CST matches the definition of Clear Channel Assessment (CCA) Mode 3 in the IEEE 802.11 Standard.

RXT is a received power threshold that is used to decide whether a packet is received correctly. In the Monarch extensions, RXT equals the received power $P_r^{(2)}(d_0)$ at a given

distance d_0, independent of the signal modulation and data rate of the transmission. If the received power level is above CST but below

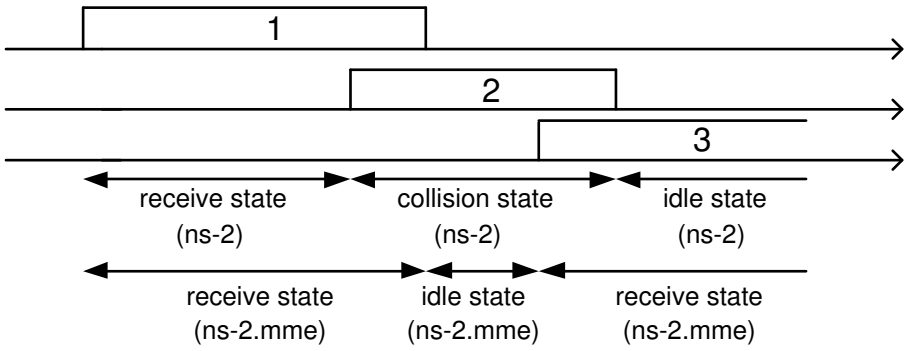
RXT, the packet is marked as a packet in error before being passed to the MAC layer (the packet is detected, but not successfully received). Otherwise, the packet is simply handed to the MAC layer.

A receiver's MAC layer is modeled as a state machine with the three states.

- **Idle State:** The MAC layer is ready to start decoding a new packet.
- **Receive State:** The MAC layer is decoding a packet.
- **Collision State:** While in receive state, the packet currently being decoded has suffered a collision.

When a node in the idle state receives a packet, the MAC layer switches to the receive state, computes the transmission time of the packet and schedules a "packet reception complete" event. If the MAC layer is not in the idle state, there are two possibilities. If the power level of the packet already being decoded is at least 10dB greater than the received power level of the new packet, a capture effect takes place, the new packet is discarded, and the receiver interface continues decoding its current packet. Otherwise, a collision occurs and both packets are dropped. If the scheduled "packet reception complete" event can occur without a collision, the MAC layer verifies that the packet is error-free, performs destination address filtering, and passes the packet up the protocol stack.

In the event of a collision, the MAC layer switches into the collision state and stays in this state until the both colliding packets have completed transmission; see Figure 3.1. This rule prevents the transmitter interface from starting a new transmission during the transmission of the colliding packet. For a transmitter, this behavior is consistent with



In both ns-2 and ns-2.mme, the receiver enters the receive state and starts to decode packet 1. In ns-2, the transmission of packet 2 causes a collision, and the receiver stays in the collision state until both colliding packets complete transmission. (In fact, the collision may be extended to the completion of packet 3 if its received power is sufficient.) In ns-2.mme, the receiver stays in the receive state until Packet 1 completes transmission, and if the received power of Packet 3 is higher than CST, the receiver again enters the receive state.

Figure 3.1: **Change of receiver states**

the CSMA/CA standard, which prevents a new transmission when there is a packet in the medium with a received power level higher than CST. However, this behavior also prevents the receiver interface from attempting to receive any new packets until the colliding packet reception ends. In Figure 3.1 for example, even if the received power of packet 3 is 10 dB higher than the received power of packet 2, the receiver cannot start the reception of packet 3. (In fact, the collision state continues until the end of packet 3; however, we should note that this is a rare event which can only take place when the RTS/CTS option is off.)

Chapter 4

Modifications

In general, packets are decoded reliably when the received power of a packet is sufficient to overcome the interference of other transmissions and thermal receiver noise. The ns-2 model of a 10 dB capture threshold for potential colliding packets is a simple abstraction of the actual packet reception process. This model works well when the transmission of each unicast packet is preceded by an RTS/CTS exchange that can be heard by most network nodes. In these conditions, the number of instantaneous interfering transmissions will be small and the probability of correct packet reception is well modeled by single packet interference comparisons. However, in the presence of multiple interfering transmissions, this model will be inaccurate. In this chapter, we describe modifications to the Monarch extensions to account for these simultaneous multiple interferers.

4.1 An SINR Interference Model

In the single user additive white Gaussian noise channel, the probability of a bit error is a monotonically decreasing function of the signal noise ratio (SNR). In systems with interference from other users, it is common practice to model the communication link quality by the signal to interference plus noise ratio (SINR) [27–29]. To formulate the SINR, we say that on communication link j, transmitter j employs power P_j to send to receiver j. We use G_{ij} to denote the power gain from the link j transmitter to the link i receiver. At

the link i receiver, the SINR is

$$\gamma_i = \frac{G_{ii}P_i}{\sum_{j\neq i}\theta_{ij}G_{ij}P_j + \eta}. \tag{4.1}$$

Note that η is the in-band receiver noise power and includes both thermal noise as well as the receiver noise figure [30]. In addition, θ_{ij} represents the fraction of transmitter j's received signal power that is projected onto the signal space of user i. For example, in a synchronous CDMA system with matched filter detection, θ_{ij} equals the normalized squared cross-correlation between the signature sequences of users i and j [31]. In general, the interference factor θ_{ij} may depend on the spreading codes, modulation formats, and data rates of the users. Analysis has also shown that θ_{ij} may also depend on such factors as the synchronism (or asynchronism) of the users' transmissions [32] as well as receiver hardware implementation design choices such as the number of bits in the analog to digital converter [33]. In certain spread spectrum systems, θ_{ij} may be reduced if the receiver employs filtering in the form of multiuser detection [31]. The interference factor θ_{ij} may also model interfering signals that overlap the frequency spectrum of user i. For example, if link j is a UWB transmitter spreading over 3–10 GHz and link i is a 22 MHz 802.11a transmission in the 5GHz band, then $\theta_{ij} = 22/7000$ would represent the fraction of the power of user j in the band of user i. In a similar way, θ_{ij} could be used to model the partially overlapping channels of 802.11b.

For a link i transmitting at data rate R_i b/s, a common model in spread spectrum systems with matched filter detection is to assume that $\theta_{ij} = \theta_i = R_i/W$, corresponding to the reciprocal of the processing gain $N_i = W/R_i$ [28]. Prior analyses of CDMA systems that concluded θ_i is proportional to $1/N_i$ were based on the assumption that both the

processing gain N_i and the number of interfering users are relatively large. For example, second generation cellular CDMA systems employ a processing gain of 128 and support 10-20 simultaneous transmissions in a single cell. Thus a CDMA analysis may not be appropriate, even for 802.11b systems operating at 1 or 2 Mb/s where the spreading factor is only 11.

In the context of 802.11 systems, the appropriate value of θ has received little attention, precisely because the carrier sense and RTS/CTS mechanisms have been designed to preclude all but very weak interfering transmissions. In the subsequent discussion, we assume a homogeneous single-rate system in which $\theta_{ij} = \theta$ for all communication links. The fundamental change to the physical layer model is that in the receive state, a receiver node i will track its SINR γ_i. Fluctuations in the SINR will determine whether a packet is received correctly. In the sequel, we refer to ns-2 with *modified Monarch extensions* to support SINR tracking as ns-2.mme.

4.2 A Model for Packet Reception

We adopt a physical model for packet reception consistent with the IEEE 802.11 protocol. When the receiver is in the idle state and the received power level of a new packet is higher than CST, the MAC layer enters the receive state and stays in this state until that packet transmission is complete. This model corresponds to the physical situation that a receiver learns the packet length and data rate from the packet header and uses this information to specify how many bits the receiver will decode. After decoding these bits, a CRC check identifies whether the bits of the packet were decoded correctly. Packets that are received correctly are then passed up the protocol stack.

The probability of correct packet reception in ns-2.mme depends on the receiver SINR,

which may vary during a single packet reception. The next modeling step is to translate the SINR into a packet error probability. This translation mechanism may embed such factors as modulation rate and error control coding. This translation may be as simple as a threshold such that the packet is correctly received only if the SINR has been above threshold during the entire packet duration [4]. Alternatively, the packet error probability may be an arbitrary function of the SINR that is derived analytically, or possibly empirically by physical layer experimentation or simulation.

In this thesis, we develop a BER based model that describes a system with uncoded packets in which the detector makes a hard decision on each transmitted bit. This BER based model probabilistically decides whether each bit in a packet is transmitted correctly based on the receiver SINR during that bit reception. At every node, the total received power P_{total} from all signal sources is stored and is updated every time a packet transmission begins or ends. SINR tracking is implemented just by tracking the total received power as follows:

- When a new packet arrives, increase P_{total} by the received power of that packet.
- When a packet completes transmission, decrease P_{total} by the received power of that packet.

If a node is receiving a packet with received power P_r and the total received power is P_{total}, the SINR is

$$\gamma = \frac{P_r}{\theta[P_{\text{total}} - P_r] + \eta}. \tag{4.2}$$

For packet decoding, we define a *segment* as a consecutive sequence of received bits over which the SINR is constant. In our BER-based model of packet reception, we determine whether a packet has errors as a function of the SINR in each packet segment. Based on

these segments, the packet reception algorithm is:

- For a given segment, find the bit error rate by using the pre-computed BER-SINR table.
- If the segment has n bits, calculate the probability

$$P_C = (1 - P_e)^n \tag{4.3}$$

  that all bits in the segment are decoded correctly.
- Throw a uniform random variable between 0 and 1. If this number is greater than P_C, mark this segment with error.
- At the end of decoding a packet transmission, check if there was a decoding error in any packet segment. If so, discard the packet; otherwise, the packet is received correctly. Note that the IEEE 802.11b standard does not use coding. Thus if there is a single segment with error, that packet will fail a CRC check.

Unlike ns-2, ns-2.mme has no need for a collision state to realize the receiver state machine; see Figure 3.1. Instead, one or more interfering transmissions in ns-2.mme will reduce the receiver SINR during packet reception and cause a packet decoding error. Whether by collision or decoding error, the first order effect of a strong interferer is the same: a transmitted packet is lost. However, there are second order differences. With SINR tracking, the probability of correct packet decoding will depend on the cumulative effect of all interfering transmissions. In addition, after completing the decoding of a received packet, an ns-2.mme receiver simply returns to the idle state, which makes it ready for starting reception of a new packet. By contrast, under ns-2, the node enters the collision state and remains in

the collission state until both colliding packets complete transmission. This difference is depicted in Figure 3.1. Generally this difference will have little impact, except in unusual cases like that of the figure, where if the received power of the packet 3 is higher than CST, the receiver can start decoding the packet.

4.3 Designing the BER-SINR look-up table

As mentioned in the previous section, we designed a BER-SINR look-up table for our packet reception model. For the BER of DBPSK we used the following equation as given in [34],

$$P_e = \frac{1}{2} e^{-\gamma_b}. \tag{4.4}$$

Here γ_b is the SINR per bit. For DBPSK, γ_b is equal to γ which is given in Equation (4.2).

For BER of DQPSK, we used the following equation which is also given in [34],

$$P_e = Q_1(a, b) - 0.5 I_0(ab) e^{-2\gamma_b}, \tag{4.5}$$

where

$$a = [2\gamma_b(1 - 1/\sqrt{2})]^{1/2}, \tag{4.6}$$

$$b = [2\gamma_b(1 + 1/\sqrt{2})]^{1/2}, \tag{4.7}$$

$Q_1(a, b)$ is Marcum Q function and I_0 is the modified Bessel function of the first kind of order 0. For DBPSK, γ_b is equal to $\gamma/2$. This is because of the fact that in 2Mbps data rate the energy used for transmitting a single bit is the half of the energy that's used in 1Mbps.

For BER of CCK, we used a simplified version of the BER computation given in [35]. In

CCK, n of the $n+2$ bits are used to select one of 2^n complex spread sequences from a table of CCK sequences and then DQPSK modulates this sequence with the other two bits. Let's denote these n bits as $B = (b_0, b_1, b_2, ..., b_n)$ which can also be denoted as $B = (B_1, B_2)$, where $B_1 = (b_0, b_1)$ and $B_1 = (b_2, ..., b_n)$. There are 2^{n+2} codewords since they are mapped from $n + 2$ binary data bits. There are 2^n different codewords determined by B_2. These 2^n codewords have ideal cross-correlation properties due to orthogonality. To find the bit error rate for CCK let's first find the probability of deciding bits B_2 correctly. Since they are orthogonal, here we used the union bound for its simplicity;

$$P_c(B_2) = (M-1)Q(\gamma_b \log_2 M). \tag{4.8}$$

where $M = 2^n$ and Q is the normal distribution function. Given that B_2 has been decided correctly, data bits B_1 are DQPSK demodulated. So, $P_b(B_1|B_2)$ is calculated as given in Equation (4.5).

B_1 consists of two data bits, hence the relation between the conditional probabilities of symbol error and bit error can be expressed as [34],

$$P_b(B_1|B_2) = \frac{2^{2-1}}{2^2-1} P_s(B_1|B_2). \tag{4.9}$$

So, the conditional probability of making correct decision on B_1 is,

$$P_c(B_1|B_2) = 1 - P_s(B_1|B_1) = 1 - 1.5P_b(B_1|B_2). \tag{4.10}$$

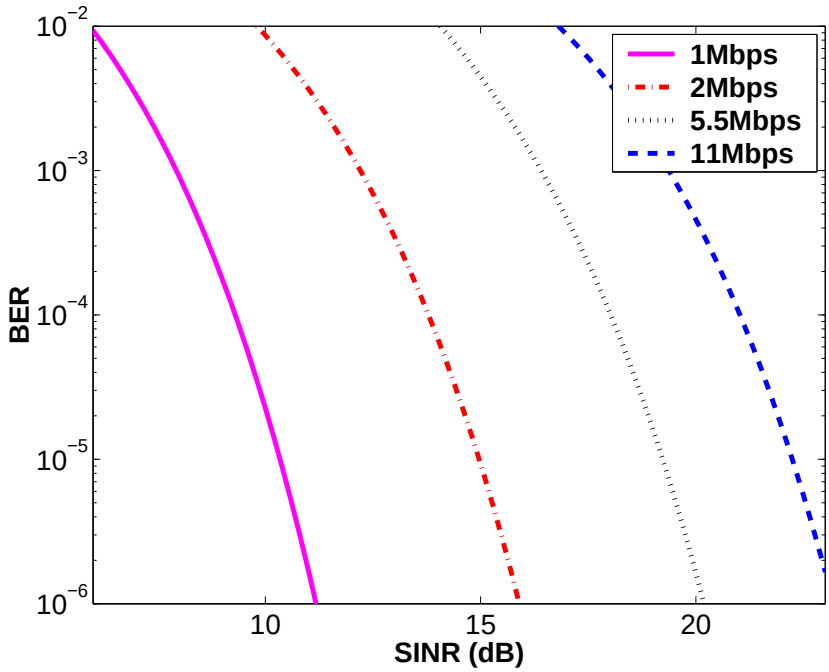


Figure 4.1: BER vs γ

The codeword can be decided correctly when both B_1 and B_2 have been decided correctly. So, the probability of a correct codeword decision is:

$$P_c(C) = P_c(B_1B_1) = P_c(B_2)P_c(B_1|B_2). \tag{4.11}$$

So, the bit error rate is,

$$P_b = \frac{2^{n+2-1}}{2^{n+2}-1}[1 - P_c(B_2)P_c(B_1|B_2)]. \tag{4.12}$$

At 5.5Mbps, $n = 2$ and $\gamma_b = \gamma/5.5$ where γ is given in Equation (4.2). At 11Mbps, $n = 6$ and $\gamma_b = \gamma/11$.

By using these equations, we created a BER-SINR look-up table spanning SINR from 0 dB to 30 dB with 0.1 dB intervals with a precision of 10^{-12} in BER. A plot of this look-up

table is shown in Figure 4.1.

4.4 A Model for Carrier Sensing

In the 802.11 protocol, there are two kinds of carrier sensing, one is called physical carrier sensing and the other is called virtual carrier sensing. Virtual carrier sensing is done by employing an RTS/CTS exchange before transmitting a unicast data packet. On the other hand, physical carrier sensing is done by using the clear channel assesment (CCA). Ns-2 handles physical carrier sensing by using the CST. However, as we explained, ns-2 is incapable of cumulative received power tracking. Thus, in ns-2 the wireless medium is reported as busy when received power of a single packet is higher than CST.

We implemented a carrier sensing system which compares the total received power at the wireless card by a power threshold. For simplicity, we will call this as "cumulative carrier sensing" (CCS) and the current implementation in the ns-2 as CS. In the original ns-2 code, physical carrier sensing is handled by a timer. When there is a packet with a received power higher than CST the timer is set to the end of that packet's transmission. So, when that packet's transmission ends, the MAC layer switches to idle state assuming no other packet is currently being transmitted or received and RTS/CTS option is turned off. We changed this implementation in the following way:

- When a packet with a received power higher than CST arrived at the node the timer is set just like in the original code.
- When this timer expires, the MAC layer checks if the total received power level at the node is lower than CST. If it's lower, the wireless medium is indicated as idle.
- If the total received power at the node is still higher than the CST, the medium

cannot be indicated as idle until the total received power drops below CST. In order to check if this is the case, after the ending of every single packet's transmission the node compares the total received power to the CST.

4.5 A Model for PLCP Preamble and Header

As explained in section 2.1, PLCP preamble and header are essential in packet reception. So, after implementing our primary packet reception model, we also implemented a secondary model which takes the PLCP preamble and header into consideration. In this model, we treated all the fields in the PLCP preamble and header as a single field that has to be decoded correctly. The details are as follows:

- When a node is receiving a packet, after receiving the PLCP Preamble and PLCP Header, which occupy the first 192 bits, it calculates the probability that all bits are decoded correctly. This is done as explained in section 4.2.
- If there's a decoding error, the MAC layer returns into the idle state. Otherwise, MAC layer starts receiving the rest of the packet, which is the MPDU. Then the same steps in section 4.2 are the followed to determine if the packet is correctly received or not.

4.6 Modifications in the ns-2 code

This section describes our actual modifications to the Monarch extensions. The main modifications are distributed between the Mac802.11 and WirelessPhy objects. In WirelessPhy, when a new packet comes, it's received power P_r is computed and the value of the total received power variable P_{total} which is stored in the MobileNode object is increased by P_r.

A "packet transmission complete" event is then scheduled. When this event occurs, P_{total} is decreased by P_r. Unlike standard ns-2, no matter what the received power P_r of a new packet is, the receive function of the Mac802.11 object is called. CST is now employed in the Mac802.11 object. That is, even if the receive function of the Mac802.11 object is called, an actual reception will begin if the received power of the new packet is higher than CST. Every time the receive function is called, using the P_{total} in the Mobilenode object, the Mac802.11 computes the SINR as shown in Equation (4.2) and executes the packet reception algorithm for the *preceding* packet segment, as explained in section 4.2. This algorithm is also executed when a "packet transmission complete" event occurs. In this case, WirelessPhy calls the receive function in Mac802.11 just to execute the packet reception algorithm. Note that the last packet segment is evaluated when the packet reception ends.

There are three important variables that are used in the packet reception algorithm:

- **Error flag:** This variable is set to 1 when a *segment* has errors. The error flag is reset to zero when a new packet arrives.
- $\text{SINR}_{\text{previous}}$: This variable is the SINR value of the previous *segment*.
- $\text{time}_{\text{lastupdate}}$: This variable indicates the last time the SINR was updated. In the previous packet segment, the number of bits n, as used in Equation (4.3), is computed using $\text{time}_{\text{lastupdate}}$ and time_{now}, the time marking the SINR change. For a link operating at rate R_i b/s,

$$n = R_i \times (\text{time}_{\text{now}} - \text{time}_{\text{lastupdate}}). \tag{4.13}$$

4.7 Radio Parameters

In the following chapter, we will use simulation to determine the effect of modifications to the physical layer models. To do this, we must select system parameters such that results from ns-2 and ns-2.mme are comparable at the same transmit power P_t despite the differences in physical layer models.

Under the reasonable assumption that the noise figure of a simulated 802.11 radio is $F = 5$ dB [30], it follows from the $2W = 22$ MHz bandwidth of an 802.11b signal and thermal noise $N_0/2$ of -174 dBm/Hz that the received in-band noise power is

$$10 \log_{10} \eta = 10 \log_{10} F + 10 \log_{10}(2W) - 174 = -95.6 \text{ dBm}$$

In this case, the RXT value of -64 dBm implies that the ns-2 simulated receiver correctly receives packets having SNR greater than 32 dB. For an 802.11b receiver operating at 1 Mb/s, this figure is conservative in that it overestimates the required SNR by perhaps 20 dB. For example, consider a 512 byte packet received with SNR $\gamma = 10$ dB. Using the equation for DBPSK bit error probability given in Equation (4.4) the bit error probability is $P_e = 0.5e^{-10} = 2.27 \times 10^{-5}$ and the probability of correct packet decoding is $P_C = (1 - P_e)^{4096} = 0.91$. Ns-2 does partly compensate for the very high RXT by setting the default transmit power to $P_t = 282$ mW (24.5 dBm), roughly 10-15 dB higher than values specified for modern cards.

In addition to RXT, CST is another parameter for which ns-2 which does not reflect recent 802.11b cards' specifications. CST should be lower than the "receiver sensitivity," a (somewhat ambiguously defined) manufacturer's specification of the lowest SNR at which a wireless card can acquire a signal and detect bits with a specified reliability. Despite

	ns-2.1b9a original parameters	ns-2.1b9a revised parameters	ns-2.mme
P_t default (dBm)	24.5	variable	variable
CST (dBm)	-78	-81, -84	-81, -84
RXT (dBm)	-64	-78	none
η (dBm)	none	none	-87
CS Range (@ $P_t = 24.5$ dBm)	550m	650m, 775m	650m, 775m
Transmit Range (@ $P_t = 24.5$ dBm)	250m	550m	550m

Table 4.1: Ns-2 parameter values.

the ambiguity of the definition, today's wireless cards are very sensitive. For example, the receiver sensitivity of a Cisco Aironet 350 card is -94 dBm for 1Mbps data rate [22] which is 16 dB below ns-2's default CST value of -78 dB. Nevertheless, as a result of the pairwise collision model, it has made little difference that ns-2 uses values for the CST, RXT, and transmitter power P_t corresponding to the outmoded 900 MHz adaptors. A more realistic set of parameters yielding the same transmit range and CS range would yield identical wireless network performance.

In general, the transmit range denotes the maximum transmitter-receiver separation such that a packet will be decoded correctly. In ns-2, the transmit range is determined by RXT threshold, the transmit power P_t, and the link gain function $G(d)$. For $P_t = 24.5$ dBm and RXT $= -64$ dBm, the range is 250 m. In ns-2.mme, the transmit range is less well defined. In the absence of interference, the SINR $\gamma = P_t G(d)/\eta$, the BER P_e, and the packet success probability P_C are continuous functions of the distance d. However, it is generally true that P_C will make a sharp transition from nearly 1 to almost 0 at a critical value of the SINR. Thus we define the transmit range for ns-2.mme as the distance where the data packet error rate is 0.5 in a zero-interference environment. Although this definition of range

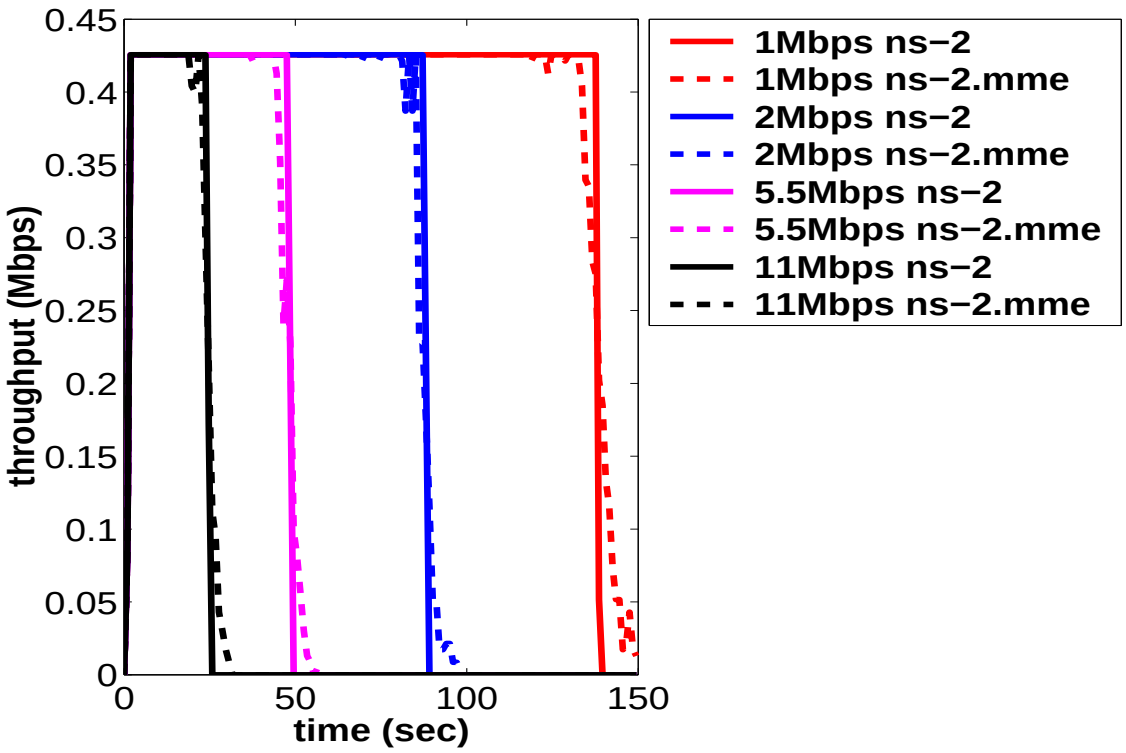


Figure 4.2: A Two Node Experiment: The range of revised ns-2 and ns-2.mme for transmit power P_t=10dBm with 512-byte packets

is somewhat arbitrary and depends on the data packet length, we will see shortly that it does make sense for the BER-based packet reception algorithm proposed here for uncoded packets. It also makes sense in a wide variety of systems with forward error correction where it is common for the bit error probability to exhibit a sharp transition.

For the BER-based decoding described in this thesis, at 1Mbps, P_C makes a very sharp transition around $\gamma = 9$ dB. In fact, for 512 byte packets, $P_C = 1/2$ at $\gamma = 9$ dB. This sharp transition can be seen in a simple simulation involving just two nodes (one transmitter and one receiver) with $P_t = 10$ dBm. When the simulation starts at $t = 0$, node 1 is at (0,0) and node 2 is at (100,0), and node 2 starts to move away from the transmitter at a speed of 1 m/s. In Figure 4.2, with 512 bytes packets, we see that the throughput at 1Mbps makes a sharp transition at $t = 140$ m, corresponding to $\gamma = 9$ dB. At first glance, a range of

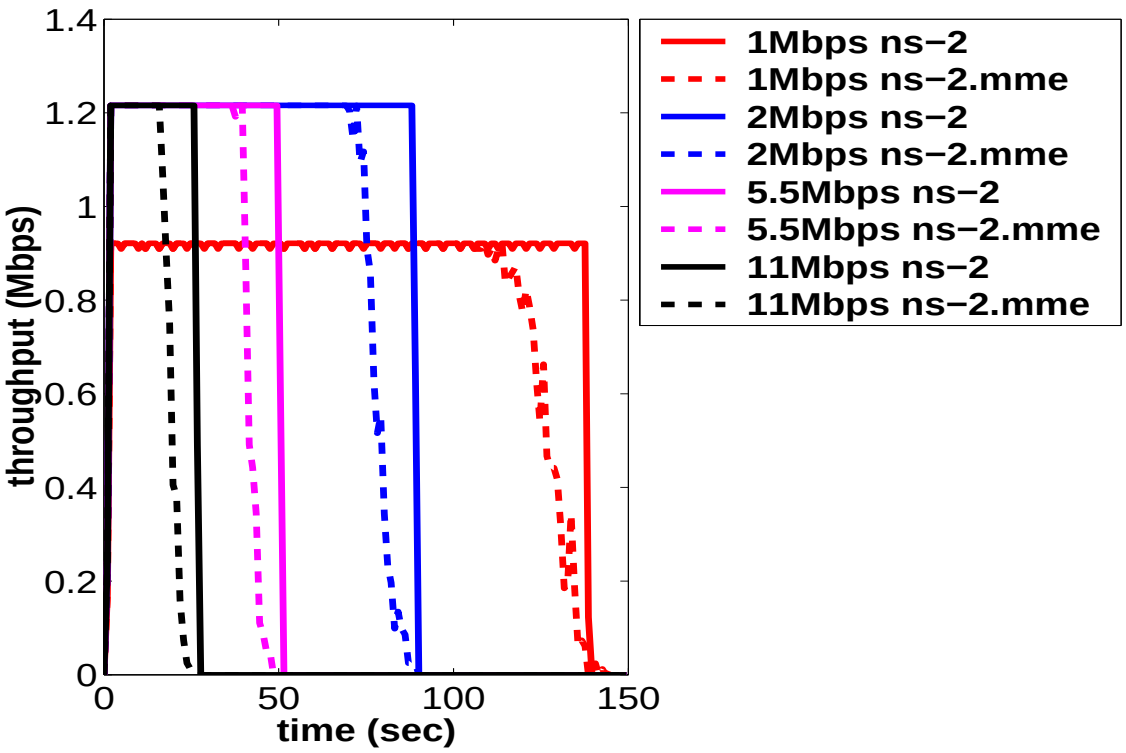


Figure 4.3: A Two Node Experiment: The range of revised ns-2 and ns-2.mme for transmit power P_t=10dBm with 1500-byte packets

240 m may seem too high at $P_t = 10$ dBm, but this is representative of the range of today's wireless cards in outdoor environments [22].

In Figure 4.2, we can also see the ranges for the other data rates as well. The sharp transition is seen at various distances depending on the data rate. However, when we look at the results of the same simulation with 1500 bytes packets we see that there's a gap between the ranges for ns-2 and ns-2.mme (Figure 4.3). This is because of the fact that our definition of range depends on the data packet length. When we use 1500 bytes packets the SINR required for all the bits to be correct is higher than the 512 bytes packets case. The opposite behavior can be also observed with much smaller packets. For example, TCP ACK packets would have a higher range in ns-2.mme than in ns-2.

Weighing these considerations against the desirability of simulation results comparable

	ns-2.1b9a revised parameters 2Mb	ns-2.1b9a revised parameters 5.5Mb	ns-2.1b9a revised parameters 11Mb
RXT (dBm)	-73	-68.5	-65.5
Transmit Range (@ $P_t = 10$ dBm)	188.5m	150m	126m

Table 4.2: Ns-2 parameter values for higher data rates.

with standard ns-2 results, we have tuned the RXT and CST in standard ns-2 and the noise power in ns-2.mme such that both simulators will give the same range for any transmit power. This is accomplished by defining 4 different RXT thresholds for each data rate in ns-2 such that the SNR values at these RXT thresholds will approximately correspond to $P_C = 1/2$ in ns-2.mme. Although we have observed that an appropriate figure for the receiver noise would be $\eta = -96$ dBm, we will perform our experiments with the inflated figure of -87 dBm. This is simply so that a reasonable transmit power of 10 mW (10dBm) yields a 240 m transmit range, roughly corresponding to the traditional 250 m range of ns-2. The radio parameters used in this thesis are summarized in Table 4.1 and Table 4.2. As most experiments will use the transmit power P_t as a control parameter, the CS range and transmit range are shown as functions of P_t in Figure 4.4.

4.8 A Simple Simulation with Four Nodes

To show the effect of interference on a wireless system, we studied a simple scenario with four nodes shown in Figure 4.5. In this scenario, all nodes have transmit power of 10 dBm and therefore a transmit range of 240 m at 1Mbps; see Figure 4.4. CST is set to -81 dBm. Therefore the carrier sensing range for the nodes is 284 meters; see Figure 4.4. The interference factor is $\theta = 1$, corresponding to a system that does no spreading. As shown in

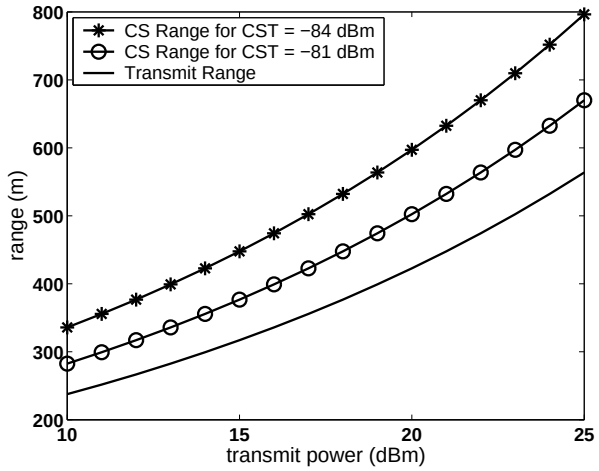
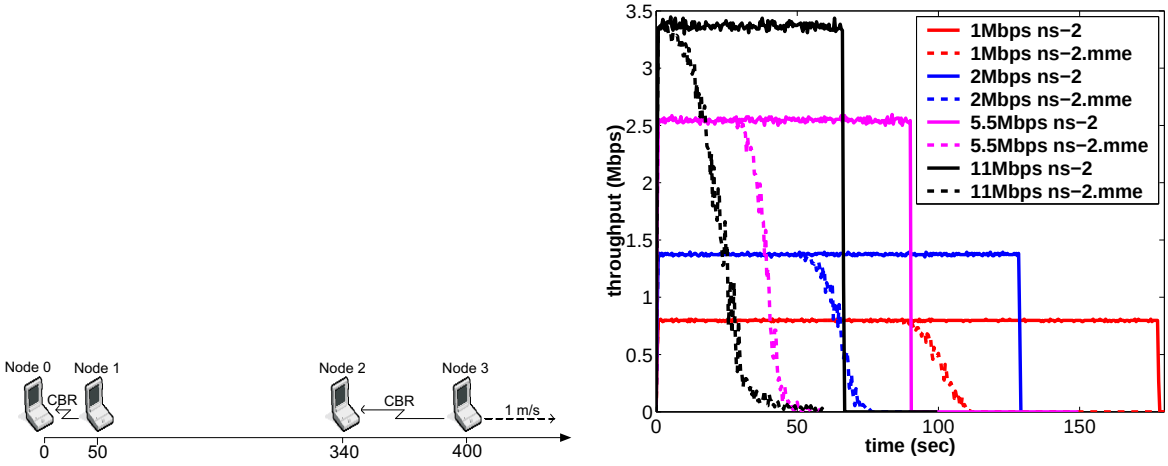


Figure 4.4: CS and transmit ranges of revised ns-2 and ns-2.mme.

Figure 4.5, Node 1 is at position $(x, y) = (50, 0)$, and is transmitting CBR data to Node 0 is at (0,0) with packet size 512 bytes and a rate of 1000 packets/second. Node 3 is at (400,0) and is transmitting CBR data with packet size 512 bytes and a rate of 1000 packets/second to Node 2 at (340,0). The simulation is run for 180 seconds and at time $t = 0$, node 3 starts to move towards position (580,0) with a speed of 1m/s. These settings imply that Node 1 cannot receive Node 2's CTS packets correctly. Neither can Node 2 receive Node 1's RTS packets correctly. In fact, they cannot even hear each other. We first simulated this scenario with ns-2.1b9a at 1Mbps where node 2 is, by design, in the transmit range of node 3 until the end of the simulation at $t = 180$s. Thus the node 3 connection remains good for the entire duration.

When we repeated this simulation with ns-2.mme, we observed in Figure 4.5 that the node 3 connection begins to degrade at $t = 90$s and that its connection is lost around $t = 110$s. That is, one interferer 290 meters away from Node 2 reduced the effective transmit range of Node 3 from 240 meters to 170 meters. This can be explained as follows. The received interference power from node 1 at node 2 is -81.45 dBm. At $t = 110$s the received



(a) Topology of the four node simulation

(b) Node 2 throughput in the four node network

Figure 4.5: Topology and the results of the simulation with four nodes

power from node 3 at node 2 is -72.17 dBm. This implies that the SINR of node 3's signal at Node 2 is around 9.2 dB. At this value, the BER of DBPSK modulation is close to 10^{-4}. This implies that for a 512 byte packet, the packet error rate is $1 - (1 - 10^{-4})^{4096} = 0.34$. Thus the connection is lost after $t = 110$ s. A similar behavior is also observed at the other data rates and the explanation is the same. Although this example was chosen to magnify the differences between ns-2.1b9a and ns-2.mme, the differences do exist. In the next section, we examine when these differences matter in larger scale ad hoc networks.

Chapter 5

Experiments with DSR and AODV

This chapter quantifies the effects of the ns-2.mme model on typical scenarios used in the evaluation of ad hoc routing protocols. In these scenarios, we evaluate the performance of two dynamic routing protocols for ad hoc networks: the Dynamic Source Routing protocol (DSR) [36] and the Ad Hoc On-Demand Distance Vector protocol (AODV) [37]. Although they are both on-demand protocols which initiate routing activities when a node in the ad hoc network wants to send a data packet to a destination, their routing mechanisms are very different. This leads to significant performance differences in various scenarios.

5.1 DSR

DSR is a source routing protocol which adds to each data packet a header that has the complete route which the packet must follow in order to reach the destination. This requires the sender to know the complete route to the destination.

Each mobile node maintains a cache that holds source routes that it has learned. When a node wants to send a packet to another node, it first checks its route cache for a source route to the destination. If a route is found, the sender uses this route to transmit the packets. If no route is found, the sender may attempt to discover one using the *route discovery* procedure.

While a node is using a source route, if the destination, or any of the other nodes on the

route, moves out of wireless transmission range of the next or previous node on the route, the route can no longer be used to reach the destination. In this case the *route maintenance* procedure informs this node. When route maintenance detects a problem with a route in use, route discovery may be used again to discover a new, correct route to the destination in case the sender doesn't already have another route to the destination in its cache.

The protocol has two main parts: the route discovery process and the route maintenance process.

5.1.1 Route Discovery

Route discovery allows any node in the ad hoc network to dynamically discover a route to any other node. A node initiating a route discovery broadcasts a route request (RREQ) packet which may be received by those nodes within wireless transmission range of it. The route request packet identifies the node, referred to as the target of the route discovery, for which the route is requested. If the route discovery is successful, the initiating node receives a route reply packet listing a sequence of network hops through which it may reach the target. In addition to the addresses of the original initiator and target of the request, each route request packet contains a route record, in which is accumulated a record of the sequence of hops taken by the route request packet as it is propagated through the ad hoc network during this route discovery. Each route request packet also contains a unique request id, set by the initiator from a locally-maintained sequence number. In order to detect duplicate route requests received, each node maintains a list of the (initiator address, request id) pairs that it has recently received on any route request.

When any host receives a route request packet, it processes the request according to the following steps:

- If the pair (initiator address, request id) contained in the RREQ packet is found in this node's list of recently seen requests, it discards this packet and does not process it further. This removes later copies of the request that arrive this node by a different route.

- If this node's address is already listed in the route record in the request, it discards the RREQ packet and does not process it further. This guarantees that no single copy of the request can propagate around a loop.

- If the target of the request matches this node's own address, then the route record in the packet contains the route by which the request reached this node from the initiator of the route request. In this case, this node returns a copy of this route in a route reply (RREP) packet to the initiator.

- If this node has a route cache entry for the target of the request, it appends this cached route to the accumulated route record in the packet, and returns this route in a RREP packet to the initiator without re-broadcasting the route request

- If none of these statements is true, this node appends its own address to the route record in the RREQ packet, and re-broadcasts the request.

The route request thus propagates through the ad hoc network until it reaches the target host, which then replies to the initiator.

In order to return the route reply (RREP) packet to the initiator of the route discovery, the target reverses the route in the route record from the RREQ packet, and use this route to send the route reply packet. This, however, requires the wireless network communication between each of these pairs of hosts to work equally well in both directions, which may not be true in some environments or with some MAC-level protocols.

5.1.2 Route Maintenance

In conventional routing protocols, nodes continuously send periodic routing updates. If the status of a link or a node changes, the periodic updates eventually reflects the changes to all other nodes, presumably resulting in the computation of new routes. However DSR does not have periodic messages of any kind from any of the mobile nodes. Instead, while a route is in use, the route maintenance procedure monitors the operation of the route and informs the sender of any routing errors.

Since wireless networks are inherently less reliable than wired networks, many wireless networks utilize a hop-by-hop acknowledgement at the data link level in order to provide early detection and retransmission of lost or corrupted packets. In these networks, route maintenance can be easily provided, since at each hop, the node transmitting the packet for that hop can determine if that hop of the route is still working. If the data link level reports a transmission problem for which it cannot recover (for example, because the maximum number of retransmissions it is willing to attempt has been exceeded), this node sends a route error packet to the original sender of the packet encountering the error. The route error packet contains the addresses of the nodes at both ends of the hop in error: the node that detected the error and the node to which it was attempting to transmit the packet on this hop. When a route error packet is received, the hop in error is removed from this nodes route cache, and all routes which contain this hop must be truncated at that point.

5.1.3 Route Caching

A node can add entries to its route cache any time it learns a new route. In particular, when a node forwards a data packet as an intermediate hop on the route in that packet, the forwarding node is able to observe the entire route in the packet. If a node forwards a

RREP packet, it can also add the route information from the route record being returned in that route reply, to its own route cache. Finally, since all wireless network transmissions are inherently broadcast, a node may be able configure its network interface into promiscuous receive mode, and can then add to its route cache the route information from any data or RREP packet it can overhear.

5.2 AODV

AODV uses a broadcast route discovery mechanism, as is also used by DSR. Instead of source routing however, AODV relies on dynamically establishing route table entries at intermediate nodes with one entry per destination, whereas DSR can maintain multiple route caches per destination. This difference pays off in networks with many nodes, where a larger overhead is incurred by carrying source routes in each data packet. AODV, uses destination sequence numbers like in DSDV to maintain the most recent routing information. Each node maintains a monotonically increasing sequence number which is used to supersede stale cached routes. AODV also features timer-based states in each node. A routing entry is deleted if not used during a specific amount of time.

5.2.1 Route Discovery

The route discovery in AODV is very similar to DSR. The main difference is the use of sequence numbers. In addition to the similar fields in DSR's RREQ, AODV's RREQ contains the pair (source sequence number, last destination sequence number known to the source). The source sequence number is used to maintain freshness information about the reverse route to the source and the destination sequence number species how fresh a route to the destination must be before it can be accepted by the source.

As the RREQ is flooded it automatically sets up the reverse path from all nodes back to the source. To set up a reverse path a node records the address of the neighbor from which it received the first copy of the RREQ. These reverse path route entries are maintained for at least enough time for the RREQ to traverse the network and produce a reply to the sender.

When a RREQ arrives at a node (possibly the destination itself) that possesses a current route to the destination, it checks the freshness of the route by comparing the destination sequence number in its own route entry to the destination sequence number in the RREQ. If the RREQs sequence number for the destination is greater than that recorded by the this node, it cant use its recorded route to respond to the RREQ. Instead, it rebroadcasts the RREQ. This node can reply only when it has a route with a sequence number that is greater than or equal to that contained in the RREQ. In this case if the RREQ has not been processed previously, the node then unicasts a route reply packet RREP back to its neighbor from which it received the RREQ. A RREP contains the following information:

As the RREP travels back to the source each node along the path sets up a forward pointer to the node from which the RREP came, updates its timeout information for route entries to the source and destination, and records the latest destination sequence number for the requested destination.

A node receiving an RREP propagates the first RREP for a given source node towards that source. If it receives further RREPs it updates its routing information and propagates the RREP only if the RREP contains either a greater destination sequence number than the previous RREP or the same destination sequence number with a smaller hop count.

5.2.2 Route Maintenance

When the next-hop link breaks, node upstream of the break sends a RRER packet with a fresh sequence number (i.e., a sequence number that is one greater than the previously known sequence number) and hop count of infinity to all active upstream neighbors. Then these nodes repeat the same process and so on. This process continues until all active source nodes are noticed. Then it terminates because AODV maintains only loop free routes and there are only a finite number of nodes in the network. Notice that in DSR, broken link information is not propagated to all caches that have that link.

5.3 Simulation Setup

In our simulations, we used the standard ns-2 traffic and mobility models. The mobility model is the random waypoint model which was first used by Johnson and Maltz in the evaluation of DSR, and was later refined by the CMU Monarch research group, which introduced the wireless model to ns-2. The refined version of random waypoint model has become the de facto standard in mobile computing research. For example, ten papers in ACM MobiHoc 2002 considered node mobility, with nine of them using the random waypoint model.

In this model, a node starts with a uniformly distributed position in a large rectangular field. After waiting for a predefined pause time, it chooses a randomly positioned destination, and move there at a random speed uniformly chosen from $(0,V_{max})$, where V_{max} is the maximum speed of a simulated mobile. Sometimes the model is described as having an average speed of $V_{max}/2$. After reaching the destination nodes stop for the same pause time, then this procedure repeats until the end of the simulation.

It is known that the spatial distribution of network nodes moving according to this

model is, in general, nonuniform [38]. This fact is claimed to impair the accuracy of simulation methodology and make it impossible to relate simulation-based performance results to corresponding analytical results. In spite of this, we employ this model due to its highly common usage.

We set the topology as 1500×300 m^2 field with 50 nodes and used a pause time higher than the simulation time (no mobility) or a zero pause time (continuous mobility) to make the simulations challenging for the routing protocols. The traffic sources are continuous bit rate (CBR) with UDP protocol or FTP with TCP protocol. In the simulations, either 512-byte data packets or 1500-byte data packets are tested.

Three important performance metrics are evaluated:

- *Packet Delivery Ratio*: This is the ratio of the data packets delivered to the destinations to those generated by the CBR sources.
- *Goodput*: This is the amount of data that has been acknowledged to the sender divided by the time when the highest TCP ACK was received. If all the packets are not acknowledged in the specified simulation time, the amount of data is divided by the whole simulation time.
- *Normalized Routing Load*: This is the number of routing packets transmitted per data packet sent to the destination. Also each forwarded packet is counted as one transmission. This metric is also highly correlated with the number of route changes that occurr in the simulation.

5.4 Simulations with UDP at 1Mbps Data Rate

In the first set of experiments, we wanted to elaborate on the effects of the interference factor θ, introduced in section 4.1, on wireless network simulations. To observe the effects of the interference factor θ more clearly, we turned off the RTS/CTS option, so that we can observe more collisions. As we explained in chapter 2, the DSSS spreading gain is highest at the 1Mbps data rate. Because of this, in these simulations we set all the 802.11b wireless interfaces to have a bit rate of 1 Mbps. Here, we also used a zero pause time (continuous mobility) to make the simulations challenging for the routing protocols. The number of source destination pairs is 15 and the sources are CBR sources with 512-byte packets. Simulations are run for 500 seconds. Each data point is an average of at least 3 runs with identical traffic scenarios but randomly generated mobility scenarios. Each of the 15 source destination pairs sends at a rate of 3 packets/s, a relatively low packet rate in order to avoid network congestion. (Higher rates will be considered in the next section.)

5.4.1 Varying Transmit Power

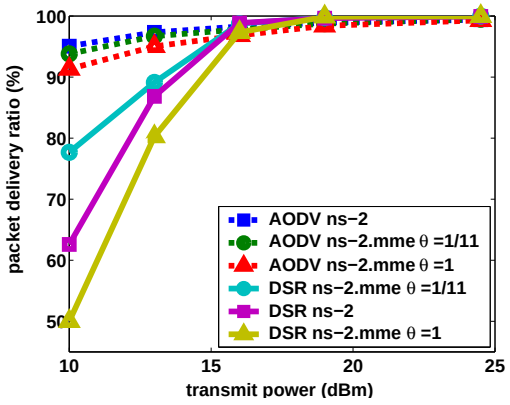
All nodes have the same transmit power, but for each experiment the transmit power is varied from 10 dBm to 24.5 dBm. Note that 24.5 dBm is the default transmit power in ns-2. As noted earlier, a transmit power of 10 mW (10dBm) roughly corresponds to the traditional 250 m transmit range of ns-2.

Our experiments focus on two extreme cases for 802.11b systems. First is the scenario in which $\theta = 1/11$. This is a first-order model for 802.11b systems operating at 1 Mb/s and employing the 11 chip Barker spreading sequence. In the second scenario, we assume $\theta = 1$, which would be a suitable approximation for higher transmission rates which do not have the interference suppression benefits afforded by spreading. As noted in section 4.1,

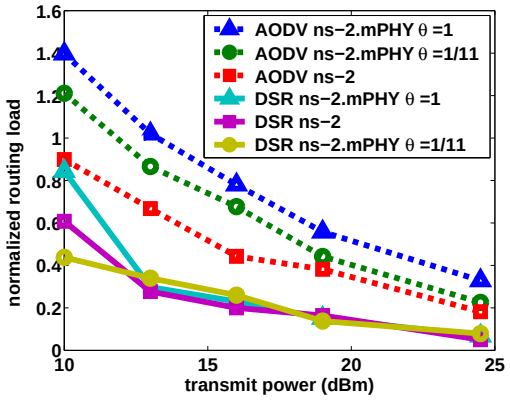
it appears that precise values of θ for 802.11b systems has not been examined. However, one can reasonably argue that $\theta = 1/11$ and $\theta = 1$ represent the extreme points of practical values.

In Figure 5.1, we see that for CST= -84 dBm, the packet delivery ratios for DSR and AODV are very similar with transmit power P_t higher than 16 dBm. When the transmit power is lower than 16 dBm, AODV outperforms DSR in terms of packet delivery ratio. However at any transmit power, DSR demonstrates significantly lower routing load than AODV. This is due to DSR's aggressive use of route caching. DSR is likely to find a route in the cache and avoid using route discovery every time a link is broken. In a previous DSR study [6], it was found that 55 percent of the route replies were from the route caches and even though 41 percent of the route replies were based on cached data contained broken routes, the DSR route maintenance was able to deliver good performance. However we see that for $P_t < 16$ dBm, caching degrades the performance of DSR. In this case, low transmit powers yield frequent link failures. As a result, DSR caches will not be up to date, and stale routes often will be chosen from the cache. However, we don't see such a big degradation in AODV, largely because of the use of sequence numbers maintained at each node to determine the freshness of the routing information. AODV also features timer-based states in each node. A routing entry is deleted if it not used during a specified amount of time. On the other hand, DSR keeps the routing entries in the cache until a link on the route is found to be broken.

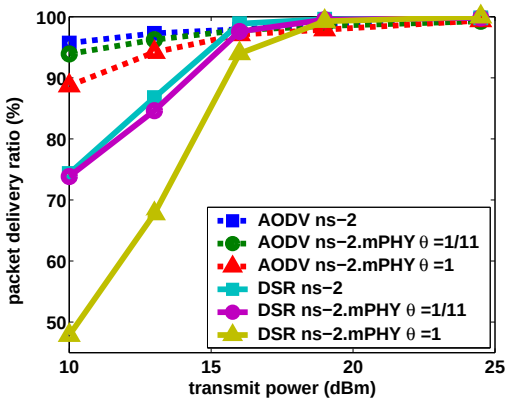
From Figure 5.1, we see that the performance of AODV is relatively insensitive to the choice of PHY layer model. On the other hand, DSR exhibits significant performance variation when we compare the ns-2, ns-2.mme with $\theta = 1$, and ns-2.mme with $\theta = 1$. We see that DSR gives the highest packet delivery ratio with ns-2.mme and $\theta = 1/11$. In this



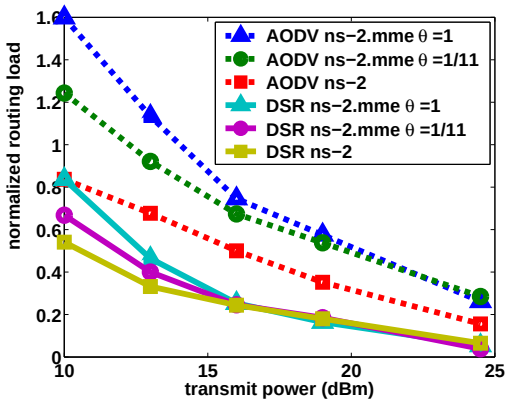
(a) Packet delivery ratio, CST=-84dBm



(b) Normalized routing load, CST=-84dBm



(c) Packet delivery ratio, CST=-81dBm



(d) Normalized routing load, CST=-81dBm

Figure 5.1: Performance at different CST levels

case, with CST = -84 dBm, there are few interfering transmissions and any interference from those transmissions is divided by 11. DSR's aggressive route caching exploits these favorable conditions. By comparison, the 10 dB threshold interference model of ns-2 is more conservative and the performance of DSR is worse under this model. Finally, under ns-2.mme with $\theta = 1$, the effect of interfering transmissions is amplified. In this case, DSR is quite sensitive to the additional interference.

As shown in [5], DSR is better than AODV in less "stressful" situations and AODV

outperforms DSR in more stressful situations. But this time, the less "stressful" situations are those scenarios with high transmit power which results in an environment with low interference. This occurs because there are fewer nodes that don't hear each other and carrier sensing results in very few interfering transmissions. This same effect also reduces the impact of the cumulative interference calculations of ns-2.mme. For the topology used in our simulations, this occurs in the region where the transmit power is higher than 20 dBm, which is high for modern wireless LANs and very high for emerging applications such as sensor networks. Here, we would like to note that even with high transmit powers, there can be situations where most nodes still fail to hear each other and cause interference.

In summary, for both AODV and DSR, the gap between ns-2 and ns-2.mme results shrinks as the transmit power increases. However, this gap is much wider for DSR at low transmit power. Thus, our most important observation is that the performance of DSR can depend strongly on the physical layer model. For example, we see in Figure 5.1 for $P_t = 10$ dBm that the choice of PHY layer model affects DSR packet delivery ratio by 50 percent. On the other hand, the performance of AODV appears to be relatively insensitive to the choice of PHY layer model.

5.4.2 The Effect of Changing CST

Prior research has noted the impact of carrier sense on the aggregate throughput. That is, the smaller the carrier sense range, the better the spatial reuse; but the interference at a receiver can also increase. Implicitly assuming a perfect MAC protocol without any overhead, Zhu et al. [39] has attempted to identify the optimal carrier sense threshold that maximizes the spatial reuse given a minimum required SINR for a regular topology. In [40] they showed that the throughput of basic CSMA scheme decreases with the increase of the

carrier sensing range because when carrier sensing range is smaller, more spatial reuse is possible. In this section, our aim is not to find the optimal carrier sense threshold. However, we would like to show when doing ns-2 simulations with various CST levels, how one can get results that contradict the results of ns-2.mme.

The next set of experiments (Figure 5.1) demonstrate the effect of increasing the CST so that the number of instantaneous transmissions will increase. In ns-2, increasing the CST value increases the packet delivery ratio because the number of instantaneous transmissions increase, but there is no SINR tracking to record the cumulative effect of interference which would actually degrade the performance. However, this effect is clearly seen in ns-2.mme. In Figure 5.1 with CST=-81dBm, we see that the gap between ns-2 and ns-2.mme with $\theta = 1$ widens. We also observe that even DSR's performance with $\theta = 1/11$ is now lower than DSR with standard ns-2. This shows that when the interference increases, ns-2's 10 dB threshold interference model becomes even more optimistic than the SINR tracking model for $\theta = 1/11$ because it underestimates the cumulative effects of interference.

5.5 Simulations with UDP at Higher Data Rates

5.5.1 PHY Layer Rates Used in the Simulations

In this section, we experiment with the PHY layer data rates. However, when using a specific rate for the unicast data packets, the choice of rate used for the broadcast data packets and the MAC control packets can dramatically affect the results. Before going into simulations, this issue needs to be discussed.

As explained in section 5.1.1 and 5.2.1 the route discovery in AODV and DSR is made by the use of RREQ packets. These RREQ packets are sent as broadcast data packets. Therefore the number of nodes participating in the route discovery and the distance between

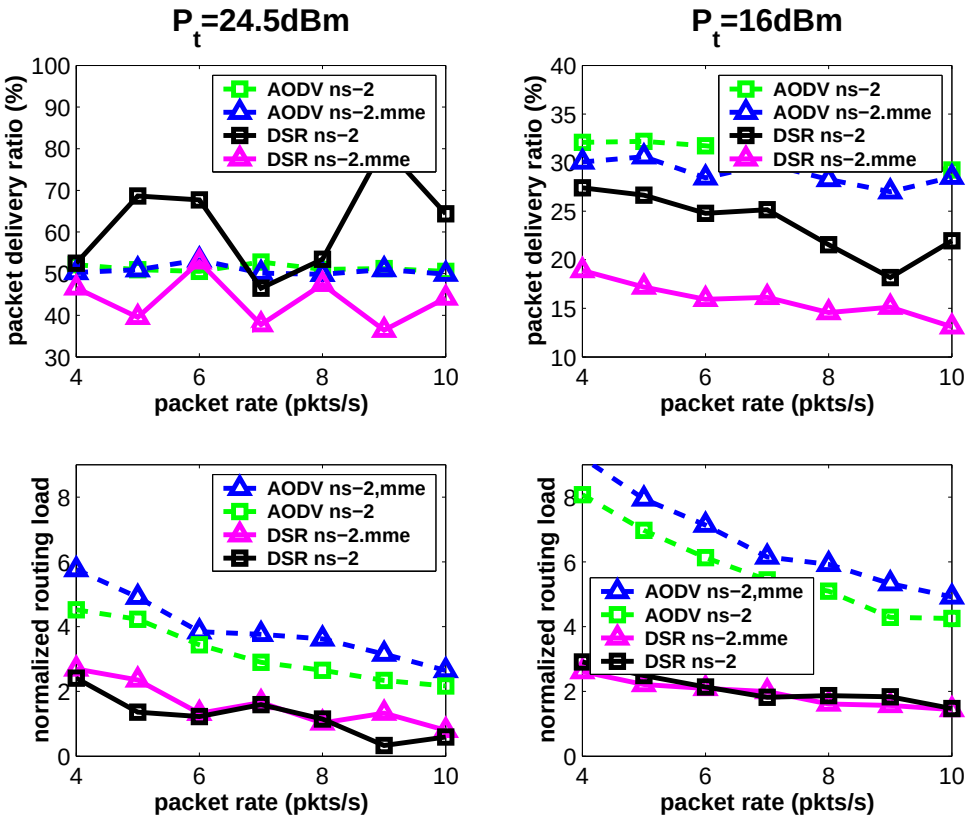


Figure 5.2: 11Mbps with broadcast data and control packets at 1Mbps.

the nodes in the chosen path is affected by the PHY rate chosen for transmitting broadcast data packets. A lower rate results in a higher range and therefore longer hops. However, in a single rate environment, the routing should be done with the rate that's used for transmitting the actual unicast data packets. For example, if the rate chosen for data packets is 11Mbps but the route is constructed with 1Mbps packets, the routing protocol may find a route that has hops that can't support an 11Mbps data rate. This may result in frequent link breakages and an unstable network behavior.

In most studies using ns-2, the ACK packets are sent at 1Mbps. However, the ACK packets can be sent at the same rate as the data packets. Because the SIFS time between sending the ACK packet after receiving the data packet is very short, it's unlikely that the channel conditions will degrade and cause the ACK packet to be lost. Moreover, the

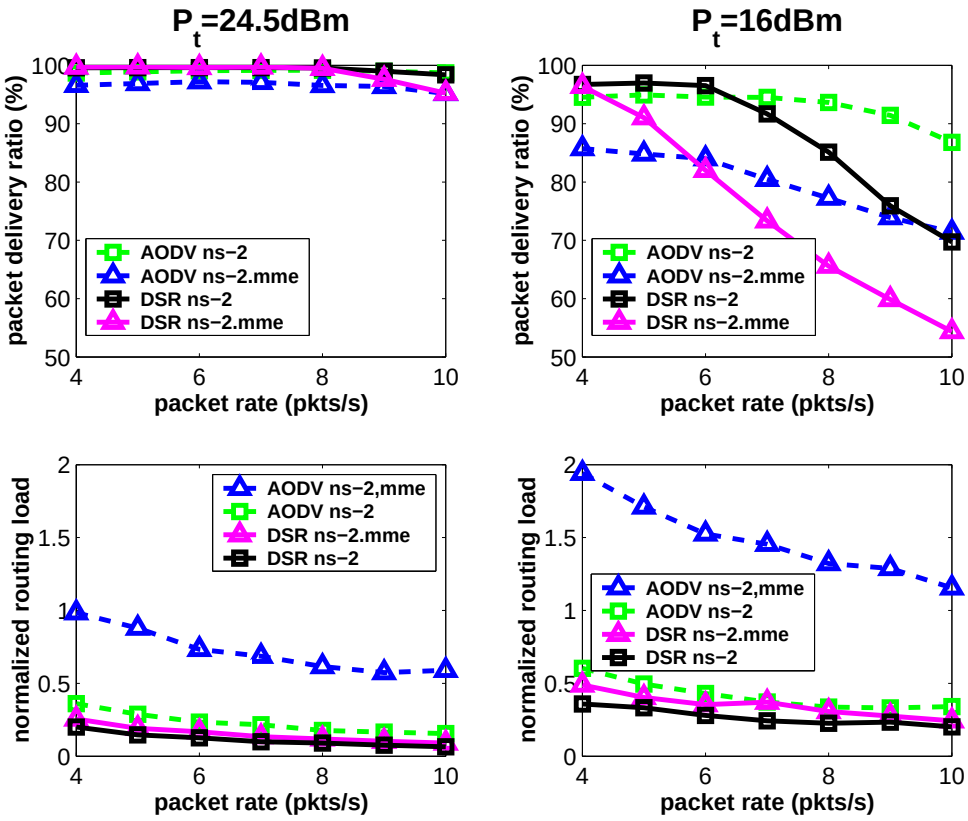


Figure 5.3: 11Mbps with 512-byte packets RTS/CTS ON

ACK packets are much shorter than the data packets, so they are more immune to interference, and therefore their transmission range is higher than data packets, as explained in section 4.7. Therefore, in all our simulations, we used the same PHY layer rate for both ACK and data packets.

Now, we examine how the choice of PHY layer rates affects the network. To illustrate this behavior, we made simulations using 11Mbps to transmit the data packets. The simulation scenario is similar to that in section 5.4. However, now the CBR data rate is changing and two different P_t levels are used. In the first set (Figure 5.2), the broadcast data packets, RTS, CTS and ACK packets are sent at 1Mbps and routing problems occur. For example, at P_t =16 dBm the communication range is 150 meters for 11 Mbps. However, the routing is done with RREQ packets sent at 1 Mbps, so the routing protocol finds routes assuming

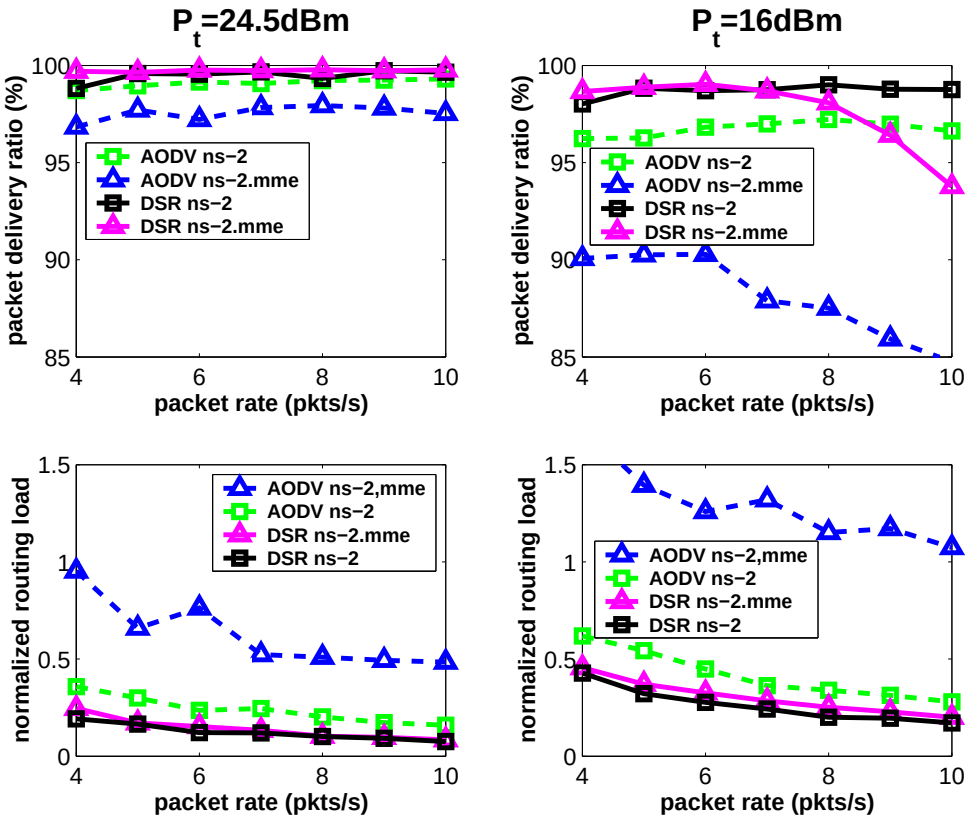


Figure 5.4: 11Mbps with 512-byte packets RTS/CTS OFF

that the communication range is 340 meters (range of 1 Mbps at P_t =16 dBm). We see that there's no correlation between the increasing packet rate and the packet delivery ratio. For both P_t levels, the packet delivery ratio is essentially a random variable. In case of DSR, we also see a great difference between the results of ns-2 and ns-2.mme. On the other hand, the performance of AODV is similar in both ns-2 and ns-2.mme. This is due to the factors that we've explained in section 5.4. Now, let's compare these results with the results of the simulations in which we used 11 Mbps for all the packets (Figure 5.3). At P_t =24.5 dBm the range of 11 Mbps is 280 meters and we see that even with 10 pkts/s, the network is not overloaded. Also, the results of ns-2 and ns-2.mme are almost the same because with this power level, there can be few multiple simultaneous transmissions. However with P_t =16 dBm, there's a big difference between the results of ns-2 and ns-2.mme. This is due

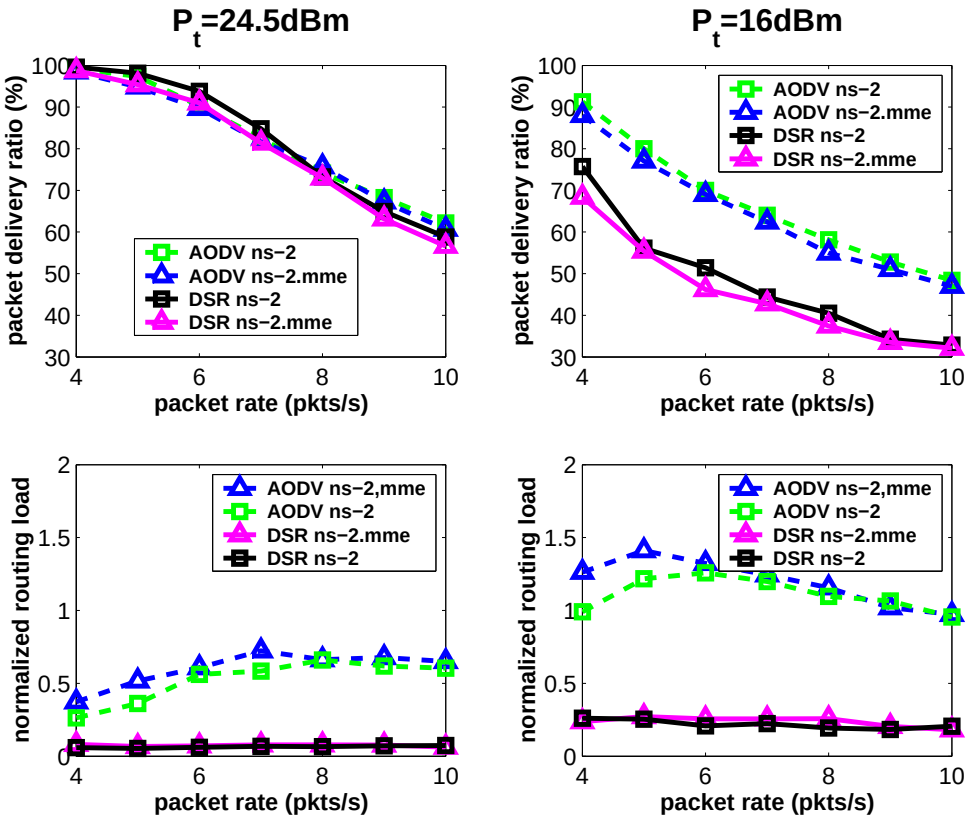


Figure 5.5: 1Mbps with 512-byte packets RTS/CTS ON

to the fact that the RTS/CTS option is turned on. We'll elaborate more on this in the next section.

5.5.2 The Effect of RTS/CTS

As we mentioned in section 3.3, 802.11b supports two types of access modes: the physical carrier sensing mechanism and the RTS/CTS based mechanism. Since transmitting RTS and CTS frames increases the overhead, there is a trade-off between such overhead and the overhead from collisions between packets in the physical carrier sensing mode. Especially in high-rate situations this can degrade the network performance because the physical headers including the PLCP preambles of data, ACK, RTS and CTS frames are transmitted at 1Mbps regardless of the data rate.

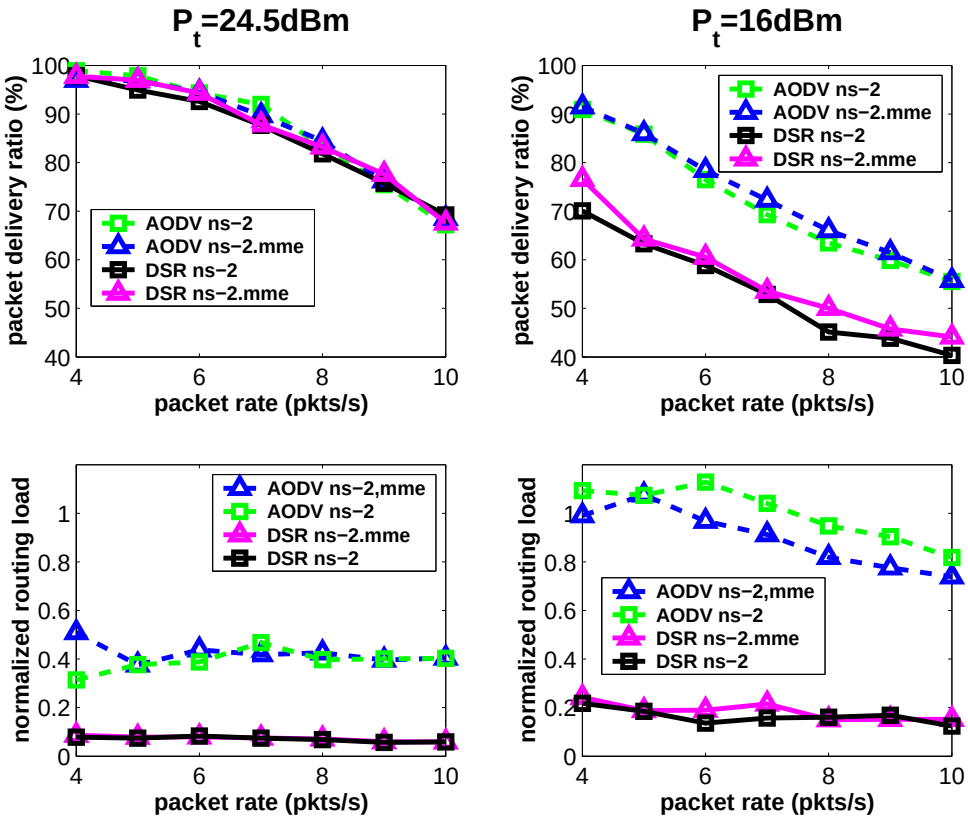
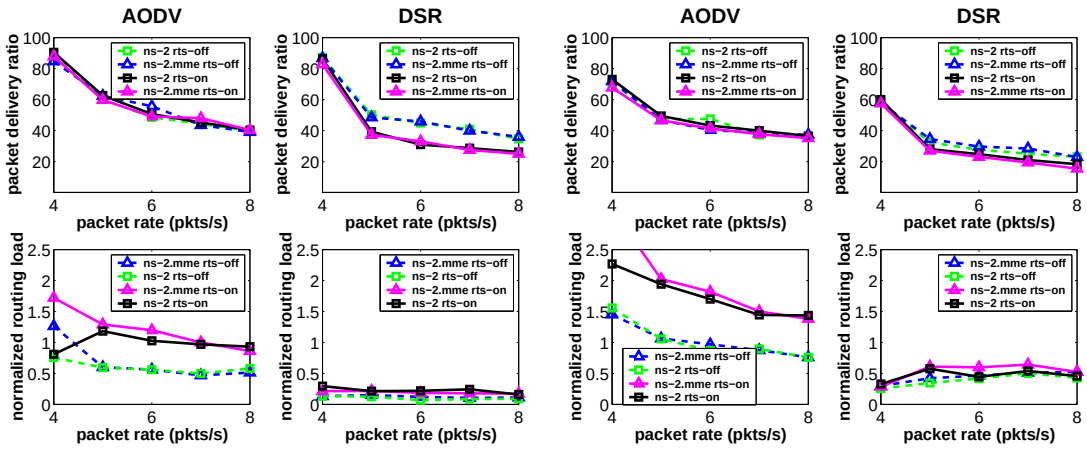


Figure 5.6: 1Mbps with 512-byte packets RTS/CTS OFF

Some researchers have studied the performance of these mechanisms. In [41] it was shown that the RTS/CTS mechanism could achieve a better performance when the hidden terminal problem is encountered. In [42] Bianchi developed an analytic model and proved that RTS/CTS mechanism performed better than the basic mechanism in most cases. However only 1Mbps data rate was used in these studies. In [43], Bianchi's model was extended and it was shown that RTS/CTS performed better than the basic mechanism with 1, 2, 5.5, 11Mbps data rates. However, the fact that the physical header is always transmitted at 1Mbps wasn't taken into account. With this fact taken into account, some simulation and analysis were provided in [44] and [45]. In [44] only 2Mbps data rate was used and it was again shown that RTS/CTS was superior to the basic mechanism. However, 11Mbps data rate was used in [45] and the results were the opposite.



(a) P_t=24.5 dBm (b) P_t=16 dBm

Figure 5.7: 1Mbps with 1500-byte packets

In the most recent study on the performance of RTS/CTS mechanism with ns-2, only TCP traffic was studied and it was shown that the RTS/CTS mechanism achieve a better performance than the basic mechanism in most cases with a data rate of 2Mbps and with 1500-byte packets [46]. However for an 11Mbps data rate, this was true only when the number of TCP connections were higher than 25.

In our simulations, we also examine whether RTS/CTS should be employed. As a result, most of our scenarios are repeated both with and without RTS/CTS.

5.5.3 Varying Offered Load

The simulations in this section demonstrate the effect of loading the network. We experiment with two different power levels, P_t=16 dBm and P_t=24.5 dBm. We again use the 50-node model and 15 sources. The packet rate of each source is slowly increased from 1 to 10, changing the total offered load to the network from 240 kb/s to 600kb/s in case of 512-byte packets and from 700 kb/s to 1760 kb/s for 1500-byte packets.

Figure 5.5 shows the packet delivery ratio and the normalized routing load for both

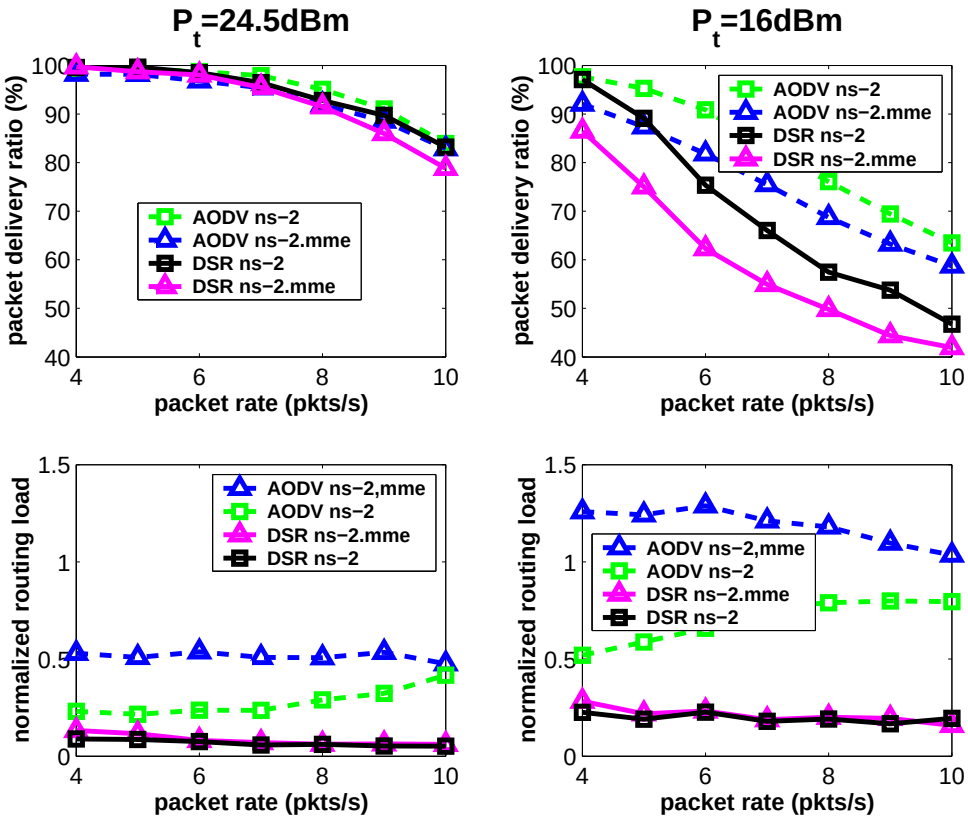


Figure 5.8: 2Mbps with 512-byte packets RTS/CTS ON

AODV and DSR at 1Mbps with 512-byte packets and RTS/CTS option turned on. The relative performance of AODV and DSR in figure 5.5 is consistent with results reported by Das, Perkins and Royer [5]. We see that DSR's packet delivery ratio is almost the same as AODV at P_t=24.5 dBm, whereas AODV is 20% better than DSR at P_t=16 dBm at every packet rate. This is due to the fact that in a "stressful" situation induced by lower transmit power, DSR fails to find reliable routes during route maintenance because of its aggressive caching strategy. As expected, AODV generates higher routing load than DSR. Also in Figure 5.5 we see that the normalized routing load has essentially no correlation with the increasing packet rate.

When we focus on the impact of the physical layer model on the packet delivery ratio, we observe that both AODV and DSR are relatively insensitive to the choice of physical

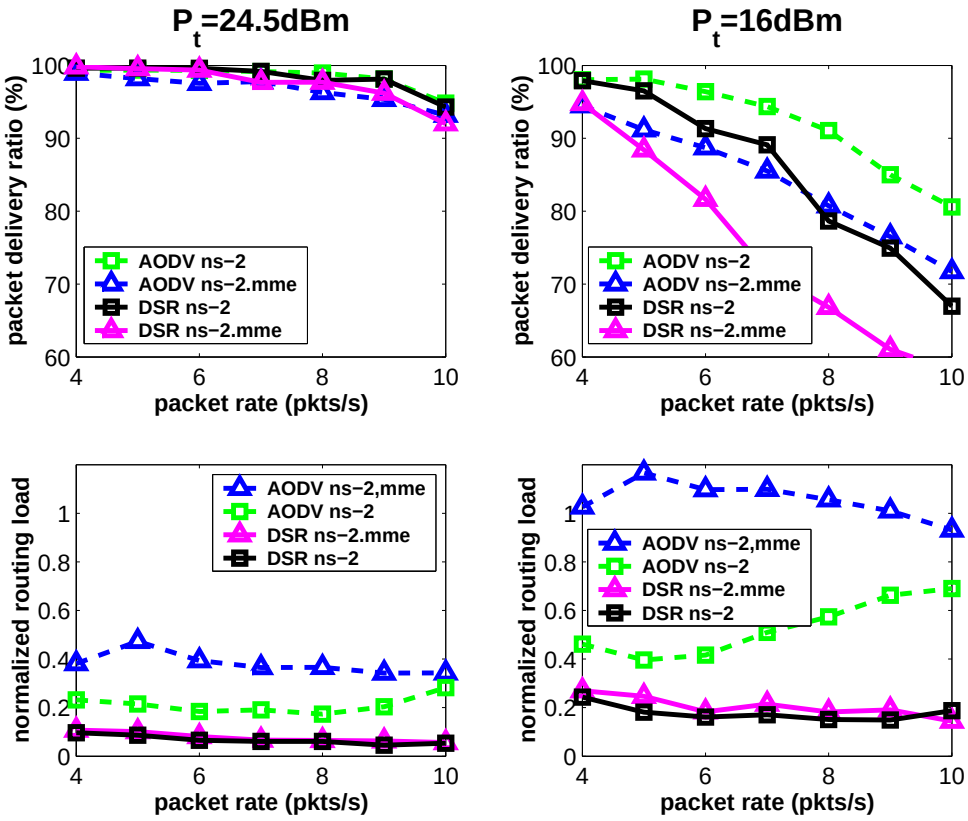


Figure 5.9: 2Mbps with 512-byte packets RTS/CTS OFF

layer model. This is because the offered load is high for 1Mbps and packet losses are high due to congestion, regardless of the physical layer model.

In Figure 5.6 we see the results of the same simulations with RTS/CTS option turned off. Here the qualitative behavior is exactly the same. However, the packet delivery ratios are 10% better than the previous simulations.

To see if RTS/CTS improves performance with longer packets, we repeated the same experiments with 1500-byte packets. In Figure 5.7 we see that when we increase the packet length, the relative overhead of RTS/CTS handshake decreases therefore the difference in performance between RTS/CTS and basic scheme is much less than the 512-byte case. However, basic CSMA scheme still performs at least as good as the RTS/CTS scheme. In fact, at 1Mbps, the relative overhead of RTS/CTS is the least. Because of this we

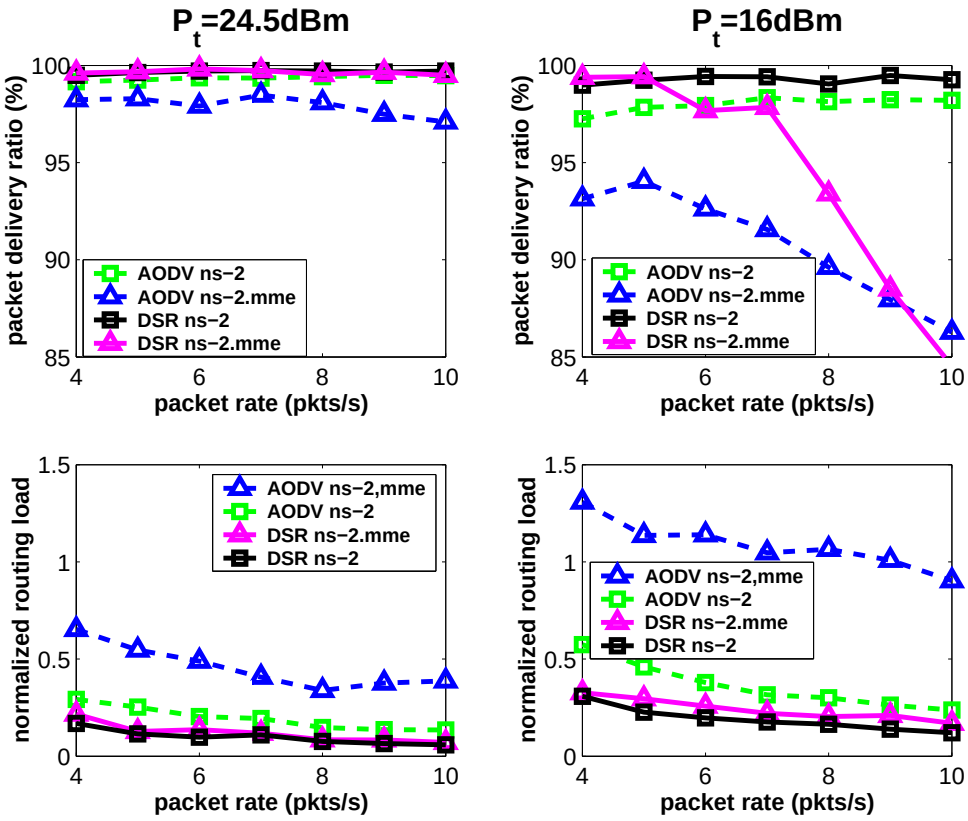


Figure 5.10: 5.5Mbps with 512-byte packets RTS/CTS OFF

would expect RTS/CTS to perform best in 1Mbps data rate. However this is not the case due to UDP's unidirectional traffic nature. Unlike TCP, there are no ACK packets being transmitted in the reverse direction. Therefore the possibility of hidden nodes is lower. And as we see in the graphs, this possibility is so low that RTS/CTS's overhead does not pay off.

5.6 Experiments with TCP

As we explained before due to its bidirectional traffic, TCP is more prone to hidden terminals. For this reason we also experimented with TCP to see how RTS/CTS performs. Ns-2 supports several versions of an abstracted TCP sender. These objects attempt to capture the essence of the TCP congestion and error control behaviors, but are not intended to

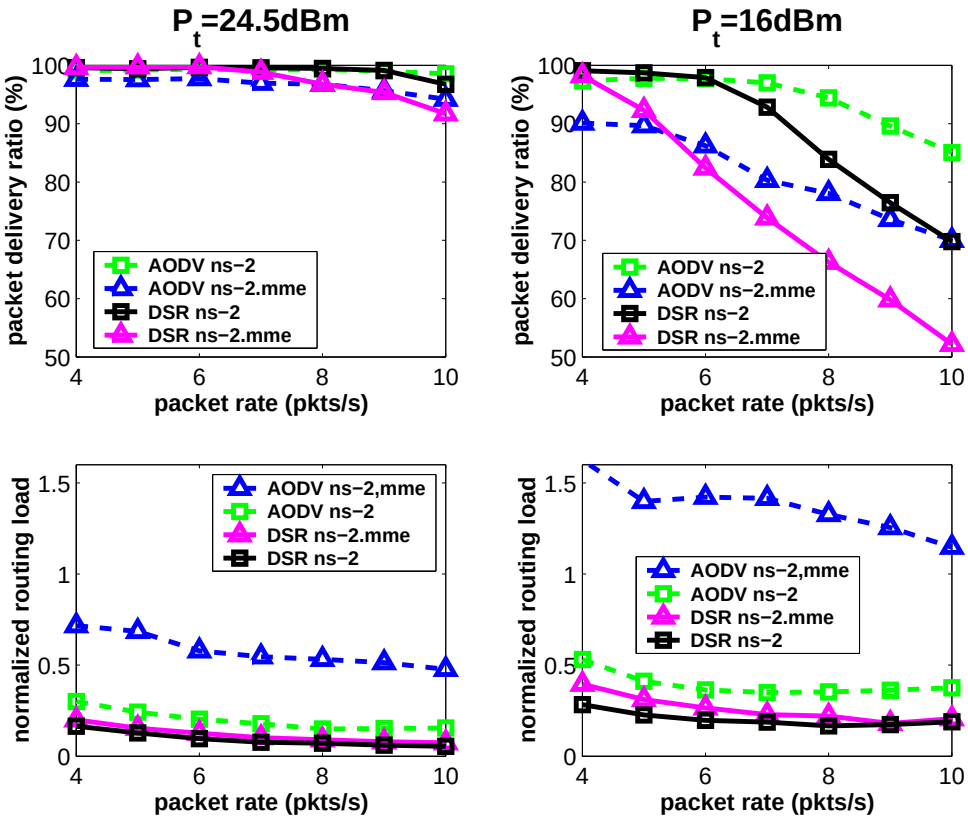
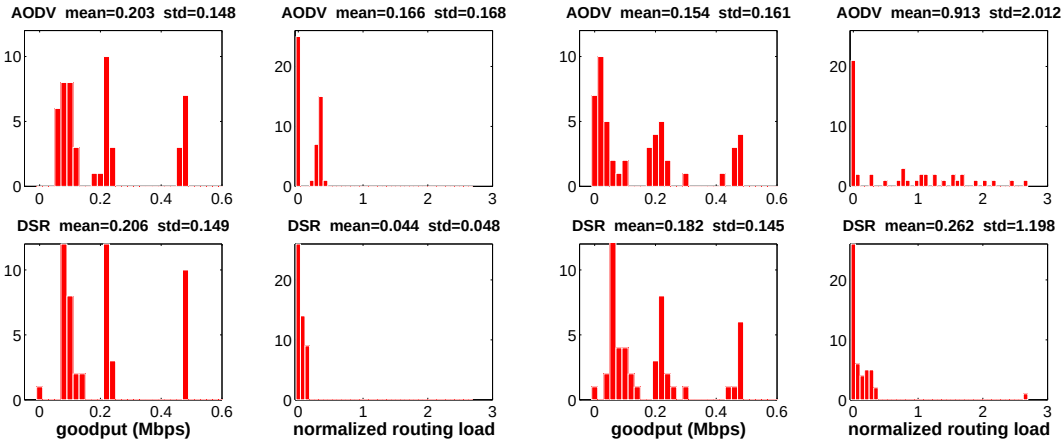


Figure 5.11: 5.5Mbps with 512-byte packets RTS/CTS ON

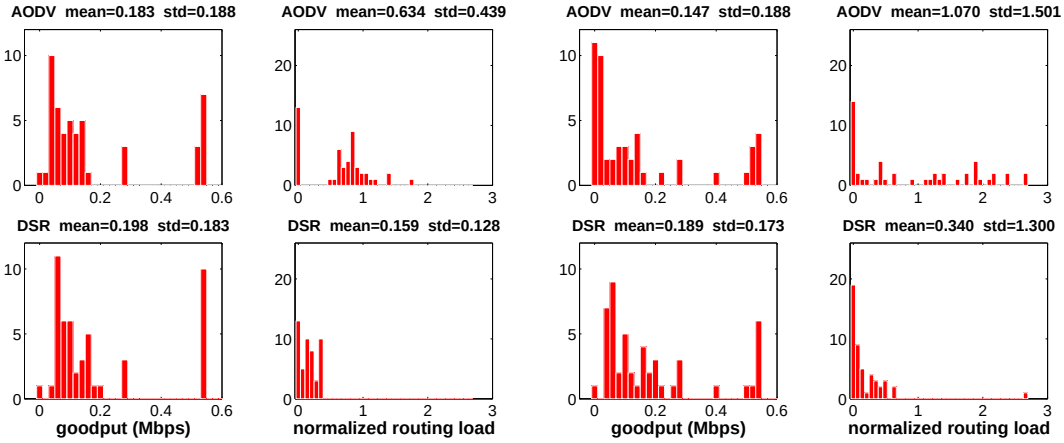
be faithful replicas of real-world TCP implementations. They do not contain a dynamic window advertisement, they do segment number and ACK number computations entirely in packet units and there is no SYN/FIN connection establishment/teardown.

In the next set of experiments, we used the base TCP sender in ns-2, which performs congestion control and round-trip-time estimation in a way similar to the version of TCP released with the 4.3BSD "Tahoe" UNIX system release from UC Berkeley. The congestion window is increased by one packet per new ACK received during slow-start and is increased during congestion avoidance. Tahoe TCP assumes a packet has been lost (due to congestion) when it observes 3 duplicate ACKs, or when a retransmission timer expires. In either case, Tahoe TCP reacts by setting *slow start threshold* to half of the current window size or 2, whichever is larger. It then initializes *congestion window* back to its initial value. This



(a) ns-2 RTS/CTS ON

(b) ns-2.mme RTS/CTS ON



(c) ns-2 RTS/CTS OFF

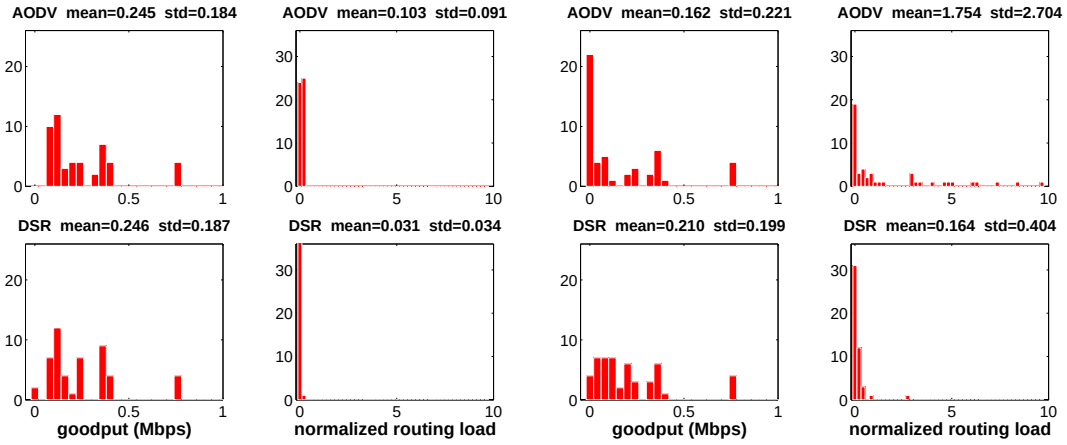
(d) ns-2.mme RTS/CTS OFF

Figure 5.12: Goodput and normalized routing load with 512 bytes packets at 1Mbps

typically causes the TCP to enter slow-start.

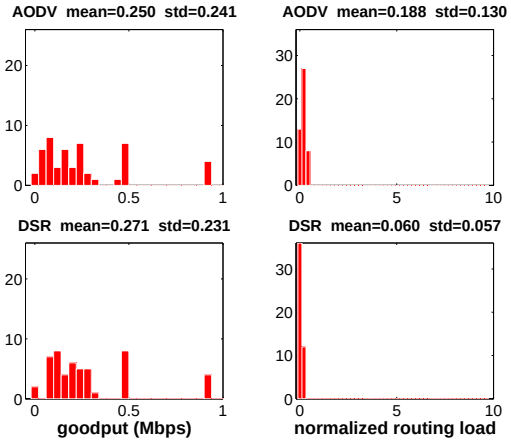
In these experiments, the topology is 1500×300 m^2 field with 50 nodes and there's only one FTP connection between two randomly chosen non-mobile nodes. To make the simulation results more understandable, we only experimented with stationery networks. In the simulations, either 512-byte data packets or 1500-byte data packets are used and the simulations are repeated 50 times.

In the first set of experiments, we used 512-byte data packets at 1 Mbps data rate

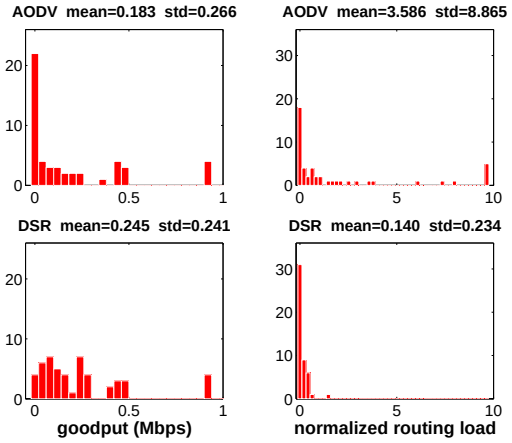


(a) ns-2 RTS/CTS ON

(b) ns-2.mme RTS/CTS ON



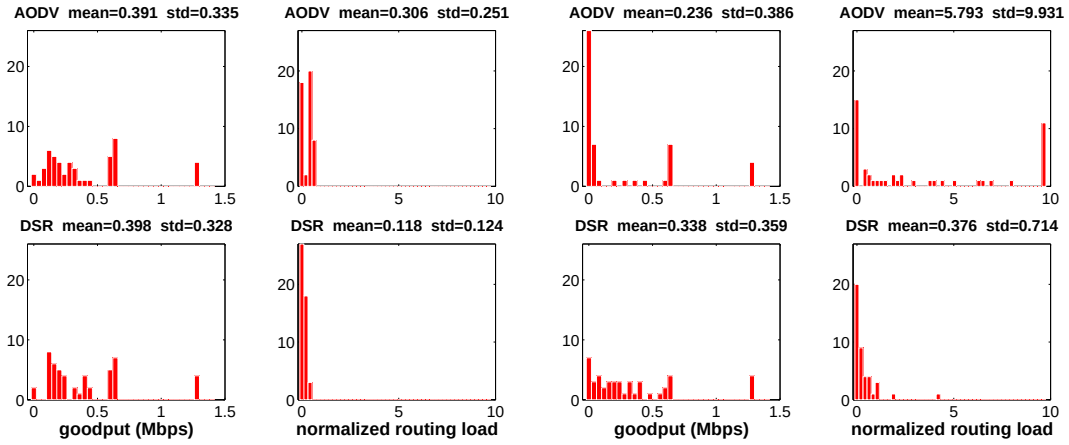
(c) ns-2 RTS/CTS OFF



(d) ns-2.mme RTS/CTS OFF

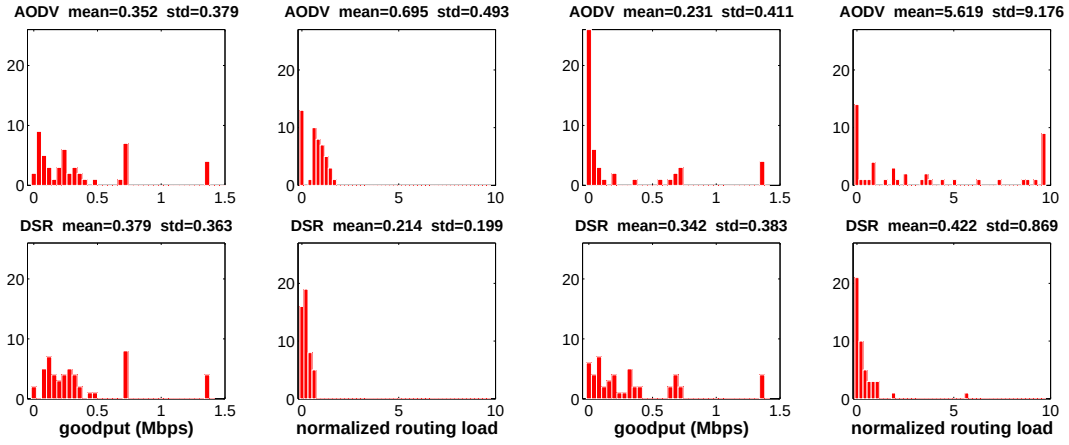
Figure 5.13: Goodput and normalized routing load with 512 bytes packets at 2Mbps

and the transmit power was 10 dBm. At 1 Mbps the range is 240 meters and given the 1500×300 m^2 field, in every simulation we expected a route to be found between the source and the destination. In 50 simulations there was only one case with zero goodput. In Figure 5.12, we see three peaks in the goodput histograms of ns-2. The first peak around 0.5Mbps belongs to the connections with one hop. With one hop communications there can't be any collisions once the route is found since there's just one FTP pair in the network. Therefore one would expect to see no difference between the goodput of ns-2



(a) ns-2 RTS/CTS ON

(b) ns-2.mme RTS/CTS ON



(c) ns-2 RTS/CTS OFF

(d) ns-2.mme RTS/CTS OFF

Figure 5.14: Goodput and normalized routing load with 1500 bytes packets at 2Mbps

and ns-2.mme at one hop. However there is a difference because the route discovery in ns-2.mme simulations takes longer than route discovery in ns-2 simulations. Therefore the mean goodput of one hop TCP connections in ns-2.mme is less than that of ns-2. Also, if you compare the 1 hop goodput distribution of the results of the experiments with RTS/CTS ON with its RTS/CTS OFF complement, you see that they are almost the same. The only difference is that throughput with RTS/CTS OFF is a little higher than with RTS/CTS ON because of the overhead of the RTS/CTS. The reason of this similarity is the fact that

during route discovery, the RREQ packets are broadcasted and therefore not preceded with an RTS/CTS exchange. Because of this, the route discovery in two different scenarios are identical. This can also be seen in Figures 5.13, 5.14, 5.18 and 5.19.

In Figure 5.12, the second peak around 0.2Mbps belongs to the connections with two hops. In this case, it's possible that the TCP source and destination pairs may not hear each others transmissions. So, collisions of TCP data packets and TCP ACK packets can occur at the intermediate nodes. Therefore when we compare the two-hop goodput in the histograms we see that when RTS/CTS is turned on, the goodput is much higher. This is true for both ns-2 and ns-2.mme. However, we can't observe the same behavior in Figure 5.13 with 2Mbps i on the second peak around 0.4Mbps due to the overhead of RTS/CTS.

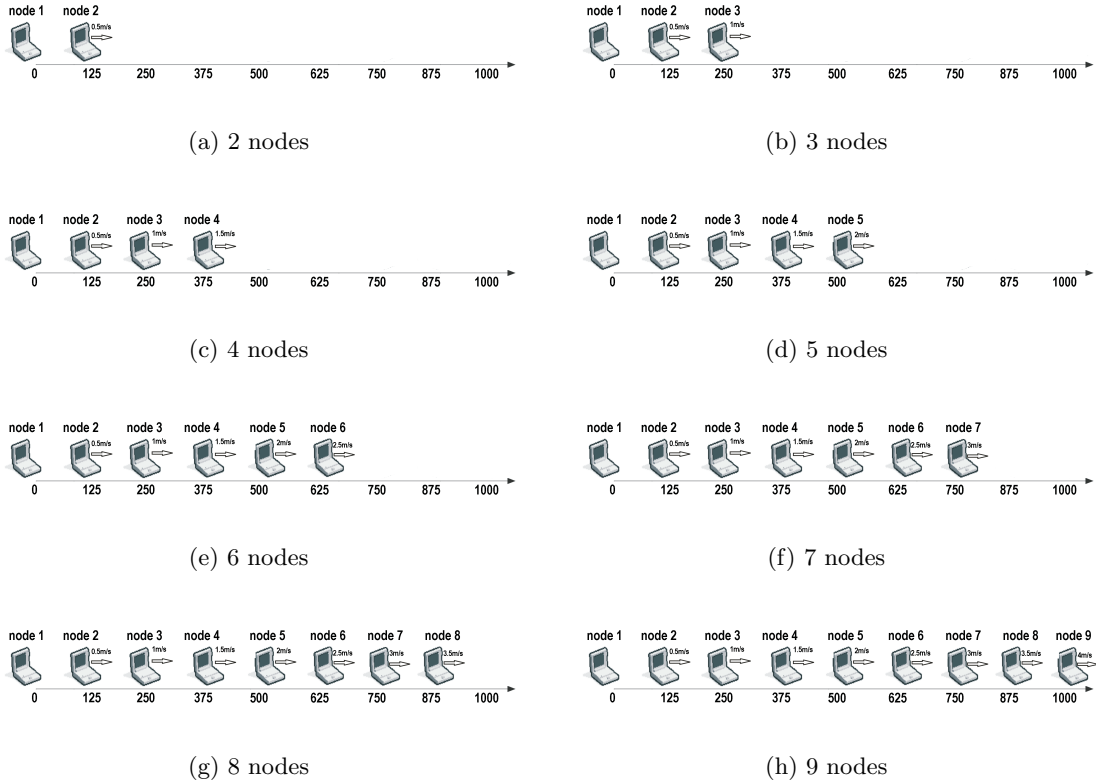
When we increase the packet size to 1500 bytes and repeat the same simulations at 2Mbps, we can again see that the two-hop goodput is higher when RTS/CTS is turned on,(Figure 5.14). This occurs because when we increase the packet length, we are also decreasing the RTS/CTS overhead. However, this is not clearly observed in ns-2.mme histograms. Actually, when we evaluate the performance of RTS/CTS at 2Mbps with 1500-byte packets we see that RTS/CTS scheme doesn't perform better in ns-2.mme simulations. On the other hand, with ns-2, we see 11% improvement with RTS/CTS scheme in AODV and 5% improvement in DSR. The improvement in DSR is less because DSR is already doing well in terms of packet delivery ratio. This is because DSR performs better than AODV in less stressful situations, as we explained in section 5.4.

We observe that, despite the fact that these are the results of stationery scenarios with just one FTP connection, there is still a big difference between the ns-2 and ns-2.mme results (Table 5.1). This is due to the TCP sender's inability to accurately determine the cause

of a packet loss. The TCP sender assumes that all packet losses are caused by congestion. Thus, when a link on a TCP route breaks, the TCP sender reacts as if congestion was the cause, reducing its congestion window and, in the instance of a timeout, backing-off its retransmission timeout (RTO). Therefore, link breakages degrade the TCP performance, and these breakages occur more often in ns-2.mme. To understand how link breakages occur in a stationery network, consider a six-hop route with the TCP sender as node 1, the TCP receiver as node 7, and intermediate nodes 2 through 6. At 2Mbps, the transmission range is 188.5 meters, so the nodes can be placed in a 1500×300 m^2 field such that there are six hops between the sender and the receiver. Even if RTS/CTS is turned on, some nodes may be transmitting simultaneously. For example, nodes 1, 4 and 7 can transmit at the same time. In this case, transmissions by 1 and 7 cause transmissions by 4 to be received in error. After a number of retries, node 4 declares the link as being broken. However, this behavior cannot be observed with ns-2.

To illustrate this difference between ns-2 and ns-2.mme, we designed eight similar experiments. In the first, there are only two nodes, node 1 at (0,0) and node 2 at (0,125). At $t = 0$ node 1 starts an ftp transmission to node 2 and node 2 starts to move towards position (1400,0) with a speed of 0.5m/s.

In every experiment, another node is added 125 meters away from the last added node as shown in Figure 5.15. Except for node 1, all nodes move towards position (1400,0) at a speed which will set the distances between the adjacent nodes always equal to each other and increase at 0.5m/s. At $t = 0$, this distance is 125m and at $t = 100$ it is 175m. In Figure 5.16 and 5.17, we plotted the latest ACKed sequence numbers as a function of time. So, the slope of the curves are proportional to the instantaneous goodput at that time. In all these experiments, RTS/CTS was turned on. We see that in all cases, the slope of the



(a) 2 nodes

(b) 3 nodes

(c) 4 nodes

(d) 5 nodes

(e) 6 nodes

(f) 7 nodes

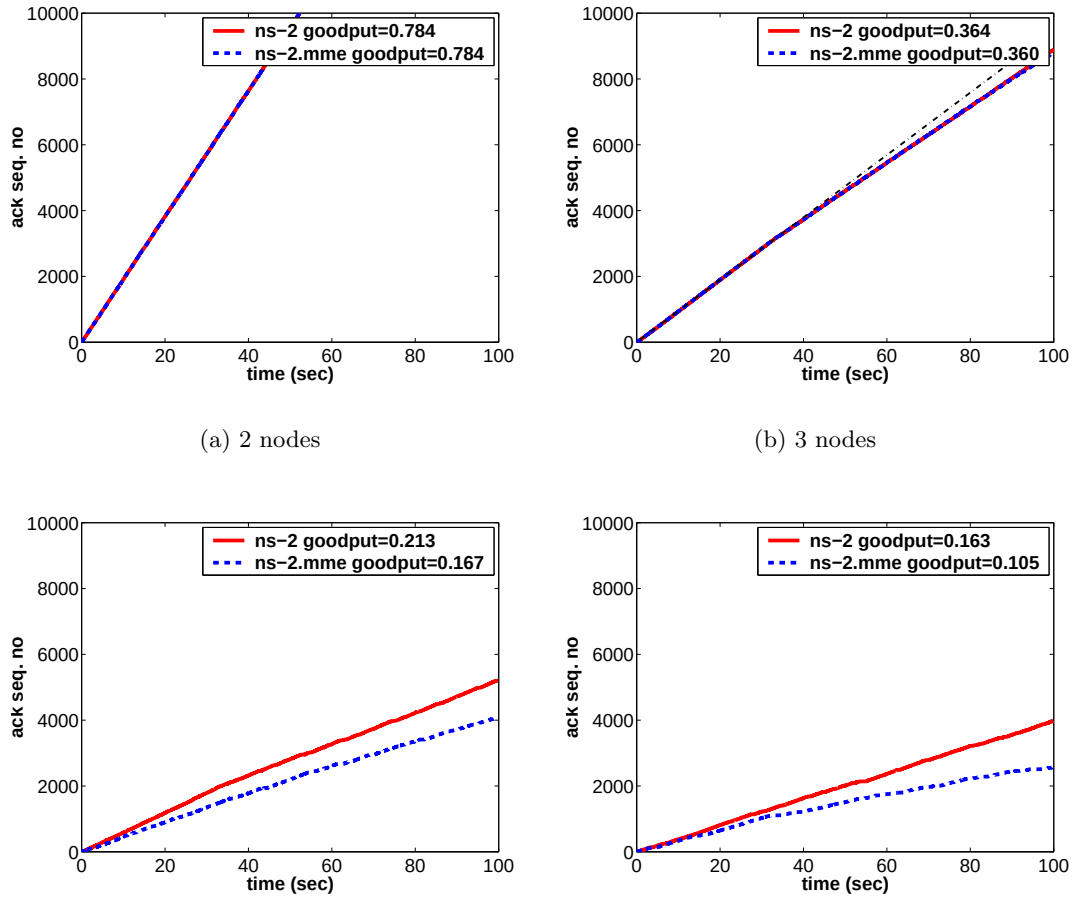
(g) 8 nodes

(h) 9 nodes

Figure 5.15: 8 similar multi-hop TCP experiments with 512 bytes packets at 2Mbps

curves in ns-2 results are almost constant throughout the entire simulation, and this slope decreases with the increasing number of hops. However in ns-2.mme, each instances of the scenarios does not have the same slope, in fact the slope decreases during the simulation until it gets close to zero.

We see in Figure 5.16 that for 2 nodes there's no difference at all between the results of ns-2 and ns-2.mme. For 3 nodes, the slope of the curves change abruptly at $t = 34$. Because after $t = 34$ node 1 is a hidden terminal for node 3 and vice versa. There's a chance that RTS of node 1 and RTS of node 3 can collide at node 2. This negligible effect causes the slopes of the curves decrease slightly at $t = 34$. As expected, ns-2's model is capable of tracking these collisions. Therefore there's no difference between the results of ns-2 and ns-2.mme. We also see that ns-2.mme's goodput drops to zero after $t = 120$ because it's



(a) 2 nodes

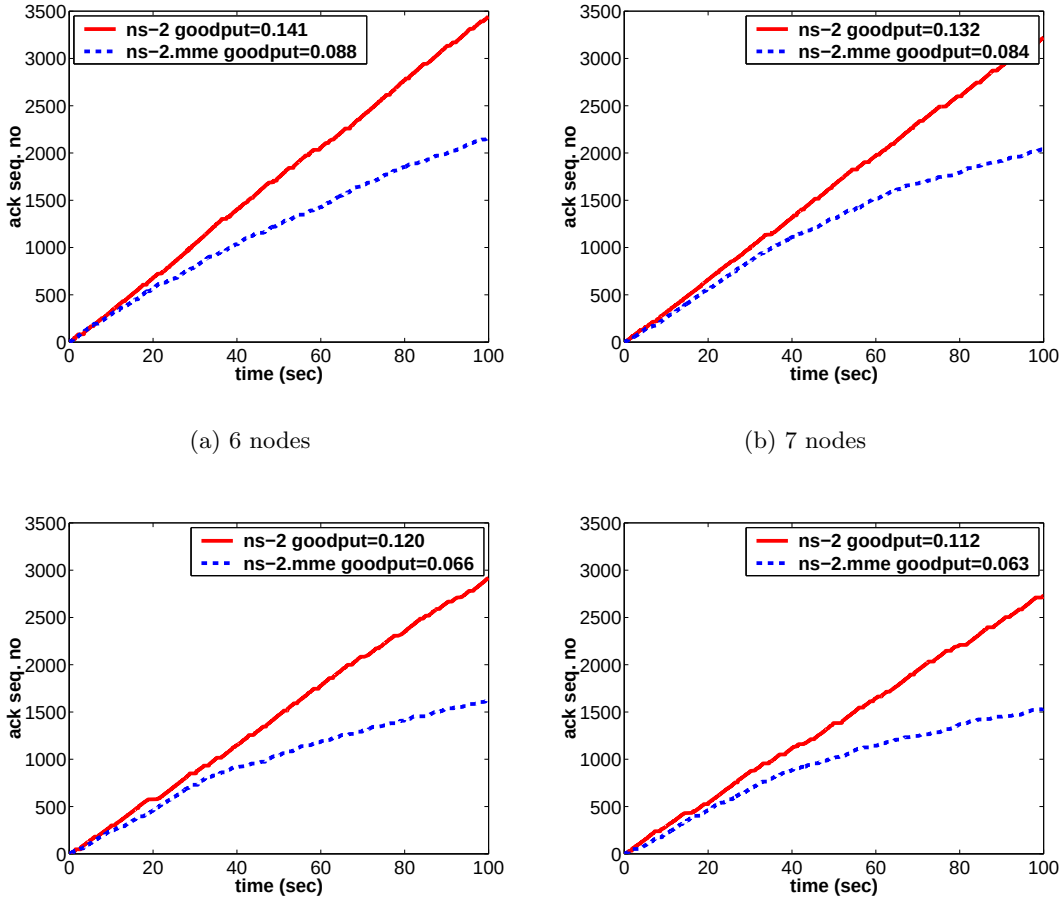
(b) 3 nodes

(c) 4 nodes

(d) 5 nodes

Figure 5.16: ACK sequence numbers vs. time with 512 bytes packets at 2Mbps

reaching the end of transmission range gradually. However this can't be observed with ns-2. In case of 4-nodes, there can be only one interferer at a time and collisions may occur at node 2 and 3. For example, when node 1 sends an RTS to node 2, node 2 broadcasts a CTS to all nodes. However, node 4 cannot receive this CTS correctly. If node 4 sends an RTS to node 3 at the same time, there will be collisions at node 2 and node 3. Due to the design of the scenario, the SINR is always 12dB at node 2 and 3 in case of a collision. At 12dB, the BER for 2Mbps is around 10^{-3}. RTS packets are 160 bits (ignoring the header) long, so this implies that in the worst case, the packet error rate will be $1-(1-10^{-3})^{160} = 0.15$.



(a) 6 nodes

(b) 7 nodes

(c) 8 nodes

(d) 9 nodes

Figure 5.17: ACK sequence numbers vs. time with 512 bytes packets at 2Mbps

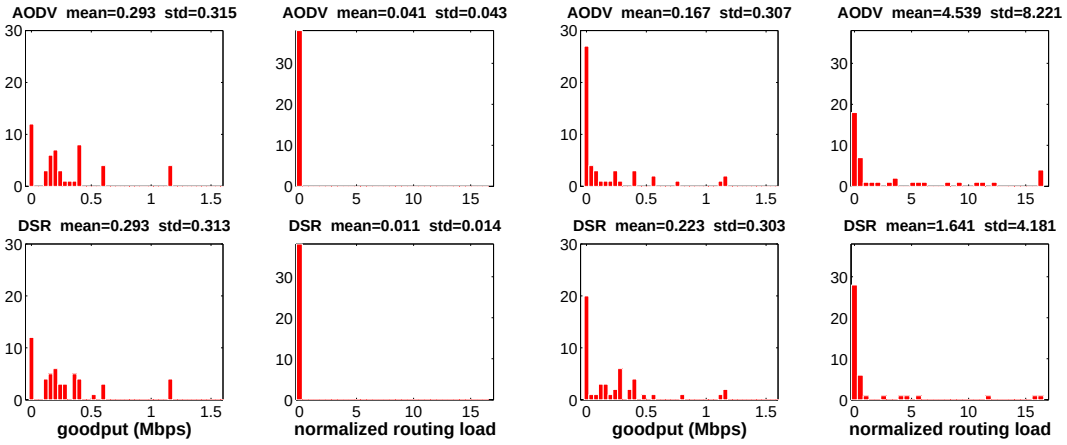
However, in Figure 5.16 we don't see such a big degradation. This is because node 1 and node 4 are not always transmitting at the same time. We also observe that with four nodes the change of slope at $t = 34$ is not clearly observable. In fact, with ns-2.mme it's almost impossible to observe this behavior. Because in ns-2.mme, the goodput is already degraded by node 4's transmissions. We also see that ns-2.mme's goodput drops to zero around $t = 120$ because it's reaching the end of transmission range gradually. Because of all these factors, ns-2.mme's goodput is 67 percent of ns-2's goodput. As expected, this difference increases when new nodes are added to the scenario.

	AODV	DSR
RTS/CTS ON	40%	15%
RTS/CTS OFF	35%	10%

Table 5.1: Goodput drop relative to ns-2 in Figure 5.14

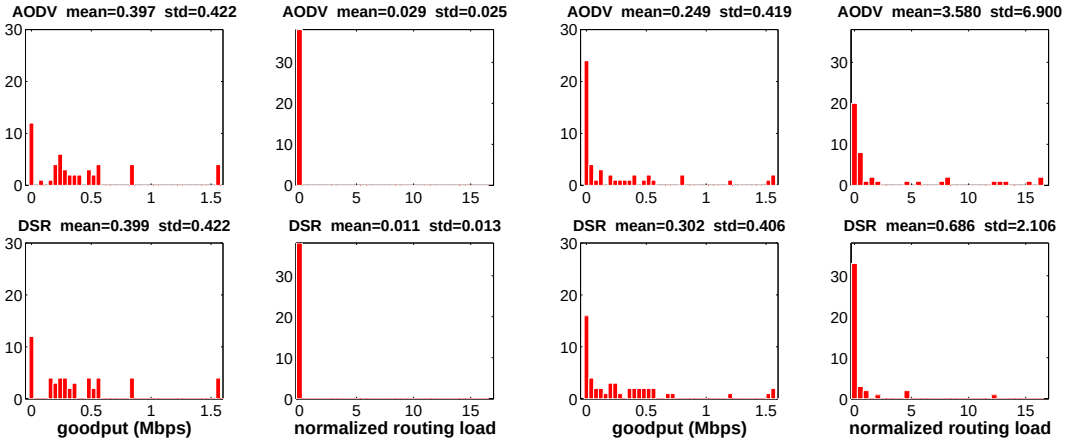
We can think of every instance of these simulations as a stationery case. In some cases we see that the slope of ns-2.mme results in Figure 5.16 and Figure 5.17 are very low. In these cases, link breakages can occur, forcing the routing protocol to look for a new route. Until a new route is discovered, no packet can be transmitted, and this causes the goodput to drop. Although there may be multiple routes from node 1 to the destination, the routing protocol may eventually rediscover the same route again since it's based on shortest path routing. The breaking and rediscovery of the path can result in much worse performance. In this case, faster route discovery results in better performance. When we look at Table 5.1, we see that the goodput drop is much less in DSR. Due to its aggressive caching strategy, DSR takes much less time to find a route than AODV.

AODV's failure to quickly find a route can be clearly seen in Figure 5.14. When we switch from ns-2 to ns-2.mme, the normalized routing load of AODV increases approximately 18 times with RTS/CTS and 8 times without RTS/CTS. The normalized routing load is usually less than 0.2 but we see that in some cases it can get as high as 10. This degradation is worse with the RTS/CTS mechanism because when RTS/CTS is off, there are more hidden nodes and ns-2's simple model can easily track the major collisions. However, when RTS/CTS is on, ns-2 is not able to track most of the collisions due to multiple interferers. On the other hand, ns-2.mme employs cumulative SINR tracking and therefore there will be collisions which RTS/CTS cannot prevent. This is also seen in Table 5.1 where goodput drop is higher when RTS/CTS is turned on.



(a) ns-2 RTS/CTS ON

(b) ns-2.mme RTS/CTS ON



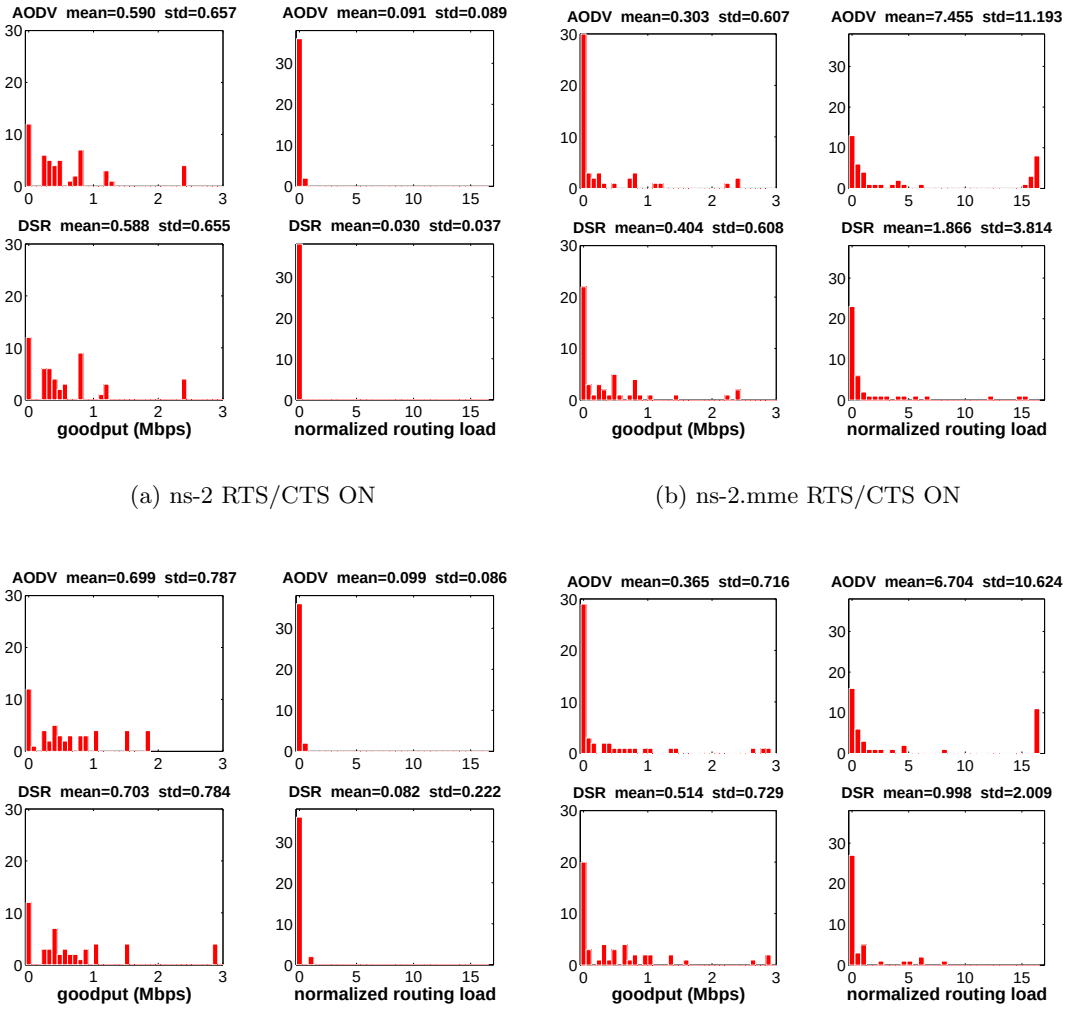
(c) ns-2 RTS/CTS OFF

(d) ns-2.mme RTS/CTS OFF

Figure 5.18: Goodput and normalized routing load with 512 bytes packets at 5.5Mbps

When we repeated the same experiments at 5.5 Mbps, we saw that in 12 experiments, the TCP sender and receiver were totally disconnected (Figure 5.18 and Figure 5.19). At first glance, this might seem too high but this is actually expected since the range of 5 Mbps data rate at 10 dBm is only 150 meters.

When we look at the performance of RTS/CTS at 5 Mbps we see that the basic access mechanism (CSMA) performs better than the RTS/CTS mechanism in every case. This is because when we increase the data rate, the overhead of RTS/CTS increases. In this final



(a) ns-2 RTS/CTS ON

(b) ns-2.mme RTS/CTS ON

(c) ns-2 RTS/CTS OFF

(d) ns-2.mme RTS/CTS OFF

Figure 5.19: Goodput and normalized routing load with 1500 bytes packets at 5.5Mbps

case, ns-2 and ns-2.mme results actually agree on whether RTS/CTS performs better than CSMA. However, with 512-byte packets at 1Mbps and 1500-byte packets at 2Mbps ns-2 results favor RTS/CTS scheme whereas there's essentially no improvement with RTS/CTS in ns-2.mme results. When we compare the performance of AODV and DSR, we see a big contradiction between the results of ns-2 and ns-2.mme. In ns-2, both routing protocols perform almost the same. However, in ns-2.mme results, DSR outperforms AODV in every example with stationery networks having single TCP session for the same reasons that we

explained before.

Chapter 6

Conclusion and Future Work

In this thesis, we examined the effects of different physical layer models on the performance evaluation of higher layer protocols. In Chapter 2 we presented the details of 802.11b PHY layer which are relevant to our modifications. In Chapter 3 we explained the fundamentals of wireless PHY layer modeling in ns-2 and in Chapter 4 we presented the flaws in this modeling and the reasoning behind our modifications. Later in Chapter 4, we presented the details of our modifications and verified our model analytically on some understandable simple simulations.

In Chapter 5, we quantified the differences between ns-2 and our model under typical large-scale scenarios used for the performance evaluation of wireless ad hoc routing protocols. In these experiments we have observed significant divergences between the results of simulations using different physical layer models. The differences are not only quantitative (not the same absolute value) but also qualitative (not the same behavior). We saw that in continuously mobile networks with UDP traffic, the choice of physical layer model can affect DSR packet delivery ratio by as much as 50 percent, whereas the performance of AODV is relatively insensitive. On the other hand we saw that in stationery networks with TCP traffic, the choice of physical layer model affects both routing protocols, especially AODV's goodput degrades by as much as 50 percent.

We also showed that when experimenting with tunable 802.11 parameters such as CST

and RTS/CTS threshold, ns-2 can give results that contradict the results of our model. We saw that in ns-2, increasing the CST value increases the packet delivery ratio because the number of instantaneous transmissions increase. However with ns-2.mme, increasing the CST value actually degrades the performance because of the SINR tracking to record the cumulative effect of interference. We also saw that ns-2's wireless PHY layer model doesn't have enough details to judge if RTS/CTS performs better than basic CSMA scheme.

We believe that this thesis makes contributions in two areas. First, we describe our modifications to the ns-2 network simulator to provide a more accurate simulation of the interactions between the PHY and the upper layers, including accumulative SINR calculations, which doesn't slow down the simulations excessively. Second, these modifications can be a starting point for the examination of a variety of other problems which currently cannot be properly addressed by ns-2 simulation. Two such examples are:

- **Transmission Rate Adaptation** All 802.11 cards support a collection of transmission rates. However, reliable decoding of a higher rate demands a higher link quality. Thus higher rate transmissions will have a shorter range, or require higher power to maintain the same range. Although power adaptive routing has been examined, the combination of power, rate and route adaptation has not been fully explored.

- **Network Simulation with other Physical Layers** With the recent FCC ruling on ultra wideband (UWB) operation [47], there is growing interest in networks of low power UWB devices [48] and coexistence of UWB devices and existing networks.

These two problems constitute the primary part of our future work.

References

[1] Opnet Modeler. http://www.opnet.com/products/modeler/home.html.

[2] The Network Simulator - Ns-2. http://www.isi.edu/nsnam/ns/.

[3] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[4] GloMoSim. http://pcl.cs.ucla.edu/projects/glomosim/.

[5] S. R. Das, C. E. Perkins, and E. M. Royer. Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks. In *Proc. of the IEEE INFOCOM 2000 Conference*, March 2000.

[6] D. Maltz, J. Broch, J. Jetcheva, and D. Johnson. The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communication*, 17, August 1999.

[7] William H. Tranter, K. Sam Shanmugan, Theodore S. Rappaport, and Kurt L. Kosbar. *Principles of Communication Systems Simulation with Wireless Applications*. Prentice-Hall, 2002.

[8] M. Bansal and G. Barua. Performance comparison of two on-demand routing protocols for mobile ad hoc networks. In *IEEE International Conference on Personal Wireless Communications*, pages 206–210, December 2002.

[9] T.D. Dyer and R.V. Boppana. Routing http traffic in a mobile ad hoc network. In *Proceedings of MILCOM*, volume 2, pages 958–963, October 2002.

[10] E.M. S. Gwalani, Belding-Royer and C.E. Perkins. Aodv-pa: Aodv with path accumulation. In *IEEE International Conference on Communications*, volume 1, pages 11–15, May 2003.

[11] N. Bai, F. Sadagopan and A. Helmy. Brics: a building-block approach for analyzing routing protocols in ad hoc networks-a case study of reactive routing protocols. In *IEEE International Conference on Communications*, volume 6, pages 3618–3622, June 2004.

[12] D. Sun and H. Man. Tcp flow-based performance analysis of two on-demand routing protocols for mobile ad hoc networks. In *Vehicular Technology Conference VTC 2001 Fall. IEEE VTS 54th*, volume 1, pages 272–275, 2001.

[13] 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 *Proc. of Fourth Annual ACM/IEEE Int. Conf. Mobile Computing and Networking*, pages 85–97, October 1998.

[14] N. Hedman B. Mielczarek P. Johansson, T. Larsson and M. Degermark. Routing protocols for mobile ad-hoc networks - a comparative performance analysis. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 195–206, 1999.

[15] S. Das, R. Castaneda, and J. Yan. Simulation based performance evaluation of mobile, ad hoc network routing protocols. *ACM/Baltzer Mobile Networks and Applications (MONET) Journal*, pages 179–189, July 2000.

[16] G. Holland and N. H. Vaidya. Analysis of tcp performance over mobile ad hoc networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 219–230, 1999.

[17] Y. C. Hu and D. Johnson. Caching Strategies in On-demand Routing Protocols for Wireless Ad Hoc Networks. page 231242, August 2000.

[18] Jay Martin Mineo Takai and Rajive Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking and computing* , pages 87–94, 2001.

[19] John Heidemann, Nirupama Bulusu, Jeremy Elson, Chalermek Intanagonwiwat, Kun chan Lan, Ya Xu, Wei Ye, Deborah Estrin, and Ramesh Govindan. Effects of detail in wireless network simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, pages 3–11, Phoenix, Arizona, USA, January 2001. Society for Computer Simulation.

[20] E. Zimnyi F. Grenez J. Dricot, Ph. De Doncker. Impact of the physical layer on the performance of indoor wireless networks. In *Proc. of the Int. Conf. on Software, Telecommunications and Computer Networks*, Split, Croatia, October 2003.

[21] J. Dricot and Ph. De Doncker. High-accuracy physical layer model for wireless network simulations in ns-2. In *Proc. of the Int. Workshop on Wireless Ad-hoc Networks, IWWAN'04*, Oulu, Finland, May-June 2004.

[22] Cisco aironet 350 series client adapters data sheet. http://www.cisco.com.

[23] Internet Engineering Task Force MANET Working Group Charter. http://www.ietf.org/html.charters/manet-charter.html.

[24] International Standard ISO/IEC 8802-11: 1999E. ANSI/IEEE Standard 802.11, 1999 Edition.

[25] Bruce Tuch. Development of Wavelan, an ISM band wireless LAN. *ATT Technical Journal*, 72:27–33, July/August 1993.

[26] Theodore S. Rappaport. *Wireless Communications: Principles and Practice, 2nd Edition.* Prentice Hall, 2001.

[27] J. Zander. Radio resource management in future wireless networks: requirements and limitations. *IEEE Communications Magazine*, 35:30–36, August 1997.

[28] Andrew J. Viterbi. *CDMA Principles of Spread Spectrum Communication.* Addison-Wesley, 1995.

[29] Nicholas Bambos, Shou C. Chen, and Gregory J. Pottie. Radio link admission algorithms for wireless networks with power control and active link quality protection. In *INFOCOM (1)*, pages 97–104, 1995.

[30] Simon Haykin and Michael Moher. *Modern Wireless Communication*. Prentice-Hall, 2004.

[31] S. Verdu. *Multiuser detection*. Cambridge University Press, 1998.

[32] M. B. Pursley. Performance evaluation for phase-coded spread-spectrum multiple-access communications part i: System analysis. *IEEE Trans. Comm.*, COM-25(8):795–799, August 1977.

[33] Ivan Seskar and Narayan B. Mandayam. A software radio architecture for linear multiuser detection. *IEEE Journal on Selected Areas in Communications*, 17(5):814–823, May 1999.

[34] J. G. Proakis. *Digital Communications*. McGraw Hill, 1989.

[35] Zhao-yang Zhang Shao-bo Liu, Aiping Huang and Zhijian Zhang. Performance analysis of cck modulation under multipath fading channel. In *Proc. of the Sixth Nordic Signal Processing Symposium - NORSIG 2004*, pages 276–279, June 2004.

[36] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Kluwer Academic Publishers*, pages 153–181, 1996.

[37] C. E. Perkins and E. M. Royer. Ad Hoc On-Demand Distance Vector Routing. In *Proc. 2nd IEEE Wksp. Mobile Comp. and Apps.*, pages 90–100, February 1999.

[38] G. Resta C. Bettstetter and P. Santi. The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, July-Sept 2003.

[39] L. Lily Yang Jing Zhu, Xingang Guo and W. Steven Conner. Leveraging Spatial Reuse in 802.11 Mesh Networks with Enhanced Physical Carrier Sensing. In *IEEE Internatioanl Conference on Communications - ICC 2004*, June 2004.

[40] B. Liang J. Deng and P. K. Varshney. Tuning the Carrier Sensing Range of IEEE 802.11 MAC. In *Proc. of IEEE Global Telecommunications Conference - Wireless Communications, Networks, and Systems - Globecom'04*, November 2004.

[41] S. Gupta H. S. Chhaya. Performance Modeling of Asycnhoronous Data Transfer Methodsof IEEE 802.11 MAC Protocol. *Wireless Networks*, 3:217–234, 1997.

[42] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE J. Select Areas Commun.*, 18(3):535–547, 2000.

[43] T. Antanakopoulos E. Ziouva. CSMA/CA Performance Under High Traffic Conditions: Throughput and Delay Analysis. *Computer Commun.*, 25:313–321, 2002.

[44] F. Ye . S. T. Sheu, T. Chen. The Impact of RTS Threshold on IEEE 802.11 MAC Protocol. In *Proc. of IEEE ICPADS 2002*, pages 267–272, December 2002.

[45] E. Gregori. R. Bruno, M. Conti. IEEE 802.11 Optimal Performance: RTS/CTS Mechanism vs. Basic Mechanism. In *Proc. of IEEE PIMRC'02*, volume 4, pages 1747–1751, September 2002.

[46] Danny H.K.Tsang Zhen-ning Kong and Brahim Bensaou. Adaptive RTS/CTS Mechanism for IEEE 802.11 WLANs to Achieve Optimal Performance. In *Proceedings of ICC*, volume 1, pages 185–190, June 2004.

[47] Federal Communications Commission. New public safety applications and broadband internet access among uses envisioned by fcc consideration of ultra-wideband technology. *Docket No. 98-153*, pages 1–118, Feb. 2002.

[48] T. Shepard. Getting the most value out of the radio spectrum. *1999 International UWB Conference*, Sep. 1999.