# ADAPTIVE PROTOCOLS FOR MOBILE AD HOC NETWORKS

A Dissertation

by

GAVIN DOUGLAS HOLLAND

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2004

Major Subject: Computer Science

# ADAPTIVE PROTOCOLS FOR MOBILE AD HOC NETWORKS

A Dissertation

by

GAVIN DOUGLAS HOLLAND

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

December 2004

Major Subject: Computer Science

## ABSTRACT

Adaptive Protocols for Mobile Ad Hoc Networks. (December 2004)

Gavin Douglas Holland, B.S., Abilene Christian University;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Nitin H. Vaidya

Recent advances in low-power technologies have resulted in the proliferation of inexpensive handheld mobile computing devices. Soon, just like the Internet empowered a whole new world of applications for personal computers, the development and deployment of robust ubiquitous wireless networks will enable many new and exciting futuristic applications. Certain to be an important part of this future is a class of networks known as “mobile ad hoc networks.” Mobile ad hoc networks (or simply “ad hoc networks”) are local-area networks formed “on the spot” between collocated wireless devices. These devices self-organize by sharing information with their neighbors to establish communication pathways whenever and wherever they are. For ad hoc networks to succeed, however, new protocols must be developed that are capable of adapting to their dynamic nature.

In this dissertation, we present a number of adaptive protocols that are designed for this purpose. We investigate new link layer mechanisms that dynamically monitor and adapt to changes in link quality, including a protocol that uses common control messages to form a tight feedback control loop for adaptation of the link data rate to best match the channel conditions perceived by the receiver. We also investigate routing protocols that adapt route selection according to network characteristics. In particular, we present two on-demand routing protocols that are designed to take advantage of the presence of multirate links. We then investigate the performance of

TCP, showing how communication outages caused by link failures and routing delays can be very detrimental to its performance. In response, we present a solution to this problem that uses explicit feedback messages from the link layer about link failures to adapt TCP's behavior. Finally, we show how link failures in heterogeneous networks containing links with widely varying bandwidth and delay can cause repeated "modal" changes in capacity that TCP is slow to detect. We then present a modified version of TCP that is capable of more rapidly detecting and adapting to these changes.

To my wife, Lee Ann.

## ACKNOWLEDGMENTS

*No man is an island, entire of itself; every man is a piece of the continent, a part of the main.*

– Meditation XVII by John Donne.

When I think on the years that have passed and recall the faces of the people that have contributed to my small 'part of the main,' I am overcome – there have been so many. I can only hope that those who I could not list here already know that I owe them my heartfelt gratitude for their help and support.

First, I am extremely grateful to my Ph.D. advisor, Nitin Vaidya, for sharing his amazing talents and time with me. His ability to instruct, guide, and inspire is truly remarkable and is only surpassed by his knowledge and passion for his work. I think I can safely speak for all who have studied under him that his devotion to his students is genuine and steadfast, a rare and precious quality in the tricky waters of higher education. Nitin exemplifies what an ideal professor and advisor should be.

I also owe a debt of gratitude to the members of my committee, Professors Riccardo Bettati, Udo Pooch, and A. L. Narasimha Reddy, for their contribution to my education through the classes that they taught, as well as through the technical guidance that they have given me on the work in this dissertation. I feel fortunate to have studied under such an accomplished group of individuals.

I am especially grateful to those that collaborated with me directly on the research herein: Nitin, Victor Bahl at Microsoft Research, Yongguang Zhang at HRL Laboratories, and Tom Henderson and Jae Kim at Boeing Phantom Works. I owe a special thank you to Yeng Lee of UCLA, who collaborated with me on the development and simulation of the MM-TCP protocol. It was an honor and a pleasure to work with them, and I look forward to continued collaboration in the future. Others that have provided consultative or technical support on the simulations in this work are K. Narayanan of Texas A&M, who assisted with Jakes' method, and Timothy Anderson and Alon Peterson of HRL, who assisted with coding, simulations, and measurements. I thank them for their invaluable support.

I have been very fortunate to have often received financial support during my graduate student career. Most notably, the work in this dissertation was supported in part by grants from the National Science Foundation and the Texas Advanced Technology Program, a gift from Microsoft Research, and a fellowship from the Department of Education. My gratitude to Nitin, Jennifer Welch, Richard Volz, and Victor for contributing this support. I also thank Victor for supporting me for six months as an intern at Microsoft Research – a very rewarding experience.

Willis Marti and Ron Szabo deserve much gratitude for providing me with continuous work opportunities whenever and however it was needed. I am especially indebted to Ron and his wife Mari for their support and friendship through the years.

I am now fortunate to be employed by HRL Laboratories LLC, for which I thank Son Dao and Bo Ryu for the opportunity. I also thank my HRL colleagues for the many profitable discussions that have led to new and interesting areas of research.

Getting an advanced degree at any institution always involves a large team of personnel to handle the logistics of the educational process. For their support, I especially thank Professor Bart Childs, Susan Spears, Sandra Morse, Elena Rodriguez, and Mashalle Mooring in graduate advising; Professor Don Friesen, Professor Wei Zhao, Professor Richard Volz, Professor Valerie Taylor, and Sherry Escalante in administration; and Mary Jane Allen, Margaret Dunaway, and Glenda Elledge in finance. A special thanks to Professor William Lively for his advice on all matters of importance.

Most importantly, I would like to thank my family for their undying love and encouragement. My father and role model, Dr. Redus Holland, has worked very hard to provide me with whatever I needed, especially when I needed it most, and taught me the importance of hard work, good humor, and family. I certainly could not have made it this far without him. My mother, Pat Fair, and her husband, Don, have always embraced me with love and laughter, providing a comfortable haven for me when I needed it. My brothers, Brennan and Kiernan, and sister-in-law Kathryn, whom I admire dearly and look up to always, have been a continuous source of love and encouragement. Finally, a heartfelt thank you is owed to my father- and mother-in-law, Joe and Jill Holder, and sister- and brother-in-law, Bonnie and Rick Miller, and their families for their love and support throughout the years. God bless you all!

Finally, none of this would have been possible without the love and support of my wonderful wife, Lee Ann, to whom this is dedicated. I thank the Lord each day for letting me walk the path of my life in the hands of one of His most precious beings. Her devotion to me and this journey has taken her through some pretty awful jobs, sad cars, and tiny apartments, and yet, through it all, she has somehow always managed a smile and a word of encouragement. It is with great pleasure that I present this to her, closing this chapter in our lives and beginning the next one anew.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

# CHAPTER I

# INTRODUCTION

Over the past three decades computers have evolved from the room-sized mainframes of the 70's to the pocket-sized PC's we have today. In the future we will see smaller and more powerful computers in everything from clothing to coffee cups, luggage to books, and mite-sized sensors to pill-sized implants. Soon, just like the Internet empowered a whole new world of applications for desktop computers, the development of robust *ubiquitous wireless networks* will enable exciting new possibilities for the computing devices of the future.

A fundamental building block of these futuristic networks will likely include the class of networks known as *mobile ad hoc networks*. Mobile ad hoc networks (or simply "ad hoc networks") are local-area networks (LANs) formed "on the spot" between collocated wireless devices. These devices form an ad hoc network by sharing information with their neighbors to establish a routing fabric, and then cooperating to relay packets throughout the fabric. Consequently, communication pathways can be formed between devices whenever and wherever they are. This "infrastructure-less" self-organizing capability makes ad hoc networks extremely appealing for future wireless networking systems. However, for ad hoc networks to be realized, many challenges must be overcome.

## 1. The Challenges of Ad Hoc Networking

One of the greatest challenges in wireless networking is how to ensure the reliable transmission of data packets over the inherently unreliable wireless channel. Wire-

The journal model is *IEEE Transactions on Automatic Control*.

less transmissions are subject to a variety of physical phenomena that degrade the transmission, making it difficult to receive reliably [100]. While modern low-power, low-noise radios and advanced technologies like OFDM and MIMO can vastly improve efficiency, reliability is still well below that of wired links [101]. Techniques for improving reliability by adapting to channel conditions, such as dynamic power and rate adaptation techniques, have been shown to be very effective [10, 46, 48]. However, most adaptation approaches have been developed for cellular networks, with base stations acting as centralized coordinators. Thus, they are not well suited to the decentralized multihop architecture of ad hoc networks. While growing interest in wireless local-area networking has resulted in the ratification of a number of standards, including IEEE 802.11 [60], HomeRF [59], and HIPERLAN [35, 36], most do not provide link layer adaptation mechanisms. Notable among these is IEEE 802.11. IEEE 802.11 is the most prominent wireless LAN standard in use today, serving as the blueprint for millions of devices worldwide. The standard includes designs for infrastructure as well as ad hoc wireless networks for several different physical layers at a number of frequencies and data rates. Support for ad hoc networks includes network configuration, creation, and access control, as well as error detection, retransmission, and collision avoidance. However, 802.11 has limited transmission range and relatively basic link layer recovery mechanisms. Subsequently, as we will show, its performance can be improved significantly through the application of dynamic adaptation mechanisms.

Further challenges exist in higher networking layers. A significant problem is the routing of packets. In ad hoc networks, the network topology can change rapidly as nodes move, causing links to dynamically break and form. This continuous change stresses the routing protocol, which must frequently and rapidly build, tear-down, and rebuild routes to maintain connectivity. While it is desirable to reuse existing technology where possible, conventional routing protocols were not designed to handle such dynamism and uncertainty. Instead, most protocol stacks such as the Internet standard TCP/IP [104] were designed to work best on relatively stable and reliable networks. Thus, most research in ad hoc networking has focused on routing protocols (e.g., LAR [77], TORA [88], DSR [19], AODV [92], DSDV [93], OLSR [23], and ZRP [51]). Many based on work produced by early DARPA packet-radio programs such as PRNet [71] and SURAN [15]. While comparisons between routing protocols have been undertaken (e.g., see [19], [102], [69], and [28]), little effort has focused on the cross-layer interactions between routing protocols and other network layers. As our research will show, however, link layer design can have a significant effect on route optimality. For instance, in networks with multirate links the link layer's handling of IP broadcast packets may cause the routing protocol to discover routes that are unusable. Although an IETF (Internet Engineering Task Force) working group has started to address such interactions (e.g., see [38], [31], [30], and [74]), many issues, such as impacts on routing, are largely unexplored. However, the group has recognized the effectiveness of cross-layer adaptability, and has very recently initiated development of an interface to facilitate link layer feedback [26].

Transport protocols, such as TCP, are also challenged by ad hoc networks. TCP is a connection-oriented protocol that provides a reliable byte stream service using built-in flow and congestion control mechanisms. However, when operating over wireless links, TCP's performance can be dramatically poor [56], [17]. The problem is that TCP assumes all packet losses are caused by congestion, and, subsequently, responds to any loss by reducing its send rate. In wireless networks, however, losses can occur frequently and suddenly for reasons other than congestion, such as wireless errors. The end result is poor average network utilization due to lower than ideal TCP send rates. Previous work in this area has focused on, for example, handoffs [20] and wireless errors [12, 9] in cellular systems, bandwidth asymmetry in cable modem and digital subscriber lines [11], and large round-trip times in satellite networks [34]. While some aspects of these approaches are applicable to ad hoc networks, many are not simply because of the differences between infrastructure and ad hoc networks. For instance, the approach in [12] (SNOOP) will only work if the forward and reverse routes are symmetric and the wireless link is unreliable (i.e., no ARQ). Furthermore, while performance analyses have been conducted for fixed multihop wireless networks, such as those in [44] and [41], few have conducted in-depth studies on TCP performance in ad hoc networks.

## 2. Contributions

In response to these challenges, we present a number of *cross-layer adaptable* protocols that dynamically adapt their behavior to match changes in network characteristics based on cross-layer feedback. Our contributions in this area include the following:

In this dissertation, we investigate new link layer mechanisms that dynamically monitor and adapt to changes in link quality by adjusting packet transmission characteristics according to certain optimization criteria; for example, by increasing transmission power when an increase in the bit error rate is detected. We show that the use of such techniques in ad hoc networks can significantly improve performance, and present a novel approach that uses common control messages to form a tight feedback control loop for adaptation of link characteristics based on receiver-perceived link conditions.

We also investigate new routing protocols that adapt to dynamic changes in link characteristics. Our work on such “link-aware” routing predominantly concentrates on protocols that take best advantage of the presence of multirate links. In particular, we present two on-demand routing protocols that consider link data rates in their route selection. In these protocols, we explore a number of different ways in which data rate information can be learned and disseminated, as well as how such information can be used to good effect.

Finally, we show, through an in-depth analysis of TCP performance in ad hoc networks, that communication outages caused by link failures and routing delays can be very detrimental to TCP performance. Specifically, we show how TCP throughput can drop dramatically because of consecutive spurious timeouts caused by link breakage and subsequent routing errors, and present our proposed solution to the problem: the Explicit Link Failure Notification (ELFN) protocol. ELFN is an adaptation protocol for TCP that uses explicit feedback messages from the link layer about link failures to adapt TCP's congestion control mechanisms to prevent spurious timeouts. We also demonstrate how link failures in heterogeneous networks containing links with widely varying bandwidth and/or delay can cause "modal" capacity shifts that are undetected by TCP, resulting in less-than-ideal throughput, and present our proposed solution, Multi-Mode TCP (MMTCP). MMTCP is an adaptation protocol for TCP that is capable of recognizing capacity shifts and adapting TCP's congestion and flow control mechanisms to respond more rapidly.

## 3. Thesis Organization

The remainder of this dissertation is organized as follows. In the next chapter, we present background information for each of the areas we have explored. This includes a discussion of the characteristics of the wireless LAN channel and the devices designed to operate over this channel. We then survey relevant ad hoc network routing protocols, with an emphasis on the DSR routing protocol that is referenced throughout this dissertation. Finally, we present an overview of the operation of TCP, and the variants that have been devised to manage the problems in wireless networks. In

Chapter III, we give a brief overview of the simulation environment and methodology that we used to analyze and demonstrate the performance of the protocols that we have developed.

The content of Chapters IV–VII form the technical core of the dissertation. In Chapter IV we present our work on multirate medium access control (MAC) protocols, including the Receiver-Based AutoRate (RBAR) protocol, which is capable of rapidly adapting the data rate to best match channel conditions. Chapter V is concerned with the how ad hoc networks with multirate links affect routing, and, conversely, how on-demand routing can be modified to opportunistically adapt route selection based on information about link data rates on a path. In Chapters VI and VII, we present novel TCP protocols that are designed to adapt to the dynamic and unpredictable topology changes prevalent in wireless ad hoc networks. In Chapter VI, we present an in-depth analysis of the behavior of TCP in the presence of link failures, and present our solution: the Explicit Link Failure Notification (ELFN) protocol. In Chapter VII, we look at the problem of how link failures in networks composed of links with wide variations in bandwidth and delay can cause modal “shifts” in capacity, and present our Multi-Mode TCP (MMTCP) protocol as a potential solution. Finally, in Chapter VIII, we conclude the dissertation by summarizing our work and presenting problems for future study.

# CHAPTER II

# BACKGROUND AND RELATED WORK

In this chapter we present background information relevant to the work presented in this thesis. The material is organized according to the order in which it is addressed in later chapters. We set the stage by giving an overview of wireless communications technology, starting first with a description of pertinent characteristics of the wireless channel. Next, we discuss the design of devices commonly used for ad hoc networking. Included in both sections are formulas used in the simulation analysis of Chapter IV. We then provide background information on protocols for ad hoc routing, focusing special attention on the Dynamic Source Routing (DSR) protocol that we mention often in Chapters V and VI. Next, we present an overview of the Transmission Control Protocol (TCP) and the performance problems it encounters when operating over wireless links. Finally, we conclude the chapter by presenting background information on techniques that have been proposed for making TCP more robust in ad hoc networks.

## A. Wireless Channel

Noisy, error-prone, and subject to frequent outages, conditions on a wireless link can be, on average, a hundred times worse than on a wired link [100]. The cause predominantly lies with the less-than-ideal transmission characteristics of the radio channel. As a signal from a radio propagates through the environment, it is dramatically attenuated and distorted by physical phenomenon such as *path loss*, *shadowing*, and *fading*. The degree and nature of that depends on many factors, such as the operating frequency, the characteristics of the terrain, and the mobility of the participating radios. Such phenomena are commonly categorized based on the timescale of their fluctuations, where path loss and shadowing are *large-scale* effects, and multipath fading is a *small-scale* effect. When we couple these effects with the inherent *noise* from the receiver's electronics and *interference* from competing transmissions, the original signal is often difficult to recover. The result is a potentially unreliable communication channel. In this section, we give an overview of these phenomena and their impact on wireless communications. First, however, we present signal quality measures and notation from communications theory. More thorough treatment of the subject can be found in [100].

### 1. Signal Quality

Named after Alexander Graham Bell [83], the $bel = \log(P_1/P_2)$ was originally a measure used to compare the relative power of one sound $P_1$ to another $P_2$. When it was discovered that the human hearing was more sensitive to lower sound levels than was previously thought, the *bel* was scaled by a factor of 10 and called a "decibel" ($dB$). The decibel is now used extensively in communications for comparing signal powers. While dB is a relative measure, when the absolute power of a signal is desired it is often handy to represent it in decibel notation in relation to a base power such as one milliwatt ($dBm$) or one watt ($dBW$).

The quality of a received signal is frequently expressed as the *signal to noise ratio* (SNR) or *signal to interference and noise ratio* (SINR), which is exactly as named: SNR is the signal power over the noise power, and SINR is the signal power over the sum of interference and noise power. Clearly, the higher the ratio the better the quality of the received signal.

When discussing transmission quality it is often beneficial to represent the received signal in a way that considers the rate at which data is being transmitted.

Thus, the *bit energy to noise ratio* ($Eb/No$) is frequently used, which is defined as

$$\frac{Eb}{No} = SNR\left(\frac{Bt}{Rb}\right) , \tag{2.1}$$

where $Rb$ is the rate at which data bits are transmitted, and $Bt$ is the bandwidth of the signal.

## 2. Path Loss

Path loss is defined as the difference in received signal power to transmitted signal power. The predominant cause is the dispersion of the radio wave as it propagates omni-directionally away from the transmitter. As the distance between radios increases, the received power falls because the energy density at a point on the wave decreases as the wave expands. Theoretically, the received power is inversely proportional to the square of the distance between the radios. This is the basis for the well known *Friis Free Space* path loss model

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} , \tag{2.2}$$

where $d$ is the distance between the radios, $P_r$ and $P_t$ are the receive and transmit powers (in Watts), $G_t$ and $G_r$ are the transmit and receive antenna gains, $\lambda$ is the carrier wavelength, and $L$ is a system loss factor.

However, this model is not accurate for many practical environments because it assumes perfect *free space* propagation. Instead, measurements show that signal power does decrease logarithmically but not always with an exponent of two. This is expressed in the widely used *Log-Distance* path loss model

$$P_l(d) = P(d_0) + 10n\log(\frac{d}{d_0}) , \tag{2.3}$$

where $P_l(d)$ is the path loss (in dB), $n$ is the *path loss exponent*, and $P(d_0)$ is the path loss (in dB) at some close-in reference distance $d_0$. Often, $P(d_0)$ is found by measurement, but Equation 2.2 can also be used. Appropriate values of $n$ have been proposed for a number of different environments. They tend to fall in the range of 2-5, where higher values represent environments with a large number of obstructions, such as city streets and office buildings.

A related model that is often referred to is the *Two Ray* path loss model, which states that power decreases more rapidly ($n = 4$) after a certain distance. It is based on the observation that as the distance between the transmitter and receiver increases the power of the signal along the line-of-sight (ray one) decreases to a point where the signal's reflection off of the ground (ray two) becomes a significant part of the received signal. Unfortunately, the second ray arrives out of phase with the first so the two interfere destructively at the receiver, causing a more rapid decline in receive power.

### 3. Shadowing

Shadowing is defined as the attenuation in power that occurs when a signal passes through an obstruction in its path, such as a building or a hill. The degree to which the signal attenuates depends on the composition of the material, its thickness, and its physical characteristics, such as its reflection, refraction, and absorption coefficients in the signal's frequency band. Extensive measurements have been made for many common materials and building structures. Results show that path loss $P_{ls}$ due to shadowing is log-normally distributed about the distance-dependent mean given by Equation 2.3, so

$$P_{ls}(d) \quad = \quad P_l(d) + X_\sigma \ , \tag{2.4}$$

where $X_\sigma$ is a zero-mean normal random variable with standard deviation $\sigma$. Attenuation due to environmental obstructions like rain, smog, and foliage can also be represented using a similar model.

## 4. Fading

Fading is a term often used interchangeably with *attenuation*. Here, it is used to mean small-scale *multipath fading*, which is the dramatic variation in signal power that occurs when transmissions from the same radio reflect off surfaces in the environment and arrive at the receiver at various times and with different phases, causing interference. The Two Ray model we described earlier is based on multipath fading, but considers only the large-scale effects on receive power.

In a multipath environment with no moving objects and a fixed transmitter, there are regions where the multipath rays combine constructively, resulting in a gain in power, or destructively, causing a fade in power. A receiver traveling through this environment will see dramatic variations in power as it moves from region to region, the rate of which is proportional to its speed. However, another speed-dependent effect arises which is related to its direction with respect to the arrival angle of each multipath ray. If the direction of the receiver is toward or away from a multipath ray, then a shift in frequency occurs, called the Doppler shift. The combination of the many Doppler-shifted multipath rays creates the rapid time-varying signal that is characteristic of multipath fading.

The rate at which a fading signal varies is often described in terms of the *coherence time* of the multipath channel. The coherence time $T_c$ is the period over which the channel can be assumed to be effectively constant. An approximation for this time is given by [100]

$$T_c(t) \approx \frac{9\lambda}{16\pi v(t)} \;, \tag{2.5}$$

where $v(t)$ is the speed along the line-of-sight between the sender and receiver at time $t$, and $\lambda = c/f_c$ is the wavelength of the carrier frequency $f_c$ ($c$ is the speed of light). Using this formula, we can see that for a radio transmitting at 2.4 GHz to a receiver moving with a line-of-sight velocity of 20 m/s (about 45 mph), the fading channel has a coherence time of about 11ms, which is about the time it takes to transmit a 1400 byte packet at 1 Mbps.

The statistical distribution of received power in a fading signal is described by two well-known distributions. The *Rayleigh* distribution is used when it is assumed that all arriving rays are approximately equal in amplitude, whereas the *Ricean* distribution is used when it is known that there is a dominant path with much greater amplitude.

## 5. Noise and Interference

Electronic devices suffer from internal temperature-dependent noise. This *thermal noise* is caused by an increase in the excitation of electrons within the component due to heat. For a single passive component, the noise power is

$$N_t \;=\; kTBt \;, \tag{2.6}$$

where $k$ is Boltzmann's constant ($1.38 \times 10^{23}$ Joules/Kelvin), $T$ is the temperature (in Kelvin), and $Bt$ is the component's bandwidth. The noise of a circuit is often given as a multiple of $N_t$, called the *noise figure*. This is usually provided by the designer or manufacturer of the circuit, based on exhaustive analysis of the individual devices or empirical measurement. For example, Intersil, a manufacturer of wireless LAN chipsets, claims a noise figure of seven for their popular Prism family of devices [116].

Interference caused by devices transmitting simultaneously in the same channel contribute to the degradation in signal quality by increasing the level of unwanted background energy. The impact on quality is dependent on how the interference combines with the received signal. For instance, multipath fading is essentially self-interference, where the result is dramatic variations in the received signal. The same could occur between two devices if their signals were highly correlated. However, in the spread spectrum techniques commonly in use today, interference appears as simple *additive white Gaussian noise* (AWGN) to the receiver. Thus, the interference can be modeled as a simple increase in the overall noise power for the duration of the interfering transmissions.

## B. Wireless Devices

The design of efficient wireless communication devices has been a hot topic of research since before the second world war. However, it wasn't until after the war (1948) that Shannon published his well known treatises on the mathematical theory of digital communication which laid the groundwork for what is now the field of information theory. Ever since, the design of wireless devices has been spurred on in pursuit of the goal of achieving the maximum limit on channel capacity $C = Bt \cdot \log_2(1 + SNR)$ that Shannon first presented fifty years ago [95]. There are a number of wireless LAN devices in the market today that provide data rates as high as 54 Mbps in a package small enough to fit in a pocket [63], [36]. In this section, we give a brief overview of relevant design aspects of these devices. The reader is referred to texts by Proakis [95] and Gibson [45] for more information.

## 1. Physical Layer

The operation of a conventional narrow-band wireless device is as follows. The incoming binary digital data stream is first passed to a channel encoder, which adds redundancy to the signal to increase its tolerance to transmission errors. A digital modulator then up-converts the digital signal into an analog signal by modulating the carrier frequency. This modulated signal is then amplified and transmitted out the antenna. At the receiver, the sequence is reversed. The received signal is passed to a digital demodulator which down-converts the analog signal into a binary digital stream, which is then fed into a channel decoder that uses the added redundancy in the stream to detect or correct errors. Here we describe in more detail the process of modulation and its impact on performance.

### a. Modulation

Higher data rates are commonly achieved by more efficient modulation schemes. Modulation is the process of translating an outgoing data stream into a form suitable for transmission on the physical medium. For digital modulation, this involves translating the data stream into a sequence of *symbols*. Each symbol may encode a number of bits, the number depending on the modulation scheme. The symbol sequence is then transmitted at a certain rate, the *symbol rate*. So, for a given symbol rate, the data rate is determined by the number of encoded bits per symbol.

The performance of a modulation scheme is measured by its ability to preserve the accuracy of the encoded data. In mobile wireless networks, path loss, fading, and interference cause variations in the received signal-to-noise ratio (SNR). Such variations also cause variations in the bit error rate (BER), because the lower the SNR, the more difficult it is for the modulation scheme to decode the received signal. Since

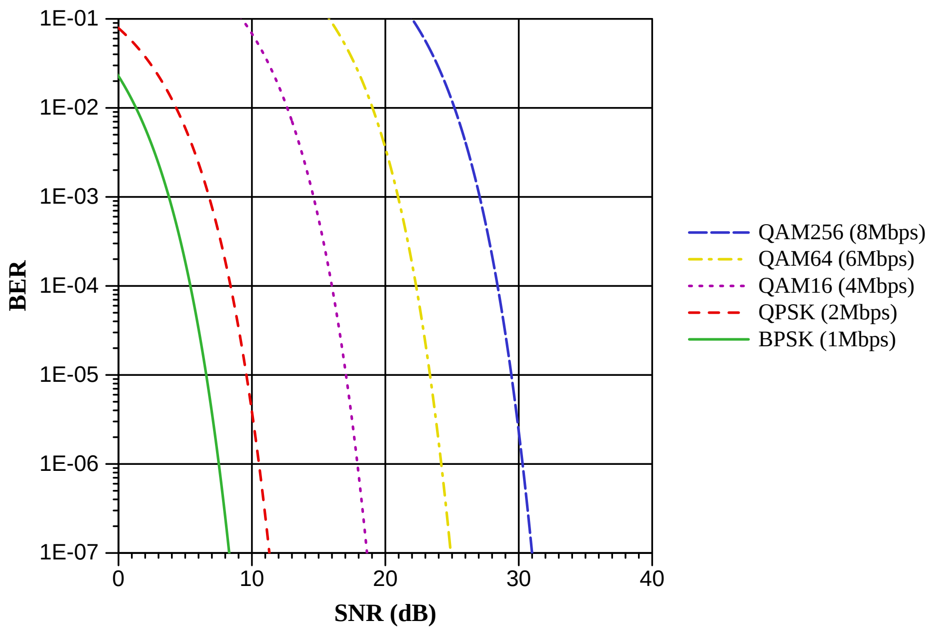


Fig. 1. *Theoretical Bit Error Rate (BER) as a function of the Signal-to-Noise Ratio (SNR) for several modulation schemes and data rates.*

high rate schemes typically use denser modulation encodings, a tradeoff generally emerges between data rate and BER: the higher the data rate, the higher the BER.

This tradeoff is illustrated in Fig. 1, which shows the theoretical BER as a function of the SNR for several different modulation schemes. Notice that for each modulation scheme the BER decreases with increasing SNR. Also notice that for a given SNR, an increase in data rate results in an increase in BER. For example, given an SNR of 10 dB, a packet transmitted at 4 Mbps using QAM16 modulation could experience a BER of 0.07, in comparison to $4 \times 10^{-6}$ for the same packet transmitted at 2 Mbps using QPSK modulation.

To illustrate the impact that this tradeoff can have on performance, Fig. 2 shows throughput as a function of distance for each of the modulation schemes shown in Fig. 1. For the sake of this illustration only large-scale path loss was modeled (using Equations 2.2 and 2.3) with a path loss exponent of 2. Notice that the lower

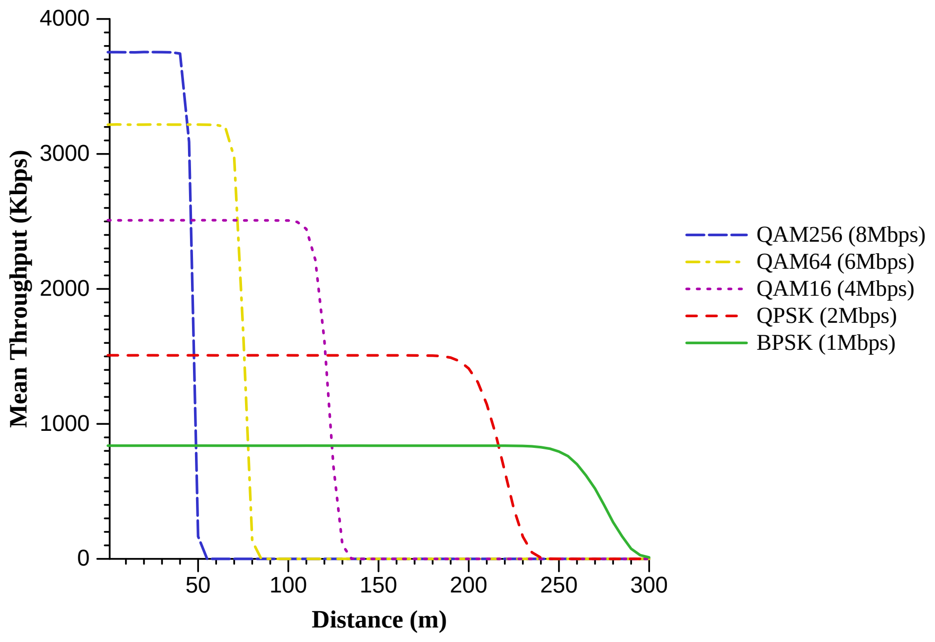


Fig. 2. *Comparison of throughput versus distance for several modulation schemes. Data was obtained by the simulation of two nodes at fixed positions, with one sending a continuous stream of UDP packets to the other. The propagation model was the log-distance path loss model, with a path loss exponent typical of an urban environment. Transmit power was constant.*

rate schemes have greater transmission ranges than the higher rate schemes. As the distance increases, the signal attenuates until the received SNR drops below the threshold required to maintain a tolerable bit error rate. This appears as a sharp drop in throughput in Fig. 2, corresponding to the steep curve in Fig. 1.

Thus, although many existing wireless LAN devices are designed with the capability of transmitting at multiple data rates, such as 802.11 [60] compliant devices like the Lucent WaveLAN II [72] and the Cisco/Aironet 350 (formerly PC4800) [3], no single data rate is optimal for all channel conditions.

b. Diversity

To enhance the resistance of the modulated signal to interference and fading, spread spectrum diversity is often used. One of the earliest spread spectrum techniques in use today had its birth in Hollywood around the start of the second world war. In 1942, an actress named Heddy Lamar patented “frequency hopping” as a technique to secure military radio transmissions. Dubbed *frequency hopping spread spectrum* (FHSS), the idea is simple: divide the channel into a number of independent frequency bands and synchronize the transmitter and receiver such that they periodically hop between bands during a transmission according to a predetermined hopping sequence. This has the advantage that interference on one frequency affects only a portion of the transmission, which can be handled by the decoder. Sequences and codes can be chosen such that overlap is minimized and manageable, allowing utilization of the channel by multiple radios. The 802.11 standard includes FHSS as one of a number of possible physical layers, for data rates up to 2 Mbps.

Another form of diversity that is widely used is *direct sequence spread spectrum* (DSSS). In DSSS, the input stream is “spread” by multiplying each data bit by a pseudo-random (pseudo-noise) bit sequence. The resulting sequence of bits (or *chips*) are then transmitted over a broad frequency band. This has the effect of spreading each data bit out in the frequency domain. At the receiver, the incoming signal is despread by bitwise multiplication of the chips by the same pseudo-noise sequence to recover the original data bit. The degree to which interference is reduced is proportional to the length of the pseudo-noise sequence. DSSS is very effective at rejecting fading and interference, and is more efficient than FHSS because it does not have the overhead inherent in switching between frequencies. The high-rate 802.11b [61] standard uses DSSS to achieve data rates as high as 11 Mbps.

Recently, a diversity technique called *orthogonal frequency division multiplexing* (OFDM) has seen greater use. In OFDM, the data bits are divided into multiple substreams and transmitted in parallel over multiple frequencies. These frequencies, or *subcarriers*, are chosen to be orthogonal, meaning that the interference between subcarriers is minimal. The result is very good interference rejection and performance. OFDM is currently used in the high-rate 802.11a [62] and 802.11g [63] standards, and the HIPERLAN [36] standard. Currently, each supports data rates up to 54 Mbps.

## 2. Data Link Layer

The wireless communication channel is a shared medium. If two or more wireless devices transmit simultaneously in the same channel their transmissions may interfere and prevent delivery of either packet. This is called a packet *collision*. Preventing collisions is an important problem in ad hoc networks because most wireless LAN radios are *half-duplex*, which means that the radio is designed to support two-way communication but only over the same radio channel, switching from send to receive as necessary. This is in contrast to *full-duplex* systems, which use two channels and two radios over which they can simultaneously send and receive. Coordination among half-duplex devices wishing to transmit is part of the responsibility of the data link layer. Other responsibilities include flow control and error recovery. This section covers salient features of link protocols for wireless devices.

### a. Medium Access Control (MAC)

MAC protocols prevent collisions by coordinating access to the channel among the devices that wish to transmit. Design goals of MAC protocols are to maximize channel utilization while maintaining an adequate level of fairness. The protocols that exist today can be classified as centralized or distributed, depending on whether the access control is the responsibility of a central node or all nodes, and guaranteed access or random access, depending on whether a node has a pre-allocated share of the channel or must contend for a share. In ad hoc networks, distributed random access protocols are most commonly used because of their suitability for dynamically changing network topologies.

The first distributed random-access protocol to be used in a wireless network was ALOHA [1]. In ALOHA, a node that is given a packet to send appends a *frame check sequence* (FCS) and immediately transmits it. The receiver acknowledges correct receipt of the packet (as indicated by the FCS) by transmitting an acknowledgment (ACK). On receipt of the ACK, the sender drops its buffered copy of the packet and continues to the packet. If no ACK arrives after some time, the sender retransmits the packet. After so many retransmission attempts, it drops the packet. This retransmission scheme is called a *Stop-and-Go Automatic Repeat Request (ARQ)* protocol, which is a commonly used error recovery scheme in wireless MACs.

The performance of ALOHA is poor in even moderately loaded networks because it has no mechanism to prevent collisions. This led to the development of *Carrier Sensing Multiple Access (CSMA)* protocols, where the node first listens to the channel (carrier sense) to determine if it is in use before it transmits its data packet. If the channel is busy, it backs off for some time and attempts to retransmit later. While it doesn't prevent collisions due to simultaneous transmissions, it significantly reduces the collision probability. Most MAC protocols in use today employ some form of carrier sensing.

In ad hoc networks, another problem arises in that a node may not always be able to sense another node's transmission. This creates an interesting situation where nodes that are unable to hear each other cause collisions at a node that can hear them both. This is the well-known *hidden terminal* problem, so named because the nodes that are unable to hear each other are essentially hidden from each other. A solution to the hidden terminal problem was devised and the resultant protocol was called *CSMA/CA*, for *Collision Avoidance*. In CSMA/CA, a node, after sensing the channel is free, signals its intent to transmit a data packet by transmitting a small *RTS (Ready to Send)* control packet to the receiver. If the receiver is capable of receiving the packet, it replies with a *CTS (Clear to Send)* control packet. On receipt of the CTS, the sender transmits the data packet. Nodes that overhear either the RTS or CTS defer their own transmissions for the duration of the data packet transmission. The purpose of this protocol is threefold: (1) to coordinate the transfer of the data packet between sender and receiver, (2) to announce the duration of the packet transfer to nodes that are in range of the sender and the receiver, and (3) to reduce the time the channel is unnecessarily in use in the instance that the first transmission collides. Since nodes maintain some state about the status of the channel based on receipt of RTS/CTS packets, the protocol is said to provide *virtual carrier sensing* to supplement the physical carrier sensing in CSMA. A number of MAC protocols have been proposed that incorporate the RTS/CTS protocol, such as SRMA [106], MACA [73], MACAW [16], FAMA [42], and the DCF protocol in 802.11.

### b. IEEE 802.11

IEEE 802.11 [60] is the de facto standard for wireless LAN devices. Part of the IEEE 802.x LAN/MAN family of standards, it serves as the blueprint for millions of devices worldwide. Defining both physical and data link layers, the 802.11 standard includes designs for infrastructure as well as ad hoc wireless networks for several different physical layers at a number of frequencies and data rates. Work on the standard is ongoing, and recent extensions include 54 Mbps OFDM-based physical layers in the

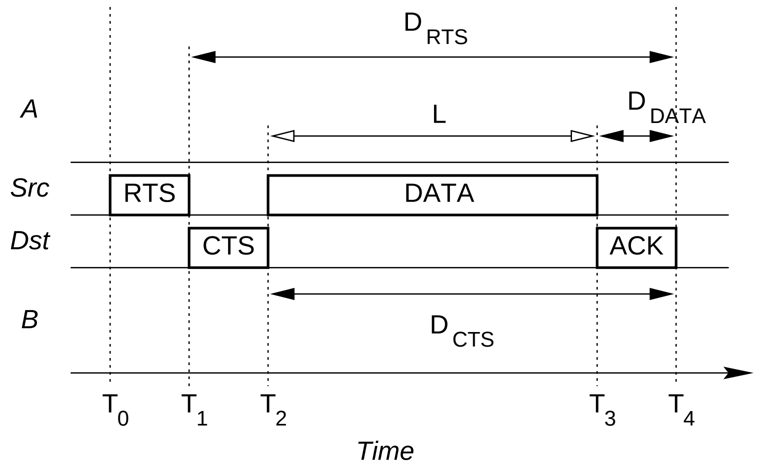


Fig. 3. *Timeline showing the RTS/CTS protocol in the IEEE 802.11 Distributed Coordination Function (DCF) for transmitting a data packet. Here, A and B are nodes that are in range of the transmitter and receiver, respectively.* $D_{RTS}$*,* $D_{CTS}$*, and* $D_{DATA}$ *are the lengths of the reservations given in the RTS, CTS and DATA packets, and L is the duration of the data packet transmission.*

5 GHz [62] and 2.4 GHz [61] ISM (unlicensed access) frequency bands. Support for ad hoc networks includes network configuration, creation, and access control, as well as error detection, retransmission, and collision avoidance. Its MAC protocol for ad hoc networks is called the *Distributed Coordination Function (DCF).*

**Distributed Coordination Function (DCF):** The Distributed Coordination Function (DCF) in 802.11 is an implementation of the RTS/CTS protocol, and is illustrated in Fig. 3, which is a time-line portraying the sequence of events that occur for a single packet transfer. Here, the source *Src* has a data packet to transmit to the destination *Dst* with a duration of length $L$. Node $A$ is in range of *Src* but not *Dst*, and node $B$ is in range of *Dst* but not *Src*. The protocol proceeds as follows. When *Src* has a packet to send, it calculates the length of time it will take to transmit the data packet at the current data rate, and then adds to that the transmission time of the CTS and ACK packets, which forms the duration of the reservation ($D_{RTS}$).

The *Src* then transmits $D_{RTS}$ in the RTS to *Dst*, using one of the rates in the *basic rate set.* The *basic rate set* is the set of rates that all nodes are required to support, which ensures that all nodes that are in transmission range are able to receive and demodulate the RTS/CTS packets. Since node $A$ is in range of *Src*, it overhears the RTS and summarily defers its own transmissions for the duration of the reservation in the RTS ($D_{RTS}$), starting from the moment that it received the RTS ($T_1$). If *Dst* is capable of receiving the data packet, it responds by transmitting a CTS packet back to *Src* containing the time remaining in the reservation ($D_{CTS}$), which it calculates by subtracting the transmission time of the CTS from $D_{RTS}$. Node $B$, overhearing the RTS, learns of the requested reservation and, like $A$, defers for length $D_{CTS}$. At this point, transmission of the data packet and subsequent ACK can now proceed without interference from $A$ or $B$. However, in the off-chance that $A$ did not receive the RTS, due to, for example, an RTS collision caused by another node, the data packet also carries the time remaining in the reservation $D_{DATA}$ to ensure that $A$ defers during the transmission of the ACK.

**Network Allocation Vector (NAV):** Since a node may overhear many different and potentially overlapping reservation requests, it needs a means by which it can efficiently manage them. This is the purpose for the maintenance of a structure called the *Network Allocation Vector (NAV).* The NAV is a data structure that stores the aggregate duration of time that the medium is presumed to be "busy," based on the reservation requests that have been received. Maintenance of the NAV is straightforward. Nodes that overhear a reservation request are free to update their NAVs without regard to any further communication, such as if the reservation was actually granted by the receiver.

# 3. Data Rate Adaptation

Rate adaptation is the process of dynamically switching data rates to match the channel conditions, with the goal of selecting the rate that will give the optimum throughput for the given channel conditions. A proven technique for wireline modems [37], rate adaptation has received attention as a technique also for use in wireless systems (e.g., [97], [10], [48], [112], [4]). In fact, the Lucent WaveLAN II and Cisco/Aironet 350 each contain proprietary rate adaptation mechanisms.

There are two aspects to rate adaptation: channel quality estimation and rate selection. Channel quality estimation involves measuring the time-varying state of the wireless channel for the purpose of generating predictions of future quality. Issues include: which metrics should be used as indicators of channel quality (e.g., SNR, signal strength, symbol error rate, BER), which predictors should be used, whether predictions should be short-term or long-term, etc. [10], [46]. Rate selection involves using the channel quality predictions to select an appropriate rate. Techniques vary, but a common technique is threshold selection, where the value of an indicator is compared against a list of threshold values representing boundaries between the data rates [108], [10].

Among the factors that influence the effectiveness of rate adaptation, of particular importance is the accuracy of the channel quality estimates. Clearly, inaccurate estimates will result in poor rate selection. Thus, it is advantageous to utilize the best information available when generating an estimate, and since it is the channel quality seen by the receiver that determines whether a packet can be received, the best information is available on the receiver. It is equally important that once estimates are generated they be used before they become outdated. Thus, it is also advantageous to minimize the delay between the time of the estimate and the time the packet is transmitted.

Few rate adaptation techniques have been designed for wireless LANs. Among those that are available, the following are most relevant. In [99], the authors present a protocol for a dual-channel slotted-aloha MAC, in which a separate control channel is used by the receiver to transmit explicit feedback to the sender, which the sender uses to adapt the rate. In [72], the authors present the “Auto Rate Fallback (ARF)” protocol for IEEE 802.11, used in Lucent’s WaveLAN II devices. In ARF, the sender selects the best rate based on information from previous data packet transmissions, incrementally increasing or decreasing the rate after a number of consecutive successes or losses, respectively. Finally, in [43], the authors propose protocol for point-to-point links, that selects transmission settings (e.g., code rate and power level) based on cached per-link information. The settings are stored in separate *transmit* and *receive* tables, which are then used by the sender and receiver to transmit and receive data packets on the link. The tables are maintained, in part, by exchanging settings in control packets, such as those in the RTS/CTS protocol. The RTS is also used by the sender to tell the receiver what settings the sender will use to transmit the data packet (which it gets from its transmit table). The receiver uses the settings in the RTS to update its receive table. If the receiver chooses, it may use the CTS to suggest a different power level, but, otherwise, no other changes to the transmit settings are made during the RTS/CTS exchange. Instead, changes to the sender’s transmit table are made by information in acknowledgment (ACK) or negative acknowledgment (NACK) packets sent at the end of the data packet transmission. These changes are then used for subsequent packet transmissions. Note that, in all three protocols ([99], [72], and [43]), rate selection is performed by the sender, and, in [72], channel quality estimation is also performed by the sender. Also note that only [72] is based on a widely used wireless local area networking standard (IEEE 802.11).

Much of the other work on rate adaptation in wireless networks has assumed a cellular network (e.g., mobile nodes communicating to a base station over a TDMA link) [10], [97], [108]. We have observed that many of these techniques have the following characteristics: (a) channel quality estimation is performed by the receiver and periodically fed to the sender either on the same channel or on a separate subchannel; (b) rate selection is performed by the sender using the feedback provided by the receiver; and (c) they often reside at the physical layer, adapting rates on a symbol-by-symbol or slot-by-slot basis, transparent to upper layers.

Although it may appear that such approaches are also applicable to ad hoc networks, several important differences exist. For instance, since single channel, half-duplex radios are generally used, simultaneous subchannel feedback is not possible. Furthermore, MAC protocols for ad hoc networks rely on accurate estimates of packet transmission times for efficient operation. Thus, if transparent physical layer rate adaptation were to be employed, it would be difficult for the MAC layer to estimate in advance how long a packet transmission will take, resulting in a potential decrease in efficiency.

## C. Routing

One of the most difficult challenges in ad hoc networks has been to develop routing protocols that can route packets through a network topology that changes frequently and suddenly. Conventional routing protocols, like RIP [84] and OSPF [86], were developed for wired networks and, as such, perform poorly in networks with a dynamic topology. In this section, we survey routing protocols for ad hoc networks, focusing on the Dynamic Source Routing (DSR) protocol that is often referred to in later chapters.

## 1. Overview

A number of routing protocols for ad hoc networks have been developed, such as the DSR protocol [19], the Ad Hoc On-Demand Distance Vector (AODV) protocol [92], the Optimized Link State Routing (OLSR) protocol [23], and the Destination-Sequenced Distance-Vector (DSDV) protocol [93]. Many of these are based on, or developed as a part of, work produced by early DARPA packet-radio programs such as PRNet [71], and SURAN [15, 80]. Most are based on the same shortest-path routing used in wired networks: prioritize routes according to their length (in hops) and give higher priority to shorter routes. The reasons being that shorter routes generally have lower end-to-end delay, because fewer routers are traversed, and higher overall throughput, because traffic is localized.

A number of other protocols consider other routing metrics, such as SSA (Signal-Strength Adaptive) [33] which selects routes based on the aggregate strength of its links, ABR (Associativity-Based Routing) [107] which chooses routes that have links with longevity, and ETX (Expected Transmission Count) [27] which selects routes with the lowest predicted retransmission probability based on online statistical monitoring. Others consider the load on a route, such as STARA (System and Traffic Dependent Adaptive Routing Algorithm) [50], INORA (INSIGNIA [2] and TORA [88]) [32], DLAR (Dynamic Load-Aware Routing in Ad hoc Networks) [81], and LWR (Load aWare Routing) [115].

Only a few routing protocols, however, consider the impact of multirate links. The work in [103] considers link data rates, selecting routes that minimize end-to-end delay. However, their algorithm is impractical for ad hoc networks because it assumes that timely and accurate network-wide link state information is available at each node, including the status of every link's queuing delay, medium access delay, and near-neighbor reachability. The overhead of disseminating and maintaining such soft state that this would require was left out of their performance analysis. In [7], the authors propose the *media time metric* to account for differences in link data rates on a route. This metric is built on the assumption that all nodes on a route contend for the same global channel. In this instance, the route that minimizes the time on the channel maximizes utilization. However, this metric does not apply to the more general case where nodes contend only within their local area.

### 2. Dynamic Source Routing (DSR)

The DSR protocol was developed by researchers at CMU [70] and is one of the protocols being considered for standardization by the Internet Engineering Task Force (IETF). In DSR, a routing header is prefixed to each packet entering the network that specifies the complete sequence of nodes on which the packet should be forwarded. This route is obtained through *route discovery.* When a node has a packet to send for which it does not have a route, it initiates route discovery by broadcasting a *route request* packet. In a request packet is stored the sequence of nodes that the request visits as it is propagated through the network. If a node, say $x$, receives a request and has a route to the destination, then it sends a *route reply* packet to the requester with the new route formed by concatenating the node sequence in the request to the route stored by $x$. To limit how far a request is propagated, a time-to-live (TTL) field is attached to every request along with a unique request identifier. A node drops any request that it has either seen before or has lived beyond its TTL.

To reduce the number of route discoveries, each node maintains a cache of routes that it has learned. A node may learn of a route through route discovery, or through other means such as snooping routes in route replies and data packets, or eavesdropping on local broadcasts. This cache is updated through *route error* messages. A route error message is sent by a node when it discovers that a packet's source route is no longer valid.

The route discovery protocol, as implemented by CMU, has two phases: a local broadcast (a *ring-0* search) followed by a propagating search. The ring-0 search is initiated in the hope that a route can quickly be found in a neighbor's cache. If a route is not found within a small amount of time, a propagating search is attempted. If this fails, the protocol backs-off and tries again, eventually giving up if a route is not found. This procedure repeats until all of the packets queued for that particular destination are dropped from the queue, or a route is found.

A packet may be dropped from the queue if a route has not been found within a pre-specified amount of time (the "Send Buffer Timeout" interval), or if the queue is full and newly arriving packets force it out. Route discoveries for the same destination are limited by the back-off and retry procedure, which is initiated per destination (versus per packet). Thus, regardless of the number of packets that need a route to the same destination, only one route discovery procedure is initiated.

Once a route is found and a packet is sent, there is the possibility that the route becomes "stale" while the packet is in flight, because of node mobility (a route is "stale" if some links on the route are broken). In such an instance, DSR uses a mechanism called *packet salvaging* to re-route the packet. When a node $x$ detects that the next link in a packet's route is broken, it first sends a route error message to the node that generated the packet's route to prevent it from sending more packets on the broken route. Node $x$ then attempts to *salvage* the packet by checking its cache to see if it knows of another route to the packet's destination. If so, node $x$ inserts the new source route into the packet and forwards it on that route; if not, the packet is dropped.

## D. Transmission Control Protocol (TCP)

Most network applications require that the data they send be delivered reliably and in order. However, the underlying fabric of many networks, including the Internet, is "best effort" – meaning that there are no guarantees that data that is sent will be received. Instead, the task of sequencing, packetizing, and tracking each application's data to ensure that it is delivered correctly is the job of end-to-end reliable transport protocols. Among the many protocols that have been devised (e.g., Delta-t [111], NETBLT [29], and XTP [113]) the de facto standard and main Internet transport is *TCP* [94], [104].

TCP is a connection-oriented protocol that provides applications with an abstract view of the network that resembles a reliable "first-in, first-out (FIFO)" data pipe. The workhorse of the Internet, TCP is the foundation for important network applications such as web browsing, e-mail, and file transfer, and carries almost all of the data exchanged on the Internet. Statistics generated by the Cooperative Association of Internet Data Analysis (CAIDA) on data from the NASA Ames Internet Exchange (AIX) showed that TCP accounted for over 80% of all the packets transferred, and over 90% of all the data [85].

TCP's success is largely attributable to its proven congestion control algorithms. These algorithms are based on sound theoretical principles that are designed to reduce the probability that a network will become over-utilized to the point of congestive collapse. It was, in fact, a series of congestive collapses in the early years of the Internet (1986) that gave rise to TCP as it is today. The first version of TCP that had adequate congestion control was called *TCP-Tahoe*. Tahoe included the slow-start, congestion avoidance, and fast retransmit protocols devised by Jacobson [65] in response to the collapses. This was followed by *TCP-Reno*, which included Jacobson's fast recovery protocol [66]. The performance and reliability of Reno has made it the most widely deployed version of TCP to date. Modifications to the fast recovery protocol to reduce the number of spurious timeouts as proposed by Hoe [52] was later added, creating a variant called TCP-NewReno, which is being deployed today. More details on the evolution of TCP can be found in [39]. In this thesis, all references to TCP imply TCP-Reno unless otherwise noted.

## 1. Overview

In this section we present an overview of the internals of the TCP protocol. We start by first discussing how TCP uses sequence numbers to track and order the data that it transmits, and the flow control mechanisms it uses to limit the rate at which data is injected into the network. We then look at the algorithms TCP uses to detect and react to congestion in the network, followed by a look at how it detects and reacts to lost data packets.

### a. Sequencing and Flow Control

To provide an end-to-end reliable, in-order, byte stream, TCP must keep track of each byte of data that it receives, until that byte is known to have been delivered to its destination. Thus, every data byte that TCP is given is assigned a unique *sequence number* in the order that it was received. This number stays with the data byte for the duration of its time in the network. Groups of bytes are arranged into *segments* for transmission by the underlying IP protocol (up to a *maximum segment size (MSS)*). Prepended to each segment is a header identifying the bytes by the sequence number of the first byte and the length of the segment. The receiver uses this sequence number to sort the segments before delivering the bytes to the application. Segments that arrive in-order are acknowledged to the sender by a *cumulative acknowledgment* packet. This acknowledgment (ACK) carries the sequence number of the last in-order byte received plus one (or, the next byte the receiver needs). This implicitly notifies that sender that all bytes up to, but not including, this sequence number have been successfully received.

To prevent the transmission of too much data into the network, a sliding window based flow control algorithm is used. Here, the sender keeps track of the last byte that was acknowledged and the last byte in the sequence that was transmitted, ensuring that the difference never exceeds the size of the window. The size of the window is bounded above by how much the receiver is able to buffer, which is called the *advertised window*. However, a second window size is maintained by the sender based on its estimate of the amount of congestion in the network, called the *congestion window*, or *cwnd* for short. To ensure neither is exceeded, the sender never sends more than the minimum of both windows.

#### b. Congestion Avoidance and Control

Management of the congestion window is the responsibility of the congestion avoidance and control algorithms. The goal of these algorithms is to maintain a window that matches precisely what the network can hold. If the network were a pipe, this capacity would be the area of its cross-section times its length. In a network, this is equivalent to the minimum link bandwidth multiplied by the end-to-end propagation delay, or the *bandwidth-delay product* as it is often called. However, maintaining this window precisely is difficult to do as other flows enter and exit the network, creating variation in the *available capacity*, which, consequently, approaches randomness as the network size grows large [18]. Instead, TCP continuously probes the limits of available capacity, rapidly backing-off if it detects congestion, in hopes of achieving an average capacity that approaches the available capacity, or at least a fair share of that capacity.

The congestion control algorithms operate in phases throughout the lifetime of the connection. At startup, the congestion window is set to one MSS since the capacity of the network is unknown. At this point, the *slow start* algorithm is in control of the congestion window. The goal of slow start is to grow the congestion window at a rapid pace (exponentially) until congestion is detected. Thus, for every ACK received, the *cwnd* is increased by two MSS. This continues until either congestion is detected or a threshold is reached, called the *slow start threshold*, or *ssthresh* for short. Once the threshold is reached, the *congestion avoidance* algorithm takes control of the congestion window. The goal of the congestion avoidance algorithm is to probe the network more slowly by only increasing *cwnd* by one MSS for every window of packets delivered.

### c. Loss Detection and Recovery

In TCP, congestion in the network is only detected by the loss of a segment, presumably dropped by the congested router at the *bottleneck link* on the path. The detection of a lost segment can happen in one of two ways. When a later segment arrives at the receiver, it transmits a *duplicate acknowledgment*, or *DUPACK*, with the sequence number of the missing segment and buffers the out-of-order segment. Since there may not always be a segment in flight to trigger a DUPACK, the sender also uses a *retransmission timer*, which it resets once for each window of packets it transmits. A segment is presumed lost if the timer expires (a *timeout*). To select an appropriate value for the timer, TCP measures the *round-trip-time*, or *rtt*, for one segment every window. These measurements are then smoothed using the low-pass filter

$$\text{srtt} = \alpha\ \text{srtt} + (1-\alpha)\text{rtt}\ , \tag{2.7}$$

where *srtt* is the smoothed round-trip time, and $\alpha$ is a smoothing factor. The retransmission timeout *rto* is then set to $\text{rto} = \beta\ \ \text{rtt}$, with $\beta = 2$ (recommended) [104].

The response to the detection of a lost segment is always a retransmission of the segment (except in the extreme case that the connection is deemed lost). However, the way in which it is detected determines how TCP adjusts its congestion window. If a timeout occurs, *ssthresh* is set to one-half of *cwnd* and cwnd = MSS. This puts slow start back in control of the congestion window as if the connection was just established. The reasoning being that a timeout could be caused by severe congestion that resulted in the loss of an entire window of segments. Setting *ssthresh* to half of the congestion window, however, retains some information about the state of the network before the loss was detected, such that if it was a transient loss, the window can be more quickly rediscovered than starting anew.

Conversely, if the loss was signaled by DUPACKs, the *fast recovery* algorithm takes control and sets $\text{ssthresh} = \text{cwnd}/2$ and $\text{cwnd} = \text{ssthresh} + 3 \times MSS$, in that order. Inflating *cwnd* by three MSS accounts for the fact that the segments that triggered the DUPACKs arrived and are buffered by the receiver, so there is room to transmit additional segments. It also provides additional segments to trigger acknowledgments should they be needed. Any further DUPACKs inflate *cwnd* by the segment size. Arrival of an ACK results in $cwnd = ssthresh$, and the start of congestion avoidance. The idea is to recover the lost segments as quickly as possible while remaining within control of the congestion avoidance algorithm.

## 2. Wireless Networks

In the previous section, it was noted that TCP uses packet loss as an indicator of congestion. In wired networks, this works well because the transmission channel is so reliable and the topology so stable that the only statistically relevant cause of loss is congestion. However, in wireless networks packets can be lost by a number of different causes, such as transmission errors, link failures, and topology changes, for which TCP's response of reducing its transmission rate is inappropriate. The result is less than ideal performance. In this section we survey proposed approaches to this problem.

Most effort to date has been focused on improving performance of TCP over "last-hop" wireless networks (e.g., SNOOP [13], HACK [14], EBSN [9], ETEN [78]). However, these approaches are not appropriate for ad hoc networks because they assume a base station is present to manage recovery efforts on the wireless link.

However, there has been some work for ad hoc networks as well; most focusing on mitigating the impact of link failures, which, as we will show, is one of the more serious problems. The dynamic nature of ad hoc networks creates a high degree of link failure and long routing delay. During these long delays, TCP can enter a cycle of consecutive timeouts resulting in very poor performance. This is documented thoroughly in our work [53], [57], and subsequently in [114]. The predominant approach to this problem has been to use some form of explicit link failure notification message (e.g., TCP-F [21], TCP/RCWE [49], TCP-BuS [75], ENIC [105]). For instance, ATCP [82] relies on the delivery of a "Destination Unreachable" ICMP to signal link failure, whereas others piggyback the message on routing packets (e.g., TCP-BuS [75]). The response to this message is generally the same: freeze TCP's congestion control mechanisms until the route has been repaired. Areas where the various approaches significantly differ are in how the notifications are delivered, how to discover that the route has been repaired, and to what state TCP should be restored. Unfortunately, there is no clear “winner” in these areas. However, what is clear that, like explicit error notification, there is a problem identifying and forwarding explicit messages to TCP senders because of the encrypted TCP headers.

One alternative to explicit messaging has been proposed, called TCP-DOOR [110]. TCP-DOOR is unique in that it uses the arrival of out-of-order packets at the end-points to signify that a route change has occurred, to which it responds by “rolling-back” TCP to a previous state if it is determined that TCP had a negative reaction to the route change. While such an approach is attractive, it is not clear how well it will work in realistic ad hoc networks, since optimal congestion windows are so small as to reduce the probability of a route change causing out-of-order packets. However, this is one approach that we will study more thoroughly.

Finally, it should be noted that the problem of long routing delays could, in many instances, be greatly reduced by enhancing the speed and accuracy of the routing protocols. The Atra [6] framework is one such attempt. Atra is a set of routing optimizations for DSR that are designed to eliminate long route repair delays. Three optimizations are proposed: Symmetric Route Pinning, Proactive Route Errors, and Route Failure Prediction. Symmetric Route Pinning forces forward and reverse routes to be symmetric end-to-end, reducing the probability of route failure by minimizing the number of links (points of failure) on the route. Proactive Route Errors is another DSR-specific optimization which forces the immediate push of route error notifications to all nodes that have sent packets across a link over some recent interval of time, rather than waiting to send notifications when packets are received for the downed link. Route Failure Prediction uses an approach similar to Signal Stability-Based Adaptive routing [33] in which signal strength is used as an indicator of impending route failure, subsequently triggering preemptive route discovery. A similar approach was also used in [47]. It should be noted that long route repair delays are especially a problem in reactive protocols, like DSR, where route caching is needed to reduce route discovery overhead. In such protocols, route caching has the tendency to cause the attempted use of stale (invalid) routes during route repair [53], resulting in longer than optimal routing delays.

# CHAPTER III

# METHODOLOGY

In this section we present the simulation environment and methodology used for the performance analysis presented in this dissertation.

## A. Simulation Environment

The simulations in this dissertation were generated using an extended version of *ns* that was developed by the Monarch Project at Carnegie Mellon University. The extensions added to *ns* by the Monarch group provide the mechanisms necessary for simulating a mobile ad hoc network, including implementations of the 802.11 wireless MAC protocol, the BSD ARP protocol, several mobile ad hoc routing protocols, and a radio propagation model. It also provides mechanisms to simulate node mobility using precomputed files that specify node motion and peer to peer reachability for the duration of the simulation. Following is a brief overview of the input parameters and files that are required by the CMU extensions. The reader is referred to [24] for a more detailed description.

The CMU extensions to the *ns* simulator include a group of OTcl scripts that are processed as inputs to the simulator as well as a collection of programs that generate new input scripts. The main input file is an OTcl script that processes command-line arguments and initializes and configures the simulation environment. This is the `cmu/scripts/run.tcl`[1] script. In the process of configuring the simulation, this script calls several other OTcl scripts, some of which are named in command-line arguments. One of these is the routing protocol configuration script, which configures

[1] All paths in this section are relative to the `ns-src/` directory in the 2.1b3 distribution of the CMU extensions.

```
set tcp_(0) [$ns_ create-connection  TCP/Reno $node_(1) TCPSink $node_(2) 0]
$tcp_(0) set window_ 32
set ftp_(0) [$tcp_(0) attach-source FTP]
$ns_ at 0 "$ftp_(0) start"
```

Fig. 4. *An example connection pattern script that creates a single TCP-Reno connection driven by an FTP data source.*

the simulated mobile nodes as routers that implement the named protocol (e.g., the DSR configuration script is `cmu/dsr/dsr.tcl`). Two other OTcl scripts, the connection pattern script and the mobility pattern script, designate the workload and network topology and dynamics for a single run of the simulation.

The *connection pattern* script designates the configuration and behavior of data connections in the network scenario to be simulated. For instance, it specifies when a data connection should be created and destroyed, the endpoints of the data connection (e.g., Node 1 to Node 2), when data flow across the connection should start and stop, the type of application data source that will send the data (e.g., FTP), and the transport protocol that the data source will use (e.g., TCP Reno). It may also specify other characteristics of the connection, such as the amount of data to send, the maximum TCP window size, and whether tracing should be enabled on variables related to the connection. An example script is shown in Fig. 4. This script sets up a single TCP-Reno connection from Node 1 to Node 2 and then sets the maximum TCP window for that connection to 32 packets. (Note that `tcp_(0)` is an example of a *mirrored object* and `window_` is an example of a *bound* variable, both of which were described earlier.) The script then attaches an FTP data source to Node 1 and schedules the data source to start sending at time 0. A program to generate connection pattern scripts is provided in the CMU distribution (`cmu/scripts/cbrgen.tcl`).

The *mobility pattern* script (also called a *scenario file* in [24]) designates the mo-

```
$node_(1) set X_ 93.909
$node_(1) set Y_ 57.308
$node_(1) set Z_ 0.000
$node_(2) set X_ 325.905
$node_(2) set Y_ 122.463
$node_(2) set Z_ 0.000
$ns_ at 0.000 "$node_(1) setdest 80.985 310.405 1.962"
$ns_ at 0.000 "$node_(2) setdest 174.895 367.713 2.101"
$god_ set-dist 1 2 1
$ns_ at 129.151 "$node_(1) setdest 297.359 448.699 1.853"
$ns_ at 137.049 "$node_(2) setdest 94.547 123.594 1.959"
$ns_ at 230.673 "$god_ set-dist 1 2 16777215"
$ns_ at 267.718 "$node_(1) setdest 217.640 420.953 2.000"
$ns_ at 268.237 "$node_(2) setdest 151.138 15.501 1.901"
```

Fig. 5. *An example mobility pattern script for two nodes moving in a 500m x 500m area at a mean speed of 2 m/s for 300 seconds.*

tion of the nodes in the network and the changes in paths between the nodes over time. Associated with each node $x$ are a set of coordinates $(x_l, y_l, z_l)$ that designate $x$'s location in the area of motion, a set of coordinates $(x_d, y_d, z_d)$ that designate $x$'s next destination, and $x$'s current speed. The initial coordinates of the nodes and their patterns of destinations and speeds for the duration of the simulation are all designated in the mobility pattern script. An example script is shown in Fig. 5, which contains commands designating the motion of two nodes moving at a mean speed of 2 m/s for 300 seconds in a square 500m x 500m area. The first sequence of commands (i.e., `$node_(1) set ...`) designate the initial placement of the two nodes at the start of the simulation. The next two commands (i.e., `$ns_ at 0.000 "$node_(1) setdest ...`) designate the $(x_d, y_d)$ coordinates ($z_d = 0$ in this version) for each node's next destination, as well as the speed at which they will travel. (The "`at 0.000`" portion of the entry designates when (in simulated time) the command will be executed.) The `setdest` commands that appear later in the script designate new destinations and speeds as dictated by the mobility model that was used to create the script. In this instance, the nodes are moving in a random-walk. Also included in the script are commands that indicate changes in the optimal path lengths (in hops) between nodes in the network (i.e., `...$god_ set-dist...`). These commands are used to track the efficiency of the routing protocol. Each "`set-dist <i> <j> <d>`" command designates the time at which the optimal (shortest) path length between Node $i$ and Node $j$ becomes $d$ hops. A value of $d = 16777215$ indicates an infinite path length, meaning no path exists between the nodes (the network is *partitioned*). A program to generate mobility pattern scripts is also provided in the CMU distribution (`cmu/setdest/setdest`). By default, this program assumes a fixed maximum transmission of 250m (the length of one hop), which is consistent with the radio that was simulated (914Mhz WaveLan).

## B. Simulation Enhancements

To the standard ns-2, we added more detailed MAC and physical layer models, including the addition of the modulation schemes and rate adaptation mechanisms that are the focus of this study, as well as the addition of a Rayleigh fading simulator for studying the impact of multipath fading. The Rayleigh fading simulator we used is based on the well known Jakes [68] simulator, which generates a continuous time-varying Rayleigh fading envelope. Additionally, we enhanced the realism of the existing network interfaces using the Intersil Prism II chipset and accompanying reference interface designs as our model. The Prism II chipset is an IEEE 802.11, direct-sequence spread-spectrum (DSSS) radio that is used in many commercially available network interfaces, including the Aironet PC4800 [3] (now known as the Cisco 350). Most of our network interface parameters were drawn directly from the Intersil documentation, including power constraints, receiver noise factors, reference antenna gains, and sensitivity thresholds. Our interest in these experiments was only to observe how the individual rate adaptation protocols reacted to the changing channel conditions, and not to evaluate the exact performance of currently available network devices. We differed slightly from the reference design of the Prism II chipsets and did not model the CCK modulation schemes, instead choosing to use the more widely known and well documented M-ary QAM modulation schemes [100]. However, similar results can be expected for CCK, MOK, and and other modulation schemes. Apart from the aforementioned changes, the nodes in our simulations were otherwise configured similar to those in [19].

Our error model was based on the detailed simulation of a Rayleigh fading channel, using the well known Jakes method. In this section, we describe, in detail, how we used this method to model packet errors in our simulations. Jakes method is a technique for simulating a signal with Rayleigh fading characteristics. The technique is based on the simulation of a finite number of oscillators with Doppler shifted frequencies, whose outputs are combined to produce the simulated Rayleigh fading signal. The resultant signal $\alpha(t) = x_c(t) + jx_s(t)$, where $x_c$ and $x_s$ are the signal's in-phase (real) and quadrature (imaginary) components, is computed as follows

$$x_c(t) = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} \cos\beta_n \cos(\omega_n t + k\beta_n) \tag{3.1}$$

$$x_s(t) = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} \sin\beta_n \cos(\omega_n t + k\beta_n) \tag{3.2}$$

where $N$ is the number of oscillators, $k = 1$, and

$$\omega_n = \frac{2\pi v}{\lambda} \cos\left(\frac{\pi n}{2N+1}\right) \tag{3.3}$$

$$\beta_n = \frac{\pi n}{N} \tag{3.4}$$

In our simulations, $N = 8$. The instantaneous gain of the channel is then the magnitude of the signal

$$|\alpha(t)| = \sqrt{x_c^2(t) + x_s^2(t)} \tag{3.5}$$

Given the gain, we computed whether a packet was received with errors using well known methods for calculating the pre-gain signal to noise ratio ($SNR$) and resultant bit error rate for the modulation schemes that were used. To compute the value of the pre-gain received signal, we used the log-distance path loss model with a path loss exponent of 2. Noise was modeled as a combination of the noise floor of the interface and the aggregate energy of neighboring transmissions that were to weak to cause a collision. For our simulations, we used a noise figure provided by Intersil for their Prism I chipset.

Finally, the received bit error rates were computed using the following bit error rate equations for the different modulation scheme that were used. For BPSK and QPSK [100]

$$P_e(t) = Q\left(\sqrt{\frac{2|\alpha(t)|^2 E_b}{N_o}}\right) \tag{3.6}$$

and for M-ary QAM

$$P_e(t) \approx 4\left(1 - \frac{1}{\sqrt{M}}\right) Q\left(\sqrt{\frac{3|\alpha(t)|^2 log_2(M) E_b}{(M-1)N_o}}\right) \tag{3.7}$$

where $E_b/N_o$ is the bit-energy-to-noise ratio of the received signal and $|\alpha(t)|$ is the instantaneous gain of the Rayleigh channel (from [68]).

Since portions of a packet may be transmitted at different modulation schemes, the probability that a packet was in error was based on separate calculations for each portion. Furthermore, since gain and noise may vary with time, we also accounted for those in our calculations by the following. For the gain, we used an approximation for the coherence time. For the noise, we accounted for the changing conditions by tracking the beginning and ending times of each of the neighboring transmissions and adjusting $SNR$ appropriately.

# CHAPTER IV

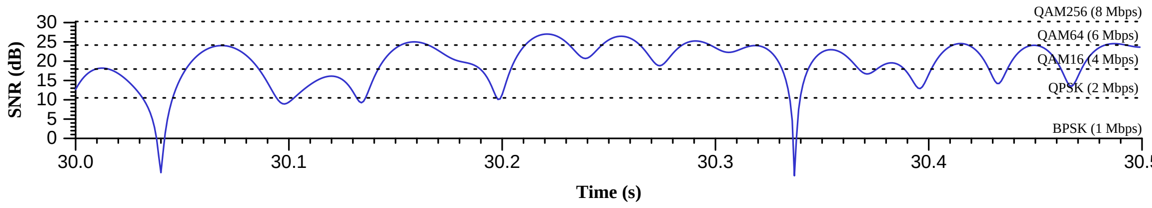
# THE RECEIVER-BASED AUTORATE PROTOCOL*

In this chapter we propose a new approach to rate adaptation for wireless local area networks. In our approach, the rate selection and channel quality estimation are located on the *receiver*, and rate selection is performed on a per-packet basis *during* the RTS/CTS exchange, just prior to packet transmission. The motivation for our approach is based on the following observations:

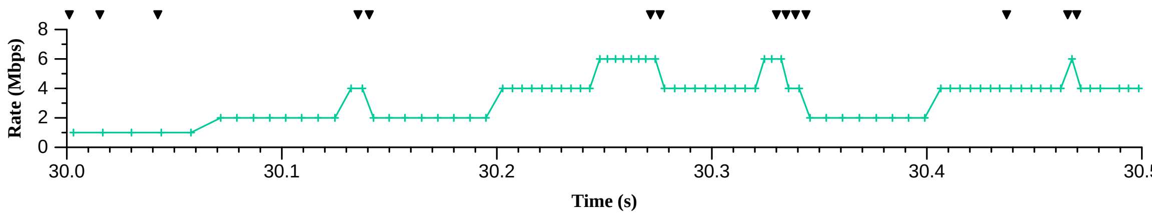
- Rate selection can be improved by providing more timely and more complete channel quality information.
- Channel quality information is best acquired at the receiver.
- Transmitting channel quality information to the sender can be costly, both in terms of the resources consumed in transmitting the quantity of information needed as well as the potential loss in timeliness of the information due to transmission delays.

To emphasize the need for better rate adaptation mechanisms, consider Fig. 6, which illustrates the behavior of the ARF protocol. Shown is the performance of ARF for a single CBR connection between two nodes in a Rayleigh fading channel. The sender is fixed and the receiver is moving at a speed of 2 m/s away from the sender. Fig. 6(b) shows the time at which packets were transmitted, and the modulation rate chosen by ARF for each packet. The tick marks along the top show the time at which packets were dropped by the receiver due to errors. Fig. 6(a) shows the

*Part of this chapter is reprinted from "A rate-adaptive MAC protocol for multi-hop wireless networks" by G. D. Holland, N. H. Vaidya, and P. Bahl, 2001, in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, pp. 236–250. © 2001 ACM, Inc.



(a) *SNR at the receiver for the packets shown in (b).*



(b) *Time and rate of packet transmission.*

Fig. 6. *Performance of ARF for a single CBR connection between two nodes in a Rayleigh fading channel. The sender is fixed and the receiver is moving at a speed of 2 m/s away from the sender.*

SNR at the receiver for the packets in (b). Also shown are thresholds representing the SNR values above which the next higher modulation rate has a theoretical mean $BER \leq 10^{-6}$. At the start, both nodes were at the same location, so the leftmost edges represent the point in time at which the two nodes were 60m apart. Studying these results, it is clear that ARF is slow to adapt to changes in SNR, as indicated by the relative dissimilarity between the upper and lower graphs. In particular, consider ARF's failure to rapidly increase the data rate after the deep fades in (a) at the 30.2s and 30.35s marks, and the attempt it makes to increase the rate in the middle of a fade at the 30.13s mark.

## A. The Receiver-Based Autorate (RBAR) Protocol

The core idea of the proposed Receiver-Based Autorate (RBAR) protocol is to allow the *receiver* to select the appropriate rate for the data packet *during* the RTS/CTS packet exchange [58]. Advantages to this approach include:

- Both channel quality estimation and rate selection mechanisms are now on the receiver. This allows the channel quality estimation mechanism to directly access all of the information made available to it by the receiving hardware (such as the number of multipath components, the symbol error rate, the received signal strength, etc.), for more accurate rate selection.
- Since the rate selection is done *during* the RTS/CTS exchange, the channel quality estimates are nearer to the actual transmission time of the data packet than in existing sender-based approaches, such as the protocol in [72] which attempts to predict channel conditions based on conditions experienced during previous data packet transmissions.
- It can be implemented into IEEE 802.11 with minor changes, as we will show in a later section.

In the remainder of this section, we present the RBAR protocol in more detail Note that although our discussion is in the context of the RTS/CTS protocol in the DCF of the 802.11 standard, the concepts are equally applicable to other RTS/CTS based protocols such as those mentioned in Chapter II.

In RBAR, instead of carrying the duration of the reservation, the packets carry the modulation rate and size of the data packet. This modification serves the dual purpose of providing a mechanism by which the receiver can communicate the chosen rate to the sender, while still providing neighboring nodes with enough information

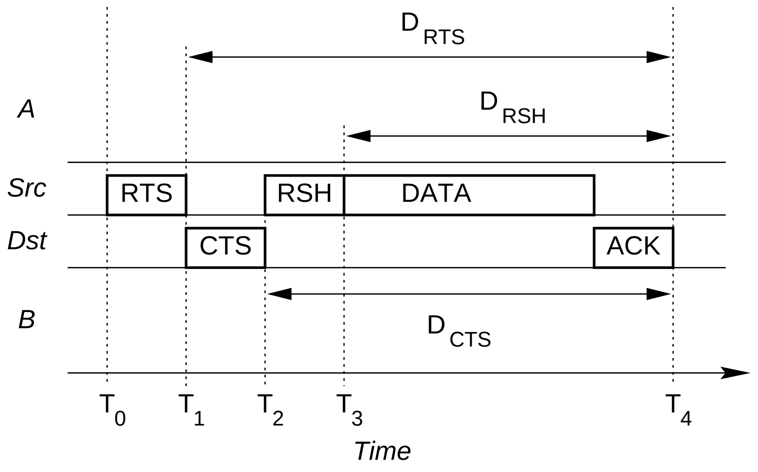
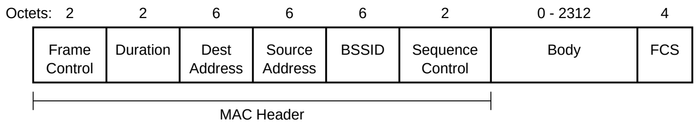


Fig. 7. *Timeline showing changes to the DCF protocol as needed for the proposed Receiver-Based Autorate Protocol.*

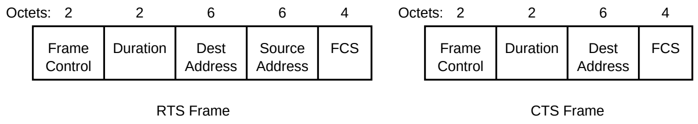
to calculate the duration of the requested reservation. The protocol is as follows.

Referring to Fig. 7, the sender *Src* chooses a data rate based on some heuristic (such as the most recent rate that was successful for transmission to the destination *Dst*), and then stores the rate and the size of the data packet into the RTS. Node $A$, overhearing the RTS, calculates the duration of the requested reservation $D_{RTS}$ using the rate and packet size carried in the RTS. This is possible because all of the information required to calculate $D_{RTS}$ is known to $A$. $A$ then updates its NAV to reflect the reservation. While receiving the RTS, the receiver *Dst* uses information available to it about the channel conditions to generate an estimate of the conditions for the impending data packet transmission. *Dst* then selects the appropriate rate based on that estimate, and transmits it and the packet size in the CTS back to the sender. Node $B$, overhearing the CTS, calculates the duration of the reservation $D_{CTS}$ similar to the procedure used by $A$, and then updates ts NAV to reflect the reservation. Finally, *Src* responds to the receipt of the CTS by transmitting the data packet at the rate chosen by *Dst*.

In the instance that the rates chosen by the sender and receiver are different, then the reservation $D_{RTS}$ calculated by $A$ will no longer be valid. Thus, we refer to $D_{RTS}$ as a *tentative* reservation. A tentative reservation serves only to inform neighboring nodes that a reservation has been requested but that the duration of the *final* reservation may differ. Any node that receives a tentative reservation is required to treat it the same as a final reservation with regard to later transmission requests; that is, if a node overhears a tentative reservation it must update its NAV so that any later requests it receives that would conflict with the tentative reservation must be denied. Thus, a tentative reservation effectively serves as a placeholder until either a new reservation is received or the tentative reservation is confirmed as the final reservation. Final reservations are confirmed by the presence or absence of a special subheader, called the *Reservation SubHeader* (RSH), in the MAC header of the data packet. The reservation subheader consists of a subset of the header fields that are already present in the 802.11 data packet frame, plus a check sequence that serves to protect the subheader. The fields in the reservation subheader consist of only those fields needed to update the NAV, and essentially amount to the same fields present in an RTS. Furthermore, the fields (minus the check sequence) still retain the same functionality that they have in a standard 802.11 header. The reservation subheader is used as follows. Referring again to Fig. 7, in the instance that the tentative reservation $D_{RTS}$ is incorrect, $Src$ will send the data packet with the special MAC header containing the RSH subheader. $A$, overhearing the RSH, will immediately calculate the final reservation $D_{RSH}$, and then update its NAV to account for the difference between $D_{RTS}$ and $D_{RSH}$. Note that, for $A$ to update its NAV correctly, it must know what contribution $D_{RTS}$ has made to its NAV. One way this can be done, is to maintain a list of the end times of each tentative reservation, indexed according to the $< sender, receiver >$ pair. Thus, when an update is required, a node can use



(a) *Data packet frame.*



(b) *RTS/CTS control packet frames.*

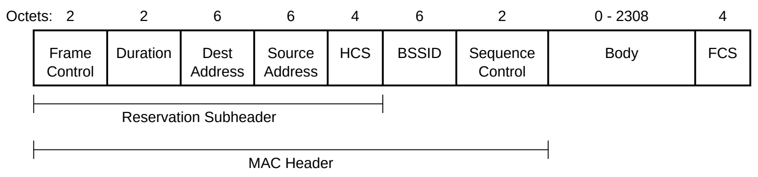
Fig. 8. *MAC frame formats used in IEEE 802.11 for ad-hoc networks (IBSS).*

the list to determine if the difference in the reservations will require a change in the NAV.

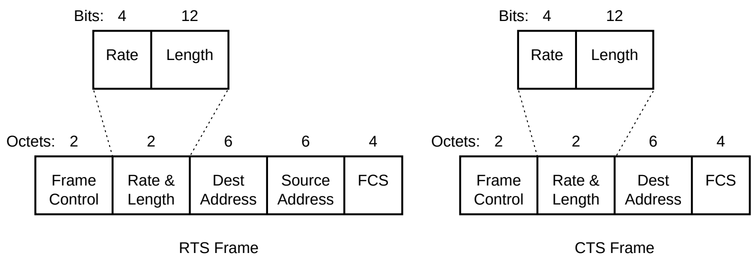
## B. Incorporation of RBAR into 802.11

In this section we describe how RBAR may be incorporated into 802.11. We start by presenting background information on the formats of the relevant 802.11 frames, and then describe in detail how these frames can be modified to accommodate RBAR.

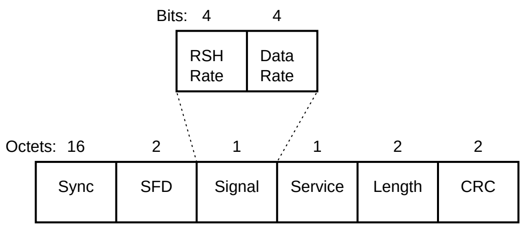
The IEEE 802.11 frame formats are shown in Fig. 8. Fig. 8(a) shows the format of the MAC frame used for sending unicast data packets in an ad-hoc network (IBSS). The *frame control* field carries frame identification information, such as the type of frame (e.g., management, control, or data), as well as protocol version information and control flags; the *duration* field contains the time remaining (in $\mu$s) until the end of the packet transfer (e.g., $D_{DATA}$ in Fig. 3); the *BSSID* is the unique network identifier; *sequence control* is a sequence number used to detect duplicate frames; and *FCS* is the frame check sequence. Fig. 8(b) shows the format of the RTS and



(a) *Data packet frame.*



(b) *RTS/CTS control packet frames.*



(c) *Physical layer (PLCP) header.*

Fig. 9. *MAC and physical layer frame formats used in the RBAR protocol.*

CTS control frames. The fields they share in common with the data frame serve the same purpose, except the duration fields contain the $D_{RTS}$ and $D_{CTS}$ values shown in Fig. 3. For a more complete description, the reader is referred to [60].

Modifications to the standard 802.11 frames for RBAR are minimal, and are illustrated in Fig. 9. A description of each modification is given next, followed by the design rationale.

1. A new MAC data frame is introduced, shown in Fig. 9(a), in which the standard

802.11 data frame has been changed to include a 32-bit check sequence positioned immediately after the source address field. The check sequence is used to protect the *reservation subheader*, which consists of the *frame control*, *duration*, *destination address*, *source address* and *address 2* fields of the header. The new frame is distinguished by other MAC frames by a unique type/subtype code in the frame control field (see [60] for more information on frame type codes).

2. The RTS and CTS control frames, shown in Fig. 9(b), have been changed to encode a 4 bit *rate* subfield and a 12 bit *length* subfield, in place of the 16 bit *duration* field in the standard IEEE 802.11 frames. The *rate* subfield uses an encoding similar to the *rate* field in the PLCP header for the 802.11a supplement standard [62], and the *length* subfield gives the size of the data packet in octets.

3. The signal field in the physical layer header (PLCP), shown in Fig. 9(c), has been divided into two 4 bit *rate* subfields, which use the similar rate encodings as those in 802.11a [62]. The first subfield, if non-empty, indicates the rate at which the reservation subheader will be transmitted, and the second subfield indicates the rate at which the remainder of the packet will be transmitted.

The rationale for the modifications shown in Figs. 9(a) and 9(b) was discussed at length in the previous section. Following, is a discussion of the modifications to the physical layer (PLCP) header shown in Fig. 9(c).

In standard 802.11, the PLCP header contains an 8 bit *signal* field that encodes the rate at which the payload of the physical frame (the MAC packet) should be transmitted. These fields are used as follows. When the physical layer has a packet to transmit, it first transmits the PLCP header at a fixed rate that is supported by all nodes (1Mbps). It then switches to the rate encoded in the *signal* field for the transmission of the payload. After verifying that the PLCP header is correct, using the *CRC*, the receiving physical layer switches to the rate given in the *signal* field to receive the packet payload. The end of the transmission is determined by the receiver from the *length* field, which stores the duration of the transmission (in $\mu$s).

In RBAR, the physical layer may be required to switch rates twice during transmission of the payload: once for the reservation subheader, and again for the remainder of the payload. To enable the use of an additional rate for the reservation subheader, our protocol requires that two rate changes occur during transmission of the data packet. Thus, instead of a single 8 bit *signal* field, we subdivide it into two 4 bit subfields, as shown in Fig. 9(c), where the first rate is used to send the reservation subheader, and the second for the remainder of the data packet. Thus, the PLCP transmission protocol is modified as follows. When the MAC passes a packet down to the physical layer, it specifies two rates: one for the subheader and one for the remainder of the packet. The physical layer then encodes the rates into the appropriate *signal* subfields shown in Fig. 9 and then transmits the packet. The receiving physical layer, after verifying that the PLCP header has been received correctly, will then switch to the first RSH rate for receipt of the subheader, and then to the data rate for the remainder of the packet. Note that, as specified in the IEEE 802.11 standard, as each byte is received, it is immediately available to the MAC. Thus, nodes that rely on the RSH to update their reservations, will be free to do so as soon as the RSH has been received.

### 1. Autorate Fallback Algorithm (ARF) Implementation

As a basis for comparison, we implemented Lucent's Autorate Fallback (ARF) protocol into the simulator. ARF is the rate adaptation scheme used in Lucent's 802.11 WaveLAN II networking devices [72]. The protocol, as specified in [72], is summarized as follows. If ACKs for two consecutive data packets are not received by the sender,

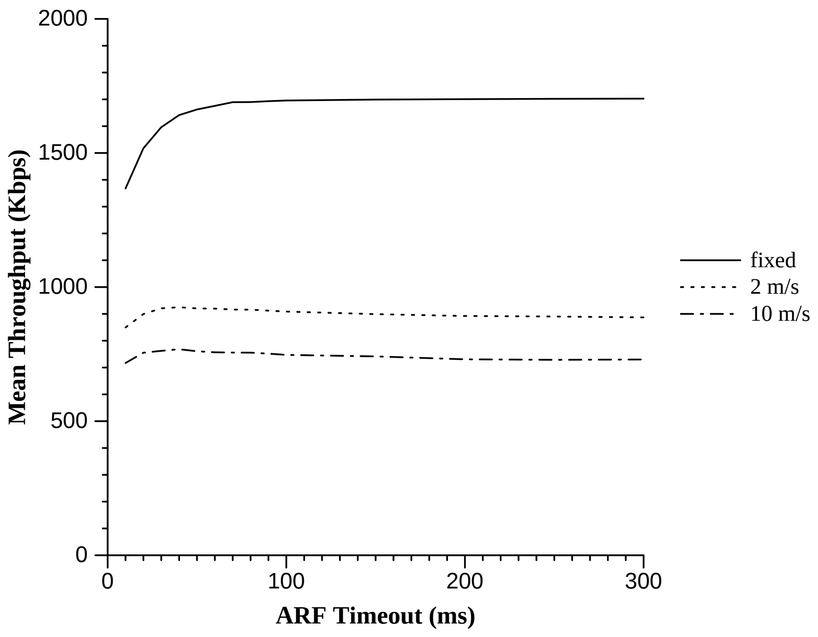


Fig. 10. *Comparison of the average throughput of ARF for various values of the timer it uses to indicate when it should attempt to increase the data rate in lieu of its usual indicator: the receipt of 10 consecutive ACKs. The curves shown are the average throughputs measured across a single CBR connection in a Rayleigh fading channel between two nodes oscillating near to far at different mean speeds. The* fixed *curve is the mean throughput between the nodes spaced at various intervals over the range of distance traveled by the oscillating nodes in the mobility curves.*

then the sender drops the transmission rate to the next lower data rate and starts a timer. If ten consecutive ACKs are received, then the transmission rate is raised to the next higher data rate and the timer is canceled. However, if the timer expires, the transmission rate is raised as before, but with the condition that if an ACK is not received for the very next packet, then the rate is lowered again and the timer is restarted. In our implementation we attempted to adhere as closely as possible to the description given in [72]. However, values for the timeout were unspecified. Therefore, prior to initiating our study, we experimented with several timeout values to determine a reasonable value for our simulations. The results of these experiments are shown in Fig. 10, which shows the average throughput as a function of the timeout value for several different mean node speeds. From these results it appears that ARF is relatively insensitive to the choice of timeout, for the given scenarios. However, there is a clear threshold region in the 40ms-60ms range, depending on the degree of mobility, beyond which there is little performance change but below which there is a noticeable drop. The drop can be attributed to the greater frequency at which packets are lost due to rate increases triggered by timeouts during times in which the channel conditions are poor. For the experiments with mobility, the peak in performance is in the 40ms range, whereas for the experiment without mobility, the performance rises sharply until the 60ms range and then levels off. The slight differences in peak values between the mobility experiments most likely represents those regions in which the timeouts are frequent enough to respond well to the variations in the Rayleigh channel, but not too frequent that the failed packet attempts significantly impact performance. Based on these results, we chose a value of 60ms for our simulations, which appears to be a reasonable compromise for the fixed and mobile simulations that we have used in our performance analysis.

### 2. Receiver-Based AutoRate Protocol (RBAR) Implementation

So far in our discussions of RBAR we have deliberately neglected to specify the channel quality estimation and rate selection protocols. This is because there are already a number of existing protocols in the literature (e.g., [10], [48], [108]), any of which may be used in RBAR. However, for our performance analysis we chose the following.

For the channel quality estimation and prediction algorithm, we used a sample of the instantaneous received signal strength at the end of the RTS reception. In practice, of course, much more accurate techniques could be used, such as those in

[10], [96], and [46].

For the rate selection algorithm, we used a simple threshold based technique. Threshold based techniques have been widely studied (e.g., [108], [10], [48]). In a threshold scheme, the rate is chosen by comparing the channel quality estimate against a series of thresholds representing the desired performance bounds of the available modulation schemes. The modulation scheme with the highest data rate that satisfies the performance objective for the channel quality estimate, is the chosen rate. The protocol we used was the following. Suppose we wish to select the modulation scheme that has the highest data rate among those with bit error rates $\leq$ 1E-5 for the estimated SNR of the next packet. The protocol would then choose the modulation scheme as follows. Let $M_1, \ldots, M_N$ represent the set of modulation schemes in increasing order of their data rate, and $\theta_i, \ldots, \theta_N$ represent the SNR thresholds (dB) at which $BER(M_i) =$ 1E-5. Choose modulation scheme

$$\begin{array}{ll} M_1 & \text{if } SNR < \theta_1 \\ M_i & \text{if } \theta_i \leq SNR < \theta_{i+1}, \quad i = 1, \ldots, N-1 \\ M_N & \text{otherwise} \end{array}$$

Notice that this protocol assumes that the values of $\theta_1, \ldots, \theta_N$ are known. In practice, however, it is difficult to determine the BER characteristics precisely, necessitating the use of approximations. For our simulations we used the BER equations found in Chapter III. The values of $theta_i$ are shown in Table I for the QAM-based radio model.

## 3. Network Configurations

In our analysis, we used two different network configurations.

Table I. *Values of SNR thresholds $\theta_i$ (dB) for the modulation schemes that were used in our QAM-based radio simulation model (BER = 1E-5).*

| Modulation | Rate (Mbps) | $\theta_i$ (dB) |
|---|---|---|
| BPSK | 1 | $< 9.6$ |
| QPSK | 2 | $< 17.1$ |
| QAM16 | 4 | $< 23.3$ |
| QAM64 | 6 | $< 29.4$ |
| QAM256 | 8 | $>= 29.4$ |

**Configuration 1:** The first configuration consisted of two identically configured nodes communicating on a single channel. One of the nodes was held in a fixed position, while the other traveled along a direct-line path to and from the fixed node in a repetitious, oscillatory motion. The length of the path was 300m, which was the maximum effective transmission range of the modulation schemes as simulated (see Fig. 2). The purpose of this configuration was to stress the rate adaptation schemes, but doing so within the bounds of a plausible scenario.

**Configuration 2:** The second configuration consisted of 20 nodes in continuous motion within a 1500x300 meter arena. For each experiment, nodes were first placed at random starting positions in the arena, and then moved during the simulation according to the *random waypoint* mobility pattern (as described in [19]). The speeds at which nodes traveled were also chosen randomly, but were held to within $\pm 10\%$ of the mean node speed for the trial. For most experiments, we used mean node speeds of 2, 4, 6, 8, and 10 m/s. Unlike in the 2 node configuration, in this configuration we were interested in observing the performance characteristics of the proposed protocols in a plausible ad hoc networking environment. Thus, the nodes were configured to use the DSR routing protocol found in [19] instead of static routing. Unless otherwise stated, all results were based on the average of 30 runs using 30 precomputed scenarios, or *patterns*. Each pattern, generated randomly, designated the placement, heading, and speed of each node over the simulated time. For each pattern, the starting position and direction of each node was random, as well as its speed and destination. When a node reached its destination, a different speed was chosen at random, uniformly distributed in an interval of $0.9v - 1.1v$, for some mean speed $v$. For experiments in which the mean speed $v$ was varied, we used the same precomputed patterns so that the same sequence of movements occurred for each experiment. For example, consider one of the patterns, let's call it $I$. A node $x$ in $I$ that takes time $t$ to move from point $A$ to point $B$ in the 5 m/s run of $I$ will take time $t/2$ to traverse the same distance in the 10 m/s run of $I$. So, $x$ will always execute the exact same sequence of moves in $I$, just at a proportionally different rate. The patterns we used had a duration of 600s at a mean node speed of 2 m/s. To provide a fair comparison, the exact same set of patterns were used for each protocol tested.

## C. Performance Evaluation

In this section we present the results of our performance evaluation.

### 1. Overhead of the Reservation Subheader

There are several sources of overhead caused by the reservation subheader. The most obvious is the addition of the four byte check sequence to the MAC header. Additional overhead is encountered when the data rate used to transmit the RTS packet is lower than the rate used to transmit the data packet. Recall that the purpose of the reservation subheader is to update the tentative reservations that were made by the the RTS packet. If a node succeeds in hearing the RTS but fails to hear the subheader, then it may defer for an incorrect amount of time. Too short a time, and its next transmission may collide with the ACK coming back for the data packet.

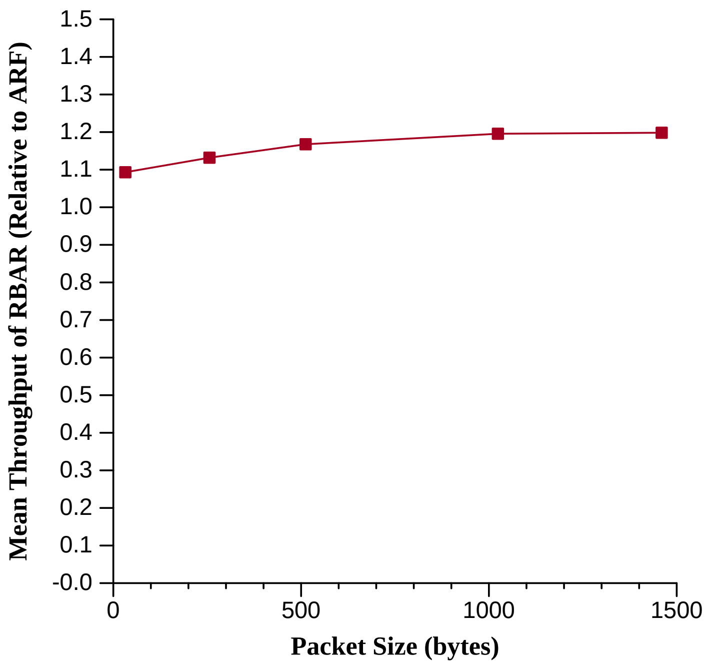


Fig. 11. *Impact of the reservation subheader on performance (relative to ARF) as a function of the packet size.*

Too long a time, and the channel may be idle. Thus, the subheader must be sent at the same or lower rate to reach those nodes that heard the RTS. The per-packet overhead of the difference in rates is easy to calculate. However, to gauge the impact that the per-packet overhead has on overall performance, we simulated the network in Configuration 1 with a single UDP connection for a range of packet sizes: 32, 256, 512, 1024, and 1460 bytes. Data was generated by an 8Mbps CBR source, and the data rate for the control packets (and, summarily, the reservation subheader), was fixed at 1Mbps. The results of these experiments are shown in Fig. 11, which presents the throughput for both protocols as a percentage of ARF's throughput. Note that, even for small packet sizes, the overhead of RBAR's reservation subheader has a relatively modest performance impact. Even for the smallest packet size (32 bytes), RBAR maintains an approximate 10% improvement over ARF.

Even when sent at a low rate, a node may still fail to receive the subheader,

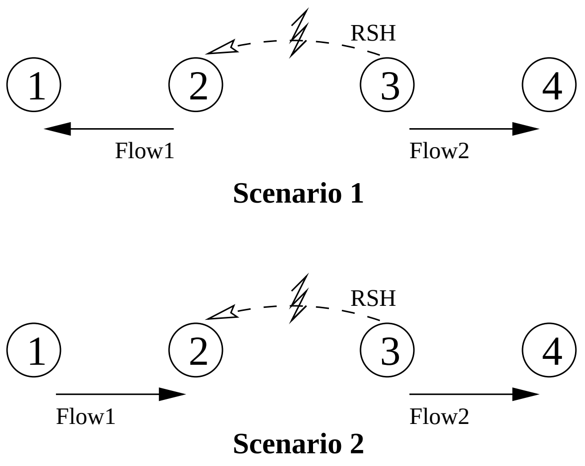


Fig. 12. *Network scenarios used to analyze the performance impact caused by the loss of the RSH subheader.*

Table II. *Mean per-flow throughput for varying reservation subheader loss probability for the network scenarios shown in Fig. 12*

| Err | Scenario 1 | | | Scenario 2 | | |
|---|---|---|---|---|---|---|
| Prob | Flow1 | Flow2 | Total | Flow1 | Flow2 | Total |
| 0.00 | 727 | 665 | 1393 | 222 | 1167 | 1389 |
| 0.05 | 708 | 690 | 1398 | 194 | 1187 | 1382 |
| 0.10 | 684 | 716 | 1400 | 187 | 1192 | 1380 |
| 0.15 | 676 | 724 | 1400 | 161 | 1220 | 1381 |
| 0.20 | 644 | 758 | 1402 | 153 | 1236 | 1389 |
| 0.25 | 634 | 761 | 1395 | 134 | 1260 | 1395 |
| 0.30 | 552 | 843 | 1394 | 123 | 1261 | 1385 |
| 0.35 | 537 | 855 | 1393 | 100 | 1285 | 1385 |
| 0.40 | 562 | 831 | 1393 | 93 | 1300 | 1393 |
| 0.45 | 498 | 896 | 1394 | 62 | 1330 | 1392 |
| 0.50 | 446 | 954 | 1400 | 47 | 1343 | 1390 |

such as, for example, when an RTS from a neighboring node collides with it or a deep fade causes excessive errors. Although such circumstances can also occur in standard 802.11, use of the reservation subheader may cause them to occur more frequently, and with more impact on performance, as touched on earlier. Thus, to gauge the sensitivity of RBAR to the loss of the reservation subheader, we simulated the networks shown in Fig. 12 for varying loss rates. In Scenario 1, the network consisted of four nodes with two flows directed away from the center of the network such that the source nodes were able to hear each other but the sink nodes were out of range of all but the source of their flow. The distance between the nodes was such that the optimal rate along each flow was 2Mbps, and the rate announced in the RTS was always 1Mbps. In Scenario 2, the network was similar except the direction of one of the flows was reversed. In both scenarios, the reservation subheaders from Node 3's packets were corrupted with varying probability, so it was expected that Flow 1 would experience a decrease in performance with an increase in the probability of loss.

The results of both experiments are shown in Table II. Each row represents the measured throughput (in Kbps) for the probability of loss shown in the leftmost column. As a basis of comparison, the measured throughput for ARF in Scenario 1 was 576Kbps for Flow 1 and 572Kbps for Flow 2, and in Scenario 2 it was 278Kbps for Flow 1 and 867Kbps for Flow 2. The difference in the throughputs between the flows in Scenario 2 is due to problems with fairness in 802.11 [109]. For Scenario 1, there is only a moderate impact on Flow 1's performance. For a 5% loss of Node 3's packets, there is only a 3% decrease in performance, and the decline stays below 10% beyond a loss of 15%. However, in Scenario 2 we see a larger impact on Flow 1, starting at a decline of 14% at a 5% loss, increasing rapidly to 38% at a loss of 15%. Thus, it is evident that situations in which reservation subheaders are lost for nodes that are on the receiving end of a flow are more sensitive to that loss, most likely because the sender on that flow is subject to repeated backoff when its RTS's are ignored with increasing probability.

### 2. Slow Changing Channel Conditions

To observe the performance of the protocols under conditions when the channel conditions are static or slow changing, we again simulated the network in Configuration 1, but the mobile node was moved in 5m increments over the range of mobility (0m - 300m), and held fixed for a 60s transmission of CBR data over a single UDP connection. Here, data was generated at a rate of 8Mbps and sent in 1460 byte packets.

The results of these experiments are shown in Fig. 13 for ARF and Fig. 14 for RBAR. Also shown are the results when the fixed rates are used. Notice that ARF fails to perform as well as the fixed rates at each distance except beyond that which is optimal for the highest rate. This is because ARF periodically tries to send data packets at the next highest rate in an attempt to gauge the channel conditions. In situations where the conditions are such that those packets are lost with high probability, then there is repeated packet loss resulting in the consistent performance degradation shown in the results.

RBAR, on the other hand, generally performs better at all distances except close in, where ARF excels. This is because of the increased impact of the reservation subheader. Recall that the reservation subheader has to be sent at one of the basic rates (in this instance, 1Mbps). Thus, at higher data rates the overhead of the subheader becomes more significant. One way to reduce this overhead is to employ a mechanism that predicts the best data rate for the channel conditions. One such technique is to simply cache the most recent rates as they are discovered. Fig. 15 shows the results when such a technique is employed. Clearly there is a significant

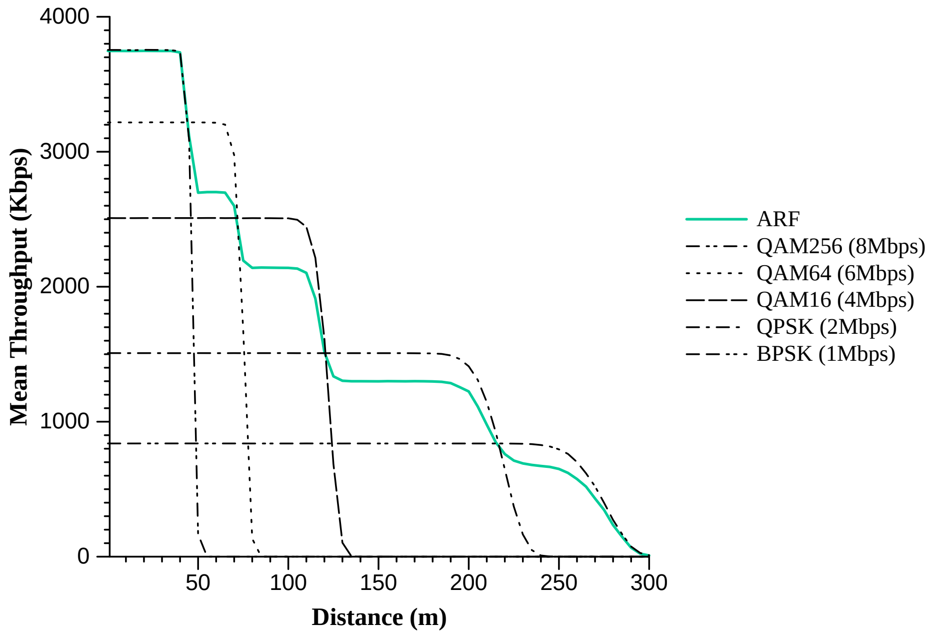


Fig. 13. *Performance of ARF for a single CBR connection between two nodes at fixed distances.*

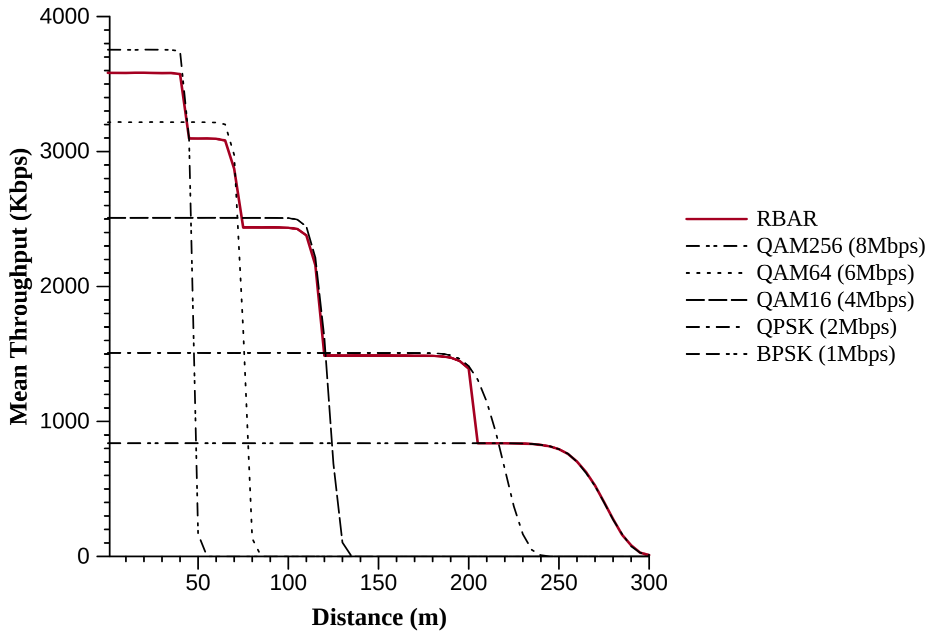


Fig. 14. *Performance of RBAR for a single CBR connection between two nodes at fixed distances.*

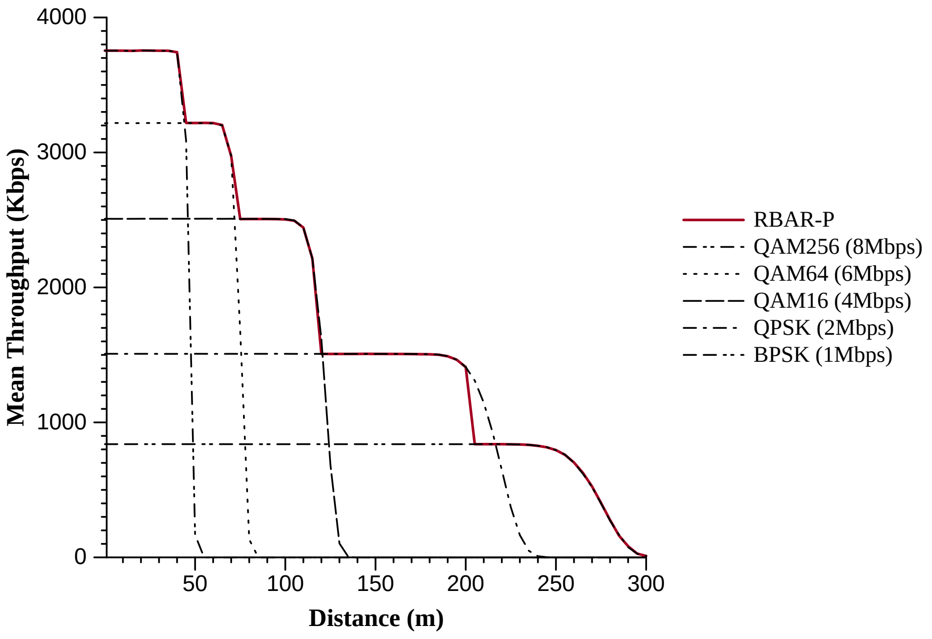


Fig. 15. *Performance of RBAR when a simple heuristic is used by the sender to try and predict the best data rate for the conditions, in an effort to reduce the frequency of the necessity for reservation subheaders.*

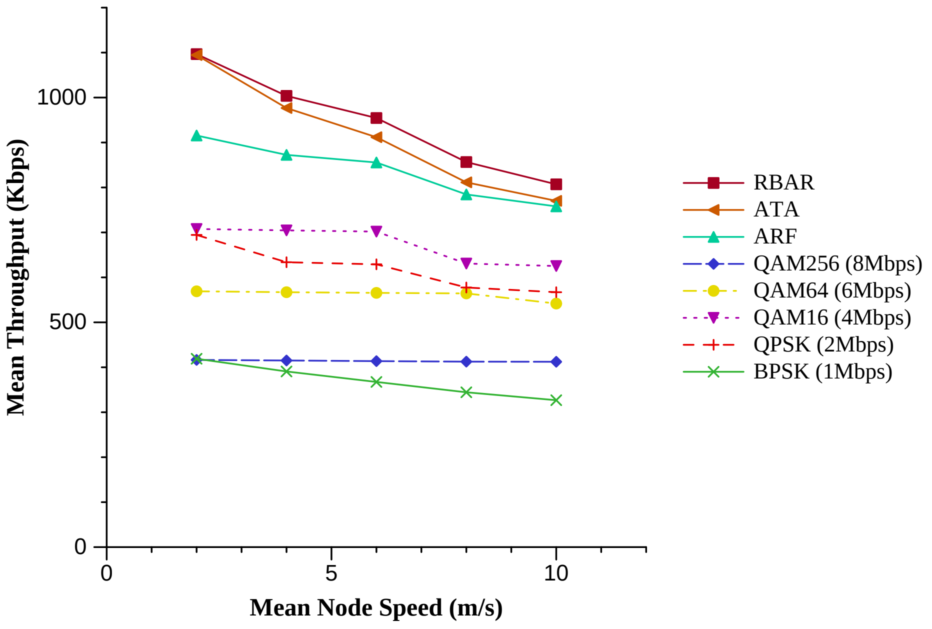


Fig. 16. *Performance for CBR traffic over a single UDP connection in a Rayleigh fading channel.*

improvement in the instance shown here, due to the high predictability of the channel. However, better techniques such as those proposed in [10] may also work well for RBAR. This is a topic of future study.

### 3. Fast Changing Channel Conditions

In a Rayleigh fading channel, variations in the wireless signal are induced at a rate that depends, in part, on the speed along the line-of-sight between the transmitter and the receiver. For a conventional local area network with nodes moving at walking speeds (e.g., node speed $\leq 2$ m/s communicating at 2Mbps over a 2.4GHz channel), changes generally occur slowly enough that the channel is effectively constant for the duration of a packet exchange (the coherence time). However, as the speed increases, changes occur much more rapidly, decreasing the predictability of the channel. Thus, by simulating a fading channel and varying the mean node speed, we can evaluate the adaptability of the protocols.

To observe this impact, we performed experiments for five different speeds, 2, 4, 6, 8, and 10 m/s, for Configuration 1 (see Fig. 16). Results were generated for a UDP connection carrying CBR traffic that was generated at a rate of 8Mbps and sent in 1460 byte packets.

- RBAR outperformed ARF for all mean node speeds, with the performance improvement ranging from 6% (10 m/s) to 20% (2 m/s).

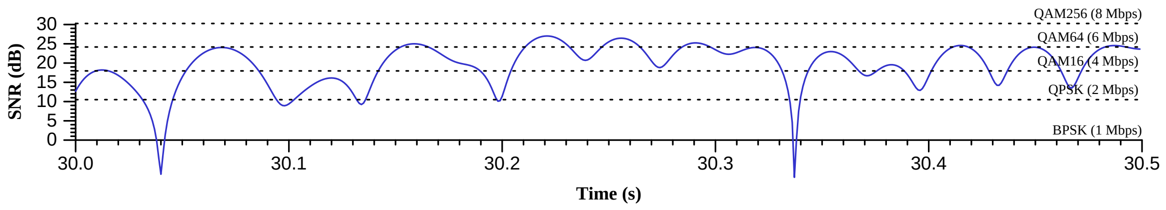
- An increase in mean node speed resulted in a decrease in performance. As expected, the increase in variability of the signal resulted in a decrease in performance.

  Also notice that the performance improvement for RBAR also decreased as the mean node speed increased. Recall that the simple channel quality prediction

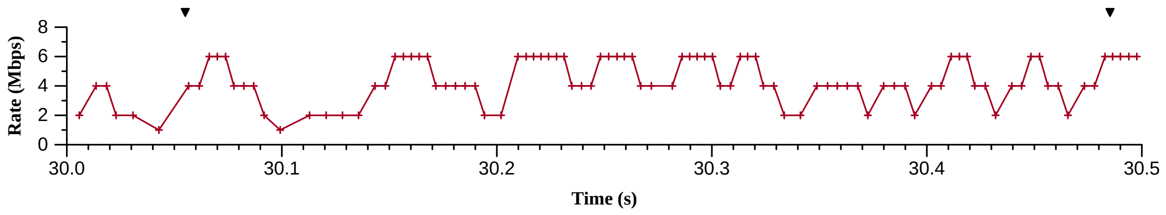
> mechanism used in RBAR for these results works best when the channel coherence time is larger than the time it takes to transmit the CTS packet and the DATA packet. For 2 m/s, the coherence time was sufficiently large that this was true for packets transmitted at all data rates. However, as the node speed increased, the coherence time shortened and the higher data rates were also affected, resulting in a decline in performance. As mentioned previously, we expect that this decline can be improved significantly with better channel quality prediction techniques.

The adaptability of RBAR to the rapidly changing channel conditions can be more clearly seen in Fig. 17. This figure shows the performance of RBAR for a single CBR connection between two nodes in a Rayleigh fading channel. Here, the sender is fixed and the receiver is moving at a speed of 2 m/s away from the sender. Fig. 17(b) shows the time at which packets were transmitted and the modulation rate chosen by RBAR for each packet. The tick marks along the top show the time at which packets were dropped by the receiver due to errors. Fig. 17(a) shows the SNR at the receiver for the packets shown in Fig. 17(b). Also shown are thresholds representing the SNR values above which the next higher modulation rate has a theoretical mean $BER \leq 10^{-6}$. At the start, both nodes were at the same location, so the leftmost edges represent the point in time at which the two nodes were 60m apart. Compared to the similar figure for ARF, it is clear that RBAR is much better at reacting and adapting to the channel conditions.

We also simulated a single TCP connection under the same conditions. These results are shown in Fig. 18. Notice that the performance improvement is more significant, which can be attributed to TCP's sensitivity to packet loss due to wireless errors.



(a) *SNR at the receiver for the packets shown in (b).*



(b) *Time and rate of packet transmission.*

Fig. 17. *Performance of RBAR for a single CBR connection between two nodes in a Rayleigh fading channel. The sender is fixed and the receiver is moving at a speed of 2 m/s away from the sender.*

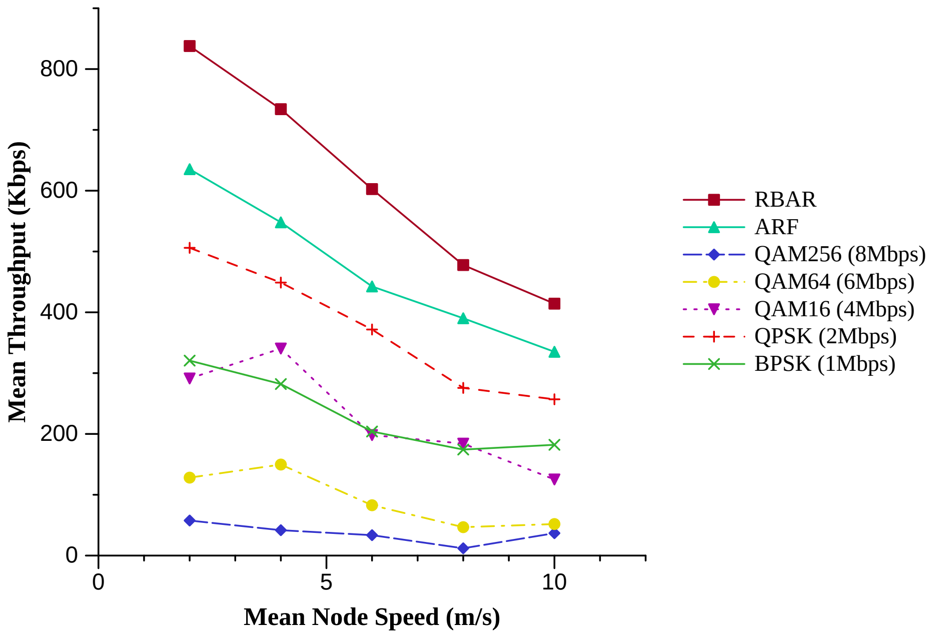


Fig. 18. *Performance for FTP traffic over a single TCP connection in a Rayleigh fading channel.*

### 4. Impact of Variable Traffic Sources

In this section, we study the impact of bursty data sources on the performance of the RBAR and ARF protocols. For this study, we performed a series of experiments using an ON/OFF traffic source, with ON ($\bar{\tau}_{on}$) and OFF ($\bar{\tau}_{off}$) times drawn from a Pareto distribution. During an ON period, data was generated at a rate of 8Mbps and sent in 1460 byte data packets, resulting in mean packet bursts ranging from $\approx 1 - 2$ packets ($\bar{\tau}_{on} = 1.5ms$) to $\approx 20$ packets ($\bar{\tau}_{on} = 30ms$). Traffic was generated for a single UDP connection across a Rayleigh fading channel. The mean node speed was 2 m/s, and we used Configuration 1.

The results of these experiments are presented in Figs. 19 and 20, for mean OFF times of 1s and 500ms respectively, which show the average delivery ratios for each protocol, where the delivery ratio is defined as the number of data packets successfully received over the total number of data packets sent.

Note that:

- RBAR outperforms ARF in all traffic conditions, with improvements ranging from 26% to 70%.
- RBAR shows the greatest improvement when the traffic is the lightest, and the least improvement when the traffic is heavy.

### 5. Multi-Hop Performance

In this section we present results for Configuration 2: 20 nodes in continuous motion within a 1500x300 meter arena. Here, we simulated a single CBR source generating traffic on a single UDP connection between two nodes in the ad-hoc network.

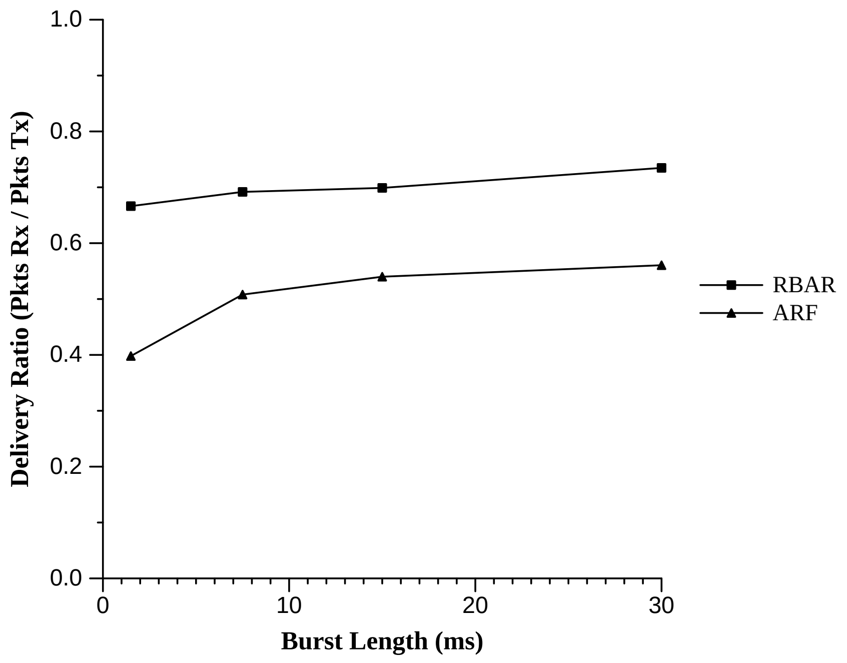


Fig. 19. *Protocol efficiency for an ON/OFF Pareto source generating traffic on a single UDP connection in a Rayleigh fading channel. The mean OFF time = 1s.*

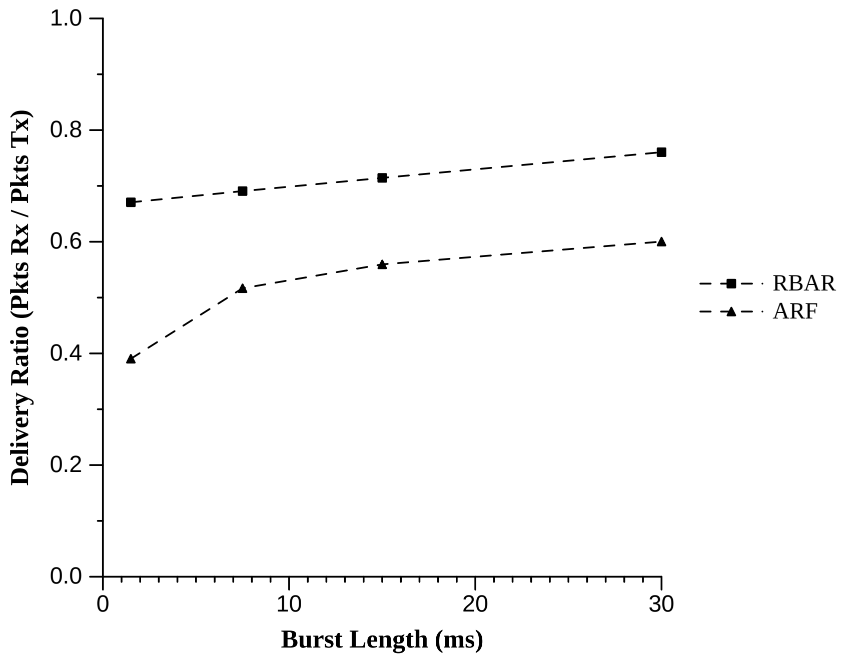


Fig. 20. *Protocol efficiency for an ON/OFF Pareto source generating traffic on a single UDP connection in a Rayleigh fading channel. The mean OFF time = 500ms.*

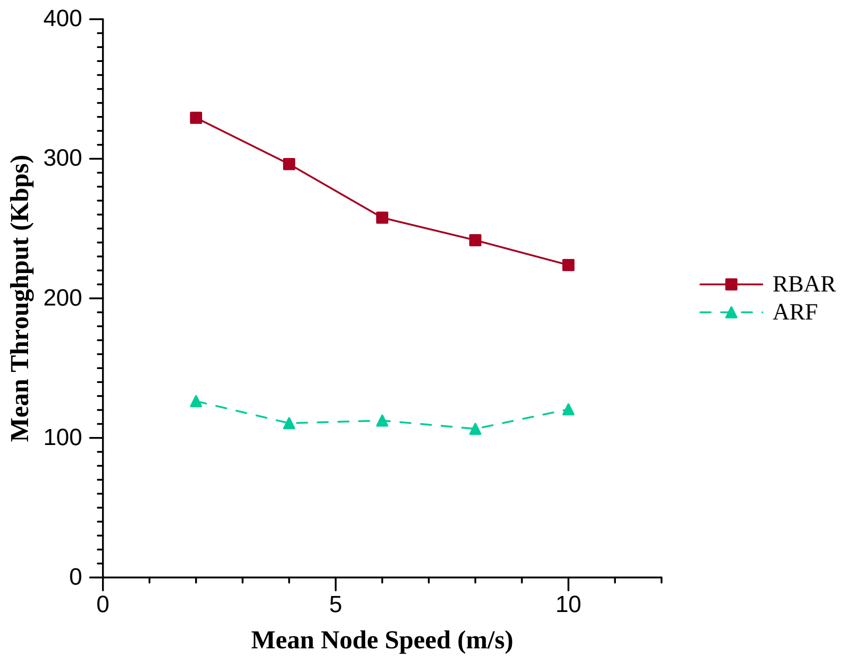


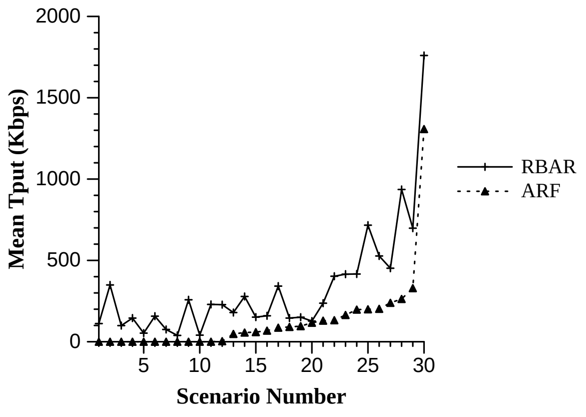
Fig. 21. *Performance comparison for a single CBR connection in a multihop network.*

The results are shown in Fig. 21 and Fig. 22. Notice that RBAR consistently outperforms ARF.

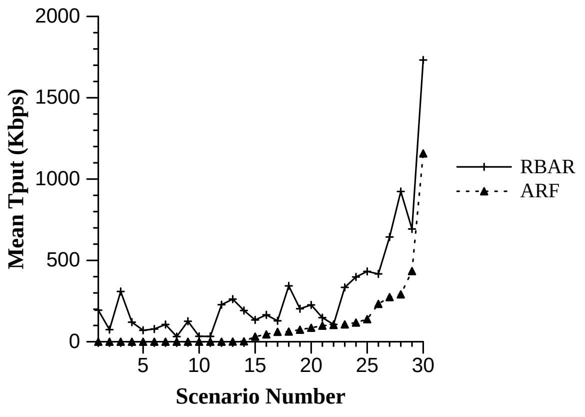
Similar results are shown in Fig. 23 for an FTP source generating traffic over a TCP connection. Clearly, the performance gains observed earlier are also applicable to a multihop scenario. We believe that ARF's increase in throughput with increased speed is due to its poor ability to select the correct rate when nodes are far apart, resulting in repeated backoff by TCP early in the simulations. However, with increased speed it may occur that the connection is established sooner due to the speed at which nodes in the sparse starting alignment are brought into range.

## D. Summary

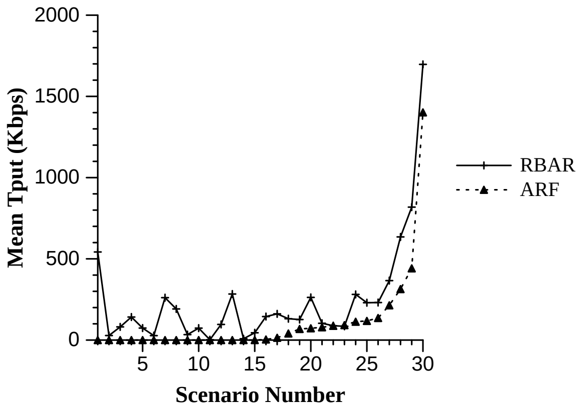
In this chapter, we addressed the topic of optimizing performance in wireless local area networks using rate adaptation. We presented a new approach to rate adaptation,



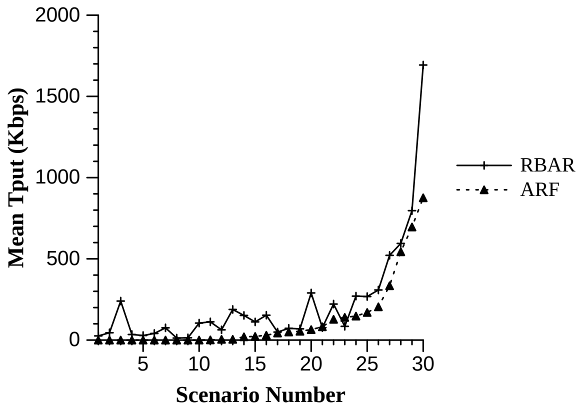
(a) *Mean speed = 2 m/s*



(b) *Mean speed = 4 m/s*



(c) *Mean speed = 8 m/s*



(d) *Mean speed = 10 m/s*

Fig. 22. *Performance comparison across multiple hops. Shown here are the results for the individual scenarios, sorted according to the increasing throughput of ARF.*

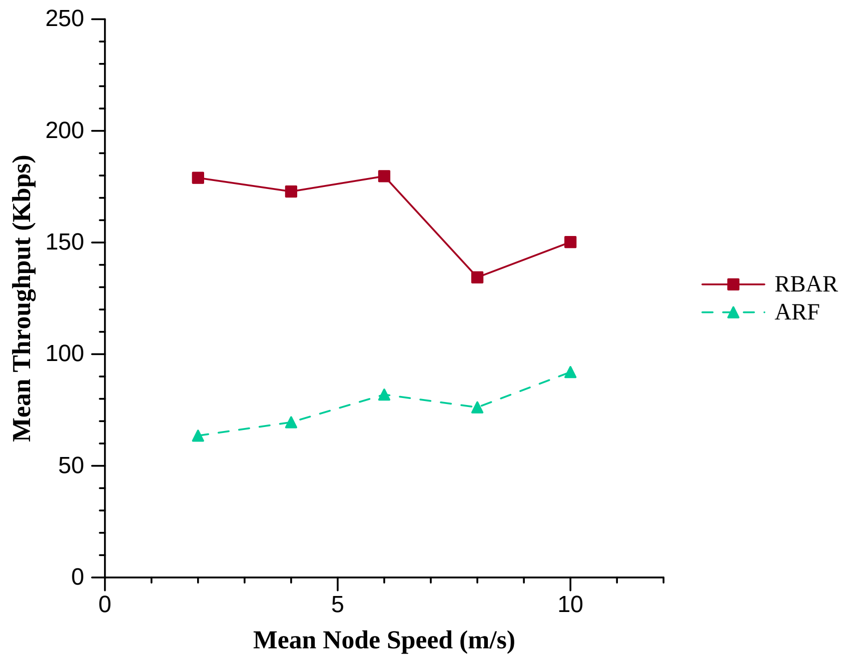


Fig. 23. *Performance comparison for a single FTP connection across a multihop network.*

which differs from previous approaches in that it uses the RTS/CTS protocol to enable receiver-based rate adaptation. Using this approach, a protocol based on the popular IEEE 802.11 standard was presented, called the *Receiver-Based AutoRate* (RBAR) protocol. Simulation results were then presented comparing the performance of the proposed protocol against the performance of an existing 802.11 protocol for mobile nodes across Rayleigh fading channels. These results showed that RBAR consistently performed well.

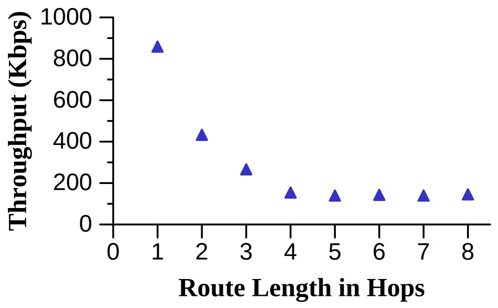
# CHAPTER V

# MULTIRATE AWARE ROUTING

In this chapter, we analyze the impact that multirate wireless links have on routing in multihop wireless networks. We then present two routing protocols that are designed to discover and prioritize routes that maximize throughput, and then show how they outperform several well-known conventional shortest-path routing protocols.

## A. Impact of Multirate, Multihop (MRMH) Paths on Routing

Packet flows that traverse multihop routes in single channel broadcast wireless networks experience a decrease in end-to-end throughput with an increase in route length. This can be seen in the simulation results in Fig. 24, which shows the end-to-end throughput for a single CBR flow across a chain network topology as a function of the route length in hops. The data rate is 1Mbps and the packet size is 1460 bytes. Nodes were spaced equidistantly, at intervals large enough that significant transmission interference did not exceed one hop. Notice the significant decrease in throughput as route length increases from one to four hops, and the relatively insignificant decrease as the route length increases from four to eight hops. The primary cause for the decrease is *self-interference*. Self-interference occurs when packets from the same flow contend with one another for the channel at different hops along the route. Consider the scenario shown in Fig. 25. Here, Node $A$ has a queue of packets to send to Node $E$. A solid circle represents the transmission range of the sender and a dotted circle represents the reception range of the receiver. For the purpose of illustration, we assume the transmission range and interference range are equivalent. The situation that is shown is the result of the following sequence of events. Node $A$ successfully reserves the channel and transmits packet $pkt_1$ to $B$. Node $B$, in its attempt to



| *Hops* | *Throughput (Kbps)* |
|---|---|
| 1 | 858.986 |
| 2 | 433.106 |
| 3 | 266.541 |
| 4 | 154.359 |
| 5 | 139.732 |
| 6 | 143.212 |
| 7 | 139.743 |
| 8 | 145.65 |

Fig. 24. *Throughput as a function of path length in hops for a single CBR flow across a "chain" network topology with 1Mbps links (802.11). The packet size is 1460 bytes. Nodes were spaced such that transmission interference did not exceed one hop.*

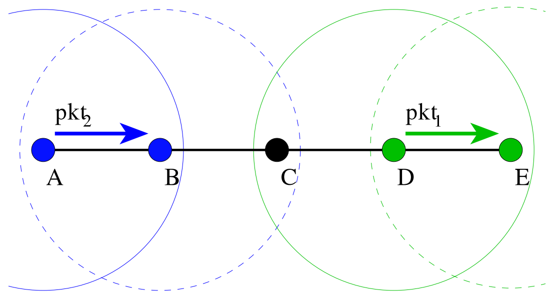


Fig. 25. *A scenario demonstrating self-interference across a multihop route. Here, Node $A$ has packets to send to Node $E$. After sending pkt$_1$, $A$ is prevented from sending pkt$_2$ while nodes $B$ and $C$ forward pkt$_1$ to $D$ because of interference at $B$.*

forward $pkt_1$ to $C$, contends with $A$ for the right to transmit. $B$ succeeds, so $A$ now contends with $C$ because $C's$ transmission of $pkt_1$ to Node $D$ will collide with $A$'s transmission of $pkt_2$ to $B$. Referring back to Fig. 24, for a one hop route ($A \to B$), there is no self-interference so the end-to-end throughput is equivalent to the channel throughput (about 900 Kbps, in this case). For a two hop route ($A \to B \to C$), the throughput is roughly one-half that of the one hop route because $A$ and $B$ must share the channel, and, similarly, the throughput for the three hop route is one-third.

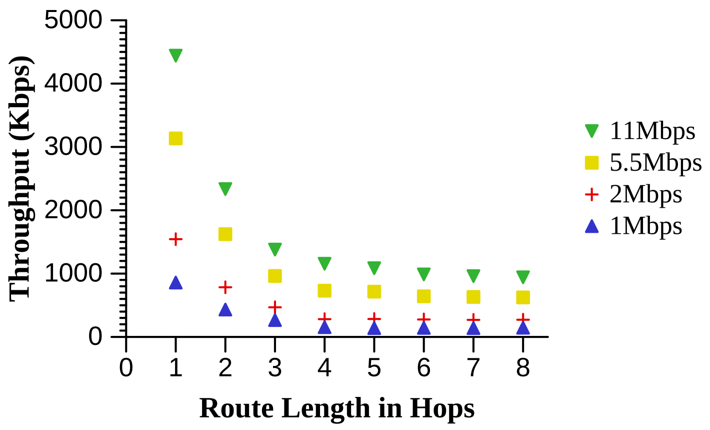


Fig. 26. *Comparison of the measured throughput of a single CBR flow across a chain network of varying length and link data rate.*

Beyond three hops the end-to-end throughput is relatively unchanged. This is because spatial reuse of the wireless channel allows the formation of a *packet forwarding pipeline* – packets propagating in lockstep along the route at equidistant intervals. For instance, in Fig. 25, if the chain were longer and $A$'s destination extended beyond $E$, say Node $G$, then $E$ could forward $pkt_1$ to $F$ while $B$ forwards $pkt_2$ to $C$, and $F$ could forward to $G$ while $C$ forwards to $D$, and so on. If the pipelining is perfectly synchronized, then additional hops beyond the third will only increase end-to-end propagation delay because, once the pipe is full, packets will arrive at the destination in regular intervals (equal to three times the per-link packet transmission time, assuming transmission interference does not extend beyond one hop). The fact that there is a drop in throughput between three and four hops suggests that 802.11 is unable to maintain a perfectly synchronized pipeline, which we will discuss later in the chapter.

For multirate, multihop (MRMH) networks, this relationship between throughput and path length has an interesting impact on routing. Usually, the route over the shortest path is the one that gives the highest throughput and lowest delay, so

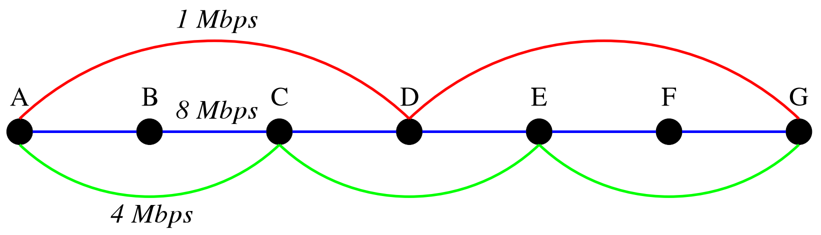


Fig. 27. *An example MRMH chain network, where transmissions at 8Mbps can reach one hop along the chain, 4Mbps one and two hops, and 1Mbps one, two, and three hops.*

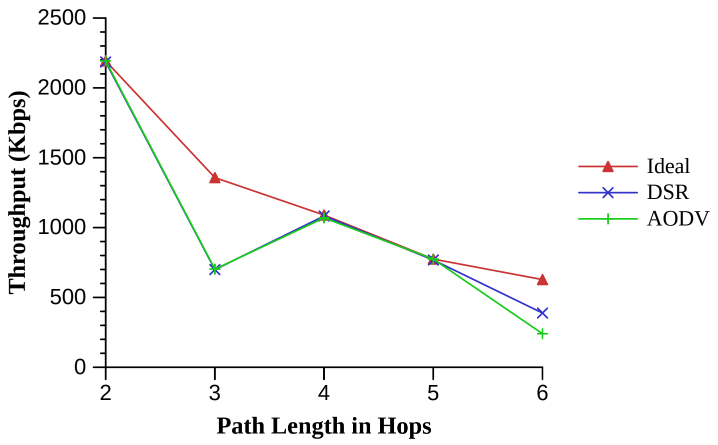


Fig. 28. *Throughput as a function of path length in hops for a single CBR flow across a MRMH chain network. Shown are results for an "ideal" multirate routing protocol, and two common shortest-path routing protocols: DSR, and AODV.*

most routing protocols give priority to the shortest routes (e.g., DSR [70], AODV [92], OLSR [23], DSDV [93], TORA [89]). In an MRMH network, however, shortest-path may not be the best strategy. Consider Fig. 26, which shows simulation results for the same network scenario as in Fig. 24 but for a number of different link data rates. Notice, for instance, that a five hop route at 5.5Mbps has higher throughput than a two hop route at 2Mbps, and an 8 hop route at 11Mbps has higher throughput than a one hop route at 1Mbps. In such instances, the shortest path may not be the best choice of route since it may result in less than ideal end-to-end throughput.

For example, consider the MRMH chain network in Fig. 27. Because of the rate/range tradeoff, transmissions at 8Mbps can reach only one hop along the chain, 4Mbps can reach one and two hops, and 1Mbps can reach one, two, and three hops. Simulation results for a single CBR flow going from Node $A$ to Node $C$ (two hops), Node $D$ (three hops), and so on, are shown in Fig. 28 for two different shortest-path routing protocols, AODV and DSR, versus the known best routes. Notice that both routing protocols fail to select the best routes for the three and six hop paths. In the case of the three hop path, both protocols select the shortest 1Mbps route from Node $A$ to Node $D$, even though the longer 4Mbps route $A \rightarrow C \rightarrow D$ has higher capacity. Similarly, in the case of the six hop path, both protocols fail to select the 4Mbps route $A \rightarrow C \rightarrow E \rightarrow G$ in favor of the shorter, lower capacity 1Mbps route $A \rightarrow D \rightarrow G$. In the case of the two, four, and five hop paths, the protocols succeed in selecting the best routes because, in all cases, the best route is also in the set of shortest paths, and since it has the highest capacity it also has the lowest delay and, therefore, is discovered and used first. However, there is no guarantee that this will always be the case. In the two hop case, both protocols might have selected the 1Mbps one hop route to $C$ instead of the 4Mbps one hop route.

## B. Routing Metric for MRMH Networks

Based on the observations in the previous section, it is easy to see that a simple upper-bound on the throughput across a route of length $n$ and rate $r$ for a flow of same-size packets is

$$\begin{array}{ll} \frac{r}{n}\,, & \text{for } 1 \leq n \leq 3; \\ \frac{r}{3}\,, & \text{for } n > 3. \end{array} \tag{5.1}$$

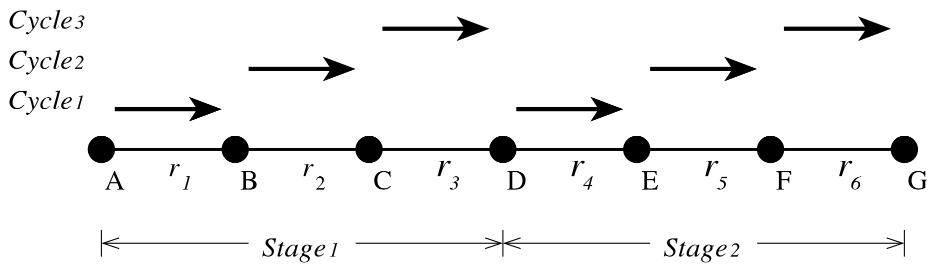


Fig. 29. *Packet pipelining across a heterogeneous route – a route that traverses links with different data rates.*

assuming transmission interference does not extend beyond one hop, and all link rates are homogeneous (identical across the path). If the rates are heterogeneous, then a more accurate bound could be derived as follows. Consider Fig. 29, which shows a route across six hops with link data rates $r_1, \ldots, r_6$. Crossing the route is a flow going from Node $A$ to Node $G$. Under ideal circumstances (e.g., optimal scheduling, transmission interference range of one hop, same size packets), the packets will traverse the route in lockstep at intervals of three hops. We refer to these intervals as *stages*. The sequence of stages starts at the source node of the flow. The route in Fig. 29 has two stages: $\{AB, BC, CD\}$ and $\{DE, EF, FG\}$ (as defined by their link sets). Every link in a stage is also a member of a *link activation set*, which is a set of links that can transmit (activate) simultaneously without interfering [8, 22]. In this example, the members of each link's activation set are those links that have the same relative position within their stage on the route. For example, links $AB$ and $DE$ are co-members of a set since they are the first links in their stages. In Fig. 29 there are three activation sets: $X_1 = \{AB, DE\}$, $X_2 = \{BC, EF\}$, and $X_3 = \{CD, FG\}$. It is easy to see that these sets are optimal for the route shown: they are the minimum number of activation sets whose union contains all the links on the route. These sets, when activated in order, are said to advance the pipeline one *cycle*. Intuitively,

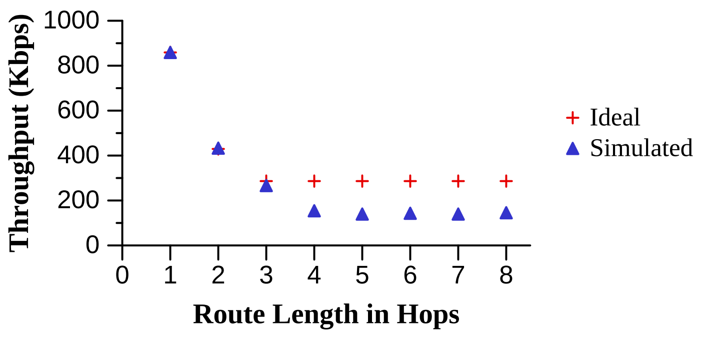


Fig. 30. *Comparison of the estimated throughput to the simulated 802.11 throughput as a function of route length n in hops for a CBR connection across a homogeneous chain network. Here, r is the maximum measured throughput across a one hop, 1Mbps link, for 1460 byte packets.*

network utilization is maximized in this example if every link in an activation set transmits for the same duration within a cycle. Such is the case for a flow of same-sized packets traversing a homogeneous route. However, in multirate networks, routes may be heterogeneous such that links in the same activation set may not transmit at the same data rate during a cycle. In this case, the duration of the cycle is determined by the link in the set with the lowest data rate, so a more accurate bound on the route throughput is

$$\begin{cases} \frac{1}{\sum_{i=1}^{n} \frac{1}{r_i}}\ , & \text{for } 1 \leq n \leq 3; \\ \frac{1}{\sum_{k=1}^{3} \frac{1}{\rho_k}}\ , & \text{for } n > 3. \end{cases} \tag{5.2}$$

where $r_i$ is the data rate of link $i$, and

$$\rho_k = \min_{i \in X_k} r_i\ ,$$

is the minimum link data rate for activation set $X_k$.

A comparison between the throughput estimate from (5.2) and the measured throughput for a homogeneous 802.11 chain network is shown in Fig. 30. The link data

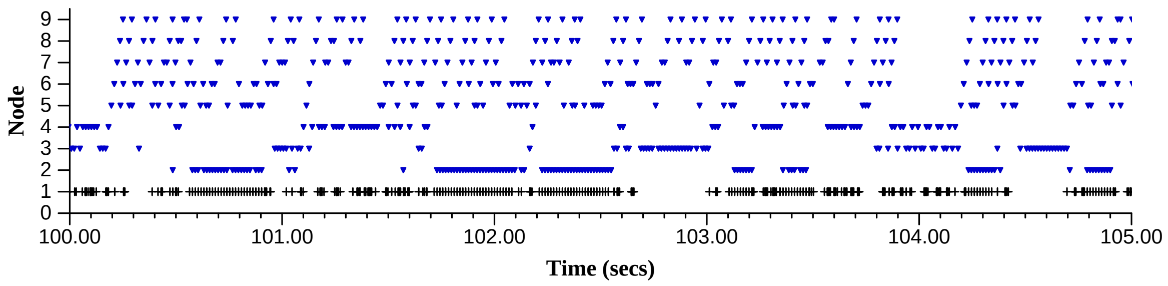


Fig. 31. *A five second snapshot of packet transmission events for a CBR flow across an eight hop chain network. The triangularly shaped points represent the successful reception of a data packet by a node along the route. The source of the flow is Node 1 and the sink is Node 9. The cross shaped points shown for Node 1 represent each attempt to send a new data packet.*

rate $r$ in (5.2) is set to the maximum measured throughput across a one hop, 1Mbps link for 1460 byte packets. Clearly (5.2) overestimates throughput for routes longer than three hops. This discrepancy appears to be caused by the MAC's inability to maintain a synchronized packet-forwarding pipeline. For example, Fig. 31 shows a five second snapshot of packet transmission events for the flow in Fig. 30. The triangularly shaped points represent the successful reception of a data packet by a node along the route. The cross shaped points shown for Node 1 represent each attempt to send a new data packet. There are several interesting things to note about this figure. There is no consistent transmission pattern on the first three hops, demonstrating that the 802.11 MAC does not maintain a synchronized pipeline. Instead, Nodes 1, 2, and 3 are able to capture the channel for relatively long periods of time. However, beyond the first four hops the packets are forwarded in a relatively stable pipeline. This is presumably the result of buffering along the route, which absorbs the large jitter in packet inter-arrival times that occur on the first three hops. To account for this discrepancy, we introduce a per-rate compensation factor $\gamma_{r_i}$ in our throughput

Table III. *Compensation factor ($\gamma$) values.*

| *1 Mbps* | *2 Mbps* | *5.5 Mbps* | *11 Mbps* |
|---|---|---|---|
| 0.50 | 0.53 | 0.64 | 0.69 |

estimate as follows

$$\begin{cases} \frac{1}{\sum_{i=1}^{n} \frac{1}{r_i}}, & \text{for } 1 \leq n \leq 3; \\ \frac{\gamma_{r_i}}{\sum_{k=1}^{3} \frac{1}{\rho_k}}, & \text{for } n > 3. \end{cases} \tag{5.3}$$

where $\gamma_{r_i}$ is based on the simulation results in Fig. 26. The values for $\gamma$ in our simulation studies are shown in Table III. In the next section, we present two routing protocols that use (5.3) to optimize route performance in MRMH networks.

## C. Proposed MRMH Routing Protocols

In this section we introduce two routing protocols, *Rate Probing* and *Minimum Rate*, based on DSR, that discover and prioritize routes according to the estimated throughput given by (5.3).

### 1. Route Selection Algorithm

Our proposed protocols use the following route selection algorithm.

1. Given a set of routes $S = \{x_1, x_2, \ldots, x_n\}$ select the subset of routes $M \subseteq S$ that have the highest maximum throughput according to Equations (5.3).

2. If $|M| > 1$ then select the subset of routes $P \subseteq M$ that have the shortest length.

3. If $|P| > 1$ then select any route $x \in P$ at random.

This algorithm gives priority first to routes with the highest maximum throughput, and then to routes of shortest length. If multiple routes remain, one is chosen at random.

Both protocols also assume the availability of cross-layer communication. In particular, the Rate Probing protocol assumes that the link layer provides the means for upper layers to designate the data rate at which an individual packet is transmitted. It also assumes that the rate at which a packet was transmitted, or failed transmission, is also provided to upper layers. The Minimum Rate protocol assumes that the network layer is capable of providing the large-scale SNR for a received packet to upper layers. This can be done, say, by tracking the strength of beacons or Hello packets from each neighbor. Finally, both protocols assume that the set of data rates supported by the link layer are made known to upper layers.

## 2. Rate Probing Protocol

Our first protocol, Multi-Rate Rate Probing (MRPRB) discovers and maintains homogeneous routes between source destination pairs using *rate probing* during route discovery. The general idea behind this algorithm is to broadcast multiple route requests at different data rates, limiting propagation of each request to only those links that support the data rate. An advantage of this approach is that it does not require explicit estimation of link quality by the link layer – successful receipt of a request is an implicit indicator. A drawback is the added overhead of multiple route requests per discovery.

We now describe the route discovery procedure.

1. The source node broadcasts multiple DSR route requests, one for each rate supported by the network, starting with the lowest rate. Each request is assigned

a unique sequence number. Contained within each route request is the rate at which the request is to be broadcasted by the link layer. Random delays are injected between requests to reduce the instantaneous load on the network.

2. Intermediate nodes (i.e., neither source nor destination) that receive a route request forward the request according to the standard DSR forwarding policies, but at the link data rate designated in the request. They also store the reverse route with its rate in their cache.

3. The destination responds to each route request with a standard DSR route reply over the reversed source route in the request. The reply carries the rate at which the request was sent. Link layers that forward the reply are not required to transmit the reply at the designated rate.

4. The source immediately transmits packets on receipt of any route reply, caching each route received along with its data rate.

5. Packets are not required to be transmitted at the data rate in the cached route. Instead, it is assumed that each link will transmit the data packet at the optimum rate.

Several items to note about this procedure are the following.

Route requests are initiated at the lowest rate to minimize the delay in finding a usable route. As mentioned previously, lower data rates have a larger effective transmission range, so starting with the lowest rate increases the probability that a valid route is found quickly. The search for better routes by subsequent route requests may then proceed in parallel with data transmissions. Minimizing route discovery delay is of special importance to delay-sensitive protocols like TCP.

Link layers are not required to forward route replies at the designated rate because it is assumed that link characteristics are bi-directional. Thus, reverse routes stored by each intermediate node and the destination are considered usable at the rate in the request at the time the request is received. An alternative is to require bi-directional confirmation by forcing the link layers to transmit the reply at the designated rate, and limiting the routes discovered by intermediate nodes to only those destinations from which they receive and forward a reply.

During route selection, it is implicitly assumed that a route in the cache is available at all rates equal to and below the rate at which the route was discovered. This enhances the effectiveness of the protocol by increasing the number of available routes from which to select. This is important for protocols that do not guarantee discovery of multiple, possibly overlapping, routes. Note that this assumption is generally valid as long as all links in the network support the same set of data rates. If this is not true, then route selection can be modified such that only routes that are discovered explicitly may be chosen.

The route maintenance procedure is as follows.

1. Discovery of an invalid route is triggered by the standard DSR mechanisms. In DSR, detection of a downed link on a packet's source route triggers transmission of a route error to the packet's sender. We modify the route error to also carry the rate at which the packet failed transmission.

2. Upon receipt of a route error with rate $r$, a node purges or truncates all routes in its cache that contain the link and have rate $\geq r$.

3. Management of the route cache is identical to that in DSR, except that routes at different data rates are maintained separately.

4. No attempt is made to track and update the changing rates of cached routes over time.

Notice that it is assumed that the link layer provides the rate at which a packet failed transmission. If this information is not available, an alternative is to simply purge all routes that contain the link.

Since no attempt is made to track changes in route rates, performance could degrade if differences in cached rates and actual rates becomes significant. An alternative is to maintain route information by periodic transmission of redundant route requests, with higher rate requests occurring more frequently than lower rate requests to account for the higher probability of link failure for higher rate routes.

## 3. Minimum Rate Protocol

Our second protocol, the Multi-Rate Minimum Rate (MRMR) protocol, gathers, during route discovery, explicit information from the link layer about the optimal data rate on each link. In this algorithm, the standard DSR route request is modified to carry the minimum rate observed over all links on the route. This rate ultimately determines the maximum end-to-end throughput that the route can provide. An advantage of this approach is that it does not incur the overhead of multiple route requests. One drawback is that it requires link quality estimates from the link layer. This can be done, say, by tracking the strength of beacons or Hello packets from each neighbor.

We now describe the route discovery procedure.

1. The DSR route request is modified to carry a small *minimum data rate* field. The source node, prior to broadcasting the DSR route request, initializes this field to the maximum supported link rate.

2. Each node that receives a route request compares the value of this field to the optimal data rate for the link on which the request was received, and updates it if the link rate is smaller. The optimal data rate is calculated based on link quality estimates from the link layer.

3. Intermediate nodes cache the reverse route in the request with the minimum rate prior to forwarding the request. The request is forwarded according to DSR's standard route request procedure.

4. The destination, on receipt of the request, caches the route and its minimum rate prior to preparing a route reply. The route reply, like the route request, is modified to carry a minimum rate field. The destination updates this field with the minimum rate received in the request prior to sending the reply according to the standard DSR route reply procedure.

5. The source node transmits packets immediately upon receipt of a valid route, caching the route with its minimum rate.

In this procedure it is assumed that the link quality is an estimate of the large-scale fading characteristics of the channel. This could be provided, say, by low-pass filtering of channel samples or by hardware capable of recognizing multipath fading characteristics of the channel.

The route maintenance procedure is identical to the Rate Probing protocol.

## D. Performance Evaluation

In this section we analyze the performance of the proposed routing protocols.

To evaluate our protocols, we performed simulations using the ns-2 simulation tool, described in Chapter III. The nodes were configured as follows. Each node had a network stack consisting of TCP and UDP transport protocols, an IP network protocol with BSD's ARP protocol (used to resolve IP addresses to MAC addresses), and an 802.11 link layer. Transmission power was fixed at 20mW, antennas were omni-directional with a gain of 1dBi, and the transmission frequency was 2.412 GHz. Packet sizes were 1460 bytes. The MAC layer in all simulations was the common WaveLAN AutoRate Fallback (ARF) protocol described in Chapter IV. Additionally, the physical layer used in this chapter differs from the modified 802.11 device as described in Chapter III, in that we have modeled the standard 802.11b 5.5Mbps and 11Mbps CCK rates, instead of the the 4, 6, and 8 Mbps QAM rates. BER models for the CCK rates are based on those in [116]. The 1 and 2 Mbps rates were used as before. For comparison, we also simulated DSR (see Chapter VI for detail on DSR).

We used 50 random topologies of 50 mobile nodes in a 300x300 meter flat region. Nodes were initially placed randomly and uniformly over the entire region. During the simulation nodes moved according to the random waypoint mobility model In the random waypoint model, each node $x$ picks a random set of coordinates as its next destination and then travels in a straight line to those coordinates at some randomly chosen speed. Once $x$ arrives at this destination, it pauses before picking another set of coordinates and continuing onward. The randomly chosen speeds were uniformly distributed in an interval of $0.9\mu - 1.1\mu$ for some mean speed $\mu$. We generated results for five different mean speeds: 2, 4, 6, 8, and 10 m/s. For traffic patterns, we simulated a single CBR connection sending at maximum link speed (11Mbps) was used to guage the routing protocol's ability to discover and maintain a high throughput route, and 5 CBR connections sending at a rate of one packet every 10ms to guage the amount of control packet overhead a protocol generates when faced with multiple flows. Unless otherwise stated, all results are based on the average of 50 simulation runs of randomly generated scenarios for each mean speed. The same sets of scenarios were used for all routing protocols to provide a fair comparison: for a given mean node speed, each routing protocol experienced identical node positions, movements, and topology changes for each scenario.

The performance metrics we used are the following:

- *Mean Throughput:* The throughput for a flow was measured by counting the number of packets successfully received by the sink over the duration of a simulation run. The mean throughput is the average throughput over all flows.

- *Routing Overhead:* The ratio of the number of routing packets transmitted to the number of data packets successfully received at the sinks of all flows in the network.

### 1. Random Topology, Single CBR Flow

In this section we present simulation results a lightly loaded network: a single CBR flow. As mentioned previously, the network is a random topology of 50 mobile nodes in a 300x300m area. The mean speed of the nodes is varied to stress the routing protocols. We also simulated fading (Rayleigh) and non-fading channels, to see what impact channel characteristics had on the relative performance between the proposed routing protocols. We first present results for a non-fading channel.

The mean throughput as a function of node speed is shown in Fig. 32. As mentioned previously, this is calculated as the number of data packets received by the flow's sink over the duration of the simulation run. Each point in the graph depicts the average over all 50 simulation scenarios. Fig. 33 is based on the same data, just shown as the percentage in improvement of the proposed MRPRB and MRMR routing protocols relative to DSR.

Fig. 32 shows that both MRPRB and MRMR are able to consistently provide

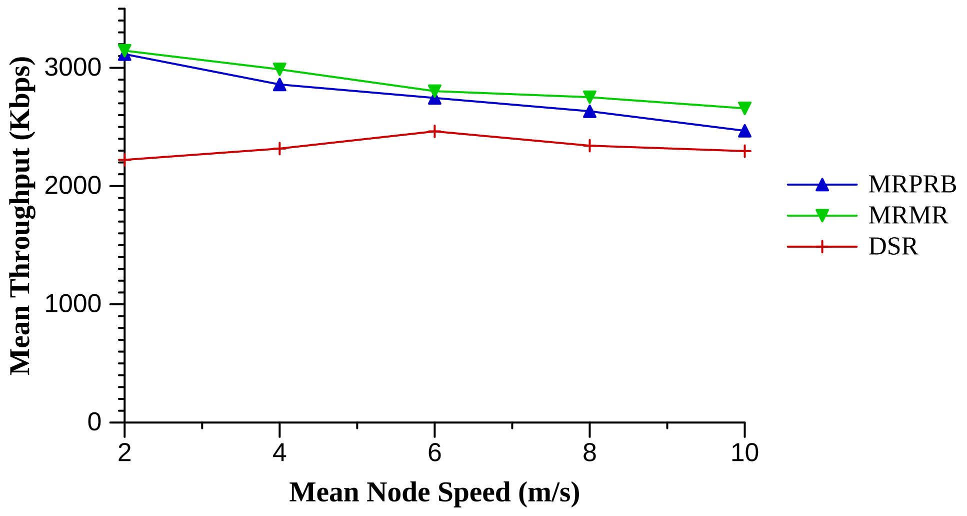


Fig. 32. *Performance comparison between Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols for a single CBR flow over a random mobile node topology and non-fading channel: average throughput.*

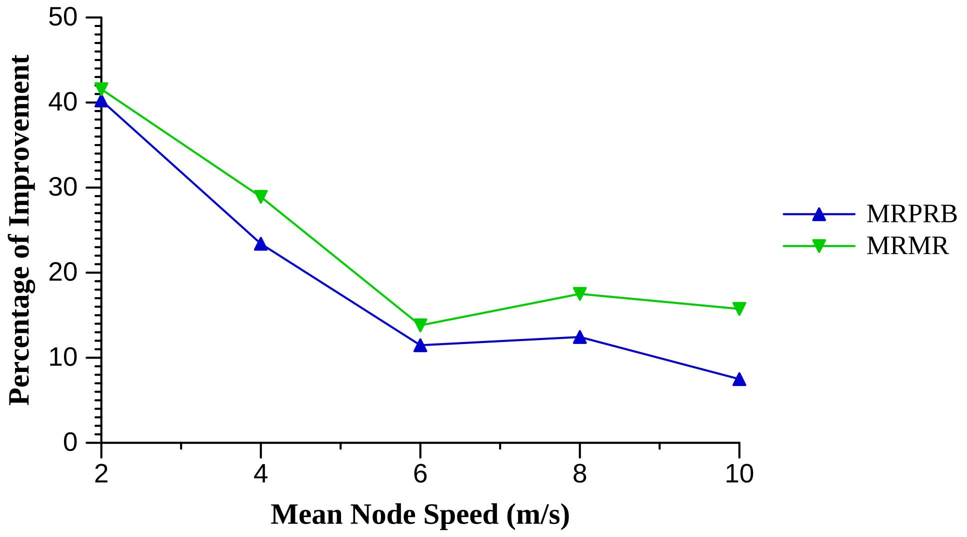


Fig. 33. *Performance comparison between Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols for a single CBR flow over a random mobile node topology and non-fading channel: percentage of improvement in average throughput over DSR.*

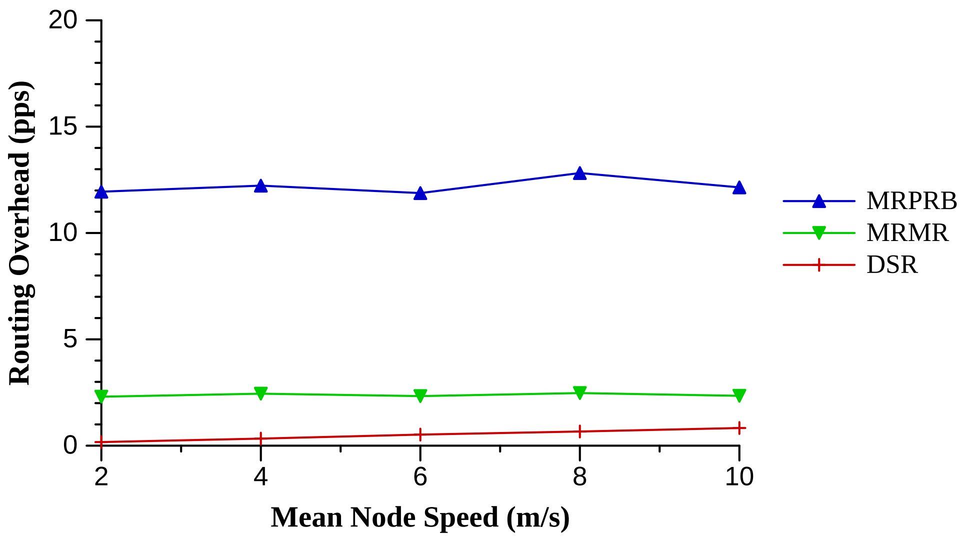


Fig. 34. *Comparison of routing overhead for Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols over a random mobile node topology and non-fading channel: low traffic load (1 CBR flow).*

higher throughput for the flow, with MRMR showing slightly better performance over all speeds. As the mean node speed increases, the rate at which links fail and the topology changes also increases so it is more difficult for MRPRB and MRMR to maintain routes. As a result, performance degrades with an increase in mean node speed. However, as seen in Fig. 33, both routing protocols are capable of providing consistent performance improvement over the full range of node speeds. Improvement was best at slower speeds, roughly 40% at 2 m/s for both protocols, but even at 10 m/s MRMR was able to maintain 15% improvement. MRPRB, on the other hand, drops below 10%. One explanation for MRPRB's poorer showing is it's relatively slow route discovery algorithm. Recall that MRPRB requires the transmission of one propagating route request at each supported data rate to discover one route. At higher speeds, links break more rapidly so this additional route discovery delay impacts its ability to provide quality routes.

Fig. 34 shows the routing overhead as the number of routing packets generated per second across the entire network. Notice that MRPRB generates much more routing traffic in the network. Nearly 3 times as much as MRMR, and 12 times as much as DSR. However, the overhead is relatively constant for both MRMR and MRPRB with speed. This is because both protocols are more proactive (than DSR) about purging routes from their route cache to prevent the build-up of "stale" routes (i.e., routes with broken links, or links with data rates that no longer match the cached route).

Figs. 35 and 36 show performance in a Rayleigh fading channel. In a fading channel, it is more difficult to guage the quality of a link because of the rapid and unpredictable changes in the signal strength. The impact of a fading channel on both protocols is significant, with MRMR dropping down from 40% to 10% improvement at 2 m/s. However, it is still able to provide improvement over the range of speeds. The impact on MRPRB is much more dramatic. Clearly, MRPRB is not well suited for use across fading channels, most likely because of its long route discovery delay. As seen in Fig. 37, all protocols generate more routing traffic than in a non-fading channel, and the traffic increases with speed. Interestingly, however, MRMR produces less traffic than DSR, which is a switch from the results for the non-fading channel. The cause is uncertain, but it may be due to the fact that MRMR favors routes with high-rate links, which naturally require higher SINR. Thus, while fading degrades the link quality (lowering the rate), it is less likely to cause a link failure than, say, a low-rate link with a much smaller SINR.

### 2. Random Topology, Five CBR Flows

In this section, we increase the number of traffic flows in the network to study how well the proposed routing protocols handle the additional routing load. As in the previous section, we start by discussing the simulation results for a non-fading channel.

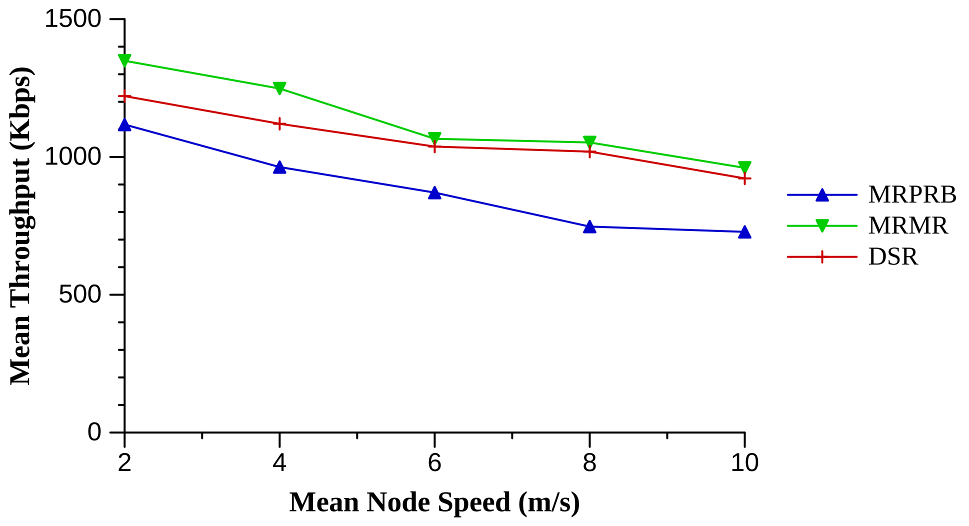


Fig. 35. *Performance comparison between Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols for a single CBR flow over a random mobile node topology and Rayleigh fading channel: average throughput.*

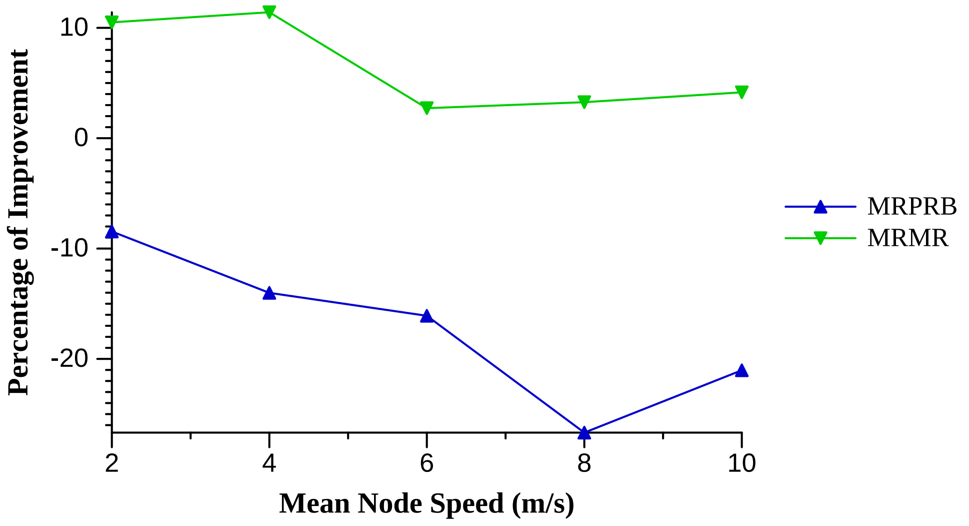


Fig. 36. *Performance comparison between Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols for a single CBR flow over a random mobile node topology and Rayleigh fading channel: percentage of improvement in average throughput over DSR.*

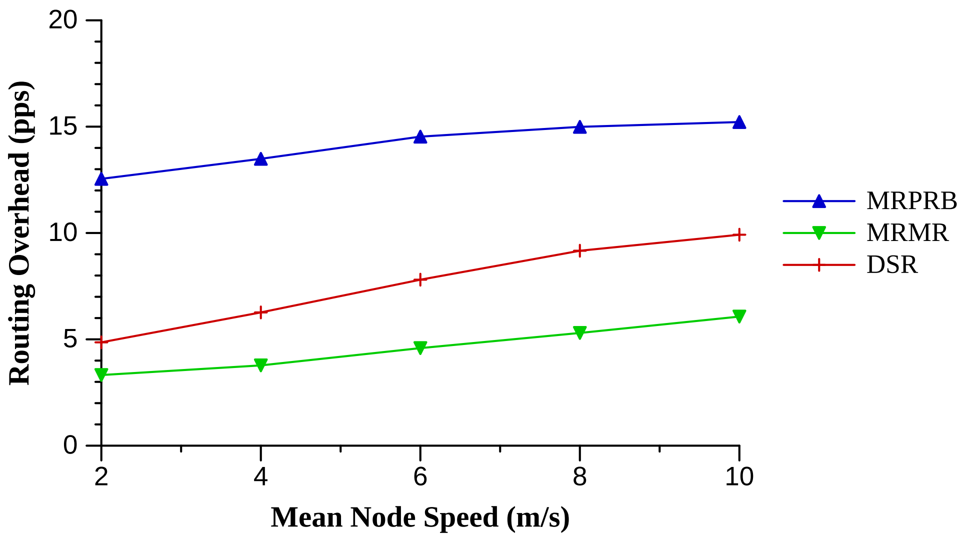


Fig. 37. *Comparison of routing overhead for Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols over a random mobile node topology and Rayleigh fading channel: low traffic load (1 CBR flow).*

Figs. 38 and 39 show the average aggregate throughput across all five CBR flows. As in the case for a single flow, both MRPRB and MRMR are able to show respectable performance improvement for the lower speeds, although not as high percentage-wise as for a single flow. The performance of MRPRB, however, is poor at higher speeds, most likely because of the increase in routing traffic, as seen in Fig. 40. MRMR, on the other hand, performs almost identically as for one flow.

Finally, we again look at performance in a Rayleigh fading channel, as shown in Figs. 41, 42, and 43. Interestingly, the performance is very similar to that of a single flow, although there is much higher routing traffic.

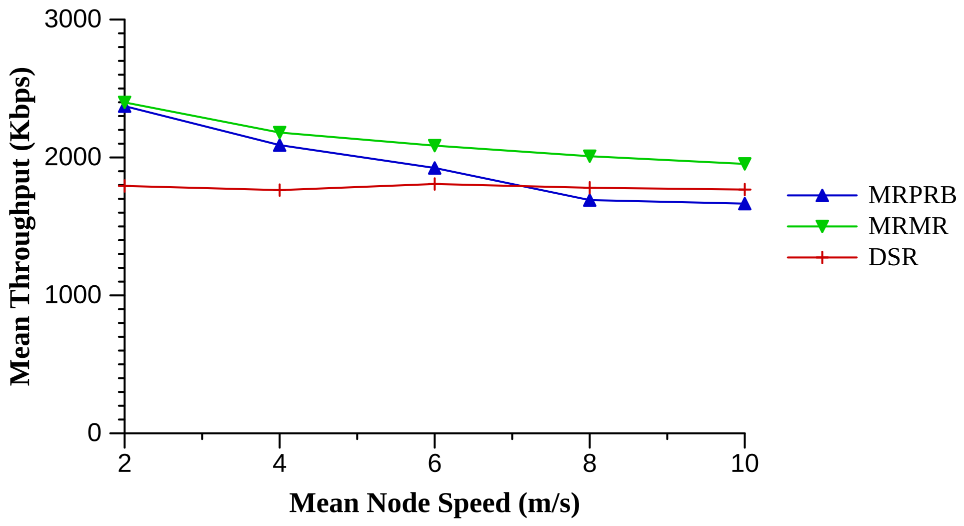


Fig. 38. *Performance comparison between Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols for 5 CBR flows over a random mobile node topology and non-fading channel: average throughput.*

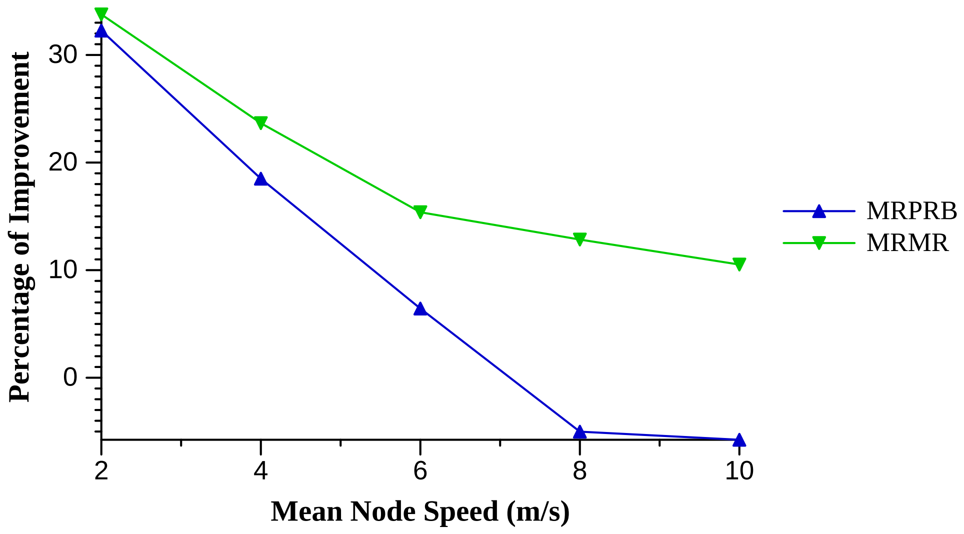


Fig. 39. *Performance comparison between Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols for 5 CBR flows over a random mobile node topology and non-fading channel: percentage of improvement in average throughput over DSR.*

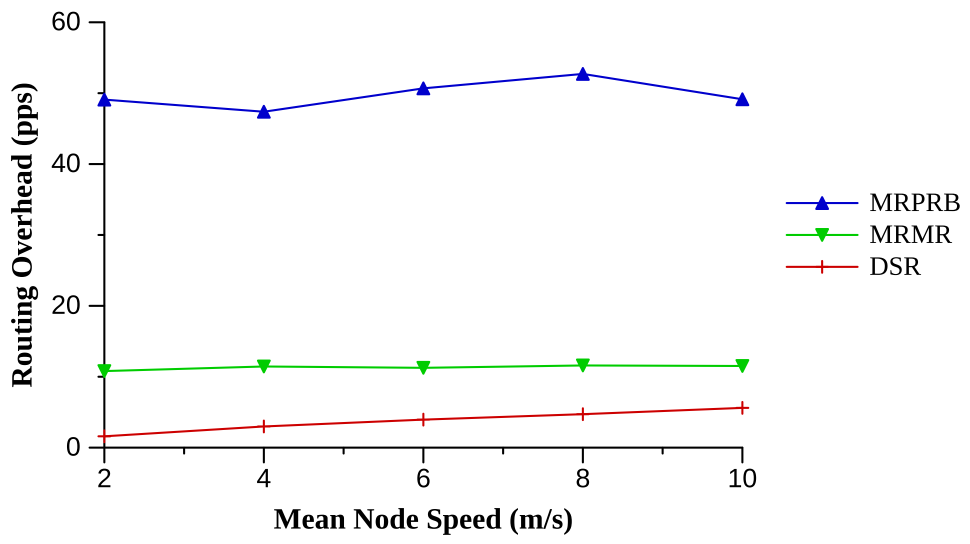


Fig. 40. *Comparison of routing overhead for Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols over a random mobile node topology and non-fading channel: moderate traffic load (5 CBR flows).*

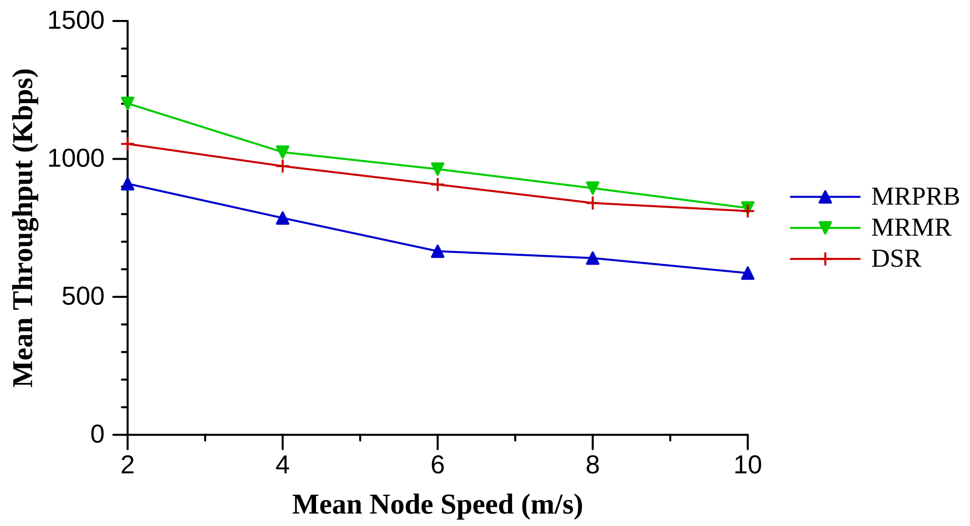


Fig. 41. *Performance comparison between Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols for 5 CBR flows over a random mobile node topology and Rayleigh fading channel: average throughput.*

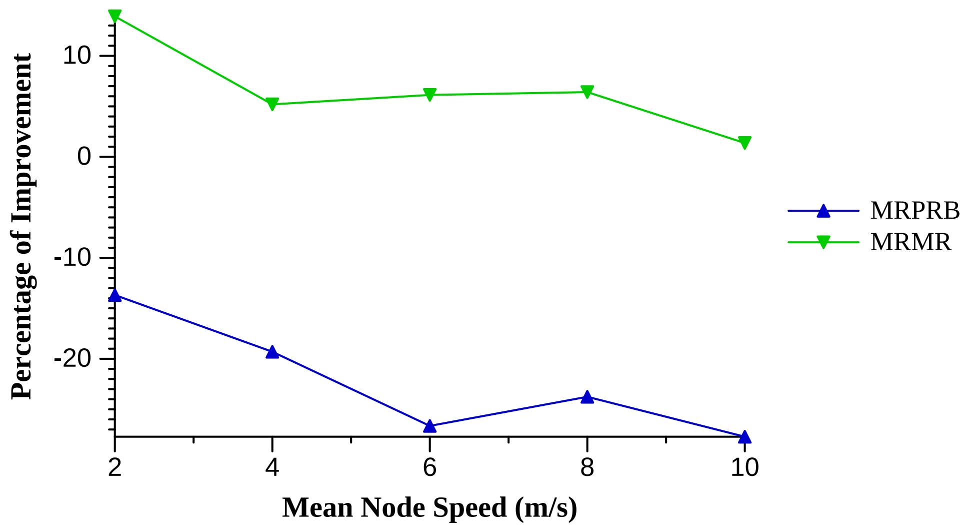


Fig. 42. *Performance comparison between Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols for 5 CBR flows over a random mobile node topology and Rayleigh fading channel: percentage of improvement in average throughput over DSR.*

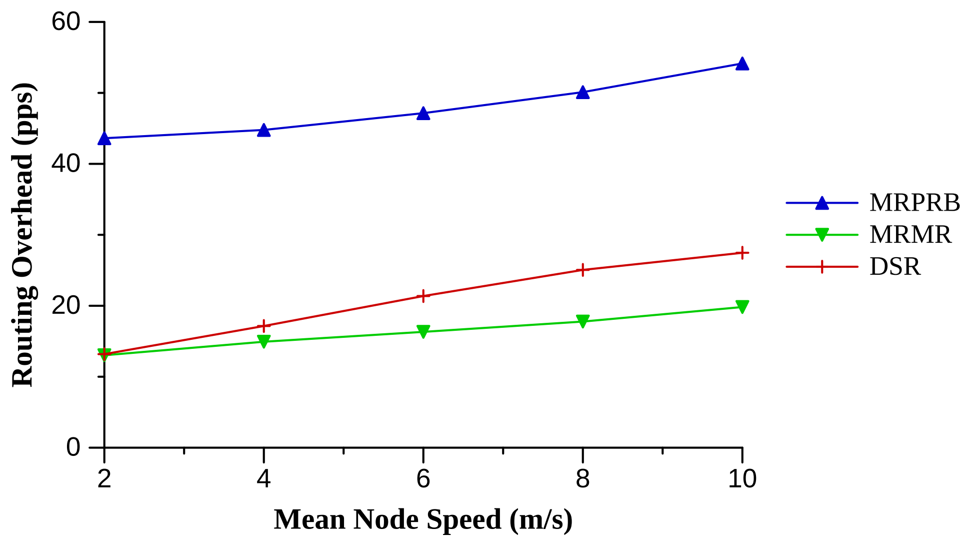


Fig. 43. *Comparison of routing overhead for Rate Probing (MRPRB), Minimum Rate (MRMR), and DSR routing protocols over a random mobile node topology and Rayleigh fading channel: moderate traffic load (5 CBR flows).*

## E. Summary

In this chapter, we analyzed the impact that multirate wireless links have on routing in multihop wireless networks. We then presented two routing protocols, Rate Probing (MRPRB) and Minimum Rate (MRMR), that are designed to discover and prioritize routes that maximize throughput. Through simulation analysis, we then showed that these two protocols can outperform a well-known conventional shortest-path routing protocol, DSR, in certain environments. Note that greater performance improvement may be possible than shown here for networks with a wider range of data rates such as those based on the 802.11g and 802.11a standards (6, 9, 12, 18, 24, 36, 48, and 54 Mbps). In the future, it is likely that advances in wireless technology will continue to increase the range of available data rates, making the proposed approaches even more attractive.

# CHAPTER VI

# EXPLICIT LINK FAILURE NOTIFICATION*

In this chapter, we address the problem of poor performance of TCP in mobile ad hoc networks. Specifically, we look at the impact of link failures due to mobility. We first present a performance analysis of standard TCP over mobile ad hoc networks, and then present an analysis of the use of explicit notification techniques to counter the affects of link failures.

## A. Simulation Environment and Methodology

The results in this chapter are based on simulations using the ns-2, as described in Chapter III.

All results are based on a network configuration consisting of TCP-Reno over IP on an 802.11 wireless network, with routing provided by the Dynamic Source Routing (DSR) protocol and BSD's ARP protocol (used to resolve IP addresses to MAC addresses). These are widely used and studied protocols, and are likely candidates for implementation in commercial ad hoc networks. Hence, understanding how they work together, and with TCP, can lead to modifications that improve performance.

Our network model consists of 30 nodes in a 1500x300 meter flat, rectangular area. The nodes move according to the *random waypoint* mobility model. In the random waypoint model, each node $x$ picks a random destination and speed in the rectangular area and then travels to the destination in a straight line. Once node $x$ arrives at its destination, it pauses, picks another destination, and continues onward.

*Part of this chapter is reprinted from "Analysis of TCP performance over mobile ad hoc networks" by G. D. Holland and N. H. Vaidya, 1999, in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Seattle, WA, pp. 219–230. © 1999 ACM, Inc.

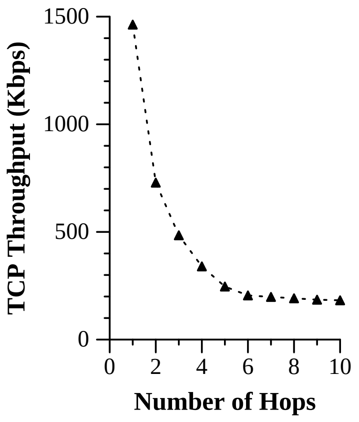
We used a pause time of 0 so that each node is in constant motion throughout the simulation. All nodes communicate with identical, half-duplex wireless radios that are modeled after the commercially available 802.11-based WaveLan wireless radios, which have a bandwidth of 2Mbps and a nominal transmission radius of 250m. TCP packet size was 1460 bytes, and the maximum window was eight packets.

Unless otherwise noted, all of our simulation results are based on the average throughput of 50 scenarios, or *patterns*. Each pattern, generated randomly, designates the initial placement and heading of each of the nodes over the simulated time. We use the same pattern for different mean speeds. Thus, for a given pattern at different speeds, the same sequence of movements (and link failures) occur. The speed of each node is uniformly distributed in an interval of $0.9v - 1.1v$ for some mean speed $v$. For example, consider one of the patterns, let's call it $I$. A node $x$ in $I$ that takes time $t$ to move from point $A$ to point $B$ in the 10 m/s run of $I$ will take time $t/2$ to traverse the same distance in the 20 m/s run of $I$. So, $x$ will always execute the exact same sequence of moves in $I$, just at a proportionally different rate. See [54] and [55] for more details on the mobility patterns.

## B. Performance Metric

In this performance study, we set up a single TCP-Reno connection between a chosen pair of *sender* and *receiver* nodes and measured the throughput over the lifetime of the connection. We use throughput as the performance metric in this chapter.

The TCP throughput is usually less than "optimal" 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



| *Hops* | *Throughput (Kbps)* |
|---|---|
| 1 | 1463.0 |
| 2 | 729.0 |
| 3 | 484.4 |
| 4 | 339.9 |
| 5 | 246.4 |
| 6 | 205.2 |
| 7 | 198.1 |
| 8 | 191.8 |
| 9 | 185.3 |
| 10 | 182.4 |

Fig. 44. *TCP-Reno throughput over an 802.11 fixed, linear, multi-hop network of varying length (in hops).*

window and, in the instance of a timeout, backing-off its retransmission timeout (RTO). Therefore, route changes due to host mobility can have a detrimental impact on TCP performance.

To gauge the impact of route changes on TCP performance, we derived an upper bound on TCP throughput, called the **expected throughput**. The TCP throughput measure obtained by simulation is then compared with the expected throughput.

We obtained the *expected throughput* as follows. We first simulated a *static* (fixed) network of $n$ nodes that formed a linear chain containing $n - 1$ wireless hops (similar to the "string" topology in [44]). The nodes used the 802.11 MAC protocol for medium access. Then, a one-way TCP data transfer was performed between the two nodes at the ends of the linear chain, and the TCP throughput was measured between these nodes. This set of TCP throughput measurements is analogous to that performed by Gerla et al. [44], using similar (but not identical) MAC protocols.

Fig. 44 presents the measured TCP throughput as a function of the number of hops, averaged over ten runs. Observe that the throughput decreases rapidly when the number of hops is increased from 1, and then stabilizes once the number of hops

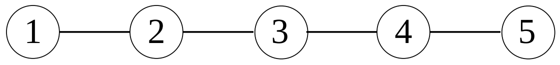


Fig. 45. *A simple multi-hop network.*

becomes large. The primary reason for this trend is due to the characteristics of 802.11. Consider the simple four hop network shown in Fig. 45. In 802.11, when link 1–2 is active only link 4–5 may also be active. Link 2–3 cannot be active because node 2 cannot transmit and receive simultaneously, and link 3–4 may not be active because communication by node 3 may interfere with node 2. Thus, throughput on an $i$ hop 802.11 network with link capacity $C$ is bounded by $C/i$ for $1 \le i \le 3$, and $C/3$ otherwise. The decline in Fig. 44 for $i \ge 4$ is due to contention caused by the backward flow of TCP ACKs. For further explanation of this trend, we refer the reader to [44]. Our objective here is only to use these measurements to determine the *expected throughput*.

The *expected throughput* is a function of the mobility pattern. For instance, if two nodes are always adjacent and move together (similar to two passengers in a car), the expected throughput for the TCP connection between them would be identical to that for 1 hop in Fig. 44. On the other hand, if the two nodes are always in different partitions of the network, the expected throughput is 0. In general, to calculate the expected throughput, let $t_i$ be the duration for which the shortest path from the sender to receiver contains $i$ hops ($1 \le i \le \infty$). Let $T_i$ denote the throughput obtained over a linear chain using $i$ hops. When the two nodes are partitioned, we consider that the number of hops $i$ is $\infty$ and $T_\infty = 0$. The expected throughput is then calculated as

$$\textit{expected throughput} = \frac{\sum_{i=1}^{\infty} t_i * T_i}{\sum_{i=1}^{\infty} t_i} \tag{6.1}$$

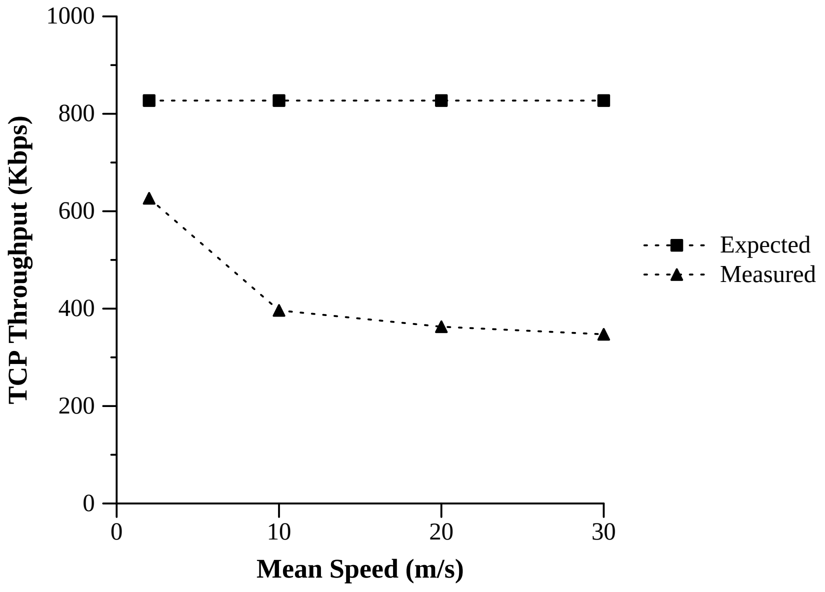
Of course, $\sum_{i=1}^{\infty} t_i$ is equal to the duration for which the TCP connection is in existence. The measured throughput may never become equal to the expected throughput, for a number of reasons. For instance, the underlying routing protocol may not use the shortest path between the sender and receiver. Also, Equation 6.1 does not take into account the performance overhead of determining new routes after a route failure. Despite these limitations, the expected throughput serves as a reasonable upper bound with which the measured performance may be compared. Such a comparison provides an estimate of the performance degradation caused by host mobility in ad hoc networks.

## C. Measurement of TCP-Reno Throughput

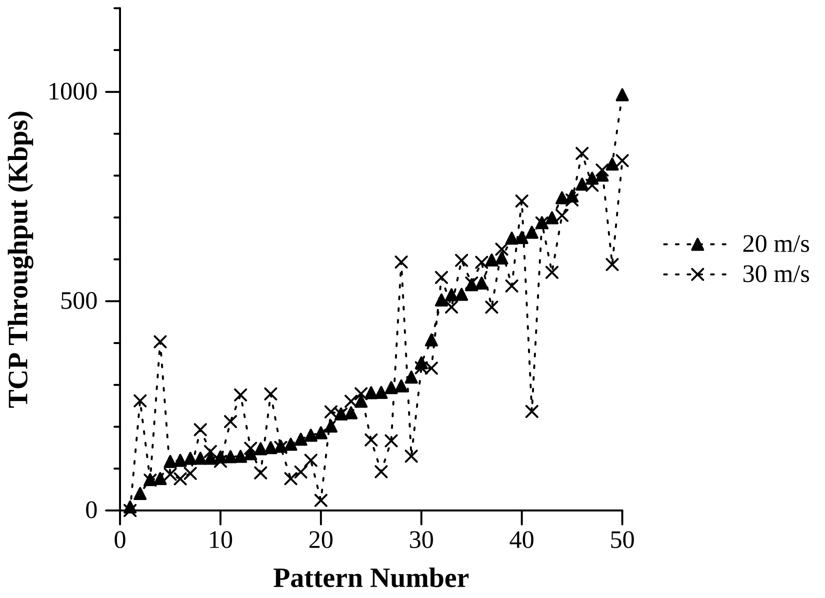
Fig. 46(a) reports the measured TCP-Reno throughput and the expected throughput as a function of the mean speed of movement.

Note that the expected throughput is independent of the speed of movement. In Equation 6.1, when the speed is increased, the values of $t_i$ for all $i$ becomes smaller, but the ratio $t_i/t_j$ for any $i$ and $j$ remains the same. Therefore, the expected throughput for a given mobility pattern, calculated using Equation 6.1, is independent of the speed.

Intuition suggests that when the speed is increased then route failures happen more quickly, resulting in packet losses, and frequent route discoveries. Thus, intuitively, TCP throughput should monotonically degrade as the speed is increased. In Fig. 46(a), the throughput drops sharply as the mean speed is increased from 2 m/s to 10 m/s. However, when the mean speed is increased from 10 m/s to 20 m/s and 30 m/s, the throughput averaged over the 50 runs decreases only slightly. This is a counter-intuitive result. The reason can be attributed, in part, to the network



(a) *Measured and expected throughput, averaged over 50 mobility patterns.*



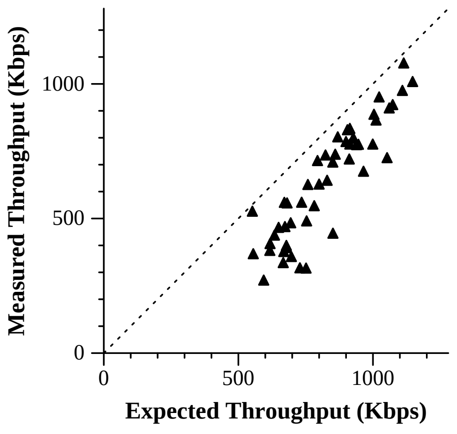
(b) *Per-pattern measured throughput for the 20 m/s and 30 m/s points in (a).*

Fig. 46. *Throughput for a single TCP-Reno connection over a mobile ad hoc network.*

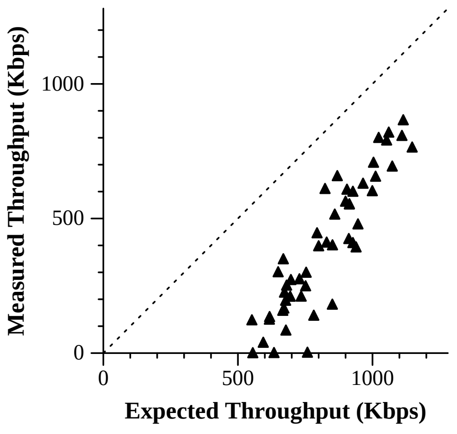
layer's problems maintaining routes for paths longer than a few hops at the higher node speeds. Thus, beyond a certain speed, throughput was commonly achieved only in the situation where the sender and receiver were within a few hops of each other. Another contributing factor to this result is that, under certain circumstances, throughput could potentially *increase* with speed. Consider, for example, Fig. 46(b), which plots the throughput for each of the 50 mobility patterns for the 20 m/s and 30 m/s mean speeds used in our simulations (the patterns are sorted, in this figure, in the order of their throughputs at 20 m/s). Observe that, for certain mobility patterns, the throughput increases when the speed is increased. This can happen, for instance, when fortuitous timing of TCP and MAC retransmissions, with regard to the state of the network (e.g., the position of the nodes in the network), results in the re-establishment of the packet flow at the higher speed but not at the slower speed. Section D discusses this anomaly in more detail.

Fig. 47 provides a different view of the TCP throughput measurements. In this figure, we plot the measured throughput versus expected throughput for each of the 50 mobility patterns. The four graphs correspond to each of the four different mean speeds of movement. Because the expected throughput is an upper bound, all the points plotted in these graphs are below the diagonal line (of slope 1). When the measured throughput is closer to the expected throughput, the corresponding point in the graph is closer to the diagonal line, and vice versa. The following observations can be made from Fig. 47:

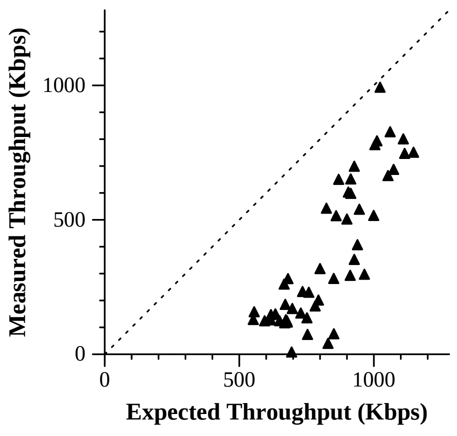
- Although, for any given speed, the points may be located near or far from the diagonal line, when the speed is increased the points tend to move away from the diagonal, signifying a degradation in throughput. Later in this chapter, we show that, using a TCP optimization, the cluster of points in this figure can be



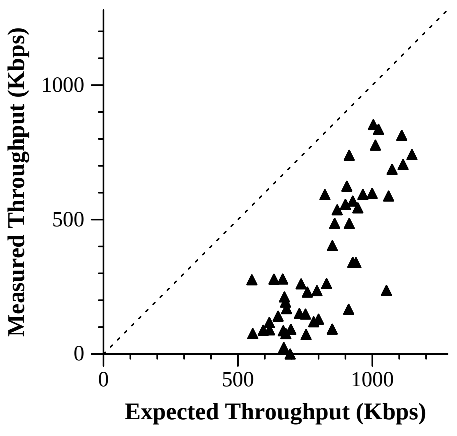
(a) *Speed = 2 m/s*



(b) *Speed = 10 m/s*



(c) *Speed = 20 m/s*



(d) *Speed = 30 m/s*

Fig. 47. *Comparison of measured and expected throughput for the 50 mobility patterns*

brought closer to the diagonal.

- On the other hand, for a given speed, certain mobility patterns achieve throughput close to 0, although other mobility patterns (with the same mean speed) are able to achieve a higher throughput.
- Even at high speeds, some mobility patterns result in high throughput that is close to the expected throughput (for instance, see the points close to the diagonal line in Figs. 47(c) and (d)). This occurs for mobility patterns in which, despite moving fast, the rate of link failures is low (as discussed earlier, if two nodes move together, the link between them will not break, regardless of their speed).

Section D provides explanations for some observations made based on the data presented in Figs. 46 and 47.

## D. Mobility Induced Behaviors

In this section, we look at examples of mobility induced behaviors that result in unexpected performance. The measured throughput of the TCP connection is a function of the interaction between the 802.11 MAC protocol, the ARP protocol, the DSR routing protocol, and TCP's congestion control mechanisms. As such, there are likely to be several plausible explanations for any given observation. Here, for each observation, we give one such explanation that we have been able to confirm using the measured data.

### 1. Some Mobility Patterns Yield Very Low Throughput

We present one observed scenario wherein loss of some TCP data and acknowledgment packets (due to route failures) results in zero throughput. Note that we measure

Table IV. *Packet trace for a 30 m/s run that experienced zero throughput.*

| *Evnt* | *Time (secs)* | *Node* | *SeqNo* | *Pkt* | *Resn* |
|---|---|---|---|---|---|
| s | 0.000 | 1 | 1 | tcp | |
| D | 0.191 | 5 | 1 | tcp | NRTE |
| s | 6.000 | 1 | 1 | tcp | |
| r | 6.045 | 2 | 1 | tcp | |
| s | 6.145 | 2 | 1 | ack | |
| D | 6.216 | 21 | 1 | ack | NRTE |
| s | 18.000 | 1 | 1 | tcp | |
| s | 42.000 | 1 | 1 | tcp | |
| s | 90.000 | 1 | 1 | tcp | |
| D | 120.000 | 15 | 1 | tcp | END |
| D | 120.000 | 16 | 1 | tcp | END |
| D | 120.000 | 25 | 1 | tcp | END |

throughput as a function of the amount of data that has been *acknowledged* to the sender. In the example scenario discussed here, no acknowledgments are received by the sender during the 120 second lifetime of the TCP connection (the average speed for this case is 30 m/s). However, the *expected throughput* for the mobility pattern in this run is 694Kbps. A path exists between the TCP sender and receiver nearly the entire time.

A condensed version of the simulation packet trace is shown in Table IV. This trace was obtained with node 1 as the TCP sender and node 2 as the TCP receiver. In the table, the *Evnt* column lists the event type – *s* denotes that a packet is sent, *r* denotes that a packet is received, and *D* denotes that a packet is dropped. The *Resn* column lists the reason why a packet is dropped – *NRTE* means that the routing protocol could not find a route and *END* means the simulation finished. The *Node*, *SeqNo*, and *Pkt* columns report the node at which the event occurred, the TCP sequence number[1] of the packet depicted in the event, and the type of packet,

[1]These are sequence numbers assigned by *ns-2* to TCP packets. *ns-2* does not number each octet individually; instead, the packets are numbered sequentially as 1, 2, etc. All references to TCP sequence

respectively.

In this scenario, the sender and the receiver nodes are initially six hops apart and stay within six hops of each other for all but 6 seconds of the 120 second simulation. For 6 seconds, the network is partitioned, with the sender and receiver nodes being in different partitions.

Soon after the first packet is sent by node 1, a link break occurs along the route that causes a partition in the network. The partition causes the first packet to be dropped (at time 0.191 seconds) by the routing protocol on node 5, which was the forwarding node that detected the link failure. Eventually, the TCP sender on node 1 times-out and retransmits the packet (at time 6.000). On the second attempt, the packet reaches the receiver, node 2, who sends a delayed acknowledgment (at time 6.145). However, the acknowledgment is sent on a route from node 2's cache that is stale (i.e., some links on the route are broken), so the acknowledgment is later dropped (at time 6.216). The remaining attempts to retransmit the packet also fail because of stale cached routes. In each instance, the packet is held by the ARP layer of a forwarding node until the end of the simulation (see the rows with *Evnt* = D and *Resn* = *END* in Table IV). Each ARP layer is left holding a packet because its attempts to resolve the IP address of the next node in the route to a MAC address fail because of mobility.

Therefore, the TCP sender is unable to receive any acknowledgment from the receiver.

---

numbers in this chapter are the *ns-2* assigned sequence numbers.

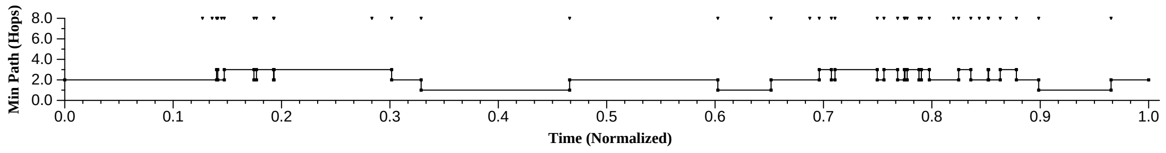
### 2. Anomaly: Throughput Increases as Speed Increases

In the example discussed in this section, TCP throughput improves by a factor of 1.5 when the speed is increased from 10 m/s to 20 m/s. In the scenario under consideration, the TCP sender and receiver were able to reach each other 100% of the time, and spent 74% of the time at most two hops away. The nodes were never more than three hops away.

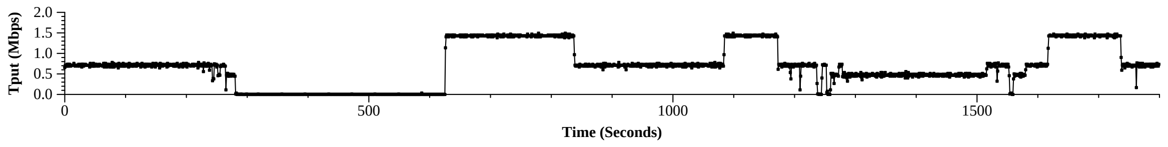
The characteristics of the connection between the TCP sender and receiver can be seen in the mobility pattern profile shown in Fig. 48(a) (see [54] for similar details on all of the patterns). The ticks shown at the top of the profile mark the points in the pattern at which the minimum path between the TCP sender and receiver changed. The curve shows the minimum path length (distance) in hops between the sender and receiver for the duration of the pattern. Notice that a change in the minimum path is not always accompanied by a change in path length (e.g., at the 0.28 mark in Fig. 48(a)), because the nodes on the path may change even though the total number of hops stays the same.

The other curves in Fig. 48 show the mean throughput over the TCP connection (averaged over 1 second) for each of the four mean node speeds. Note that, as mentioned in Section A, the sequence of moves that each node makes is identical, regardless of the mean speed.

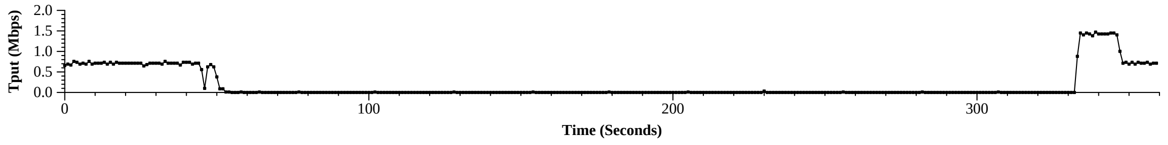
The only difference is that a distance covered by a node, say $x$, over time $t$, such as in figure $(b)$, takes $x$ a time of $t/2$ to cover in figure $(c)$. This is analogous to a movie in which the time taken to show the same number of frames at rate $r$ takes half the time to show at rate $2r$. Thus, the mobility pattern profile shown in $(a)$ can be used as a reference point for the other curves in Fig. 48. Note that the variations in the throughput for curves $(b) - (e)$ are correlated to the path length in $(a)$ because



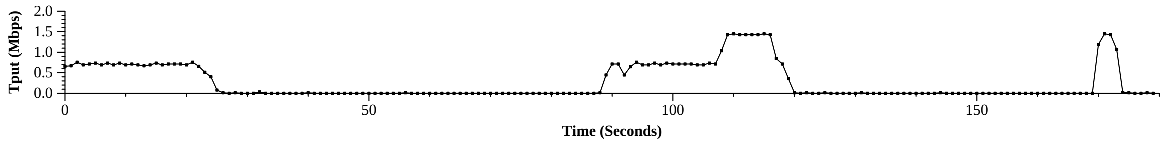
(a) *Mobility pattern profile.*



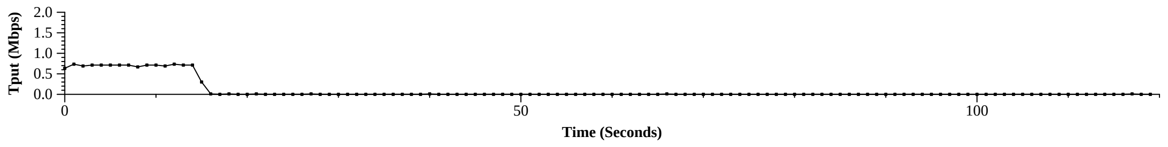
(b) *Mean speed = 2 m/s*



(c) *Mean speed = 10 m/s*



(d) *Mean speed = 20 m/s*



(e) *Mean speed = 30 m/s*

Fig. 48. *TCP-Reno performance for mobility pattern No. 20, demonstrating that an increase in mean node speed may result in an increase in mean throughput. The ticks at the top of (a) denote changes on the minimum path between the TCP sender and receiver. The curves in (b) – (e) show the measured throughput for the connection, averaged over 1 second intervals.*

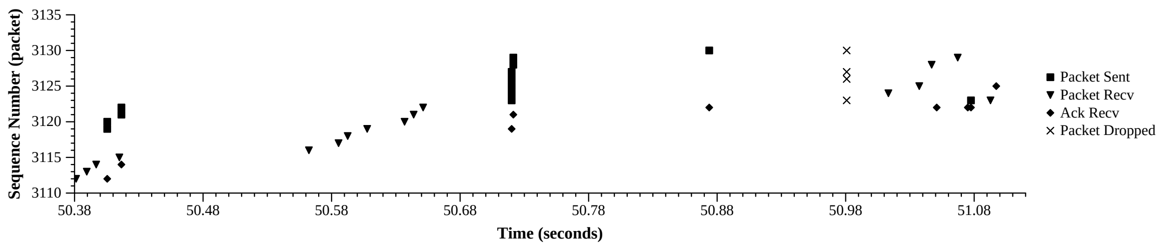


Fig. 49. *Detailed packet plot showing the beginning moments, around the 50s mark in Fig. 48(c), at which a sequence of path changes, shown in Fig. 48(a), causes TCP to repeatedly timeout and back-off.* Packet Sent *and* Packet Recv *indicate the time at which a TCP data packet with the indicated* ns-2 *sequence number was sent by the sender and arrived at the receiver, respectively,* Ack Recv *indicates the time at which a TCP acknowledgment was received by the sender with the indicated sequence number, and* Packet Dropped *indicates the time at which a data packet with the indicated sequence number was dropped.*

of the effect shown in Fig. 44, which we discussed earlier.

**Discussion of Fig. 48(c):** In the 10 m/s run, the routing protocol uses symmetric forward and reverse routes (of optimal length) between the TCP sender and receiver for the first 50s of the simulation, resulting in good initial throughput. However, the sequence of path changes around the 50s mark causes the TCP sender to back-off, from which it fails to recover, until the final 30s of the simulation. The details of the packet activity around the moment at which the initial back-off occurs is shown in Fig. 49. Leading up to the failure, the forward and reverse routes are symmetric and optimal in length (two hops). Around the 50.4s mark, the route breaks (because of mobility) at the link between the intermediate node and the TCP receiver. This results in the queuing of nearly a full window of packets at the intermediate node. The intermediate node salvages the queued packets, then successfully delivers them to the receiver on a new forward route (seen around the 50.58s mark). After detecting the failed link, the receiver chooses a new reverse route for sending acknowledgments, which is different than the forward route. However, the reverse route that it chooses is also stale, so several acknowledgments are lost before salvaging results in the arrival of two of the acknowledgments at the TCP sender around the 50.72s mark. These acknowledgments trigger a burst of packets from the sender, which are immediately queued by the forwarding node at the next hop in the path, because, although the reverse route is good, the forward route is now broken by mobility. Another acknowledgment arrives later (around the 50.87s mark), resulting in the queuing of another packet. Meanwhile, the forwarding node, which now has the full window queued, repeatedly tries to salvage the packets. This finally results in the loss of half of the packets (around the 50.98s mark) by ARP, which fails to determine the MAC address of the node over the next hop in the salvaged route because the node has moved away. However, half of the packets are successfully salvaged on an alternate route and delivered (seen between the 51.0s and 51.08s marks), generating a sequence of duplicate acknowledgments (*dupacks*) from the receiver signifying the packet loss. After the third and fourth dupacks arrive, the TCP sender enters fast recovery and retransmits the lost packet (at the 51.08s mark), but the lost packets cause the sender to timeout. The retransmission of the lost packet by the sender results in a brief burst of packets (seen as a small bump at the 51s mark), but the routes break quickly thereafter, as the path changes from two to three hops, resulting in lost packets that cause the sender to timeout again.

For all subsequent timeouts, except one, stale routes result in packet losses even though the TCP sender and receiver are never more than three hops distance from each other. The one exception occurs around the 333s mark, at which time a retransmitted packet results in the re-establishment of packet flow when the nodes are one hop away.

**Discussion of Fig. 48(d):** The 20 m/s run shares many of the characteristics of the slower 10 m/s run, but results in higher throughput because a retransmission late in the pattern (around the 90s mark) succeeds in briefly re-establishing the flow of packets. Initially, the data flow is quickly stalled (around the 25s mark) because of the loss of a full window of packets, which is caused by the same sequence of link changes in the pattern that affected the 10 m/s run. The throughput, again, degrades when repeated route failures induce packet losses, causing the TCP sender to timeout and back-off. However, unlike the 10 m/s run, the packet flow is re-established later in the pattern (at the 88s mark) when a retransmitted packet results in the discovery of a good route when the nodes are only two hops apart. This success is why the 20 m/s run is able to transfer data at 1.5 times the rate of the 10 m/s run, for the same mobility pattern.

## 3. Summary and Observations

In this section, we present a summary of the effects of mobility on TCP performance that we observed in the previous examples and in our other experiments.

From the previous examples, it is clear that the characteristics of the routing protocol have a very significant impact on TCP performance. Most notable were the problems caused by the caching and propagation of stale routes. Even in relatively slowly changing topologies, the inability of the TCP sender's routing protocol to quickly recognize and purge stale routes from its cache resulted in repeated routing failures. Allowing intermediate nodes to reply to route requests with routes from their caches complicated this problem, because they often responded with stale routes. This was further amplified by the fact that other nodes could overhear or snoop the stale routes in the replies as they were propagated, spreading the bad information to caches in other nodes. We saw the effects of this problem in our simulations. For instance,

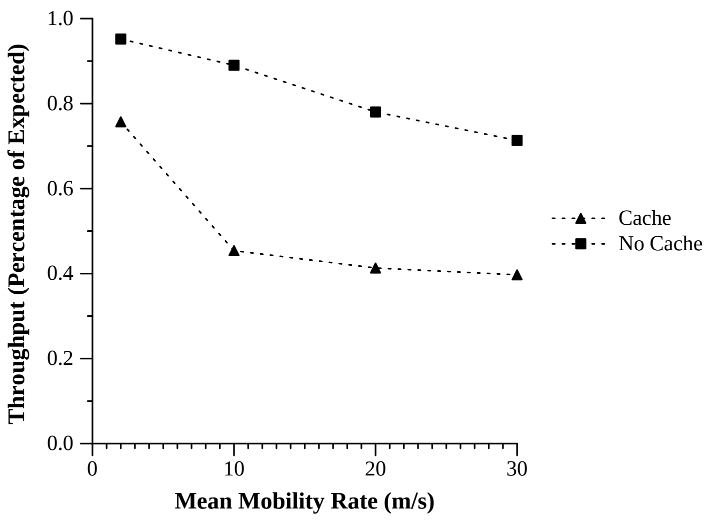


Fig. 50. *A comparison of TCP-Reno performance when DSR route replies from caches are, and are not, allowed.*

in the simulation run presented in our first example (Section 1), the TCP sender tried to use the same stale route three times because it received the route repeatedly from other nodes. In the latter two tries, the stale route came to the TCP sender by way of salvaging. The stale route that was used was a two hop route between the TCP sender and receiver. In each of the two instances, a neighboring node salvaged a packet from the TCP sender using the stale route, which the node had stored in its route cache. The neighboring node then sent the packet on the next hop in the salvaged route, back to the TCP sender. The result was that the TCP sender ended up trying to forward its own packet on a route that it had earlier determined was stale. However, we believe that these problems can potentially be solved using more effective cache maintenance strategies, including simple techniques like dynamically adjusting the route cache timeout mechanism depending on the observed route failure rate, the use of negative route information (mentioned in [70]), or the use of signal strength information.

Alternatively, replying from caches can be turned off altogether. This has a

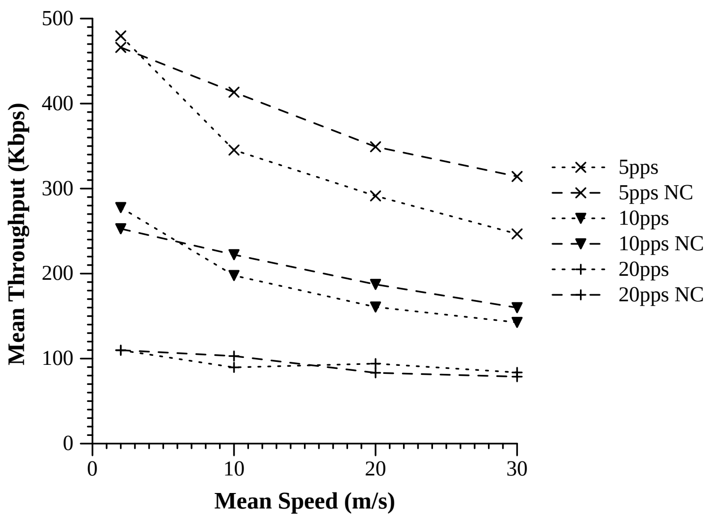


Fig. 51. *A comparison of TCP-Reno performance when DSR route replies from caches are, and are not (NC), allowed, and additional traffic is in the network: 10 CBR connections, each sending 5, 10, and 20 packets per second.*

startling improvement in performance, as shown in Fig. 50. However, these results are for a single TCP connection in a network with no other data traffic. In a network with multiple data sources, the additional routing traffic introduced when replies from caches are not used could degrade performance. Therefore, we simulated the same TCP connection in a network containing multiple CBR data sources to gauge the impact of the additional routing overhead when replies from caches are not allowed. The results for TCP throughput are shown in Fig. 51. We looked at three different levels of network traffic using ten CBR connections across eight nodes (not including the TCP sender or receiver), each sending 512 byte packets at mean rates of 5, 10, and 20 packets per second (pps). Start times were staggered. For low mobility in the presence of CBR traffic, disallowing route replies from caches results in slightly lower TCP performance than when route replies from caches are allowed. This is due, in part, to the impact of the additional routing overhead. For moderate to high mobility, however, we see a consistent improvement in performance, which is

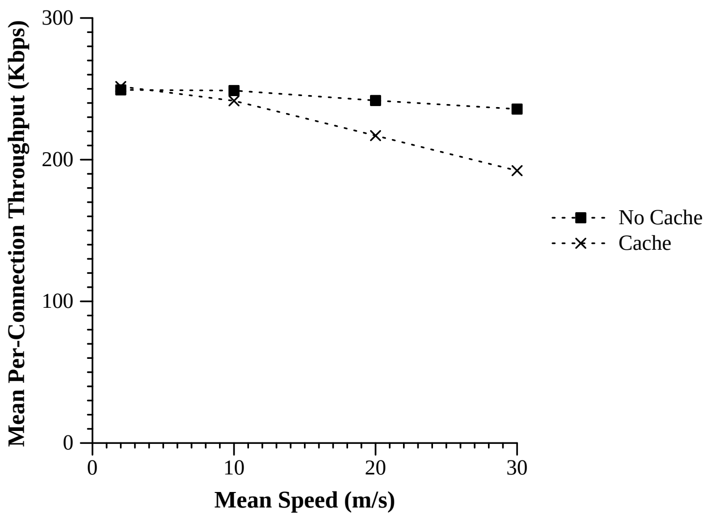


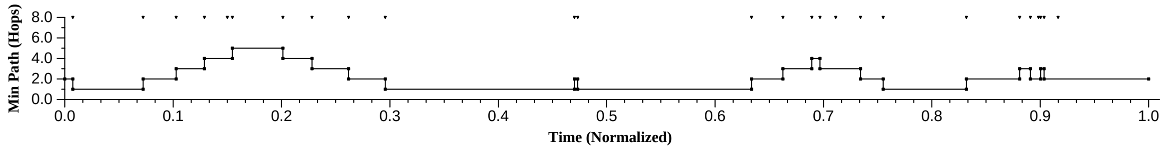
Fig. 52. *A comparison of TCP-Reno performance of five TCP connections when DSR route replies from cache are, and are not, allowed.*

clearly evident in the 5 pps and 10 pps curves. We observed that this improvement occurred because the steady traffic provided by the CBR connections increased the accuracy of cached routes by steadily exercising routes in the network, facilitating quick detection of broken links and, summarily, the purging of stale routes. However, for increasing levels of traffic the performance improvement decreases due to the additional routing traffic, until, for the 20 pps curve, disallowing route replies results in worse performance at 5 and 10 m/s, and only slight improvement at 20 and 30 m/s.

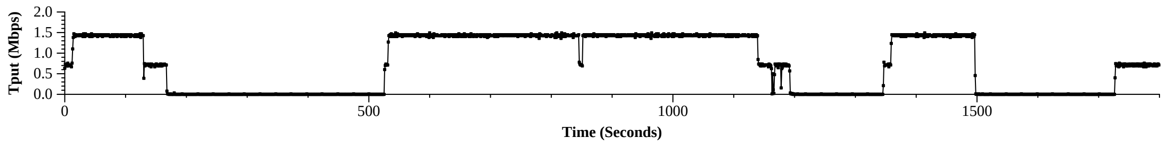
We also looked at a scenario where multiple TCP connections share the network. This data is presented in Fig. 52, which shows the mean throughput over five TCP connections (each between separate pairs of nodes) for 30 patterns. Here, also, we see the same trend as in the previous figure.

Another interesting effect of a routing protocol's behavior with respect to mobility was observed in our second example (Section 2). The fact that the TCP data flow was lost at the same point in the mobility pattern for both runs raised questions about what characteristic of the pattern was causing the failure. From Fig. 48(a), it is clear that the rapid sequence of path changes at the 0.13 mark caused all four runs to fail. Upon further inspection, we observed that the routing protocol regularly failed when the minimum path increased in length. This is apparent in the results shown in Fig. 53.

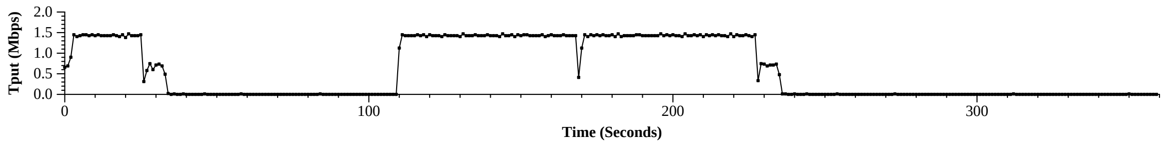
In the first few moments of the mobility pattern, shown in Fig. 53(a), the TCP sender and receiver move closer to each other, shortening the path between them from two hops to one (around mark 0.01). A few moments later (around mark 0.07), they slowly diverge to a distance of five hops. In the TCP throughput measurements shown in $(b) - (e)$, it is evident that the data flow across the TCP connection is maintained when the path is shortened, but is lost when the path is lengthened. This happens several times in the pattern, independent of the mean speed of the nodes. Most notably, $(b)$ shows that even while traveling at a slow speed of 2 m/s, a path change from one hop to two (around the 1500s mark) can stall the data flow. This behavior can be attributed, in part, to the routing protocol. As the TCP sender and receiver move closer to each other, DSR can often maintain a valid route by shortening the existing route, and often does so before a failure occurs. However, as the TCP sender and receiver diverge, the increase in path length eventually causes a route failure because DSR does not attempt to lengthen a route until a failure occurs. The route failure and subsequent route discovery process often result in the restoration of the route only after the TCP sender has repeatedly timed-out and backed-off, stalling the data flow. This is further magnified by the caching and propagation of stale routes, as mentioned previously. However, intuition suggests that this is not a problem that is unique to DSR, but will most likely be a problem for other reactive protocols as well. Thus, perhaps a metric of routing protocol performance should not only measure the protocol's ability to recognize optimal routes, but also to quickly adjust an existing



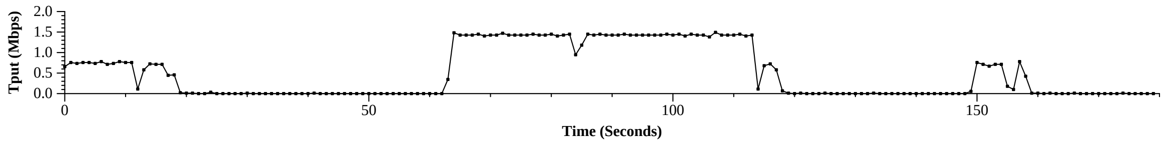
(a) *Mobility pattern profile.*



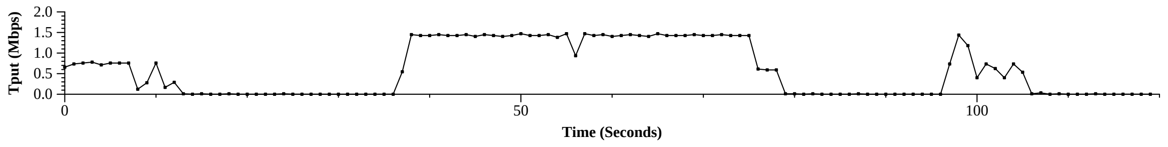
(b) *Mean speed = 2 m/s*



(c) *Mean speed = 10 m/s*



(d) *Mean speed = 20 m/s*



(e) *Mean speed = 30 m/s*

Fig. 53. *TCP-Reno performance for mobility pattern No. 46, showing that an increase in the minimum path length between the TCP sender and receiver consistently results in the loss of data flow across the connection. The ticks at the top of (a) denote changes on the minimum path between the TCP sender and receiver. The curves in (b) – (e) show the measured throughput for the connection, averaged over 1 second intervals.*

route, albeit non-optimally.

Another characteristic of DSR that we observed affecting TCP performance was the route request retransmission back-off algorithm. In DSR, if a route request does not generate a timely reply, the requester times-out and retransmits the request. Each timeout results in exponential back-off, which is limited to some fixed maximum value. If this value is too large, then route requests may occur too infrequently to recognize available routes in time to prevent TCP's retransmission timer from backing-off to a large value, but if it is too small, then the frequent route requests may cause network congestion. The maximum value suggested in [70] may not be suitable for good TCP performance.

Based on these observations, it might be suggested that instead of augmenting TCP/IP, it would be better to improve the routing protocols so that mobility is more effectively masked. Clearly, extensive modifications to upper layer protocols is less desirable than a routing protocol that can react quickly and efficiently such that TCP is not disturbed. However, regardless of the efficiency and accuracy of the routing protocol, network partitioning and delays will still occur because of mobility, which cannot be hidden. Thus, in the next section, we analyze some simple modifications to TCP/IP to provide TCP with a mechanism by which it can recognize when mobility induced delays and losses occur, so that it can take appropriate actions to prevent the invocation of congestion control.

## E. TCP Performance Using Explicit Feedback

In this section, we present an analysis of the use of explicit feedback on the performance of TCP in dynamic networks. The use of explicit feedback is not new, and has been proposed as a technique for signaling congestion (e.g., ECN [40], FECN/BECN [87], DECBit [98]), corruption due to wireless transmission errors (e.g., EBSN [9], ELN [12]), and link failures due to mobility (e.g., [20], SCPS-TP [25, 34], TCP-F [21]). Our interest in this section is analyzing the performance of an explicit signaling mechanism, which we refer to as Explicit Link Failure Notification (ELFN) [57], [53]. Although the TCP-F paper studies a similar idea, the evaluation is not based on an ad hoc network. Instead, they use a black-box, that does not include the evaluation of the routing protocol.

The objective of ELFN is to provide the TCP sender with information about link and route failures so that it can avoid responding to the failures as if congestion occurred.

There are several different ways in which the ELFN message can be implemented. A simple method would be to use a "host unreachable" ICMP message as a notice to the TCP sender. Alternatively, if the routing protocol already sends a route failure message to the sender, then the notice can be piggy-backed on it. This is the approach we took in this analysis. We modified DSR's route failure message to carry a payload similar to the "host unreachable" ICMP message. In particular, it carries pertinent fields from the TCP/IP headers of the packet that instigated the notice, including the sender and receiver addresses and ports, and the TCP sequence number. The addresses are used to identify the connection to which the packet belongs, and the sequence number is provided as a courtesy.

TCP's response to this notice is to disable congestion control mechanisms until the route has been restored. This involves two different issues: what specific actions TCP takes in response to the ELFN notice, and how it determines when the route has been restored.

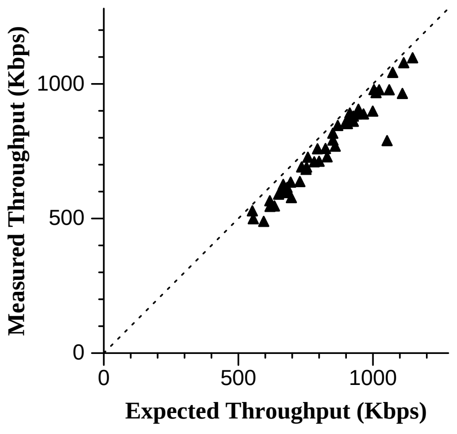
We used the following simple protocol. When a TCP sender receives an ELFN, it disables its retransmission timers and enters a "stand-by" mode. While on stand-by, a packet is sent at periodic intervals to probe the network to see if a route has been established. If an acknowledgment is received, then it leaves stand-by mode, restores its retransmission timers, and continues as normal. For this study, we elected to use packet probing instead of an explicit notice to signal that a route has been re-established.

To see what could be achieved with this protocol, we studied variations in the parameters and actions and measured their effects on performance. In particular, we looked at the following:

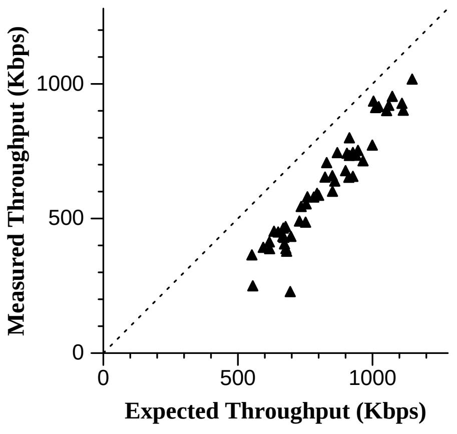
- Variations in the length of the interval between probe packets.
- Modifications to the retransmission timeout value (RTO) and congestion window upon restoration of the route.
- Different choices of what packet to send as a probe.

The results of these studies are presented below. Each curve is based on the mean throughput for the 50 different mobility patterns we used earlier.

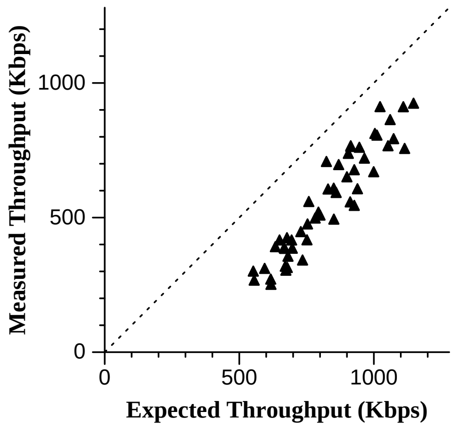
Fig. 54 is analogous to Fig. 47, except that the results in Fig. 54 are based on simulations in which TCP-Reno was modified to use ELFN (with a 2s probe interval). Clearly, the use of ELFN has improved the mean throughput for each of the speeds, as evidenced by the closer proximity of the measured pattern throughputs to the expected throughput line. The tighter clustering of the points also suggests that the use of ELFN techniques improves throughput across all patterns, rather than dramatically increasing just a few. However, notice that for one pattern performance was worse when ELFN was used. In Fig. 47(c) there is a pattern which has a measured throughput very near to its expected throughput (i.e., it is very close to the line), which is not present in Fig. 54(c). In this instance, the unusually good performance of TCP was a consequence of fortuitous timing of packet retransmissions, with



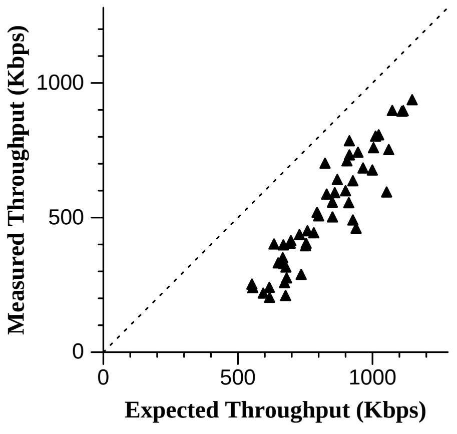
(a) *Speed = 2 m/s*



(b) *Speed = 10 m/s*



(c) *Speed = 20 m/s*



(d) *Speed = 30 m/s*

Fig. 54. *Per-pattern performance of TCP with ELFN using a 2s probe interval.*

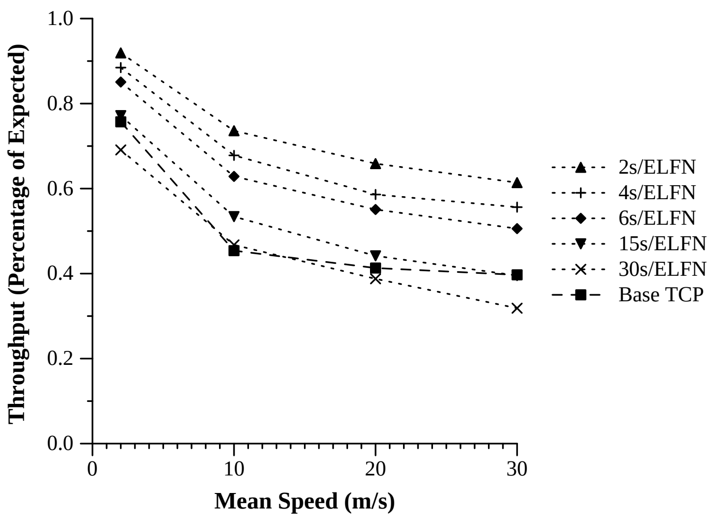


Fig. 55. *Performance comparison between basic TCP-Reno and TCP-Reno with ELFN using varying probe intervals.*

regard to the state of the network, that did not occur when ELFN was used. This is further evidence of the complex nature of TCP. The general trend, however, shows a performance improvement when ELFN is used.

Fig. 55 shows the measured throughput as a percentage of the expected throughput for various probe intervals. Based on these results, it is apparent that the throughput is critically dependent on the time between probe packets. This dependency exists because increasing the time between probes delays the discovery of new routes by the length of the interval. Thus, it is no surprise that if the probe interval is too large, then the throughput will degrade below that of standard TCP, as shown by the results for probe intervals of 30s. Intuitively, if the probe interval is too small, then the rapid injection of probes into the network will cause congestion and lower throughput. Thus, instead of a fixed interval, perhaps choosing an interval that is a function of the RTT could be a more judicious choice. However, based on the sensitivity of the throughput to the interval size, the function must be chosen very carefully.

In addition to varying the probe intervals, we also looked at the performance

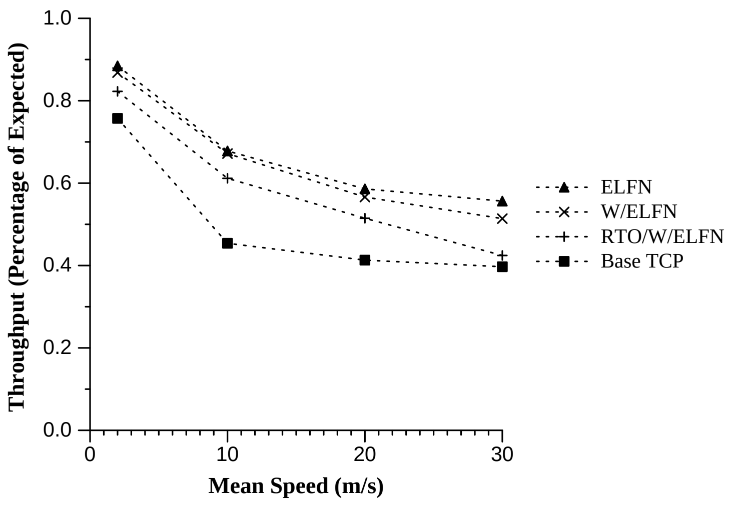


Fig. 56. *Performance comparison of different window and RTO modifications in response to the receipt of an ELFN message.*

advantages of adjusting the congestion window and/or retransmission timeout (RTO) after the failed route had been restored. These results are shown in Fig. 56. In the figure, *ELFN* represents the case where no changes are made to TCP's state because of ELFN. Thus, TCP's state (congestion window, RTO, etc.) are the same after the route is restored, as it was when the ELFN was first received. *W/ELFN* represents the case where the congestion window is set to one packet after the route has been restored, and *RTO/W/ELFN* represents the case where the RTO is set to the default initial value (6s in these simulations) and the window is set to one after the route is restored. Adjusting the window seemed to have little impact on the results. This is believed to be due to the fact that the optimal window (the bandwidth/delay product) of the simulated network is a relatively small number of packets, so it takes only a few round trips to ramp up to the optimal window after a failure. However, altering the RTO had a more significant impact on throughput. We suspect that this is due to a combination of factors, but is most probably caused by the frequency at which routes break, coupled with ARP's proclivity, as implemented, to silently drop

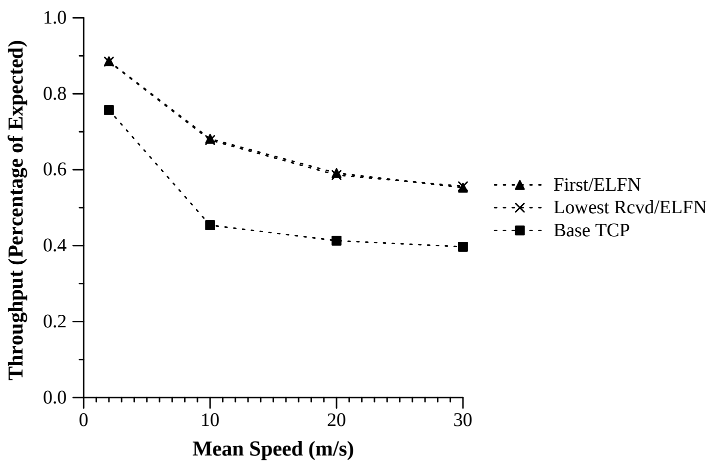


Fig. 57. *Performance comparison between basic TCP-Reno and TCP-Reno with ELFN using different choices for the probe packet.*

packets. Thus, if a restored route immediately breaks again and results in a failed ARP lookup, then the sender will likely timeout. Given the length of the timeout, it does not take many such occurrences to dramatically affect performance.

We also took a brief look at the impact that the choice of probe packet had on performance, which is shown in Fig. 57. We considered two possibilities: always send the first packet in the congestion window (*First/ELFN* in the figure), or retransmit the packet with the lowest sequence number among those signaled as lost in the ELFNs that were received (*Lowest Rcvd/ELFN*). The first approach is intuitive, the second approach was chosen with the optimistic thinking that perhaps some packets in the window did get through, and, if the route is restored quickly, then the next packet in sequence will be in flight. However, as shown by the results, this had almost no impact whatsoever. We suspect that this has to do with the fact that routes, once broken, were rarely restored quickly. In addition, as shown in Section D, the presence of different forward and reverse routes equalizes the two approaches when only the forward link breaks, since those packets that did get through before the break are acknowledged via the reverse channel. Thus, the lowest sequence number of the packets lost would also happen to be the first in the window.

Finally, we looked at how well the ELFN protocol performs in networks that contain multiple data sources by repeating the set simulations that were used for the throughput curves with cache replies enabled in Fig. 51 and Fig. 52, only now using ELFN as well. The results with CBR traffic are shown in Fig. 58, and the results for multiple TCP connections are shown in Fig. 59. Each curve is the average over 30 patterns. The ELFN protocol used 4s probes. Based on these results, it appears that similar performance benefits can be expected in congested networks, as in the uncongested network.

## F. Summary

In this chapter, we investigated the effects of mobility on TCP performance in mobile ad hoc networks and observed that TCP throughput drops significantly with node movement. The cause was identified as TCP's inability to recognize the difference between link failure and congestion, resulting in unnecessary delays. We then showed several specific examples of TCP behavior, one of which resulted in zero throughput, the other, in an unexpected rise in throughput with an increase in speed. We also introduced a new performance metric, *expected throughput*, that accounts for the differences in throughput when the number of hops varies. Based on this analysis, we then presented the Explicit Link Failure Notification (ELFN) protocol for TCP. The objective of ELFN is to provide the TCP sender with information about link and route failures so that it can avoid responding to the failures as if congestion occurred. Through simulation analysis, we showed that ELFN can significantly increase the expected throughput of a TCP connection in ad hoc networks.

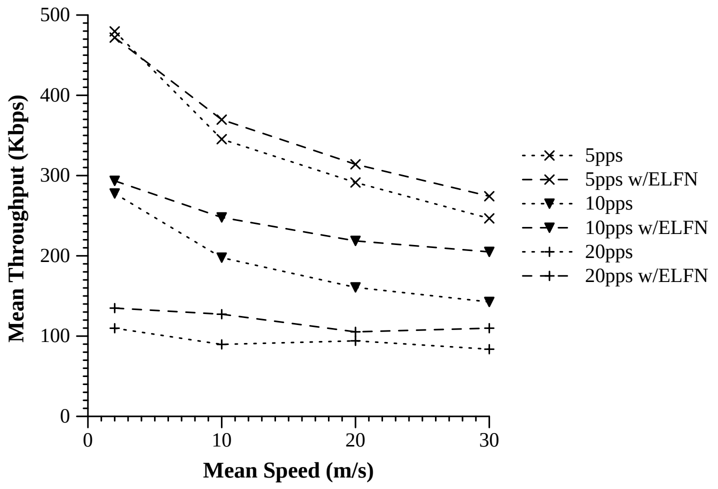


Fig. 58. *Performance comparison of TCP-Reno and TCP-Reno with ELFN when additional traffic is in the network. The additional traffic is provided by 10 CBR connections, each sending 5, 10, and 20 packets per second (pps).*

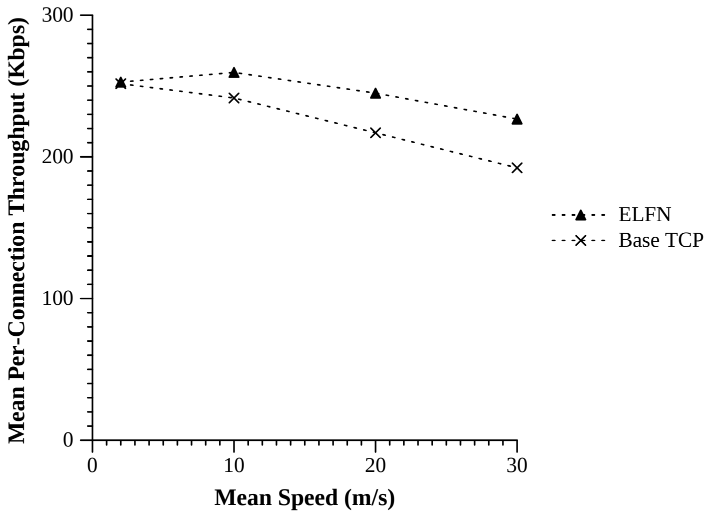


Fig. 59. *Performance comparison of TCP-Reno and TCP-Reno with ELFN for five concurrent TCP connections.*

# CHAPTER VII

# MULTI-MODE TCP

In this chapter, we address problems with TCP that arise when used over wireless networks that contain a mixture of broadband and legacy links. The large differences in bandwidth and delay between, for example, a 100 Mbps satellite link and a 64 Kbps dialup can result in end-to-end connection behavior that cause TCP to perform poorly. The cause is that, to the end-user, connections across these networks could exhibit a "modal" behavior as periodic link breakages result in repeated oscillations in the end-to-end bandwidth and delay. For example, a connection between two legacy nodes may periodically be routed across a broadband satellite link when they lose direct communication with each other. For TCP connections, these wide and sudden variations in capacity result in an under-utilization of the network because TCP's rate control mechanism is not designed to react quickly to rapid changes in capacity. Since TCP is the most widely used transport protocol, it is worthwhile to devise end-to-end solutions that maximize its performance for future wireless broadband networks. In this chapter, we present a modified version of TCP, called *Multi-Mode TCP* that is designed to capitalize on the "modal" behavior of broadband wireless networks. MM-TCP does this by recognizing these modes and saving the state of the congestion control mechanisms for rapid recall if the mode is revisited later. This chapter is organized as follows. We first present a performance analysis of standard TCP-Reno over multi-modal broadband networks, and then present and evaluate MM-TCP.

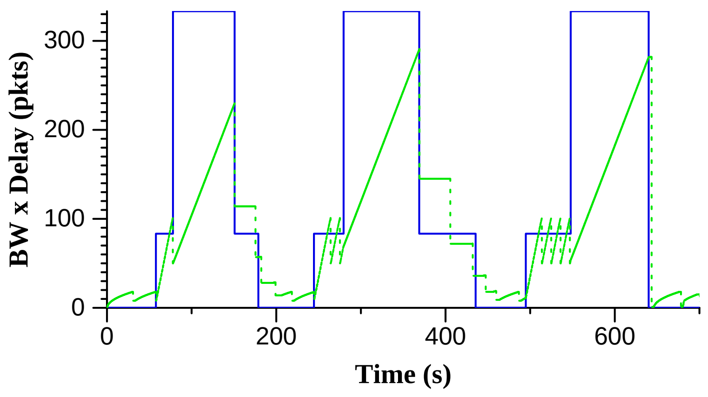


Fig. 60. *Performance of TCP-Reno across a connection with "modal" changes in bandwidth-delay. Shown is TCP-Reno's congestion window (dotted line) over the connection bandwidth-delay (solid line).*

## A. Performance of TCP-Reno

Fig. 60 illustrates the reaction of TCP-Reno to "modal" oscillations in available capacity. Available capacity for a connection is frequently expressed as the *bandwidth-delay product*, which is the capacity of the network as seen by the connection endpoints, and is calculated as $bw \times rtt$, where $bw$ is the available capacity on the bottleneck link on the path and $rtt$ is the round-trip time (delay). Here, a single TCP-Reno connection is spread across a network that experiences modal shifts in bandwidth-delay between 1, 83, and 333 packets, which is represented by the solid line in the figure. The dotted line represents the sender's *congestion window*, which is the maximum number of unacknowledged packets the sender is allowed to have outstanding in the network (assuming the receiver allows that many).

Notice that when there is a sharp *increase* in capacity, the sender does not increase its congestion window to take advantage of it. This is because TCP-Reno's sending rate is governed by an additive-increase, multiplicative-decrease (AIMD) congestion avoidance algorithm. This is a conservative algorithm, designed to prevent congestive collapse, that limits rate increases to one packet per $rtt$. The result, in this instance, is that TCP-Reno's throughput is less than "optimal" for the network conditions.

Also notice that when there is a sharp *decrease* in capacity, TCP-Reno does not always lower its sending rate as quickly. This is fine, as long as the sender does not try to inject more packets into the network until the window is lowered below the available capacity, since the network is now in a congested state. However, in some circumstances the sudden decrease in capacity results in a dramatic increase in the $rtt$, which can cause TCP-Reno's congestion control mechanism to assume that packets are lost (as triggered by a retransmission timeout $RTO$) and retransmit them. These are called "spurious" retransmissions, and as many as an entire congestion window of packets may be needlessly retransmitted, again resulting in under-utilization of the network.

Finally, notice that even though the capacity changes are repetitive, TCP-Reno's response is the same for each.

In the next section, we present the a modified version of TCP-Reno, called Multi-Mode TCP, that is designed to take greater advantage of sharp increases in capacity, while limiting spurious retransmissions as much as possible when there are share decreases in available capacity, by caching information about recent modes for later recall and rapid response.

## B. Multi-Mode TCP

The purpose of MM-TCP is to capitalize on the "modal" behavior that occurs on connections over heterogeneous links that have widely varying bandwidth and delay characteristics. MM-TCP does this by identifying modes and caching state variables that can be reused later when the mode is revisited. The variables cached (see Ta-

Table V. *State variables cached by MM-TCP for each network mode.*

| *Variable* | *Description* |
|---|---|
| $cwnd$ | Congestion window size |
| $ssthresh$ | Slow-start threshold |
| $srtt$ | Smoothed round-trip time average |
| $rttvar$ | Mean deviation of round-trip time samples |
| $rto$ | Retransmission timeout |
| $backoff$ | Retransmission timer backoff multiplier |

ble V) are those that are used to determine the sending rate and congestion detection. In the next section, we describe how these variables are used and managed.

### 1. Mode Definition

As discussed in the previous section, it is the widely varying bandwidth and delay characteristics that cause the "modal" behavior seen in Fig. 60. In this section, we present how a mode is characterized and detected.

TCP is designed to continuously probe for the maximum available capacity by increasing its transmission rate until it detects congestion, then cutting its rate down and starting again. This behavior results in the characteristic sawtooth pattern, seen in Fig. 60 (around the 500s mark), that oscillates around the available capacity. To identify the network *mode* that the connection is in, we filter the bandwidth $bw$ and delay $d$ estimates to remove as much of the oscillation as possible. We then track the smoothed estimates over time to see if they stay within a given range of values that define the boundaries, or *window*, of a mode. If the estimates consistently stay within a fixed window for some number of samples $k$, then that window is declared as a new mode. More specifically, for a given window size of $w_{bw}$ for the bandwidth and $w_d$ for the delay, and sample size $k$, a new mode is declared if

$$(bw_0 - \tfrac{w_{bw}}{2}) < bw_i < (bw_0 + \tfrac{w_{bw}}{2}), \quad 0 < i \leq k \tag{7.1}$$

$$(d_0 - \tfrac{w_d}{2}) < d_i < (d_0 + \tfrac{w_d}{2}), \qquad 0 < i \leq k \tag{7.2}$$

are true. Once the mode has been declared, the window is centered around the mean of the most recent $k$ samples. We then save $bw_0$ and $d_0$ for later retrieval when searching modes.

Determining the correct window size is important because if too large a window is chosen the protocol may not recognize all of the modes (i.e., the window may span across multiple modes), but if too small a window is chosen the protocol may frequently and unnecessarily identify false modes. So the granularity of the window is very important. Finding the correct window, though, is challenging because the range of variation in the samples is dependent on the size of the bandwidth-delay. Thus, we define $w_{bw}$ and $w_d$ on separate log scales with bases $B$ and $D$, so Equations (7.1) and (7.2) become

$$(\log_B[bw_0] - \tfrac{w_{bw}}{2}) < \log_B[bw_i] < (\log_B[bw_0] + \tfrac{w_{bw}}{2}), \quad 0 < i \leq k \tag{7.3}$$

$$(\log_D[d_0] - \tfrac{w_d}{2}) < \log_D[d_i] < (\log_D[d_0] + \tfrac{w_d}{2}), \qquad 0 < i \leq k \tag{7.4}$$

Therefore, the granularity of the bandwidth window is controlled by the log base $B$ and the granularity of the delay window by log base $D$. In practice, it is difficult to find fixed values for $B$, $D$, $w_{bw}$, and $w_d$ that give optimal boundaries between all modes for all network conditions, so for the purpose of the analysis in this report we define the boundaries dynamically as follows. We used a very simple algorithm where the size of $w_{bw}$ and $w_d$ were fixed at 0.6, and the default base for both $B$ and $D$ was 2. If the estimated delay drops below 500 ms, then $D$ was changed to 3. Similarly, if the estimated bandwidth dropped below 800 Kbps, then $B$ was changed to 3. This algorithm is based on our experience, which showed that performance increased if we used coarser granularity at lower bandwidth-delay because of the higher relative variability in the $(bw, d)$ estimates. Investigation of techniques to vary both window size and granularity automatically according to the variance of the estimates is a topic of future work.

To obtain stable estimates of (bw, d) for a mode, we use exponentially weighted moving average (EWMA) filters, similar to the approach in [76]. These filters take the form:

$$E(t) = gain * E(t-1) + (1 - gain) * O(t) \tag{7.5}$$

where $E(t)$ is the newly generated estimate, $E(t-1)$ is the prior estimate (initially, $E$ is equal to the observed value when we decide to create a new mode), and $O(t)$ is the observation where *ssthresh* is reset. For stable bandwidth and delay estimations, *gain* is set to 0.9.

A sender takes the following approach when creating a new mode, or updating an existing mode.

1. Verify if this is a "known" mode. That is, check to see if the current (bw, d) estimate maps to a mode that was visited earlier. If so, use the stored TCP variables. If no modes satisfy this condition, use the standard TCP algorithms to obtain the values for the TCP variables.

2. Wait for $R$ (say $R = 2$) round trip times and save the set of TCP variables. Save a new set of TCP variables every *rtt* from then on, and keep track of the last $S$ (say $S = 5$) samples.

3. If the current *cwnd* is greater than twice *ssthresh*, change *ssthresh* to *cwnd*/2

instantly. This results in a quick update of the TCP variables to more efficiently represent the current network conditions.

4. Upon a mode change, average the values for each individual TCP variables from the past $S$ sample sets and save those values in a candidate TCP variables set. If the visits to this mode lasted less than $R$ *rtt*s, or *ssthresh* has never been changed/updated, ignore the visit to this mode.

5. If a candidate TCP variable set was obtained, compare that with the previous $C-1$ (say $C=3$) candidate sets obtained from the previous visits to this mode and save the one that has the smallest value for *ssthresh*. If this mode has not been visited before, save the candidate set as the TCP variable set for the current mode. If the maximum number of modes is exceeded, throw out the least recently visited mode.

## 2. Mode Change Detection

In order to detect a mode change, the sender must constantly monitor a set of parameters that give some indication of the current network condition. As discussed in the previous section, these parameters are the round trip delay (d) and the bandwidth (bw) of the bottleneck link in the forward path. In our work, we have limited our modifications to the TCP sender. The challenge is to find reliable mechanisms to obtain accurate estimates of both parameters. To do this, we estimate the bandwidth first, and then use the bandwidth estimate to determine the delay.

### a. Bandwidth Estimation

To estimate the bottleneck bandwidth, we use the Round-Trip Time Measurement (RTTM) technique described in [64]. In RTTM, the sender and receiver exchange timestamps reflecting when the packet and ACK were sent. At the receiver, the sender's timestamp is also echoed back so that the sender can use the difference between the receive times to estimate the bandwidth on the forward link. The idea is that when packets cross the bottleneck link they will be spread out in time by the transmission time of the first packet. Assuming this spacing is maintained through the remainder of the path, the timestamp given them by the receiver can then be used by the sender to compute the bandwidth using the calculated delay and the size of the packets. In our implementation, this is done for every two packets transmitted back-to-back by the sender. Although this approach has been shown to overestimate the bandwidth [91], alternative forms of estimation such as Packet Bunch Modes [5], Tailgaiting [79], and SLoPS [67] were deemed to be too inefficient and/or complex for use as estimators of TCP connection state, since they were generally designed for use as network management tools.

As a topic of future work, we intend to investigate heuristics that attempt to diminish the impact of the overestimation of the RTTM approach, such as that given in [90], and/or develop alternative methods for online bandwidth estimation based on ideas from the aforementioned network management tools.

#### b. Delay Estimation

Standard TCP implementations that rely on one *rtt* measurement on one segment per window lead to unacceptably slow estimation in networks with a high bandwidth-delay product (pipe size). Therefore, we use the timestamp TCP option to allow an *rtt* measurement for every segment transmitted, and then employ the stable filter in Equation 7.5 again to smooth the estimate (referred to as *my_rtt*), where the current observation is obtained on every received segment. However, the *my_rtt* estimator includes packet-propagation delays, packet-queuing delays, packet-processing delays, and transmission delay, which causes variations in a wide range since the queuing delay, unlike the other three delays, can alter from packet to packet. So, we propose a queuing delay estimation method, and apply this method to gain a flat end-to-end delay that attempts to exclude the unwanted queuing delay.

To estimate the queuing delay, we use a simple fluid–flow model to represent the end-to-end path between sender and receiver. In this model, the sequence of hops from sender to receiver can be represented as a single service queue with queuing delay $E\_Dq$ and available bandwidth $E\_BW$. $E\_BW$, which is the transmission rate, is also the rate at which bits are pushed out the queue. The ratio *buffer*/$E\_BW$ often plays an important role in estimating the extent of the queuing delay. To estimate the queuing delay, we assume that the queue is empty when the number of unacknowledged packets ($no\_notacked\_packet$) is equal to $ssthresh$, and bits increase as the number of $not\_acked\_packets$ increases. Note that, by design, $ssthresh$ is intended to be a good estimator of the available bandwidth. To stabilize the estimate, we use a simple $Flip - Flop$ filter (similar to that in [76]), consisting of two EWMA filters. One of the filters is *agile*, with a gain of 0.09, which is used to quickly detect transient shifts in network conditions (i.e., a mode shift), and the other is *stable*, with a gain of 0.91, to smooth out noise when the network is in a stable mode. The stable filter is the default, and the agile filter is selected when observations are unusually noisy (like at the beginning of the TCP session and between mode shifts).

$$E\_Dq(t) = gain * E\_Dq(t-1) + (1-gain) * Dq \tag{7.6}$$

In Equation 7.6, the current observation $Dq$ is defined as:

$$\begin{aligned} Dq &= (no\_notacked\_packet - e\_ssthresh) \\ &\quad *packet\_size/E\_BW \end{aligned}$$

$$\begin{aligned} & if\ no\_notacked\_packet > e\_ssthresh && (7.7) \\ Dq &= 0; \\ & if\ no\_notacked\_packet <= e\_ssthresh && (7.8) \end{aligned}$$

where $e\_ssthresh$ is equal to $max(ssthresh, cwnd/2)$.

Note that, in situations where we overestimate or underestimate the queuing delay, estimate works for our purposes since the outgoing rate is always equal to the transmission rate, and our aim is *only* to find a *flat* end-to-end delay.

Finally, the flat end-to-end delay estimation is calculated using the stable EWMA filter in Equation 7.5 where the current observation is equal to $my\_rtt - E\_Dq$.

## 3. Mode Change Decision

Based on the discussion in the previous section, we have (bw, d) estimates describing the current network conditions. Using these, the sender can make decisions to change or not change modes, and, summarily, use or not use the TCP variable set associated with the modes.

In order to make sure a mode change has occurred, the sender may need to observe a sequence of estimates that consistently indicate such a change. Only then it will be able to make a safe decision that, in fact, the network conditions have changed to the extent that a new mode is required. This will reduce the probability of a "miss" and/or "false detect". The recommended approach is to have the sender verify an estimate pair (bw, d) and check whether or not it maps to a mode different from the current mode. If it maps to a different mode, a counter is increased by 1 and compared to a threshold $k$ (say $k = 3$). If the counter is equal to $k$, the sender has enough evidence to switch to a new mode. In case it maps to more than one mode, due to modes overlapping, the most recently visited mode has higher priority to be selected. Otherwise it waits for the next estimate pair and repeats the procedure. If at any time an estimate pair maps to the current mode or another existing mode, the counter is set to 0.

## 4. Mode Change Adaptation

If the network conditions present path characteristics (bw, d) that the TCP sender has seen before and identified as a mode, the TCP sender declares one RTT as a switching period and quickly adapts the mode's TCP variables to those previously stored for the new mode.

The adaptation algorithm is implemented as follows.

1. If the current *ssthresh* is less than the new mode's *ssthresh*,
   (a) Set *ssthresh* to the mode's *ssthresh*.
   (b) If the current *cwnd* is less than the new *ssthresh* (i.e., the sender will be in *slow start*), then to avoid sending a burst of data packets that might cause congestion, we temporarily change the rate at which the slow start algorithm increases *cwnd*. Typically, the slow start algorithm increases *cwnd* at a rate of 1 segment for every ACK received (exponential growth). Here, if the new mode's *cwnd* is larger than the current *ssthresh* by some threshold $t$, we temporarily change the rate of growth to some lower value $r < 1$ segment per ACK. A challenge is selecting values of $r$ and $t$ that achieve rapid growth (hence, good performance), but don't overload the network. In our simulations, we set $r = 0.1$ and $t = 50pkts$, which gave reasonably good performance. In the future, we will investigate how to determine these values dynamically and store them as part of the state of the mode. Of course, once *cwnd* is equal to *ssthresh*, the slow start rate

is reset to 1 for normal TCP operation.

2. If the current *ssthresh* is greater than the new mode's *ssthresh*
   (a) Set *ssthresh* to the new mode's *ssthresh*.
   (b) If the *cwnd* is greater than the new mode's *cwnd*, set *cwnd* to the new mode's *cwnd*.
3. If *srtt* is less than the new mode's *srtt*, set *srtt*, *rttvar*, *backoff* and *rto* to the new mode's values.
4. During the switching period, each time another duplicate ACK arrives, the TCP sender retransmits the next missing packet (since each duplicate ACK means that a packet has left the network and changing link condition causes a lot of missing packets in the forward path).

## C. Performance Evaluation

In this section we present simulation results of the performance of MM-TCP, and compare it to the performance of TCP-Reno. In the next section we describe the simulation model used to evaluate the protocols, which is followed by the simulation results.

### 1. Simulation Model

The results in this chapter are based on simulations using the ns-2 simulator, described in Chapter III.

Here, the network was built using simple duplex links. For single-hop experiments, dynamic bandwidth and delay changes were scripted on the duplex link. For multi-hop experiments, we used a fixed topology and induced path changes through

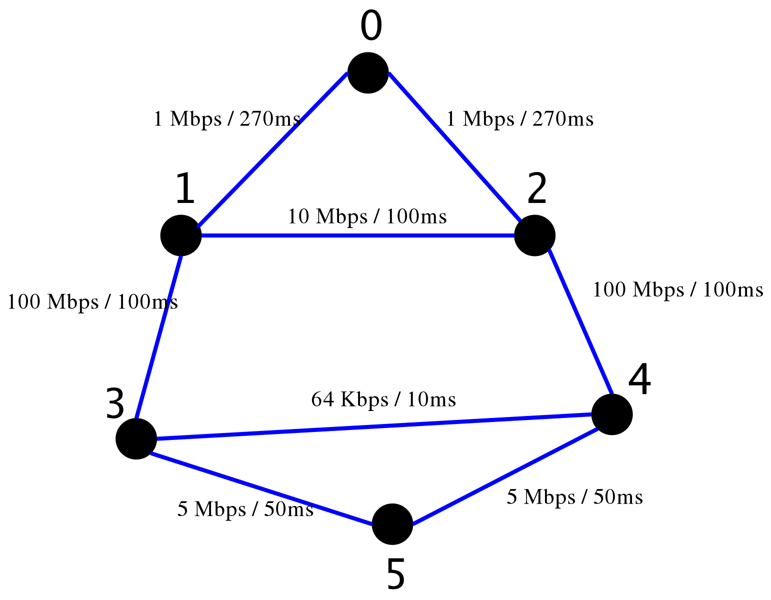


Fig. 61. *Network topology used in this study. Link rates and transmission delays correspond to network configuration 1.*

random link failures. TCP-Reno is used as a basis for comparison for all results. Traffic was generated using a continuously backlogged FTP application. Routing was "omniscient", manually and instantaneously reconfigured after every link change, and were calculated using an all-pairs shortest path algorithm. Although route changes were instantaneous, packets in queues of downed links were lost, reflecting, to some extent, the kind of packet losses that might be expected when using an "actual" routing protocol.

In our analysis, we used three different network configurations. Each configuration consisted of six nodes arranged in the topology shown in Fig. 61. Link data rates were fixed, but delay and failure characteristics were varied, resulting in end-to-end bandwidth-delay products ranging from 1 to 166 packets for Network 3, 1 to 333 packets for Network 2, and 1 to 500 packets for Network 1. Also, time intervals between modes were compressed by 1/2 for Network 2 and Network 3, as compared to Network 1. A simple link impairment algorithm was used to simulate random link failures. The algorithm behaved as follows. For some random time interval with mean $T$, a link would toss a coin and flip its state (up or down), with the constraint that a link cannot go down if doing so would disconnect the network. All results are for a single FTP connection from Node 3 to Node 4. The packet size was 1500 bytes. Each network configuration was simulated for 50 runs. Specific details of the network configurations are as follows.

- **Network 1:** Link data rates and transmission delays were set as shown in Fig. 61. $T = 600s$ for links $(0, 1)$ and $(0, 2)$, and $T = 300s$ for all other links.
- **Network 2:** Identical to Network 1, but delays on links $(1, 3)$ and $(2, 4)$ were changed to 50ms. $T = 300s$ for links $(0, 1)$ and $(0, 2)$, and $T = 150s$ for all other links.
- **Network 3:** Identical to Network 1, but the delays on links $(1, 3)$ and $(2, 4)$ were 25ms, and the delay on link $(1, 2)$ was 50ms. $T$ was the same as in Network 2.

### 2. Simulation Results

Fig. 62 illustrates the reaction of MM-TCP to the same "modal" oscillations in available capacity that were shown in Fig. 60. Notice that each time a higher mode is revisited, MM-TCP adapts more quickly to the increase in capacity. Fig. 63 shows the values of $backoff$ for both TCP-Reno and MM-TCP to the same "modal" oscillations in available capacity that were shown in Fig. 62. Note that $backoff$ is the value that is multiplied by the $retransmissiontimeout(RTO)$ estimate to determine the value for the next timeout interval. Its value is doubled for every consecutive timeout that occurs, up to a maximum of 64. Notice that although MM-TCP suffers a timeout

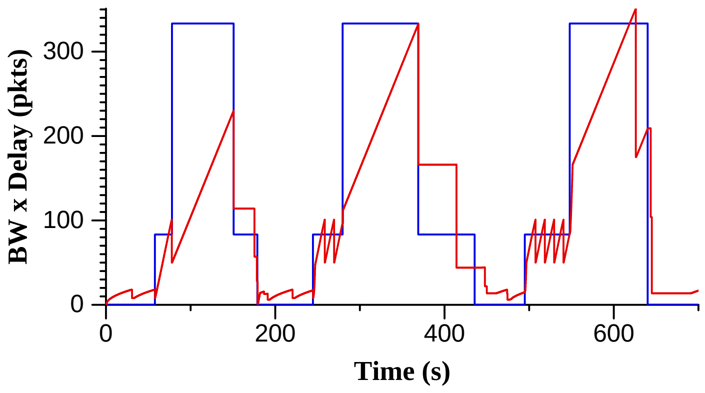


Fig. 62. *Performance of MM-TCP across a the same "modal" connection shown in the Introduction. Shown is MM-TCP's congestion window (dotted line) over the connection bandwidth-delay (solid line).*

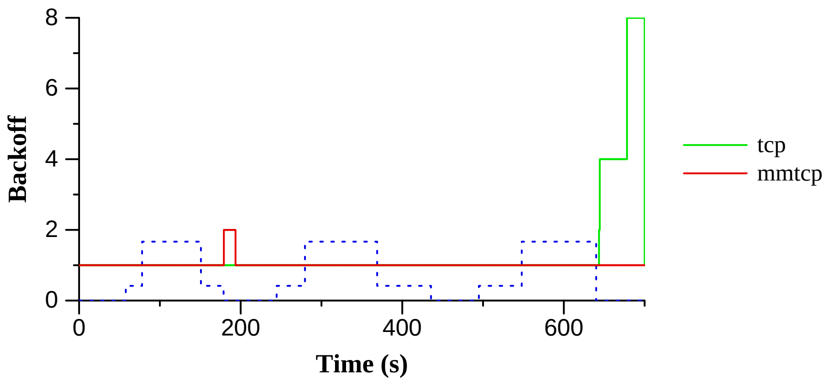


Fig. 63. *Comparison of values for the $backoff$ multiplier for MM-TCP and TCP-Reno.*

after the first decrease in capacity, later decreases in capacity are detected and timeouts are avoided, whereas TCP-Reno suffers consecutive timeouts at the large drop around time 650s.

Fig. 64 shows the percentage improvement for each simulation scenario for the three multi-hop network configurations. Note that in almost all cases MM-TCP improves performance over standard TCP-Reno. Also notice that in some cases the performance improvement is dramatic.

Fig. 65 shows a comparison of the “optimal” throughput for the scenarios simulated. Here, the optimal throughput is defined to be the sum of the total bandwidth-delay product over the lifetime of the connection. Notice that in general MM-TCP is closer to optimal than TCP-Reno, and that even in cases such as for Network 3, where TCP-Reno performs very close to optimal, MM-TCP still shows better overall performance (as signified by the tighter “grouping” of measurements.

## D. Summary

In this chapter, we presented a modified version of TCP, called *Multi-Mode TCP* that is designed to capitalize on the “modal” behavior of wireless networks. MM-TCP does this by recognizing these modes and saving the state of the congestion control mechanisms for rapid recall if the mode is revisited later. We gave a performance analysis of standard TCP-Reno over multi-modal wireless networks, and then presented results that showed the efficacy of MM-TCP for long-fat pipes in modal networks.

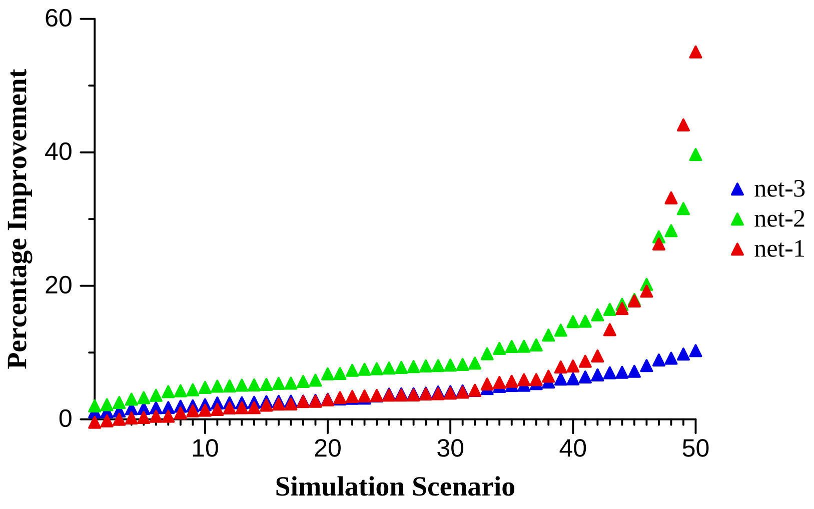
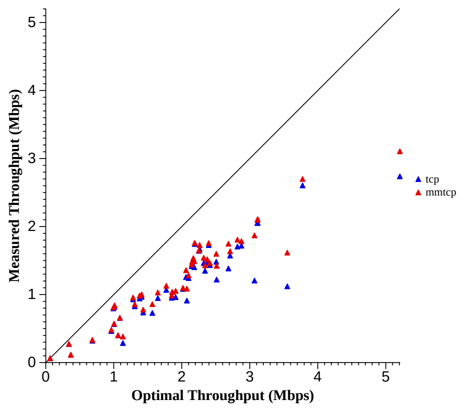
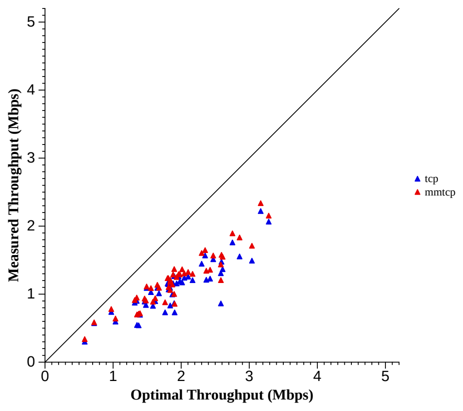


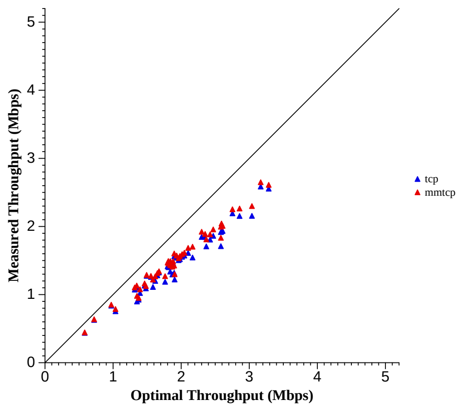
Fig. 64. *Percentage of improvement of MM-TCP over standard TCP-Reno for each of the three network configurations simulated. The results are sorted according to increasing improvement for each series of results.*



(a) *Network 1*



(b) *Network 2*



(c) *Network 3*

Fig. 65. *Comparison of measured and optimal throughput for the 50 scenario patterns.*

# CHAPTER VIII

# CONCLUSIONS AND FUTURE DIRECTIONS

In this chapter, we conclude this dissertation by summarizing the contributions of our work and suggested directions for future work.

## A. Summary

In this dissertation, we presented a number of *cross-layer adaptive* protocols that dynamically adapt their behavior to match changes in network characteristics based on cross-layer feedback.

- We investigated new link layer mechanisms that dynamically monitor and adapt to changes in link quality by adjusting packet transmission rates according to certain optimization criteria. We showed that the use of such techniques in ad hoc networks can significantly improve performance, and presented a new approach, the Receiver-Based AutoRate (RBAR) protocol, that uses common control messages to form a tight feedback control loop to adapt link characteristics based on receiver-perceived link conditions.

- We showed how conventional shortest-path routing protocols may not perform well in multirate, multihop networks because they fail to consider the data rates of the links on a route. We then present two on-demand routing protocols, Rate Probing (MRPRB) and Minimum Rate (MRMR), that discover, prioritize, and select routes based on their estimated end-to-end throughput according to the data rates on the route.

- We then presented a thorough analysis of TCP performance in ad hoc networks, showing that communication outages caused by link failures and routing delays can be very detrimental to TCP performance. Specifically, we showed how TCP throughput can drop dramatically because of consecutive spurious timeouts caused by link breakage and subsequent routing errors, and presented a proposed solution to the problem, the Explicit Link Failure Notification (ELFN) protocol, which uses explicit feedback messages from the link layer about link failures to adapt TCP's congestion control mechanisms to prevent spurious timeouts.

- Finally, we demonstrated how link failures in heterogeneous networks containing links with widely varying bandwidth and/or delay can cause "modal" capacity shifts that are undetected by TCP, resulting in less-than-ideal throughput, and present our proposed solution, Multi-Mode TCP (MMTCP), which is capable of recognizing capacity shifts and adapting TCP's congestion and flow control mechanisms to respond more rapidly.

## B. Future Directions

There are a number of interesting areas of future work that were identified as a result of the work in this dissertation.

More research is needed to better understand the complex interactions between TCP and lower layer protocols when used over mobile ad hoc networks, and to find solutions to the problems caused by these interactions.

- One such problem that we identified was the interaction between TCP and ARP. The ARP in the extensions is based on a BSD implementation, with a one-packet queue and no request timeout mechanism. Thus, packets were regularly dropped or held indefinitely while awaiting resolution. A more advanced ARP needs to be employed, such as one that will provide for the queuing of multiple packets awaiting resolution, with a timeout mechanism to promptly signal failure.

- Another problem we identified was the significant impact that route cache management has on TCP performance. The results suggest that more aggressive cache management protocols are needed to counter the effects of mobility, such as the use of adaptive route cache timeouts, negative information, or signal strength information.

- Interest in the deployment of long-haul high bandwidth terrestrial, orbital, and inter-planetary wireless networks suggests that research on MM-TCP and similar protocols should be continued. In particular, improvements to the mode adaptation algorithm, as well as analysis of performance in congested networks. Also, moving MM-TCP's modal monitoring capability into an external module that can be used to monitor routes for short-lived connections, like HTTP traffic, would be advantageous.

The development of "intelligent" adaptive network protocols is also an interesting area of future research. In particular, the coordinated interaction of distributed learning and reasoning algorithms with adaptive network protocols shows great promise towards providing networks that can autonomously configure and adapt to harsh and challenging environments.

## REFERENCES

[1] N. Abramson, "The ALOHA system – another alternative for computer communications," in *Fall Joint Computer Conference*, vol. 37, Montvale, NJ, pp. 281–285, American Federation of Information Processing Societies (AFIPS), 1970.

[2] G.-S. Ahn, A. T. Campbell, S.-B. Lee, and X. Zhang, "INSIGNIA," draft-ietf-manet-insignia-01, Internet Engineering Task Force (IETF), Reston, VA, October 1999.

[3] Aironet, *PC4800 User Guide*, 1998. accessed: August 2001, http://www.aironet.com/support/ftp/.

[4] S. M. Alamouti and S. Kallel, "Adaptive trellis-coded multiple-phase-shift keying for rayleigh fading channels," *IEEE Transactions on Communications*, vol. 42, no. 6, pp. 2305–2314, June 1994.

[5] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proceedings of the ACM Conference on Communications Architectures and Protocols (SIGCOMM)*, Cambridge, MA, pp. 263–274, August 1999.

[6] V. Anantharaman and R. Sivakumar, "A microscopic analysis of TCP performance over wireless ad-hoc networks," poster paper, Marina Del Rey, CA, June 2002. accessed: November 2004, http://portal.acm.org/citation.cfm?id=511375.

[7] B. Awerbuch, D. Holmer, and H. Rubens, "High throughput route selection in multi-rate ad hoc wireless networks," Tech. Rep., Computer Science, Johns Hopkins University, Baltimore, MD, March 2003. Version 2.

[8] D. J. Baker, J. E. Wieselthier, and A. Ephremides, "A distributed algorithm for scheduling the activation of links in a self-organizing, mobile, radio network," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Philadelphia, PA, pp. 2F.6.1–2F.6.5, June 1982.

[9] B. S. Bakshi, P. Krishna, D. K. Pradhan, and N. H. Vaidya, "Improving performance of TCP over wireless networks," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, Baltimore, MD, pp. 365–373, May 1997.

[10] K. Balachandran, S. R. Kadaba, and S. Nanda, "Channel quality estimation and rate adaption for cellular mobile radio," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 7, pp. 1244–1256, July 1999.

[11] H. Balakrishnan, V. N. Padmanabhan, and R. H. Katz, "The effects of asymmetry on TCP performance," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Budapest, Hungary, pp. 77–89, September 1997.

[12] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," in *Proceedings of the ACM Conference on Communications Architectures and Protocols (SIGCOMM)*, Stanford, CA, pp. 256–269, August 1996.

[13] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Berkeley, CA, pp. 2–11, November 1995.

[14] R. K. Balan, B. P. Lee, K. R. R. Kumar, L. Jacob, W. K. G. Seah, and A. L. Ananda, "TCP HACK: TCP header checksum option to improve performance over lossy links," in *Proceedings of IEEE INFOCOM*, vol. 1, Anchorage, AL, pp. 309–318, April 2001.

[15] D. A. Beyer, "Accomplishments of the DARPA survivable adaptive networks SURAN program," in *Proceedings of the IEEE Military Communications Conference (MILCOM)*, Monterey, CA, pp. 855–862, October 1990.

[16] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LAN's," in *Proceedings of the ACM Conference on Communications Architectures and Protocols (SIGCOMM)*, London, UK, pp. 212–225, August 1994.

[17] S. Biaz, G. D. Holland, Y.-B. Ko, and N. H. Vaidya, "Evaluation of protocols for wireless networks," in *Parallel and Distributed Processing Techniques and Applications*, vol. 4, Las Vegas, NV, pp. 2150–2156, July 1999.

[18] S. Biaz and N. H. Vaidya, "Is the round-trip time correlated with the number of packets in flight?," Tech. Rep. 99-006, Department of Computer Science, Texas A&M University, College Station, TX, March 1999.

[19] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, “A performance comparision of multi-hop wireless ad hoc network routing protocols,” in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Dallas, TX, pp. 85–97, August 1998.

[20] R. Caceres and L. Iftode, “Improving the performance of reliable transport protocols in mobile computing environments,” *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, pp. 850–857, June 1995.

[21] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, “A feedback based scheme for improving TCP performance in ad-hoc wireless networks,” in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, Amsterdam, The Netherlands, pp. 472–479, May 1998.

[22] I. Chlamtac and A. Lerner, “Fair algorithms for maximal link activation in multihop radio networks,” *IEEE Transactions on Communications*, vol. COM-35, no. 7, pp. 739–746, July 1987.

[23] T. Clausen and P. Jacquet, “Optimized link state routing protocol,” draft-ietf-manet-olsr-11, Internet Engineering Task Force (IETF), Reston, VA, July 2003.

[24] CMU Monarch Project, *Extension to the* ns *simulator*, November 1998. accessed: November 2004, http://www.monarch.cs.cmu.edu.

[25] Consultative Committee for Space Data Systems (CCSDS), *Space Communications Protocol Specifications - Transport Protocol (SCPS-TP)*, September 1997. accessed: November 2004, http://www.scps.org/scps/.

[26] M. S. Corson, “A triggered interface,” draft-corson-triggered-00, Internet Engineering Task Force (IETF), Reston, VA, May 2002.

[27] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing,” in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, San Diego, CA, pp. 134–146, September 2003.

[28] S. Das, C. E. Perkins, and E. M. Royer, “Performance comparison of two on-demand routing protocols for ad hoc networks,” in *Proceedings of IEEE INFOCOM*, vol. 1, Tel-Aviv, Israel, pp. 3–12, March 2000.

[29] L. Z. David Clark, Mark Lambert, "NETBLT: A bulk data transfer protocol," RFC 998, Internet Engineering Task Force (IETF), Reston, VA, March 1987.

[30] S. Dawkins, G. Montenegro, M. Kojo, and V. Magret, "End-to-end performance implications of slow links," RFC 3150, Internet Engineering Task Force (IETF), Reston, VA, July 2001.

[31] S. Dawkins, G. Montenegro, M. Kojo, V. Magret, and N. H. Vaidya, "End-to-end performance implications of links with errors," RFC 3155, Internet Engineering Task Force (IETF), Reston, VA, August 2001.

[32] D. Dharmaraju, A. Roy-Chowdhury, P. Hovareshti, and J. S. Baras, "INORA: a unified signaling and routing mechanism for QoS support in mobile ad hoc networks," in *International Conference on Parallel Processing Workshops (ICPPW)*, Vancouver, B.C., Canada, pp. 86–93, August 2002.

[33] R. Dube, C. D. Rais, K.-Y. Wan, and S. K. Tripathi, "Signal stability-based adaptive routing (SSA) for ad hoc mobile networks," *IEEE Personal Communications*, vol. 4, no. 1, pp. 36–45, February 1997.

[34] R. C. Durst, G. J. Miller, and E. J. Travis, "TCP extensions for space communications," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Rye, NY, pp. 15–26, November 1996.

[35] European Telecommunications Standards Institute (ETSI), Sophia Antipolis, France, *Broadband Radio Access Networks (BRAN); High Performance Radio Local Area Network (HIPERLAN) Type 1; Functional Specification*, July 1998. accessed: November 2004, http://www.palowireless.com/hiperlan2/.

[36] European Telecommunications Standards Institute (ETSI), Sophia Antipolis, France, *Broadband Radio Access Networks (BRAN); High Performance Radio Local Area Network (HIPERLAN) Type 2; System Overview*, February 2000. accessed: November 2004, http://www.palowireless.com/hiperlan2/.

[37] M. V. Eyuboglu, G. D. Forney, P. Dong, and G. Long, "Advanced modulation techniques for V.Fast," *European Transactions on Telecommunications*, vol. 4, no. 3, pp. 9–22, May 1993.

[38] G. Fairhurst and L. Wood, "Advice to link designers on link automatic repeat request (ARQ)," draft-ietf-pilc-link-arq-issues-02, Internet Engineering Task Force (IETF), Reston, VA, August 2002.

[39] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *Computer Communication Review*, vol. 26, no. 3, pp. 5–21, July 1996.

[40] S. Floyd, "Tcp and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 10–23, October 1994.

[41] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proceedings of IEEE INFOCOM*, vol. 3, San Francisco, CA, pp. 1744–1753, March 2003.

[42] C. L. Fullmer and J. J. Garcia-Luna-Aceves, "Solutions to hidden terminal problems in wireless networks," in *Proceedings of the ACM Conference on Communications Architectures and Protocols (SIGCOMM)*, Cannes, France, pp. 14–18, September 1997.

[43] J. H. Gass, M. B. Pursley, H. B. Russell, R. J. Saulitis, C. S. Wilkins, and J. S. Wysocarski, "Adaptive transmission protocols for frequency-hop radio networks," in *Proceedings of the IEEE Military Communications Conference (MILCOM)*, vol. 2, Boston, MA, pp. 282–286, October 1998.

[44] M. Gerla, K. Tang, and R. Bagrodia, "TCP performance in wireless multi-hop networks," in *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, New Orleans, LA, pp. 41–50, February 1999.

[45] J. D. Gibson, Ed., *The Mobile Communications Handbook*, $2^{nd}$ ed. Piscataway, NJ: IEEE/CRC Press, 1999.

[46] D. L. Goeckel, "Adaptive coding for time-varying channels using outdated fading estimates," *IEEE Transactions on Communications*, vol. 47, no. 6, pp. 844–855, June 1999.

[47] T. Goff, N. B. Abu-Ghazaleh, and D. S. Phatak, "Analysis of TCP performance on ad hoc networks using preemptive maintenance routing," in *Proceedings of the IEEE International Conference on Parallel Processing (ICPP)*, Valencia, Spain, pp. 232–239, September 2001.

[48] A. Goldsmith and S. G. Chua, “Adaptive coded modulation for fading channels,” *IEEE Transactions on Communications*, vol. 46, no. 5, pp. 595–602, May 1998.

[49] M. Günes and D. Vlahovic, “The performance of the TCP/RCWE enhancement for ad-hoc networks,” in *Proceedings of the IEEE International Symposium on Computers and Communications (ISCC)*, Taormina-Giardini Naxos, Italy, pp. 43–48, July 2002.

[50] P. Gupta and P. R. Kumar, “A system and traffic dependent adaptive routing algorithm for ad hoc networks,” in *IEEE Conference on Decision and Control*, San Diego, CA, pp. 2375–2380, December 1997.

[51] Z. J. Haas and M. R. Pearlman, “The zone routing protocol (ZRP) for ad hoc networks,” draft-ietf-manet-zrp-01, Internet Engineering Task Force (IETF), Reston, VA, August 1998.

[52] J. Hoe, “Improving the start-up behavior of a congestion control scheme for TCP,” in *Proceedings of the ACM Conference on Communications Architectures and Protocols (SIGCOMM)*, Palo Alto, CA, pp. 270–280, August 1996.

[53] G. D. Holland and N. H. Vaidya, “Analysis of TCP performance over mobile ad hoc networks,” in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Seattle, WA, pp. 219–230, August 1999.

[54] G. D. Holland and N. H. Vaidya, “Analysis of TCP performance over mobile ad hoc networks : Part I : Problem discussion and analysis of results,” Tech. Rep. TR99-004, Department of Computer Science, Texas A&M University, College Station, TX, February 1999.

[55] G. D. Holland and N. H. Vaidya, “Analysis of TCP performance over mobile ad hoc networks : Part II : Simulation details and results,” Tech. Rep. TR99-005, Department of Computer Science, Texas A&M University, College Station, TX, February 1999.

[56] G. D. Holland and N. H. Vaidya, “Impact of routing and link layers on TCP performance in mobile ad hoc networks,” in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, pp. 1323–1327, September 1999.

[57] G. D. Holland and N. H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *ACM/Baltzer Wireless Networks*, vol. 8, no. 2, pp. 275–288, March 2002.

[58] G. D. Holland, N. H. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multi-hop wireless networks," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, pp. 236–251, August 2001.

[59] HomeRF Working Group, *HomeRF Specification*, July 2002. accessed: November 2004, http://www.palowireless.com/homerf.

[60] IEEE, *Std 802.11 Information Technology Telecommunications And Information Exchange Between Systems, Local and Metropolitan Area Networks, Specific Requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* New York, NY: IEEE Press, November 1997.

[61] IEEE, "High-speed physical layer in the 2.4 GHz band," in *Std 802.11 Information Technology Telecommunications And Information Exchange Between Systems, Local and Metropolitan Area Networks, Specific Requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, New York, NY: IEEE Press, September 1999.

[62] IEEE, "High-speed physical layer in the 5 GHz band," in *Std 802.11 Information Technology Telecommunications And Information Exchange Between Systems, Local and Metropolitan Area Networks, Specific Requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, New York, NY: IEEE Press, September 1999.

[63] IEEE, "Further higher data rate extension in the 2.4 GHz band," in *Std 802.11 Information Technology Telecommunications And Information Exchange Between Systems, Local and Metropolitan Area Networks, Specific Requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, New York, NY: IEEE Press, June 2003.

[64] V. Jacobson, R. Braden, and D. Borman, "TCP extensions for high performance," RFC 1323, Internet Engineering Task Force (IETF), Reston, VA, May 1992.

[65] V. Jacobson, “Congestion avoidance and control,” in *Proceedings of the ACM Conference on Communications Architectures and Protocols (SIGCOMM)*, Stanford, CA, pp. 314–329, August 1988.

[66] V. Jacobson, “Modified TCP congestion avoidance algorithm,” email to end2end-interest@isi.edu, April 1990. accessed: November 2004, ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

[67] M. Jain and C. Dovrolis, “End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput,” in *Proceedings of the ACM Conference on Communications Architectures and Protocols (SIGCOMM)*, Pittsburgh, PA, pp. 295–308, August 2002.

[68] W. C. Jakes, Ed., *Microwave Mobile Communications.* Piscataway, NJ: IEEE Press, 1994.

[69] P. Johanson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, “Scenario-based performance analysis of routing protocols for mobile ad-hoc networks,” in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Seattle, WA, pp. 195–206, August 1999.

[70] D. B. Johnson, D. A. Maltz, and Y.-C. Hu, “The dynamic source routing protocol for mobile ad hoc networks,” draft-ietf-manet-dsr-08, Internet Engineering Task Force (IETF), Reston, VA, April 2003.

[71] J. Jubin and J. D. Tornow, “The DARPA packet radio network protocols,” *Proceedings of the IEEE: Special Issue on Packet Radio Networks*, vol. 75, no. 1, pp. 21–33, January 1987.

[72] A. Kamerman and L. Monteban, “WaveLAN-II: A high-performance wireless LAN for the unlicensed band,” *Bell Labs Technical Journal*, pp. 118–133, Summer 1997.

[73] P. Karn, “MACA – a new channel access method for packet radio,” in *ARRL/CRRL Amateur Radio Computer Networking Conference*, vol. 1, London, Ontario, Canada, pp. 134–140, ARRL, September 1990.

[74] P. Karn, “Advice for internet subnetwork designers,” draft-ietf-pilc-link-design-15, Internet Engineering Task Force (IETF), Reston, VA, July 2002.

[75] D. Kim, C.-K. Toh, and Y. Choi, "TCP-BuS: Improving TCP performance in wireless ad hoc networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, New Orleans, LA, pp. 1707–1713, June 2000.

[76] M. Kim and B. Noble, "Mobile network estimation," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, pp. 298–309, June 2001.

[77] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Dallas, TX, pp. 66–75, October 1998.

[78] R. Krishnan, M. Allman, C. Partridge, and J. P. G. Sterbenz, "Explicit transport error notification (ETEN) for error-prone wireless and satellite networks," Tech. Rep. 8333, BBN Technologies, Cambridge, MA, February 2002.

[79] K. Lai and M. Baker, "Measuring bandwidth," in *Proceedings of IEEE INFOCOM*, vol. 1, New York, NY, pp. 235–245, March 1999.

[80] G. S. Lauer, "Packet-radio routing," in *Routing in Communications Networks*, M. E. Steenstrup, Ed., ch. 11, pp. 351–396, Upper Saddle River, NJ: Prentice-Hall, 1995.

[81] S.-J. Lee and M. Gerla, "Dynamic load-aware routing in ad hoc networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, vol. 10, Helsinki, Finland, pp. 3206–3210, June 2001.

[82] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300–1315, July 2001.

[83] R. G. Lyons, *Understanding Digital Signal Processing.* Boston, MA: Addison-Wesley, 1997.

[84] G. Malkin, "RIP version 2: Carrying additional information," RFC 1723, Internet Engineering Task Force (IETF), Reston, VA, November 1994.

[85] S. McCreary and K. C. Claffy, "Trends in wide area IP traffic patterns," 2000. accessed: November 2004, http://www.caida.org/outreach/papers/2000/.

[86] J. Moy, "OSPF version 2," RFC 2328, Internet Engineering Task Force (IETF), Reston, VA, April 1998.

[87] P. Newman, “Traffic management for ATM local area networks,” *IEEE Communications Magazine*, pp. 44–50, August 1994.

[88] V. D. Park and M. S. Corson, “A highly adaptive distributed routing algorithm for mobile wireless networks,” in *Proceedings of IEEE INFOCOM*, vol. 3, Kobe, Japan, pp. 1405–1413, April 1997.

[89] V. D. Park and M. S. Corson, “Temporally-ordered routing algorithm (TORA) version 1: Functional specification,” draft-ietf-manet-tora-spec-00, Internet Engineering Task Force (IETF), Reston, VA, November 1997.

[90] C. Partridge, D. Rockwell, M. Allman, R. Krishnan, and J. Sterbenz, “A swifter start for TCP,” Tech. Rep. 8339, BBN Technologies, Cambridge, MA, March 2002.

[91] V. Paxson, “End-to-end internet packet dynamics,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 277–292, June 1999.

[92] C. E. Perkins, E. Belding-Royer, and S. Das, “Ad Hoc On-Demand Distance Vector (AODV) routing,” RFC 3561, Internet Engineering Task Force (IETF), Reston, VA, July 2003.

[93] C. E. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers,” in *Proceedings of the ACM Conference on Communications Architectures and Protocols (SIGCOMM)*, London, UK, pp. 234–244, August 1994.

[94] J. Postel, “Transmission control protocol,” RFC 793, Internet Engineering Task Force (IETF), Reston, VA, September 1988.

[95] J. G. Proakis, *Digital Communications*, 4$^{th}$ ed. New York, NY: McGraw-Hill, 2000.

[96] M. B. Pursley and C. S. Wilkins, “Adaptive transmission for direct-sequence spread-spectrum communications over multipath channels,” *ACM/Baltzer Wireless Networks*, vol. 7, no. 2, pp. 69–77, April 2000.

[97] X. Qiu and K. Chawla, “On the performance of adaptive modulation in cellular systems,” *IEEE Transactions on Communications*, vol. 47, no. 6, pp. 884–895, June 1999.

[98] K. Ramakrishnan and R. Jain, "Congestion avoidance in computer networks with a connecitonless network layer: Part IV: A selective binary feedback scheme for general topologies," Tech. Rep. DEC-TR-510, DEC, August 1987.

[99] R. Ramanathan and M. Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *ACM/Baltzer Mobile Networks and Applications*, vol. 3, no. 1, pp. 101–119, June 1998.

[100] T. S. Rappaport, *Wireless Communications: Principles and Practice.* Upper Saddle River, NJ: Prentice Hall, 1996.

[101] T. S. Rappaport, A. Annamalai, R. M. Buehrer, and W. H. Tranter, "Wireless communications: Past events and a future perspective," *IEEE Communications Magazine*, vol. 40, no. 5, pp. 148–161, May 2002.

[102] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, April 1999.

[103] S.-T. Sheu, Y. Tsai, and J. Chen, "MR$^2$RP: the multi-rate and multi-range routing protocol for IEEE 802.11 ad hoc wireless networks," *ACM/Baltzer Wireless Networks*, vol. 9, no. 2, pp. 165–177, March 2003.

[104] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols.* Boston, MA: Addison-Wesley, 1998.

[105] D. Sun and H. Man, "ENIC - an improved reliable transport scheme for mobile ad hoc networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 5, San Antonio, TX, pp. 2852–2856, November 2001.

[106] F. A. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part ii - the hidden terminal problem in carrier sense multiple-access modes and the busy-tone solution," *IEEE Transactions on Communications*, vol. COM-23, no. 12, pp. 1417–1433, December 1975.

[107] C.-K. Toh, "Associativity based routing for ad hoc mobile networks," *Wireless Personal Communications Journal*, vol. 4, no. 2, pp. 103–139, March 1997.

[108] T. Ue, S. Sampei, N. Morinaga, and K. Hamaguchi, "Symbol rate and modulation level-controlled adaptive modulation/TDMA/TDD system for high-bit-rate wireless data transmission," *IEEE Transactions on Vehicular Technology*, vol. 47, no. 4, pp. 1134–1147, November 1998.

[109] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed fair scheduling in a wireless LAN," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Boston, MA, pp. 167–178, August 2000.

[110] F. Wang and Y. G. Zhang, "Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response," in *Proceedings of the ACM International Conference on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, Lausanne, Switzerland, pp. 217–225, June 2002.

[111] R. W. Watson, "The Delta-t transport protocol: Features and experience," in *Proceedings of the IEEE Conference on Local Computer Networks (LCN)*, Mineapolis, MN, pp. 399–407, October 1989.

[112] W. T. Webb and R. Steele, "Variable rate QAM for mobile radio," *IEEE Transactions on Communications*, vol. 43, no. 7, pp. 2223–2230, July 1995.

[113] XTP Forum, *Xpress Transport Protocol Specification: XTP Revision 4*, March 1995. accessed: November 2004, http://www.ca.sandia.gov/xtp.

[114] S. Xu and T. Saadawi, "Revealing and solving the TCP instability problem in 802.11 based multi-hop mobile ad hoc networks," in *Proceedings of the IEEE Vehicular Technology Conference (VTC) – Fall*, Atlantic City, NJ, pp. 257–261, October 2001.

[115] Y. Yi, T. J. Kwon, and M. Gerla, "A load aware routing (LWR) based on local information," in *Proceedings of the IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, vol. 2, San Diego, CA, pp. G–65–G–69, September 2001.

[116] J. Zyren and A. Patrick, "Tutorial on basic link budget analysis," Tech. Rep. AN9804.1, Intersil, Milpitas, CA, June 1998.

# VITA

Gavin Douglas Holland

HRL Laboratories, LLC
3011 Malibu Canyon Road
Malibu, CA 90265

## Education

Ph.D., Computer Science, Texas A&M University.
M.S., Computer Science, Texas A&M University.
B.S., Computer Science, Abilene Christian University.

## Recent Publications

Gavin D. Holland and Nitin H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *ACM/Baltzer Wireless Networks*, vol. 8, no. 2, pp. 275–288, March 2002.

Gavin D. Holland, Nitin H. Vaidya, and Paramvir Bahl, "A rate-adaptive MAC protocol for multi-hop wireless networks," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, pp. 236–251, August 2001.

Gavin D. Holland and Nitin H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Seattle, WA, pp. 219–230, August 1999.

Saad Biaz, Gavin D. Holland, Young-Bae Ko, and Nitin H. Vaidya, "Evaluation of protocols for wireless networks," in *Parallel and Distributed Processing Techniques and Applications*, vol. 4, Las Vegas, NV, pp. 2150–2156, July 1999.

Gavin D. Holland and Nitin H. Vaidya, "Impact of routing and link layers on TCP performance in mobile ad hoc networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, pp. 1323–1327, September 1999.