# NOTE TO USERS

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Guihua Jia

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Performance Evaluation of Congestion Control Protocols and Loss Differentiation Algorithms over Wireless Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Prof. A. Boukerche

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Prof. S. Shirmohammadi

Prof. G. Warner

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

# Performance Evaluation of Congestion Control Protocols and Loss Differentiation Algorithms over Wireless Networks

by

Guihua Jia

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For the M.A.Sc. degree in

Electrical and Computer Engineering

School of Information Technology and Engineering

Faculty of Engineering

University of Ottawa

*Your file Votre référence*
*ISBN: 978-0-494-49215-4*
*Our file Notre référence*
*ISBN: 978-0-494-49215-4*

# Abstract

Congestion control protocols for wireless networks must be efficient, TCP-friendly, and robust to random wireless loss. Based on these criteria we have evaluated several different congestion control protocols, such as TCP Westwood, TFRC, MULTFRC, RAP, and IFTP. In wireless network environments, most of these protocols do not work well since wireless losses are counted as congestion losses. Therefore, it is necessary to extend these congestion control protocols with end-to-end Loss Differentiation Algorithms (LDA). We evaluate several existing LDA schemes, including Biaz, mBiaz, Spike, ZigZag, ZBS, PLC, SPLD, and TD, with simulation results showing different drawbacks for each scheme.

We thus propose a new LDA scheme: the mSpike scheme. The mSpike scheme classifies the loss type according to the mean and deviation of the relative one-way trip time. The simulation results show that the mSpike scheme has better performance and fewer problems in most of the situations evaluated. We also test the combination of MULTFRC and mSpike under different wireless lossy environments. The simulation results show that they have a high utilization of the available bandwidth. Therefore, this combination would be a good choice for applications with high bandwidth requirements.

# Acknowledgements

I take great pleasure in thanking the many people who have helped me in my studies at the University of Ottawa. First, I would like to thank my supervisor, Professor Azzedine Boukerche, for his invaluable guidance and advice. I would like to thank Professor Shervin Shirmohammadi and Professor Gabriel A. Wainer for their careful reviewing on my thesis and serving on the exam committee, as well as Professor Voicu Groza for serving as the chair of the exam committee. I also would like to thank Dr. Richard Pazzi Werner for his kindly help in my research work.

Finally, I would like to extend my thanks to my parents, my husband and my kids, whose love and encouragement have always sustained me.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

The currently most popular TCP network is designed to provide connectionless packet service so that it is very flexible and robust. But its disadvantage is that it is difficult to "provide good service under heavy load"[56]. The continuous growth of the popularity of computer and network applications makes this problem more serious. Therefore, an efficient congestion control protocol is very important for these networks.

Usually, congestion control is performed in two parts of the network: the router and the end systems. The router uses queue management to deal with congestion. Traditionally, most routers use a simple drop-tail queuing policy. When the queue is full, any arriving packet will be dropped. There are also many Active Queue Management (AQM) schemes in the router, such as Random Early Detection (RED) [58], Random Exponen-

tial Marking (REM) [5], and so on. In the end systems, TCP is used most frequently for the congestion control.

The basic TCP protocol was first defined in RFC 793 (1981) [52] and did not have any congestion control mechanisms. In October 1986, first congestion collapse occurred due to the rapid growth of Internet [21]. RFC 2581 (1999) [43] later specified in detail four TCP congestion control algorithms: *slow start*, *congestion avoidance*, *fast retransmit* and *fast recovery*. These algorithms were initially proposed by Jacobson in [66, 65], and were first standardized to combine with TCP in [67].

The standard TCP uses a (1, 0.5) Additive Increase Multiplicative Decrease (AIMD) congestion control mechanism. The TCP sender increases *cwnd* by 1 packet per new ACK and decreases it to 1/2 of the previous *cwnd* upon congestion. General AIMD (GAIMD) in [53] is a general case AIMD($\alpha$,$\beta$) that additively increases *cwnd* by $\alpha$ and multiplicatively decreases *cwnd* by $\beta$. In [4], a Dynamic TCP-friendly AIMD (DTAIMD) algorithm is proposed that adjusts $\alpha$ adaptively and dynamically based on the current *cwnd* to enhance the TCP-friendliness of GAIMD($\alpha$,$\beta$) [4].

TCPs send data at a highly fluctuating rate [4]. On the other hand, TCPs also provide reliable services with retransmission, which is not necessary for most multimedia applications. That is because the multimedia streaming is tolerant to certain data losses, but is very sensitive to delay [69].

User Datagram Protocol (UDP) [51] is commonly used for multimedia applications. However, a major drawback of UDP is that it has no any form of congestion control, a

schedule that is very important for preventing the network from becoming fully congested. If many people transmit video streaming over the internet at the same time without using any congestion control method, the routers would overflow and the Internet would become totally blocked. Therefore, the lack of congestion control is a serious problem in UDP [2].

Many protocols have been proposed to improve the congestion control performance, such as the Video Transport Protocol (VTP) [71], TCP-Friendly Rate Control (TFRC) [29, 28], Internet Friendly Transport-level Protocol (IFTP) [22, 24, 23], Rate Adaptation Protocol (RAP) [54, 15], Multiple TFRC (MULTFRC) [17, 19], and so on.

All of the protocols listed above reduce their sending rate in the presence of packet losses since they treat them as an indication of congestion. However, in a mixed wired and wireless network, packet loss can be due to error-prone wireless links and not be related to the network congestion at all. Thus, an efficient Loss Differential Algorithm (LDA) would be very helpful for these protocols if it could properly distinguish most wireless losses from losses due to congestion. Any packet losses classified as wireless losses do not need to be counted by the congestion control protocol. For example, the TFRC receiver does not need to include them in its calculation of the loss event rate. Therefore, the congestion control protocol will not dramatically decrease its sending rate when wireless losses are present in the network.

In the past few years, many kinds of LDA schemes have been proposed. Among them, the following end-to-end solutions are the most popular and most efficient: Biaz

[9], mBiaz [14], SPLD [50], Spike [14], PLC [36], TD [20], ZigZag [14], and ZBS [14]. But these schemes also have different problems and limits.

## 1.2 Contributions

The main contribution of this thesis is the design and implementation of a new LDA scheme, mSpike, to improve support for congestion control protocols over wireless networks. The works we have done include the following:

- The implementation and evaluation of several existing congestion control protocols, such as TCP Westwood [12], TFRC [29, 28], IFTP [22, 24, 23], RAP [54, 15], and MULTFRC [17, 19]. These protocols are implemented in the *ns*2 - network simulator. Through a simulation study, their performance is evaluated in a mixed wired and wireless network topology. The simulation results show that, in all the conditions tested, TFRC and MULTFRC can achieve a higher throughput than others, while most protocols have good TCP-friendliness. However, none of these protocols works well with lossy wireless links.

- The implementation and evaluation of several existing end-to-end Loss Differential Algorithms (LDA), such as Biaz [9], mBiaz [14], SPLD [50], Spike [14], PLC [36], TD [20], ZigZag [14], and ZBS [14]. These algorithms are implemented in the *ns*2 - network simulator. These LDA schemes can work together with the congestion control protocols to improve their performance in wireless lossy networks. However,

the analysis and simulation results show that they have different problems and limits.

- The design and implementation of a new LDA scheme, a modified Spike scheme (mSpike), to assist in packet loss classification. The mSpike scheme is based on the mean and deviation of the Relative One-way Trip Time (ROTT). Our simulation results show that mSpike is a very good LDA scheme when it works with TFRC. Furthermore, we also extended MULTFRC with the mSpike scheme. The simulation results show that this combination can achieve high bandwidth utilization even in environments with high wireless error rates.

## 1.3 Thesis Organization

This thesis is organized as follows.

In Chapter 2, we define and briefly survey different TCP and TCP-friendly congestion control protocols. They are TCP Westwood, TFRC, MULTFRC, RAP, and IFTP. Compared to existing surveys, we pay more attention to the performance of these protocols over wireless lossy networks.

Chapters 3 discusses the loss differentiation algorithms. Many previous works are introduced, including the Biaz, mBiaz, SPLD, ZigZag, ZBS, Spike, PLC, and TD schemes.

In Chapter 4, we propose a modified Spike scheme (mSpike) to assist in packet loss classification. It is based on the mean and deviation of the relative one-way trip time,

and avoid the drawbacks of previously proposed LDA schemes.

In Chapter 5, the parameters and network topology used in our simulation are introduced. We also introduce the wireless error models.

All the simulation works we have done are introduced in Chapters 6 and 7. The simulation tool we have used is the *ns*2 - a Network Simulator. The congestion control protocols introduced in Chapter 2, the LDA schemes in Chapter 3, and the proposed mSpike scheme in Chapter 4 have all been evaluated and tested with *ns*2.

In Chapter 8, we present a conclusion for this thesis and briefly discuss the future work needed on this topic. TCP-friendly congestion control protocols combined with loss differentiation algorithms over wireless networks are an attractive area of research and there are many issues still waiting to be investigated in the future.

# Chapter 2

# Congestion Control Protocols over Wireless Networks

## 2.1 Problem Formulation

Transmission Control Protocol (TCP) is a reliable end-to-end window-based transport protocol. Indeed, the majority of the traffic on the Internet is controlled by it [11]. TCP additively increases the congestion window when no congestion is detected, and multiplicatively decreases the window when congestion is detected [43]. TCP assumes that every packet loss is due to congestion. However, in wireless networks, a noisy, fading radio channel is the main cause of packet loss. As a consequence, TCP reduces its transmission rate drastically when wireless losses are present [12].

For improving TCP's performance in lossy wireless networks, Claudio et al propose

the TCP Westwood (TCPW) protocol in [12]. This protocol only shows some modification on the sender side of the original TCP Reno and NewReno protocols. The TCP Westwood sender continuously measures the end-to-end bandwidth it has used during the connection. When congestion is detected, it adjusts the congestion window and slow start threshold according to the estimated bandwidth value. The congestion is indicated by three duplicate acknowledgments or a timeout [12].

However, the improvement of TCP Westwood over the earlier protocols is very limited. On the other hand, TCP and TCP variants provide reliable services with retransmission, this is usually not necessary for most multimedia applications.

The UDP protocol for multimedia applications does not have any kind of congestion control algorithm. Therefore, whenever there is congestion in the network, the UDP flows do not respond by lowering their throughputs. Instead, they would be very aggressive and might starve other TCP and TCP-friendly flows [2].

The popularity of multimedia applications and realtime games is growing continuously. Since neither TCP nor UDP provide good service for multimedia applications, many other protocols have been proposed as follows.

The Rate Adaptation Protocol (RAP) first introduced in [54] is a TCP-friendly rate-based protocol for non-reliable realtime streams. RAP uses an Additive Increase and Multiplicative Decrease (AIMD) algorithm to adjust the transmission rate. In [15], some modifications are proposed to improve RAP's performance in a lossy wireless network.

The window-based AIMD approaches used by TCP will cause burstiness when the

congestion window (*cwnd*) increases or decreases rapidly [10]. Rate-based protocols, like RAP, can avoid this problem by sending packets out evenly. Therefore, they usually produce flows that are smoother than those in window-based schemes [4].

An Internet Friendly Transport-level Protocol (IFTP) is proposed in [22] by ElAarag and Bassiouni. IFTP is also a rate based flow control protocol and is similar to RAP. However, RAP only emulates TCP in the steady state since it uses AIMD, which is a steady-state model, while IFTP can imitate all the stages of TCP [22]. IFTP-W is introduced in [24] to improve the performance of IFTP in wireless lossy environment. It distinguishes between application data that is error sensitive and error insensitive. Another paper [23] discusses further how the error-sensitive section length affects on IFTP-W's performance. In a recent paper [35], ElAarag et al propose IFTPV to enhance the IFTP's performance by being faithful to TCP Enhanced Veno [72, 32].

TCP-Friendly Rate Control (TFRC) is another rate-based congestion control mechanism. It is designed for "unicast flows operating in a besteffort Internet environment" [29, 28]. TFRC flows are "reasonably fair when they compete for bandwidth with TCP flows" [30]. Another important character of TFRC is that it has a much lower fluctuation of throughput than other protocols. Therefore, it is more suitable for multimedia applications for which a relatively smooth sending rate is important [31].

In [17] and [19], Multiple TFRC (MULTFRC) has been proposed to open multiple simultaneous TFRC connections between a sender and a receiver. MULTFRC not only avoids modifications to the network infrastructure, but also results in almost a full

utilization of the wireless channel.

As the usage of wireless networks increases, the need to develop a good protocol for the use of multimedia applications over wireless networks also increases. In the remainder of this chapter, we will introduce the protocols mentioned above in more detail. Their performance over wireless networks will also be evaluated with the network simulator *ns2* [1].

## 2.2 TCP-Friendly Congestion Control Protocols

### 2.2.1 TCP Westwood

TCP Westwood (TCPW) is a sender-side only modification of the TCP Reno and NewReno congestion control protocol that improves upon the performance of TCP in mixed wired and wireless networks [16]. TCP Westwood is based on an end-to-end bandwidth estimation to set the congestion window (*cwnd*) and slow start threshold (*ssthresh*) after either three duplicate acknowledgments or a timeout [12]. Many papers [12, 33, 40, 34, 45, 13] discuss this protocol, and more information can be found on this website [49]. TCP Westwood+ is an evolution of TCP Westwood. It copes with the ACK compression effect, which may cause TCP Westwood to overestimate the bandwidth [34].

The following is an introduction to TCP Westwood according to [12].

### The end-to-end bandwidth estimate

The end-to-end bandwidth estimation is the key idea of TCP Westwood, which uses it to control the congestion window and slow start threshold. The TCPW sender performs the bandwidth estimation by monitoring the returning ACKs [12].

Let us assume that the TCPW sender receives an ACK at time $t_i$, with the information that the TCPW receiver has received $d_i$ bytes data after the last ACK sent. Thus, we get one bandwidth sample $b_i$ of this connection as follows:

$$b_i = \frac{d_i}{t_i - t_{i-1}} \tag{2.1}$$

where $t_{i-1}$ is the time at which the previous ACK was received at the TCPW sender [13].

A discrete-time filter has been used on these bandwidth samples $b_i$ and $b_{i-1}$ for filtering out the high-frequency components and the noise due to delayed acknowledgments [13]. At time $t_i$, the filtered estimate of the available bandwidth $\hat{b}_i$ is shown as follows:

$$\hat{b}_i = \alpha_i \hat{b}_{i-1} + (1 - \alpha_i)(\frac{b_i + b_{i-1}}{2}) \tag{2.2}$$

where $\hat{b}_{i-1}$ is the last filtered estimate of the available bandwidth at time $t_{i-1}$, $\alpha_i = (2\tau - \Delta_i)/(2\tau + \Delta_i)$, $\Delta_i = t_i - t_{i-1}$, $1/\tau$ is the cutoff frequency of the filter [12].

The coefficient $\alpha_i$ is dependent on the interarrival time $\Delta_i$. When $\Delta_i$ increases, $\alpha_i$ decreases. Thus, the last value $\hat{b}_{i-1}$ has less significance, but the last two samples $b_i$ and $b_{i-1}$ have more significance [12].

For example, considering a constant interarrival time $\Delta_i = t_i - t_{i-1} = \tau/10$, we get:

$$\hat{b}_i = \frac{19}{21}\hat{b}_{i-1} + \frac{2}{21}(\frac{b_i + b_{i-1}}{2}) \tag{2.3}$$

The new estimated bandwidth comes from about 90% of $\hat{b}_{i-1}$, and 10% of the average of $b_i$ and $b_{i-1}$ [12].

After a time $\tau/m(m \geq 2)$ without ACKs, the filter will assume that it has received a null sample $b_i = 0$. The delayed and cumulative ACKs also have an effect on the estimated bandwidth [12]. We will not discuss about the details here.

**The Westwood algorithm**

During the slow start and congestion avoidance phases, the congestion window (*cwnd*) updating scheme remains unchanged. It still functions identically to the standard TCP Reno or NewReno. Its difference from them is in how to set up the congestion window and slow start threshold is set up after a congestion is detected. The congestion is indicated by three DUPACKs or a timeout.

After three DUPACKs are received, the TCPW sender sets up the slow start threshold *ssthresh* according to Equation 2.4 [12].

$$ssthresh = (\hat{b}_i * RTT_{min})/seg\_size \tag{2.4}$$

where $\hat{b}_i$ is the bandwidth estimate, $RTT_{min}$ is the minimum measured round trip time, and *seg_size* is the length of the payload of a TCPW segment in bits [12, 34]. If ($cwnd > ssthresh$), then $cwnd = ssthresh$.

After the timeout timer expires, the TCPW sender also sets up the slow start threshold *ssthresh* according to the same equation 2.4. However, at the same time, the congestion window *cwnd* is set to 1. Therefore, the basic Reno and NewReno is still followed [12].

Whenever the TCPW sender detects a congestion (3 DUPACKs or timeout), it sets the *ssthresh* value according to the estimated available bandwidth as shown in Equation 2.4. In this way, it is unlike TCP Reno, which simply halves the current *ssthresh* values. Therefore, faster recovery is granted in TCPW, which can also have a better throughput [12].

In our simulation experiments, TCP WestwoodNR has also been evaluated. It is a sender-side modification of the most popular version of TCP: TCP NewReno.

### 2.2.2 RAP

The Rate Adaptation Protocol (RAP) was first introduced in [54]. In [15], some modifications were proposed to help RAP function well in a lossy wireless network. RAP uses an Additive Increase and Multiplicative Decrease (AIMD) algorithm, similar to that used in TCP, to adjust the transmission rate. Therefore, it is a fair and end-to-end TCP-friendly congestion control protocol. RAP is "well suited for unicast playback of realtime streams

and other semi-reliable rate-based applications" [54].

The primary implementation of RAP is on the sender side. The RAP sender uses timeout and acknowledgment (ACK) from the receiver to detect packet loss. All losses are considered to be congestion losses. If no congestion is detected, the RAP sender increases the transmission rate periodically. Otherwise, it decreases the transmission rate immediately after detecting congestion [54].

According to [54], the transmission rate $R$ of the RAP sender is changed by adjusting the inter-packet-gap ($IPG$) since $R = S_p/IPG$, where $S_p$ is the packet size. $IPG$ is adjusted according to Equation 2.5 for additively increasing the transmission rate, where $C$ is a constant of time that is selected depending on how often the rate will be changed. $IPG$ is adjusted according to Equation 2.6 for multiplicatively decreasing the transmission rate, where it usually chooses $\beta = 0.5$.

$$IPG = \frac{IPG * C}{IPG + C} \tag{2.5}$$

$$IPG = \frac{IPG}{\beta} \tag{2.6}$$

According to the suggestion in [44], rate-based protocols should not adjust their rates more than once per $RTT$ so that over-reacting to congestion can be prevented. On the other hand, however, changing the rate infrequently will lead to unresponsive behavior [54]. Thus, RAP adjusts the IPG once per $SRTT$ according to Equation 2.5, where

$C = SRTT$. $SRTT$ is the estimate of $RTT$ based on the Jacobson/Karel's algorithm as shown in Equation 2.7, where $\gamma = 0.5$.

$$SRTT = SRTT + \gamma * (RTT_i - SRTT) \tag{2.7}$$

In a low bandwidth environment, the long delay before an ACK packet is received will lead to oscillation in the transmission rates [15]. Thus, a fine-grained rate adaptation algorithm is used to make RAP more stable and responsive. According to [54], $FRTT$ is the short term exponential moving average as defined in Equation 2.8, where $\alpha = 0.9$, and $XRTT$ is the long term exponential moving average as defined in equation 2.9, where $\beta = 0.01$. The instant feedback of the network is defined as $Feedback = \frac{FRTT}{XRTT}$. Then, $IPG$ is modulated by this fine-grain feedback signal as shown in Equation 2.10.

$$FRTT = (1 - \alpha) * FRTT + \alpha * RTTi \tag{2.8}$$

$$XRTT = (1 - \beta) * XRTT + \beta * RTTi \tag{2.9}$$

$$IPG_{fine} = IPG * Feedback \tag{2.10}$$

Ref. [15] examines two possible problems in RAP when it is used in a lossy wireless network. One problem is that the transmission rate has the potential to increase explosively after a link error. The other problem is that the protocol could get stuck in a freeze mode [15]. The solutions for these problems are also proposed in this paper.

For the first problem, it uses an error ratio defined in Equation 2.11 to detect the link error. After the link error occurs, it limits $IPG$ by $IPG \geq IPG_{min}$ [15]. $IPG_{min}$ is defined in Equation 2.12, where $S_p$ is the packet size in bytes, and $S_{ack}$ is the ACK packet size.

$$error\_ratio = \frac{XRTT}{RTT} \tag{2.11}$$

$$IPG_{min} = \frac{RTT}{S_p + S_{ack}} * S_p * 0.9 \tag{2.12}$$

$IPG$ will be reset to normal after it detects a packet drop or the estimated backoff time as shown in Equation 2.13 [15].

$$T_{backoff} = \frac{XRTT}{S_p + S_{ack}} * S_p * 13.5 \tag{2.13}$$

A delayed-purge method is proposed for solving the second problem. The timeout packets will be marked as delayed-purge and kept in the transmission history. "They

will be cleared after an acknowledgment with sending time later than the delayed-purge packets is received" [15].

In our simulation experiments, the performance of RAP in a wireless environment has been evaluated. But the two problems discussed in [15] cannot be simulated so the solutions have not been verified in our simulation.

### 2.2.3 IFTP

An Internet Friendly Transport-level Protocol (IFTP) is proposed in [22] by ElAarag and Bassiouni. IFTP is a rate based flow control protocol for solving the TCP-friendly problem [22]. IFTP-W proposed in [24] is a variant of IFTP that is designed to improve the performance of IFTP in wireless lossy environments. IFTP-W allows the application to distinguish between error sensitive and error insensitive data [24]. Another paper [23] further shows how the error-sensitive section length can have an effect on IFTP-W's performance. In a recent paper [35], ElAarag et al propose IFTPV, which enhances the IFTP's performance by being faithful to TCP Enhanced Veno [72, 32].

IFTP is an Internet friendly protocol for multimedia applications. It uses a relaxed ACK, similar to the RAP protocol [22]. IFTP does not provide any retransmission for lost packets. The implementation of IFTP has four parts [22]: 1) Slow start, 2) Congestion avoidance, 3) Packet loss, and 4) Timeout.

Let us define the following parameters: $S_p$ is the packet size in bytes, $\tau$ is the transmission time of a single packet in seconds, $IPG$ is the inter-packet gap in seconds, $SRTT$

is the smoothed round trip time in seconds, $n$ is the number of packets successfully sent in an adjustment period that is equal to $SRTT$.

**Slow start phase**

During this phase, the sender recalculates the inter-packet gap $IPG$ according to Equation 2.14 in every round trip time $SRTT$. Assume that IFTP sends $n$ packets in the $i^{th}$ period. Then, the number of packets to be transmitted in the $i + 1^{th}$ period will be doubled [22].

$$IPG_{i+1} = \frac{SRTT}{2 * n} - \tau \tag{2.14}$$

When $n$ reaches a threshold (*ssthresh*), the IFTP sender exits the *slow start* phase and enters the *congestion avoidance* phase [22]. For being TCP-friendly, IFTP chooses the same *ssthresh* as TCP based on the bandwidth-delay product [38].

**Congestion Avoidance Phase**

In the congestion avoidance phase, the IFTP sender linearly increases the number of packets to be transmitted in every $SRTT$ [22]. The inter-packet gap $IPG$ is updated according to Equation 2.15. This continues until $n$ reaches the maximum window [22].

$$IPG_{i+1} = \frac{SRTT}{n + 1} - \tau \tag{2.15}$$

### Packet Loss

If packet loss is detected, the IFTP sender cuts the number of packets by half for being TCP-friendly. Then, the corresponding inter-packet gap $IPG$ is recalculated using Equation 2.16. At the same time, the IFTP sender also "halves the *ssthresh* value or sets it to 2, whichever is larger" [22].

$$IPG_{i+1} = \frac{2 * SRTT}{n} - \tau \tag{2.16}$$

The IFTP sender will leave the packet loss phase if an ACK is received with a larger sequence number than or equal to the last transmitted. If no ACK is received within a timeout period, the IFTP sender enters the timeout phase [22].

### Timeout

In this phase, the IFTP sender cuts the *ssthresh* value by half, and sets $n = 1$, $IPG_{i+1} = SRTT - \tau$ [22]. The timeout timer is set up according to Equation 2.17, where $RTTvar$ is defined in Equation 2.18, and $backoff$ is initially 1 and doubles after a timeout occurs. The maximum value of backoff is 64.

$$T_{timeout} = backoff * (SRTT + 4 * RTT_{var} + 1) \tag{2.17}$$

$$RTT_{var} = \frac{3}{4} * RTTvar + \frac{1}{4} * (RTT - SRTT) \tag{2.18}$$

There is no retransmission for lost packets in IFTP. Thus, IFTP can get a little higher bandwidth than TCP since the latter retransmits all lost packets.

IFTP interprets all packet drops as congestion losses, and thus reduces the transmission rate. As a consequence, it does not work well in wireless networks. IFTP-Wireless (or IFTP-W) is proposed in [24] to improve the performance of IFTP in a wireless environment.

IFTP-W divides the data packets into error sensitive and error insensitive sections. At the IFTP-W receiver, the corrupted packets are only dropped when the error occurs in error sensitive sections. When bit errors occur in error insensitive sections, the packet will not be dropped [24]. In [39], the same method has been proposed for use in UDP, and has proven to be useful.

In [23], the authors discuss the performance of IFTP-W when the length of the error-sensitive section varies. The paper's conclusion is that the IFTP-W's performance will increase when the length of the error-sensitive section decreases [23].

"Both IFTP and IFTP-W are designed to be faithful to TCP Reno so that they are TCP-friendly. But TCP Reno is inherently inefficient in wireless networks" [35]. In a recent paper [35], ElAarag et al propose IFTPV, which is based on TCP Enhanced Veno [72, 32]. The simulation results in [35] show that IFTPV outperforms IFTP and IFTP-W in wireless environments.

In our simulation experiments, only IFTP has been evaluated.

### 2.2.4 TFRC

TCP-Friendly Rate Control (TFRC) is a congestion control mechanism designed for "unicast flows operating in a besteffort Internet environment" [29, 28, 30, 31]. The TFRC version we discuss here and evaluate in my simulation is *RFC 3448, Proposed Standard, January 2003* [30]. A revision of RFC 3448 was proposed in March 2007 and can be found in [62]. "TFRC is reasonably fair when competing for bandwidth with TCP flows, but has a much lower variation of throughput over time compared with TCP, making it more suitable for applications such as streaming media where a relatively smooth sending rate is of importance" [31]. TFRC's sending rate is controlled by the loss event rate, which is estimated by the receiver.

According to the standard proposed in [30], TFRC works as follows:

- The receiver calculates the loss event rate $p_l$ and sends feeback message to the sender.
- The sender uses feedback messages to measure the round-trip time ($rtt$).
- Then, the sender uses the loss event rate $p_l$ and $rtt$ to calculate the acceptable transmit rate according to the throughput equation 2.20.
- The sender adjusts its transmit rate to match the calculated rate.

One of the most important functions of the TFRC receiver is to "perform an accurate and stable measurement of the loss event rate $p_l$" [30]. It is based on the detection of lost or marked packets according to the sequence numbers of received packets. It is defined that $p_l = 1/I_{mean}$, where $I_{mean}$ is the average loss interval. The $I_{mean}$ is calculated as follows [30]:

if $(i < n/2)$ $w_i = 1$;

else $w_i = 1 - (i - (n/2 - 1))/(n/2 + 1)$;

$I_{tot0} = 0$;

$I_{tot1} = 0$;

$W_{tot} = 0$;

for $(i = 0$ to $n - 1)$ {

$I_{tot0} = I_{tot0} + I_i * w_i$;

$W_{tot} = W_{tot} + w_i$;

}

for $(i = 1$ to $n)$ {

$I_{tot1} = I_{tot1} + I_i * w_{i-1}$;

}

$I_{tot} = max(I_{tot0}, I_{tot1})$;

$I_{mean} = I_{tot}/W_{tot}$;

At [30] states, "The receiver sends feedback packets to the sender at least once per $rtt$, or once per received data packet when the send rate is less than one packet per $rtt$. Whenever a new loss event is detected, the receiver will also send a feedback".

The TFRC sender estimates the round trip time $rtt$ according to the Equation 2.19, where $\gamma$ is a filter constant. The recommended value is $\gamma = 0.9$ [30].

$$rtt = \gamma * rtt + (1 - \gamma) * rtt_{new} \tag{2.19}$$

The throughput equation recommended for TFRC is shown in Equation 2.20, which is a simplified version of the throughput equation for Reno TCP from [37].

$$R_{calc} = \frac{S_p}{rtt * \sqrt{2 * n * p_l/3} + t_{RTO} * (3 * \sqrt{3 * n * p_l/8} * p_l * (1 + 32 * p_l^2))} \tag{2.20}$$

where:

$R_{calc}$ is the transmit rate in bytes/second.

$S_p$ is the packet size in bytes.

$rtt$ is the round trip time in seconds.

$p_l$ is the loss event rate between 0 and 1.0.

$t_{RTO}$ is the TCP retransmission timeout value in seconds.

$n$ is the number of packets acknowledged by a single TCP acknowledgment.

The TFRC updates the sending rate $R$ as follows [30]:

if $p_l > 0$, $R = max(min(R_{calc}, 2 * R_{recv}), S_p/t_{mbi})$;

else if $(t_{now} - t_{ld} >= rtt)$, $R = max(min(2 * R, 2 * R_{recv}), S_p/rtt)$; $t_{ld} = t_{now}$.

where $R_{recv}$ is the measured receiving rate, $t_{mbi}$ is 64 seconds.

More details about the TFRC protocol can be found in [30], which is the version we evaluated in our simulation.

### 2.2.5 MULTFRC

In [17, 19], Multiple TFRC (MULTFRC) is proposed. MULTFRC is designed to open multiple simultaneous TFRC connections between a sender and a receiver so that the available bandwidth can be fully utilized. In [47], a similar protocol called MULTCP is proposed that uses multiple TCP connections for the same application to "provide resilience against short-term insufficient bandwidth".

According to [57], the sending rate of a TCP connection can be expressed as that shown in Equation 2.21, where $R$ is the sending rate in bytes/s, $S_p$ is the packet size in bytes, $RTT$ is the round trip time in seconds, $p_l$ is the end-to-end packet loss rate, and $k$ is a constant factor between 0.7 and 1.3 [19]. The average throughput measured in the receiver will be $R * (1 - p_l)$.

$$R \leq \frac{k * S_p}{RTT * \sqrt{p_l}} \tag{2.21}$$

This model for TCP also can be used for TFRC. Figure 2.1 is a typical scenario for streaming over wireless connections [17]. It will be used for the following analysis. Let us define the available bandwidth of the wireless link as $B_w$. The end-to-end packet loss rate $p_l$ should be a combination of $p_w$ and $p_e$, as shown in Equation 2.22, where $p_w$ is the packet loss rate caused by wireless link error and $p_c$ is the packet loss rate caused by congestion.

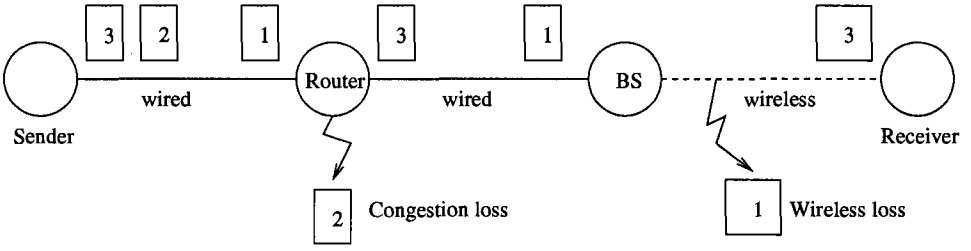


Figure 2.1: Typical scenario for streaming over wireless [17]

$$p_l = p_w + (1 - p_w) * p_c \tag{2.22}$$

If there is no congestion loss in the connection, we will get $RTT = RTT_{min}$, $p_l = p_w$. Then, the upper bound of the sending rate $R_{max}$ should be as follows [17]:

$$R_{max} = \frac{k * S_p}{RTT_{min} * \sqrt{p_w}} \tag{2.23}$$

If $R_{max} < B_w$, the wireless channel is underutilized. This is often the case. By using multiple connections, it can achieve optimal performance [17]. It has been proved in

[17, 19] that the optimal number of connections $n_{opt}$ should be that shown in Equation 2.24 for cases with the fixed packet size $S_p$.

$$n_{opt} = \lfloor B_w * \frac{RTT_{min} * \sqrt{p_w}}{k * S_p} \rfloor \tag{2.24}$$

"The practical implementation of the above strategy is to adjust the number of connections according to the measured round trip time" [17], as shown in Figure 2.2. In the MULTFRC sender, the RTT Measurement Module (RMM) measures the average round trip time $rtt_{ave}$ within a monitored interval. The computation of $rtt_{ave}$ is shown in Equation 2.25, where $m$ is the number of $rtt$ samples received within the monitored interval. Then the RMM sends this information to the Connection Controller Module (CCM) [17].

$$rtt_{ave} = \frac{\sum_{i=1}^{m} rtt_i}{m} \tag{2.25}$$

Based on the inputs from the RMM, the CCM adjusts the number of connections according to the following [17]:

$$n = \begin{cases} n - \beta & if\ rtt_{ave} - rtt_{min} > \gamma * rtt_{min} \\ n + \alpha/n & otherwise \end{cases}$$

where $\gamma$ is a preset parameter, and $n$ is the number of connections and should be an integer. Thus, in the implementation of MULTFRC, $n$ is "quantized to its closest integer" [17].

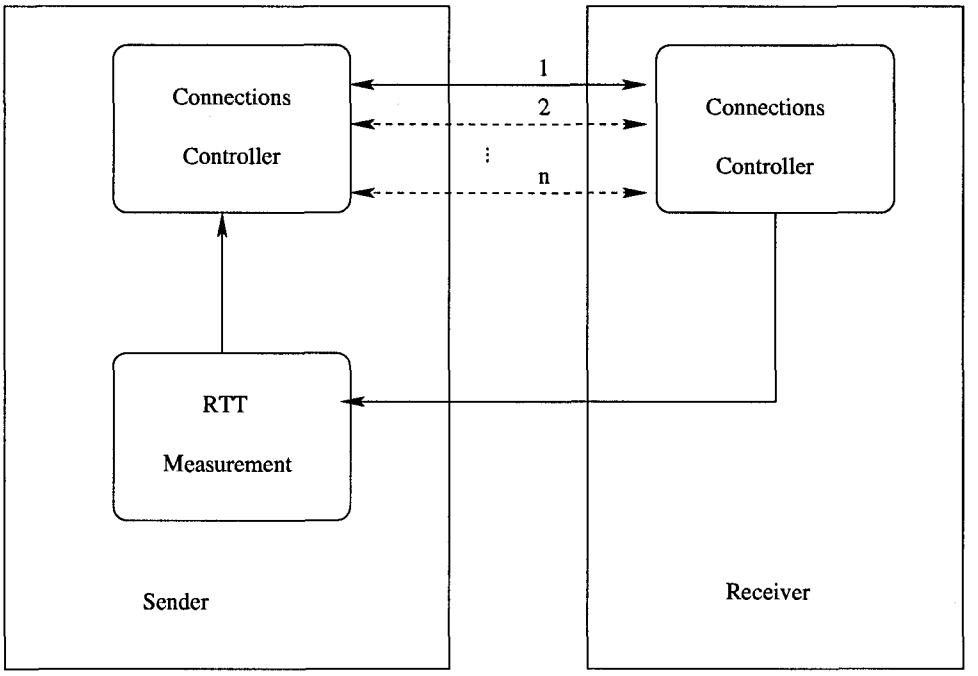


Figure 2.2: MULTFRC system framework [17]

MULTFRC is an end-to-end scheme, so it does not require any modification to the network infrastructure. All the necessary modifications can be done at the application layer. Therefore, it is easy to use MULTFRC for different applications. MULTFRC also makes possible the full utilization of the wireless bandwidth if the number of connections and the packet size are chosen properly [17].

However, MULTFRC also has some apparent drawbacks. Because the number of connections must be an integer, the quantization may result in low bandwidth utilization and large rate fluctuations. On the other hand, operating multiple TFRC connections in one application will have a large overhead. It may cause low utilization and also require more system resources, such as more memory space or higher power consumption [18].

In [18], Chen et al propose an alternative end-to-end approach to MULTFRC, called All-In-One TFRC (AIO-TFRC). AIO-TFRC physically creates only one connection, but the sender's transmit rate on this connection is increased to $n$ times of one normal TFRC rate. The $n$ is the optimal number of TFRC connections without any quantization. It could be a non-integer. Thus, the quantization effect can be avoided, and the overhead is also reduced in AIO-TFRC [19].

In our simulation experiments, only MULTFRC has been evaluated.

## 2.3 Summary

TCP Westwood is a modified TCP that works toward better performance in wireless lossy environments. But TCP and TCP variants provide unnecessary retransmissions for multimedia applications. UDP is more efficient for multimedia services, but it does not have any kind of congestion control.

RAP and IFTP are two similar rate-based TCP-friendly protocol. TFRC is a equation based congestion control mechanism. MULTFRC opens multiple simultaneous TFRC connections to get better bandwidth utilization.

All of these protocols are proposed for best-effort, unicast multimedia applications without retransmission. They treat any packet loss as a congestion loss. However, in a wireless network, packet loss is not always related to network congestion. Instead, it is mostly due to wireless link errors caused by noisy or fading wireless channels. Thus, most of these protocols do not work well in wireless environments. Although some further

modifications have been proposed for these protocols to improve their performance in wireless networks, the improvements are not very efficient.

In the following chapter, we will introduce some Loss Differentiation Algorithms (LDAs). They can be applied easily to these congestion control protocols for improving their performance in wireless environments.

# Chapter 3

# Loss Differentiation Algorithms

## 3.1 Overview

From the discussion in the last chapter, we know that almost all congestion control protocols reduce the sending rate when they detect the presence of packet loss. However, in a hybrid network in which the wired and wireless links are mixed together, the packet loss can be either congestion loss or wireless loss. The wireless loss could be caused by wireless channel fading, signal collision or interferences. Sometimes, the wireless loss rate can be even higher than the congestion loss rate. Since the wireless losses are not related to the network congestion at all, the congestion control protocol will overreact to these losses, thus making the network underutilized.

The Loss Differentiation Algorithm (LDA) is an algorithm aimed at discriminating between wireless losses and congestion losses. In recent years, several different LDA

schemes have been proposed. They use different techniques and work in different parts of the networks to classify the loss types. For a wireless network or wired/wireless hybrid network, it is necessary to extend the congestion control protocol with a LDA that can correctly tell the wireless losses from the congestion losses. If a packet loss is classified as a wireless loss, the congestion control protocol does not need to reduce the sending rate. For example, the TFRC receiver does not need to include a wireless loss in its calculation of the loss event rate. A Video Transport Protocol (VTP) was introduced in [71] and [70]. In [71], it is proven that a VTP equipped with a loss discrimination algorithm works efficiently in wireless networks.

In the past few years, many LDA schemes have been proposed including Biaz [9], mBiaz [14], SPLD [50], Spike [14], PLC [36], TD [20], ZigZag [14], and ZBS [14]. Among them, the Biaz and mBiaz methods in [14, 9] use packet inter-arrival time at the receiver to classify losses. SPLD (Statistical Packet Loss Discrimination) in [50] is a new proposed scheme that is based on the statistical value of the inter-arrival time of received packets. Spike in [14] and PLC (Packet Loss Classification) in [36] use two predefined thresholds on relative one-way trip time (ROTT) to differentiate losses. ZigZag in [14] classifies losses based on the mean and deviation of ROTT for different number of lost packets. ZBS in [14] is a hybrid algorithm that switches between the Biaz, mBiaz, ZigZag, and Spike schemes depending on different situations. The TD (Trend-and-Loss-Density-based) scheme proposed in [20] uses the trend of the ROTT curve and the loss density to determine the loss type.

There are also many other LDA solutions. In [27], Su proposes a fuzzy pattern recognition algorithm based on the Relative Oneway Delay (ROD) of the received packet to differentiate the loss type. In [59], Non Congestion Packet Loss Detection (NCPLD) is proposed that classifies the loss type according to a defined *kneepoint* value of the Round Trip Time (RTT). Flip Flop, proposed in [6], uses a Flip Flop filter to discriminate loss types based on the average and variance of RTT. A loss predictor, $Vegas$, is defined in [8] and is used to classify the loss type based on the RTT and throughput. In [55], some methods are proposed to enhance the performance of the NCPLD, Flip Flop, and Vegas schemes. In [7], Dhiman proposes a Bayesian approach to estimate the loss type, using "Maximum Likelihood Ratio tests to evaluate TCP as a classifier of the type of packet loss" [7]. In [42, 41], Stephane proposes a cross-layer LDA scheme that "acts at the TCP layer but uses 802.11 MAC parameter, $AckFailureCount$, to realize the targeted loss differentiation" [41].

All the LDA schemes we mentioned above are end-to-end solutions. If we want to add them to work together with the congestion control protocols, the modifications only need to be done at the sender and receiver sides. We do not need to change the routers or the intermediate nodes. In [60], a link-level truncated Automatic Repeat reQuest (ARQ) scheme is proposed to reduce the packet losses visible to the transport layer protocol. However, its computation is very complicated.

In the remainder of this chapter, we will give a detailed introduction to these end-to-end LDA schemes, specifically Biaz [9], mBiaz [14], SPLD [50], Spike [14], PLC [36],

TD [20], ZigZag [14], and ZBS [14]. Their performance is evaluated with the network simulator *ns*2. The details can be found in Chapter 7.

## 3.2 Previous Work

### 3.2.1 Biaz Scheme

The Biaz scheme proposed in [9] uses packet inter-arrival time to classify congestion loss and wireless loss. When the wireless link is the bottleneck, the base station (BS) will buffer the packets. As shown in Figure 3.1(a), the packet inter-arrival gap at the receiver will be the same $T$ when no packet is lost. Figure 3.1(b) shows that the inter-arrival gap between packet 1 and packet 3 is comparable to $T$ when packet 2 is lost due to congestion in the router. In Figure 3.1(c), packet 2 is lost when it is transmitted over the wireless link. In this situation, the inter-arrival gap between packet 1 and packet 3 is almost $2T$.

The Biaz scheme in [9] is based on these above observations from Figure 3.1. The number of consecutive packet(s) lost is $n$. $T_i$ is the instantaneous packet inter-arrival time of the first packet received after the loss. $T_{min}$ is the minimum packet inter-arrival time observed so far. If Formula 3.1 is satisfied, then these $n$ packets are counted as wireless loss packets. Otherwise, they are counted as congestion loss.

$$(n+1)T_{min} < T_i < (n+2)T_{min} \tag{3.1}$$

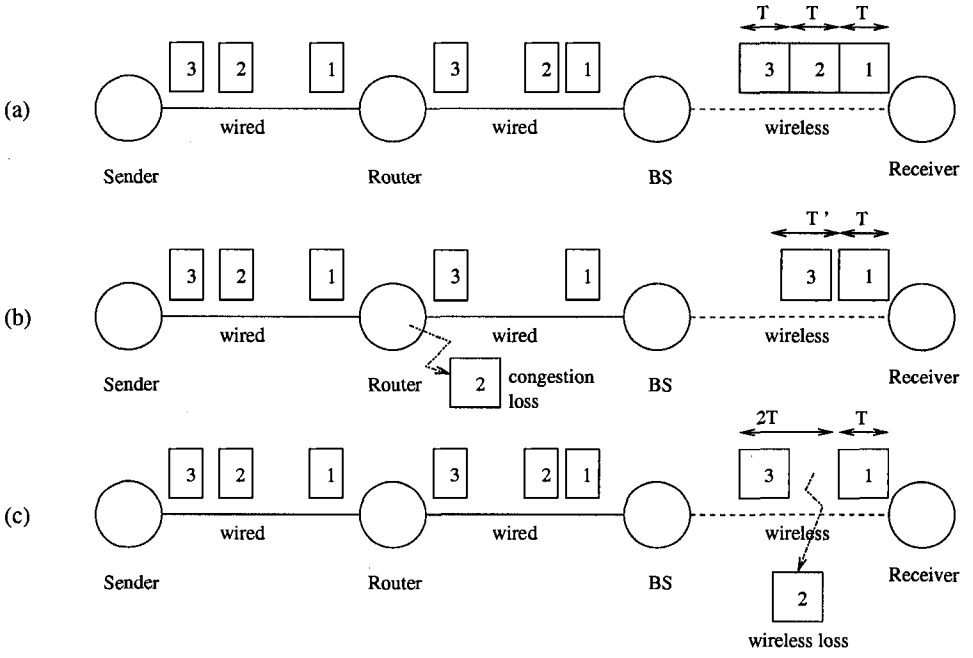


Figure 3.1: Inter-arrival gap [9]

The Biaz scheme only works in the case where the last hop of the connection is the only wireless link and the bottleneck among all links [9]. Another limit of this scheme is that it can classify the loss type only when a single stream exists on the connection path.

### 3.2.2 mBiaz Scheme

The mBiaz scheme proposed in [14] is a variant of the Biaz scheme that makes a slight change in the upper boundary. It assumes that if Formula 3.2 is satisfied, then these $n$ packets are counted as wireless loss packets. Otherwise, the losses are counted as losses due to congestion.

$$(n+1)T_{min} < T_i < (n+1.25)T_{min} \tag{3.2}$$

This modified Biaz scheme does not work well in our experiments since the upper limit is too strict.

### 3.2.3 SPLD Scheme

The SPLD scheme is newly proposed by Min et al in [50]. It is shown in Figure 3.2. It also uses the packet inter-arrival time to discriminate the loss type, like Biaz and mBiaz; however, the SPLD scheme has a lower misclassification rate than Biaz and mBiaz when there are multiple streams on the link [50].

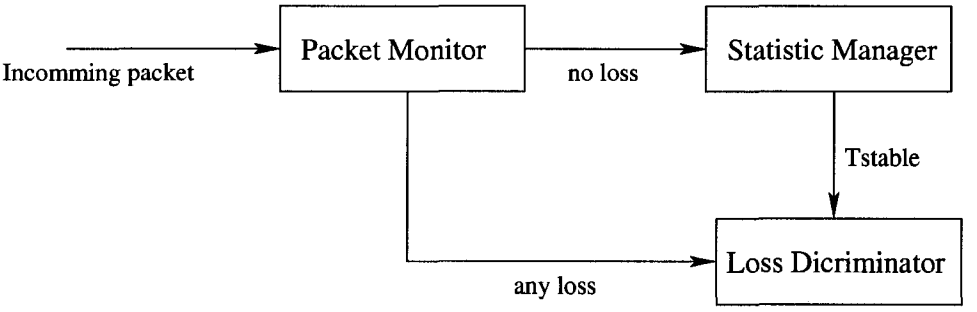


Figure 3.2: SPLD scheme [50]

According to [50], the SPLD scheme works in the following way:

When a packet is received by the receiver, the packet monitoring module will check its sequence number to see whether there is any packet loss. If no loss is detected, the statistic manager module updates the stable average inter-arrival time $T_{stable}$. In [50], no

detailed information is given about how to update the value of $T_{stable}$. In our simulation, it is updated according to Formula 3.3, where $\alpha = 1/32$. $T_i$ is the instantaneous packet inter-arrival time.

$$T_{stable} = (1 - \alpha) * T_{stable} + \alpha * T_i \tag{3.3}$$

If there is any packet loss, the loss discriminator module will classify the loss type according to the current average inter-arrival time $T_{cur}$. $T_{cur}$ is defined in Formula 3.4, where $n$ is the number of packet losses. If $T_{cur} \geq T_{stable}$, the losses are wireless losses. Otherwise, they are congestion losses.

$$T_{cur} = \frac{T_i}{n} \tag{3.4}$$

This SPLD scheme is based on the statistical values of the inter-arrival time of these received packets [50]. The main difference of the SPLD scheme from the Biaz and mBiaz schemes is that the SPLD scheme uses the stable average inter-arrival time $T_{stable}$ to classify the loss type. On the other hand, the Biaz and mBiaz schemes are based on $T_{min}$, which is the minimum packet inter-arrival time observed so far. Therefore, the SPLD scheme can discriminate the packet loss type in both single and multiple streams, while the Biaz and mBiaz schemes only work for a single flow.

### 3.2.4 Spike Scheme

The Spike scheme in [14] differentiates wireless losses from congestion losses according to the relative one-way trip time (ROTT). According to [14], the Spike scheme works as follows:

If the connection is currently not in the spike state, and the current *ROTT* is larger than *spike_start* as shown in Formula 3.5, the scheme enters the spike state. Otherwise, if the connection is currently in the spike state, and the current ROTT is less than *spike_end* as shown in Formula 3.6, then this scheme leaves the spike state [14]. This scheme is shown in Figure 3.3.

$$spike\_start = rott_{min} + \alpha * (rott_{max} - rott_{min}) \tag{3.5}$$

$$spike\_end = rott_{min} + \beta * (rott_{max} - rott_{min}) \tag{3.6}$$

The receiver classifies the loss based on the current state. If the packet loss happens during the spike state, it will be classified as a congestion loss; otherwise, it will be classified as a wireless loss [14]. In Formula 3.5 and 3.6, $rott_{max}$ and $rott_{min}$ are the maximum and minimum relative one-way trip times observed so far. The parameters $\alpha = 1/2$ and $\beta = 1/3$ are selected in our simulation .

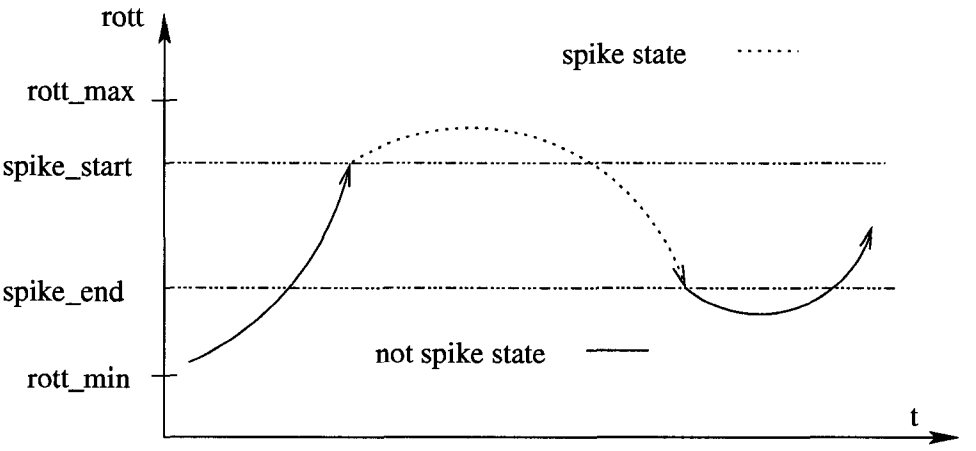


Figure 3.3: Spike Scheme [14]

### 3.2.5 PLC Scheme

The PLC scheme proposed in [36] is similar to the Spike scheme. It also differentiates the wireless losses from congestion losses according to the relative one-way trip time (ROTT). As shown in Figure 3.4, the PLC scheme works as follows according to [36]:

When the receiver finds packet losses according to the sequence number, it will check the *ROTT* of the received packet. If it is smaller than *spike_end* as shown in Formula 3.6, the losses are assumed to be wireless losses. If it is larger than *spike_start* as shown in Formula 3.5, however, the losses are classified as congestion losses.

For the *ROTT* between *spike_end* and *spike_start*, a trend detection algorithm based on Formula 3.7 runs to classify the loss type.

$$S_f = (1-\gamma)S_f + \gamma I(D_i > D_{i-1}) \tag{3.7}$$

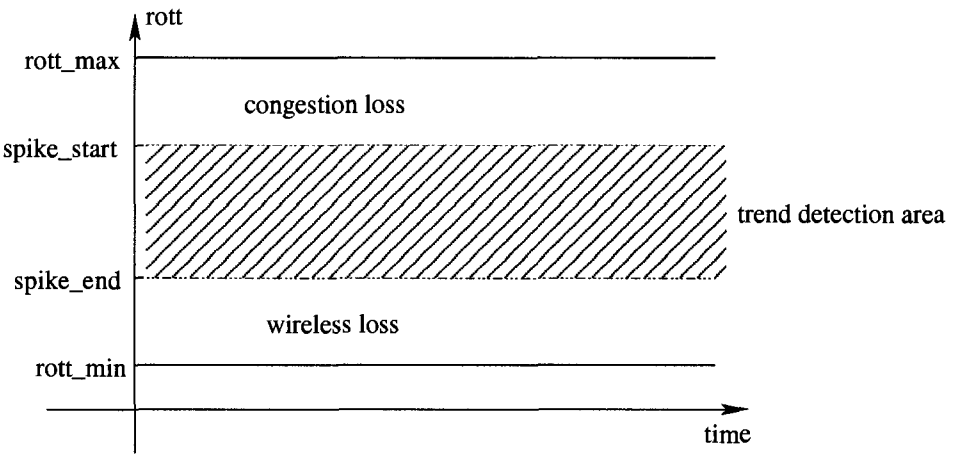


Figure 3.4: PLC Scheme [36]

where $D_i$ is the $ROTT$ of the $i^{th}$ packet; $I(X) = 1$ if $X$ is valid, otherwise it is 0; $\gamma$ is the smoothing factor.

$S_f$ will be from 0 to 1. If there is a strong decreasing trend, it will approach 0. If there is a strong increasing trend, it will approach 1. Otherwise, it will remain around 0.5. A threshold is defined as $S_{th}$. In our simulation, $S_{th} = 0.4$. If $S_f > S_{th}$, the packet loss is assumed to be congestion loss [36]. Otherwise, the packet loss is classified as a wireless loss. The parameters $\alpha = 0.8$ , $\beta = 0.3$, $\gamma = 1/30$ are selected in our simulation.

### 3.2.6 ZigZag Scheme

The ZigZag scheme described in [14] classifies loss type according to the number of losses $n$, ROTT, and the mean and deviation of ROTT. The mean $rott_{mean}$ and the deviation $rott_{dev}$ are defined in Formula 3.8 and Formula 3.9. $\alpha = 1/32$ is selected in our simulation.

$$rott_{mean} = (1 - \alpha) * rott_{mean} + \alpha * rott \tag{3.8}$$

$$rott_{dev} = (1 - 2\alpha) * rott_{dev} + 2\alpha * |rott - rott_{mean}| \tag{3.9}$$

According to [14], the packet loss is classified as wireless loss if

$$(n = 1 \bigcap rott_i < rott_{mean} - rott_{dev})$$

$$\text{OR } (n = 2 \bigcap rott_i < rott_{mean} - rott_{dev}/2)$$

$$\text{OR } (n = 3 \bigcap rott_i < rott_{mean})$$

$$\text{OR } (n > 3 \bigcap rott_i < rott_{mean} + rott_{dev}/2)$$

Otherwise, it is assumed to be a congestion loss [14].

### 3.2.7 ZBS Scheme

The ZBS scheme is a hybrid algorithm proposed in [14]. It is designed to choose different schemes based on whether the bottleneck wireless link is shared. According to [14], the ZBS scheme works as follows:

When $n$ flows share the bottleneck link, the average packet inter-arrival time $T_{avg}$ will be almost $n * T_{min}$, where $T_{min}$ is the minimum inter-arrival time [14]. $T_{avg}$ is updated according to Formula 3.10.

$$T_{avg} = 0.875 * T_{avg} + 0.125 * T_i/n \tag{3.10}$$

Let $T_{narr} = T_{avg}/T_{min}$. If $rott_i < rott_{min} + 0.05 * T_{min}$, the Spike scheme is used. Otherwise, it selects one of the base algorithms according to the value of $T_{narr}$ as shown in Figure 3.5.

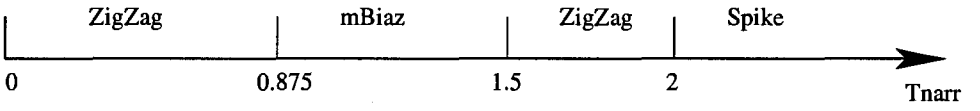


Figure 3.5: ZBS Scheme [14]

### 3.2.8 TD Scheme

The TD scheme in [20] is a Trend and Loss Density based scheme. It uses "1) the trend to indicate where the packet loss happens, i.e., if it is around the peak of the ROTT curve, and 2) the loss density to examine how often the packet loss happens, i.e., if there is any other packet loss around this loss" [20].

According to [20], the TD scheme works as follows:

Given a received packet sequence that contains $n$ packets, the least square method is used to approximate the $ROTT$ curve of these $n$ packets. Let us define the $ROTT$ curve as $f(x) = a + bx + cx^2$, where $x$ is the sequence number of the packet. The Equation 3.11 shows the least square error [46].

$$\sum_{i=1}^{n}[1-f(x_i)]^2=\sum_{i=1}^{n}[1-(a+bx_i+cx_i^2)]^2 \tag{3.11}$$

After the first derivatives of equation 3.11 in terms of all the unknown variables (a, b, c), we get the expanded equations 3.12, 3.13 and 3.14. Computations of unknowns (a, b, c) are done using matrices.

$$\sum_{i=1}^{n}1=a\sum_{i=1}^{n}1+b\sum_{i=1}^{n}x_i+c\sum_{i=1}^{n}x_i^2 \tag{3.12}$$

$$\sum_{i=1}^{n}x_i=a\sum_{i=1}^{n}x_i+b\sum_{i=1}^{n}x_i^2+c\sum_{i=1}^{n}x_i^3 \tag{3.13}$$

$$\sum_{i=1}^{n}x_i^2=a\sum_{i=1}^{n}x_i^2+b\sum_{i=1}^{n}x_i^3+c\sum_{i=1}^{n}x_i^4 \tag{3.14}$$

According to the $f(x)$ we receive and the position where the packet loss happened, the trend of the lost packets can be classified as Up, Down, Convex Up, Convex Down, Concave Up or Concave Down, as shown in Figure 3.6.

The loss density is measured according to the autocorrelation function $f_{cor}(S,j)$ of the lost packet $j$ as shown in Formula 3.15. $S$ is the received packet sequence with $m$ packets. $I(k)$ is 1 if the $k^{th}$ packet in $S$ is lost, otherwise it is 0.

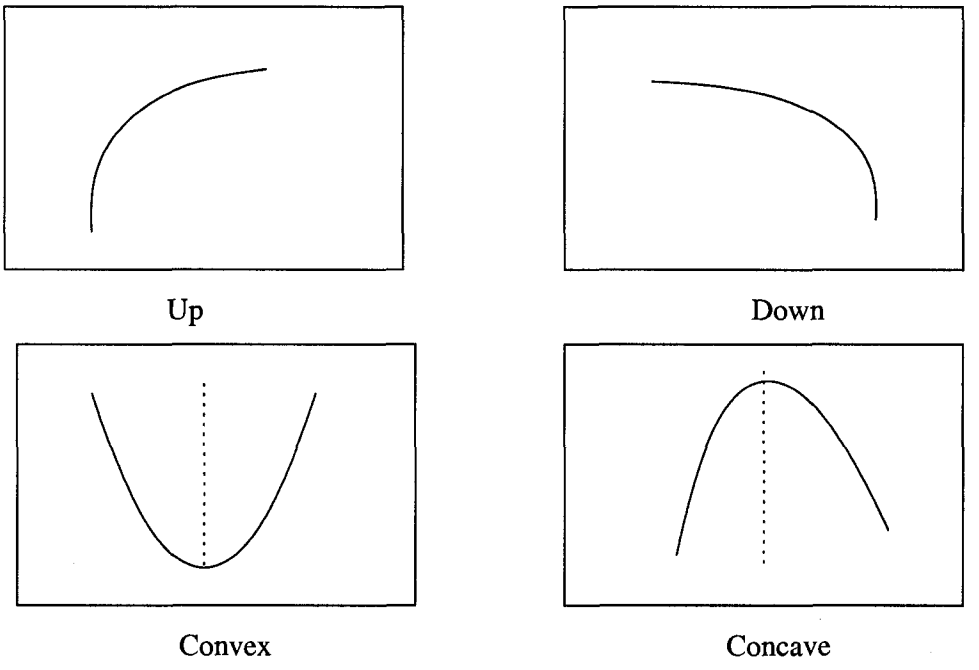


Figure 3.6: Type of Curve [20]

$$f_{cor}(S,i) = \frac{\sum I(k) * I(k-j)}{m} \tag{3.15}$$

The loss density of a lost packet $f_d$ is defined in Formula 3.16, where $S_i$ is the packet sequence that contains the lost packet $i$ at the center and has $r$ packets on both of its sides, $m = 2r + 1$. In our simulation, $r = 7$.

$$f_d(S_i,m) = \sum_{j=1}^{r} \frac{f_{cor}(S_i,j)}{m} \tag{3.16}$$

According to the trend and loss density, a different threshold $f_{th}$ has been applied to classify the loss type. Above the threshold $f_{th}$, the loss is regarded as congestion loss. 1)

If there is only one packet loss in $S_i$, it is always classified as wireless loss; 2) If the trend is Down, ConvexD or ConcaveD, we set up $f_{th} = 0$; 3) If the trend is ConvexU, $f_{th}$ is set equal to the loss density when there are 5 packets lost in the packet sequence $S_i$; 4) If the trend is Up or ConcaveU, $f_{th}$ is set as the loss density when there are 2 packets lost in the packet sequence $S_i$ [20].

## 3.3 Summary

The Loss Differentiation Algorithm (LDA) is dedicated to distinguish the wireless losses from congestion losses. The schemes described above use different techniques in different parts of the networks to discriminate the loss types.

The Biaz, mBiaz, and SPLD methods are all based on the inter-arrival time at the receiver side, while the Spike and PLC schemes use two thresholds based on relative one-way trip time to differentiate loss types. The ZigZag scheme is based on the number of lost packets and the difference of ROTT from its mean to discriminate the wireless losses from congestion losses. The ZBS scheme is a hybrid LDA scheme that switches between Biaz, mBiaz, ZigZag, and Spike schemes depending on different situations. Lastly, the TD scheme uses the trend of the ROTT curve, and the loss density to discriminate the loss type; however, its realization is very complicated.

# Chapter 4

# Proposed Solution - mSpike

## 4.1 Motivation

In a wireless or mixed wired/wireless network, congestion control protocols that mistreat wireless losses as congestion losses will cause a low utilization of the network bandwidth. This is because these protocols must reduce their sending rate after any congestion in order to be TCP-friendly. Extending the congestion control protocol with a good LDA scheme is very necessary for networks containing wireless links.

In the last chapter, we introduced some previous works on LDA schemes. Among them, the Biaz, mBiaz and SPLD methods are all based on the inter-arrival time at the receiver side [14, 50]. They do not need the timestamp information present in the received packets. Therefore, these schemes can have a smaller overhead than others that need the timestamp in the packet's headers. But the drawback of these schemes

is that they only work well in very limited situations. For example, all of them are based on the assumption that the last hop is a wireless link that is the bottleneck of the whole connection [14]. The Biaz and mBiaz schemes can not function at all when any competitive streaming is present. The SPLD scheme uses the statistical information of the inter-arrival time to classify the loss type, therefore, it still works when there are multiple streams. However, the accuracy of its classification is not very high.

The Spike and PLC schemes use two predefined thresholds on relative one-way trip time (ROTT) to differentiate wireless losses from congestion losses [14, 36]. These two important thresholds are totally based on the observed maximum and minimum ROTT values. Once an occasional maximum or minimum value has been reached by interference or anything else, they can no longer recover from it. On the other hand, these two schemes do not consider the wireless losses that can occur in the presence of high ROTT.

The ZigZag scheme is based on the number of lost packets and the difference between ROTT and its mean [14]. The number of lost packets is related to the congestion loss rate. In a severe congestion state with a high loss rate, the number of lost packets will be large. In a situation involving a low congestion loss rate, the number of lost packets will most likely be one. It is hard to find a distinct boundary for the number of lost packets related to congestion loss. The ZigZag scheme has a high misclassification rate for misclassifying wireless losses as congestion losses.

The ZBS scheme is a hybrid LDA scheme that switches between Biaz, mBiaz, ZigZag and Spike schemes depending on different situations [14]. Since it is difficult for a single

LDA scheme to work well under different conditions, it is a good idea to use different LDA solutions for different environments. However, this method results in another challenge regarding how to find a proper parameter to determine which LDA scheme should be used. The ZBS scheme has been proven in our simulation not to have a good performance.

The TD scheme proposed in [20] uses the trend of the ROTT curve, and the loss density to discriminate the loss type. The main drawback of this is that its realization is more complicated than other LDA schemes. It requires more memory to store the information, and more power to perform the calculations. These calculations would also cause longer delays.

Since all of the LDA schemes we have discussed do not have a satisfactory performance, we propose a modified Spike scheme (mSpike) to assist packet loss classification. The mSpike scheme is based on the mean and deviation of ROTT to decide the threshold values of spike states.

## 4.2 Background Knowledge

The Relative One-way Trip Time (ROTT) is the end-to-end delay of a packet traveling from a source node to a destination node. It is determined by measuring the difference between the timestamp in the received packet's header from the current time at the receiver side. Since some error may exist between the sender's clock and the receiver's clock, we call the delay measurement *relative* one-way trip time [14].

According to [64], the ROTT mainly includes four delay types: propagation delay,

transmission delay, processing delay, and queuing delay.

- Processing delay is the time used for examining the packet header, checking bit errors, and determining the output link.
- Transmission delay is the time spent transmitting the packet to the physical link.
- Propagation delay is the time required for the packet to travel through all of the physical links.
- Queuing delay is the time that the packet spends in the queue for transmission.

Furthermore, the processing delay and transmission delay are usually small. For a high-speed network, they are only a few microseconds. The propagation delay may vary from a few microseconds to hundreds of milliseconds. But for a given path and fixed packet size, the processing delay, transmission delay, and propagation delay are almost constant. In this situation, the main reason that ROTT varies is the variation of the queuing delay.

Additionally, the queuing delay varies depending on the congestion levels and queue sizes in all the routers of the connection. The queuing delay of a given path can vary significantly from packet to packet. It can range from milliseconds to microseconds in actual practice. If all the queues along the transmit path are empty and there is no other packet currently in transmission, then the packet's queuing delay is zero. On the other hand, if there is heavy traffic on the given path, then there will be many packets waiting to be transmitted. The congestion loss will happen when any queue is full along

the path. In this situation, the packet's queuing delay will be large and thus its ROTT will also be large. Therefore, we conclude that a large value of ROTT indicates that congestion may exist along the connection path.

On the other hand, the motivation of using one-way delay instead of round-trip delay to classify the loss type is based on the analysis performed in [3]: (1) The forward data packet may travel along the different paths from the reverse feedback packet ("asymmetric paths"). Therefore, different routers may be used for the forward and reverse packets. The round-trip delay will mix the performance of two distinct paths together. (2) "Even when the two paths are symmetric, they may have radically different performance characteristics due to asymmetric queuing" [3].

## 4.3 Proposed mSpike

Based on the above analysis, we propose a new loss classification scheme: mSpike. It works as shown in Figure 4.1. It is derived from the Spike scheme and ZigZag scheme in [14]. The original Spike scheme runs based on the maximum and minimum ROTT values observed so far. Therefore, it cannot recover from any incidental error that may be caused by wireless interference or anything else [14].

The mSpike scheme is based on the exponential mean of ROTT and its deviation. They are calculated according to Formula 4.1 and Formula 4.2, which are similar to those used in the ZigZag scheme. The exponentially averaged ROTT values used by mSpike are resistant to the incidental extreme values of ROTT. This is because only the most

recent values of ROTT affect the exponential mean of ROTT and its deviation according to Formula 4.1 and 4.2. The updating rate of mean and deviation of ROTT is controlled by the exponential decaying factors $\alpha$ and $\beta$ in Formula 4.1 and 4.2. The occasional extreme value will only have an effect within a limited time.

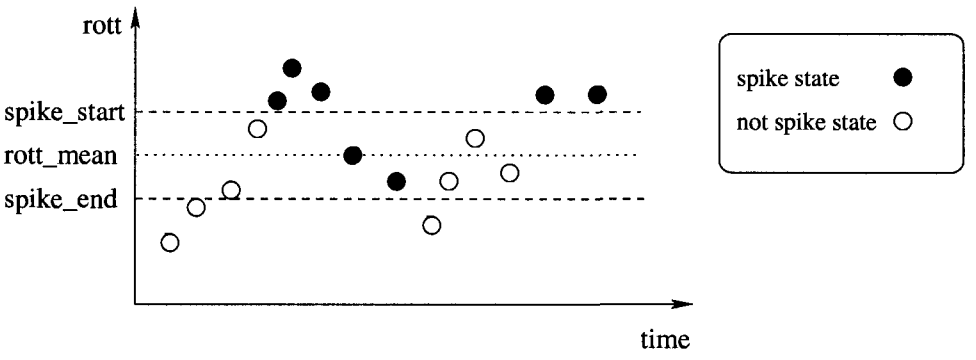


Figure 4.1: The mSpike Scheme

$$rott_{mean} = (1 - \alpha) * rott_{mean} + \alpha * rott \tag{4.1}$$

$$rott_{dev} = (1 - \beta) * rott_{dev} + \beta * |rott - rott_{mean}| \tag{4.2}$$

In my simulation, $\alpha = 1/64$, $\beta = 1/16$ are selected. We experimented with different values of $\alpha$ and $\beta$. Finally, we found $\alpha = 1/64$ and $\beta = 1/16$ provide good results in most of the situations we simulated. Actually, they can be different values according to different network environments. If the ROTT values have large fluctuations, it would be better to choose smaller $\alpha$ and $\beta$ values so that we can get a smoother mean and

deviation of ROTT. If the variation of ROTT values is very small, choosing larger $\alpha$ and $\beta$ would be better.

The mSpike scheme sets up the threshold values of spike states according to the mean and deviation values of ROTT determined in Formula 4.1 and Formula 4.2. The *spike_start* is calculated using the equation shown in Formula 4.3. The *spike_end* is calculated using the equation shown in Formula 4.4, where the parameters $\gamma$ and $\lambda$ are values within [0,1]. They are set up to 1/4 in our simulation.

$$spike\_start = rott_{mean} + \gamma * rott_{dev} \tag{4.3}$$

$$spike\_end = rott_{mean} - \lambda * rott_{dev} \tag{4.4}$$

In the mSpike scheme, the spike state is decided similarly to that in the Spike scheme [14]. When a packet has been received at the receiver, the receiver calculates this packet's ROTT value *rott* according to the current time and the time stamp in the header of the received packet. If $rott > spike\_start$, this scheme is in the spike state. If $rott < spike\_end$, this scheme is in the non-spike state. If $spike\_end \leq rott \leq spike\_start$, this scheme keeps its last state unchanged. This means that if the scheme was in the spike state when the last packet was received, it is still in spike state now; conversely, if it was in the non-spike state, it remains in the non-spike state.

When a packet loss is detected at the receiver, the receiver classifies the loss type according to the current spike states. If the packet loss happens in the non-spike state, it is assumed to be a wireless loss. If it happens in the spike state, however, the receiver checks the loss density. If only one packet is lost within a distance of $n$ packets, it is regarded as a wireless loss. Otherwise, multiple losses are considered to be congestion losses. This is because congestion loss usually happens in bursts. In our simulation, $n = 20$.

As those used in the Spike scheme [14], the parameters $\gamma$ and $\lambda$ in Formula 4.3 and Formula 4.4 can also be adjusted according to different practical requirements. If $\gamma$ is larger or $\lambda$ is smaller, the probability of misclassifying a congestion loss as a wireless loss will be larger. A higher throughput would be achieved. But, as a consequence, the congestion loss rate could also be higher. This situation is suitable for applications that require high throughput, low delay, and have high tolerance for packet losses. However, this may cause problems with TCP-friendliness. On the other hand, if $\gamma$ is smaller or $\lambda$ is larger, the probability of misclassifying wireless losses as congestion losses would be larger. This will cause low efficiency in the loss differentiation. It means that most wireless losses could not be classified. Therefore, it would cause a low utilization of the available wireless network bandwidth. In our simulation, $\gamma$ and $\lambda$ are set up as $1/4$, which provides good results in most of the situations we simulated.

# Chapter 5

# Simulation Parameters

## 5.1 Topology and Network Parameters

We use the Network Simulator ns-2.30 [1] to evaluate the performance of these congestion control protocols and loss differentiation algorithms.

### Topology

The basic network topology used in the simulation is shown in Figure 5.1. The links from the senders to the Router and the link from the Router to the Base Station are all wired links. The last links from the Base Station to the receivers are wireless links. The MAC layer protocol running on the wireless links is $IEEE802.11b$. Thus, these wireless links share the wireless bandwidth.

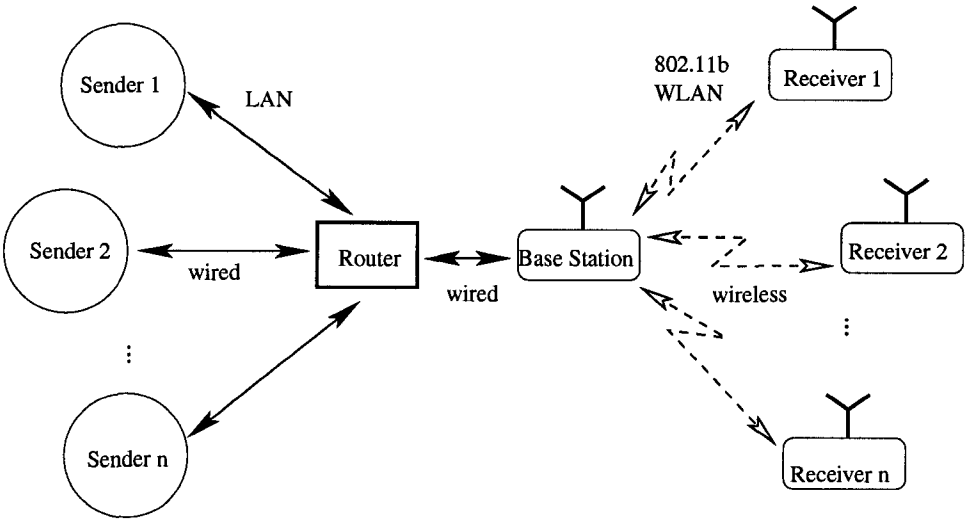


Figure 5.1: Simulation Topology

### Bandwidth

We tested these protocols and LDA schemes with n=1,2,4,6 or 8 traffic flows in the network. In this way, the n streams compete for bandwidth from the Router to the Base Station. The bandwidth of all the wired links is 100 Mbps. The bandwidth of the wireless channel is set to either 2Mbps or 11 Mbps. The congestion can happen both at the Router and at the Base Station.

### Delay

The total propagation delay of the wired links is from 10 ms to 150 ms.

### Packet Size

The packet size is 1000 bytes.

### Queue

The queuing policy is DropTail. The size of the queue in the router is 50 packets, while the size of the queue in the BS is 10 packets.

### Test Conditions

The simulation time is 500 seconds. Every set of experiments was repeated 5 times with different random seeds, and the results were averaged. For different Congestion Control Protocols and LDA schemes, the same random seeds were used.

The parameters set up in out *ns2* code are as follows:

*set opt(chan) Channel/WirelessChannel ;# channel type*

*set opt(prop) Propagation/TwoRayGround ;# radio-propagation model*

*set opt(netif) Phy/WirelessPhy ;# network interface type*

*set opt(mac) Mac/802_11 ;# mac layer protocol*

*set opt(ifq) Queue/DropTail/PriQueue ;# interface queue type*

*set opt(ll) LL ;# link layer type*

*set opt(ant) Antenna/OmniAntenna ;# antenna model*

*set opt(ifqlen) 50 ;# max packet in ifq*

*set opt(adhocRouting) NOAH ;# routing protocol*

*set opt(cp) "" ;# connection pattern file*

*set opt(sc) "" ;# node movement file*

*set opt(x) 100 ;# x coordinate of topology*

*set opt(y) 100 ;# y coordinate of topology*

*Mac/802_11 set RTSThreshold_ 3000 ;# disable RTS/CTS*

*Mac/802_11 set ShortRetryLimit_ 0 ;# disable retransmit*

*Mac/802_11 set LongRetryLimit_ 0*

*#DSSS (IEEE802.11b):*

*Mac/802_11 set SlotTime_ 0.000020 ;# 20us*

*Mac/802_11 set SIFS_ 0.000010 ;# 10us*

*Mac/802_11 set PreambleLength_ 144 ;# 144 bit*

*Mac/802_11 set PLCPHeaderLength_ 48 ;# 48 bits*

*Mac/802_11 set PLCPDataRate_ 1.0e6 ;# 1Mbps*

*Mac/802_11 set dataRate_ 11.0e6 ;# 11Mbps*

*Mac/802_11 set basicRate_ 1.0e6 ;# 1Mbps*

## 5.2 Wireless Loss Model

There are two types of error models that have been used in our simulation. One is the uniform error model, while the other is the two-state Markov error model.

### Uniform error model

The uniform error model is set up as the following *tcl* code:

```
proc UniformErr {} {
    global rw
    set err [new ErrorModel]
    $err unit pkt
    $err set rate_ $rw
    $err ranvar [new RandomVariable/Uniform]
    $err droptarget [new Agent/Null]
    return $err
}
```

### Two-state Markov chain model

Gilbert/Elliot's two-state Markov chain model [63, 26, 25] is a popular model for simulating the bursty nature of wireless multipath fading channels [36]. It is shown in Figure 5.2, where $P_{xy}$ is the probability of transition to the state $y$ from the current state $x$ [36].

The Markov model assumes that there are two states, good and bad. When the packet is in the good state, there is no packet loss. When it is at the bad state, however, the packet loss probability is $\alpha$. The following is the code to set up the Markov error model in our simulation.

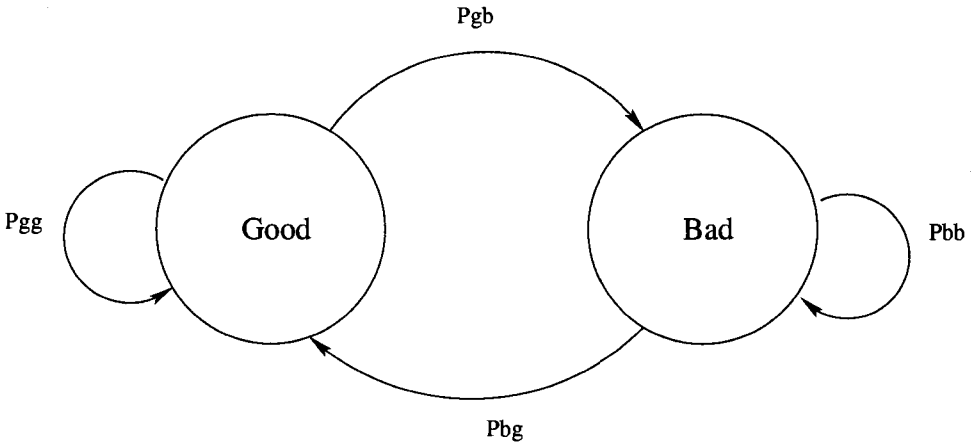


Figure 5.2: Markov Model [36]

```
proc TwoStateMarkovErr {} {
    global rw
    set tmp0 [new ErrorModel/Uniform 0 pkt]
    set tmp1 [new ErrorModel/Uniform 0.25 pkt]
    # Array of states (error models)
    set m_states [list $tmp0 $tmp1]
    # Durations for each of the states, tmp1 and tmp2, respectively
    set m_periods [list 1 0.5]
    # Transition state model matrix
    set m_transmx { {0.805 0.195} {0.78 0.22} }
    set m_trunit pkt
    # Use time-based transition
```

```
    set m_sttype time
    set m_nstates 2
    set m_nstart [lindex $m_states 0]
    set em [new ErrorModel/MultiState $m_states $m_periods $m_transmx $m_trunit
            $m_sttype $m_nstates $m_nstart ]
    return $em
}
```

These two wireless error models are set up in our *tcl* code as follows:

```
if {$error_t == 0} {
    $ns_ nodeconfig IncomingErrProc UniformErr
} else {
    $ns_ nodeconfig IncomingErrProc TwoStateMarkovErr
}
```

These models are inserted over the incoming wireless channels [48]. All of the wireless error rates used in our simulations are based on packets. The retransmission in the MAC layer has been disabled so that we can determine the exact wireless error rate as we set up.

# Chapter 6

# Simulation of Congestion Control Protocols over Wireless Network

## 6.1 Throughput

### 6.1.1 Throughput with One Flow

The network topology is shown in Figure 5.1. First, we measure the throughputs when there is only one connection between *sender*1 and *receiver*1. We ran the different congestion control protocols on the network, varying the wireless link error rate between 0%, 1%, 2%, 4% and 8%. The error model assumed here is the uniform error model in which the time between back-to-back errors is uniformly distributed.

Figures 6.1, 6.2 and 6.3 show the results of the average throughput versus error rate of the wireless link using different protocols. Figure 6.1 shows when the total propagation

delay of the wired link is set to 10 ms. Figure 6.2 shows the results when the total propagation delay of the wired link is set to $50ms$. Finally, Figure 6.3 shows the results when the total propagation delay of the wired links is set to 100 ms.

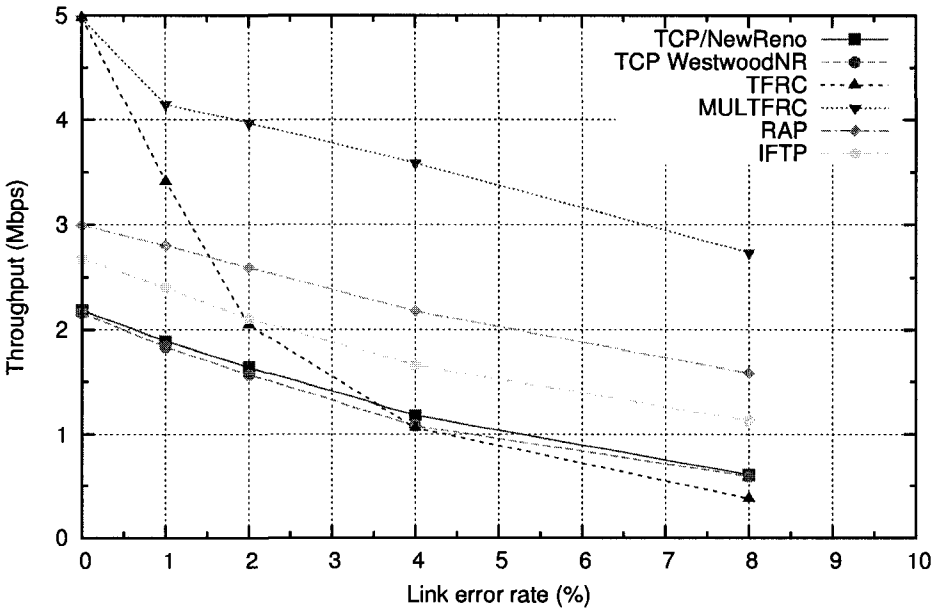


Figure 6.1: Throughput versus wireless error rate with 10 ms propagation delay

The results in Figure 6.1 show that TCP WestwoodNR almost has the same throughput as that of TCP NewReno without any gain. However, when the end-to-end propagation delay increases, we can see the difference. Figures 6.2 and 6.3 show the results for propagation delays of 50 ms and 100 ms. From them we can see the TCP WestwoodNR has slightly better throughput than TCP NewReno. When the wireless error rate increases, both of them have lower throughput because both assume that all losses are congestion losses.

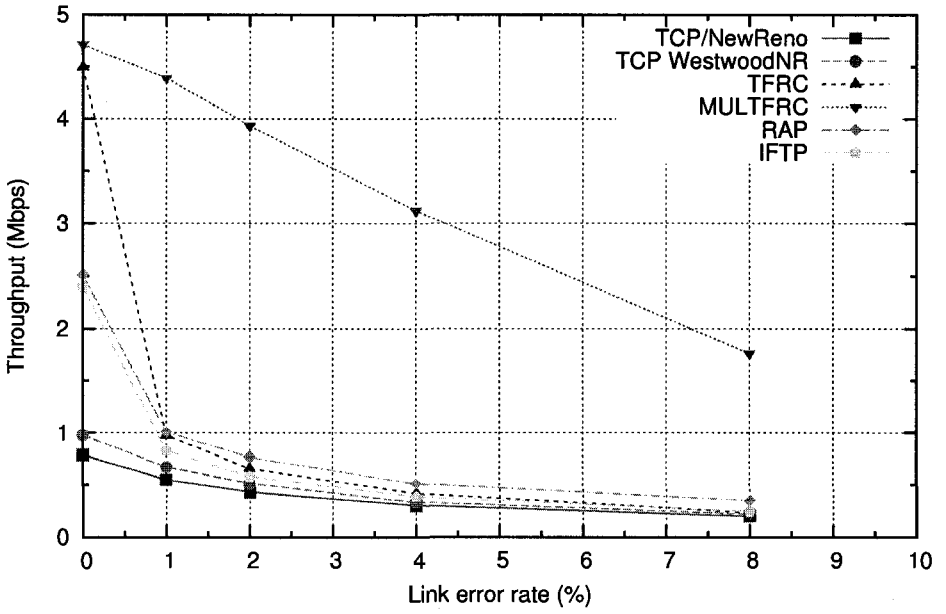


Figure 6.2: Throughput versus wireless error rate with 50 ms propagation delay

In all three figures (6.1, 6.2, and 6.3), TFRC has very high throughput when the wireless link is error free. However, its performance decreases quickly when errors exist in the wireless link. When the end-to-end propagation delay is high, it decreases more quickly. In Figure 6.2, we can see that TFRC's throughput decreased by 80% when there is only 1% error on the wireless link. In Figure 6.3, TFRC's throughput decreased further by 90% with only 1% error on the wireless link.

MULTFRC has the highest throughput in all these situations. It still shows good performance when errors exist in the wireless link. Its main drawback is the complexity of operating multiple connections in one application: it could consume more power, cause more congestion loss, or result in longer delays.

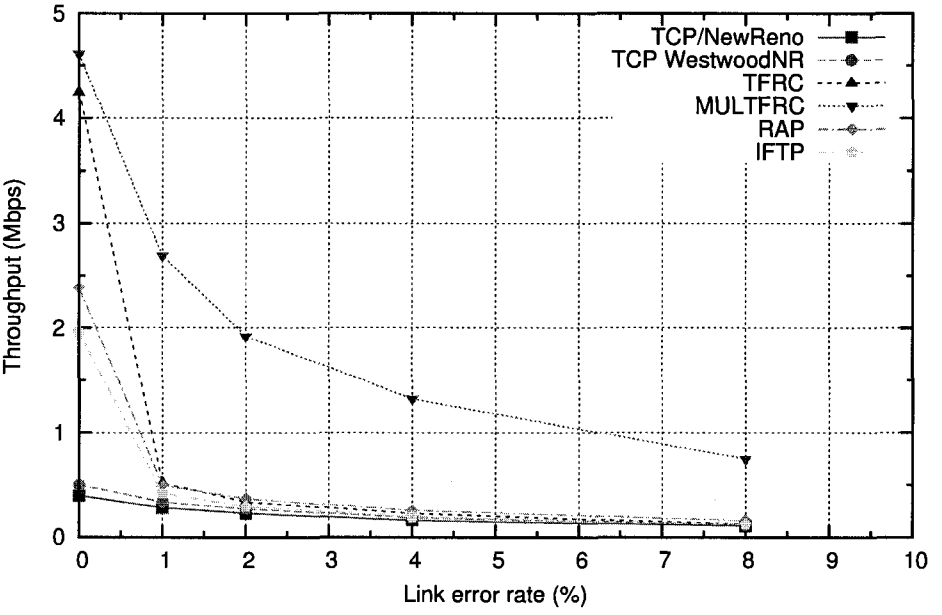


Figure 6.3: Throughput versus wireless error rate with 100 ms propagation delay

RAP and IFTP are two very similar protocols. Therefore, they have very similar performance. According to the figures, they have higher throughputs than those of TCP NewReno and TCP WestwoodNR. However, they are still lower than those of TFRC and MULTFRC, especially when the wireless link is error free. When the end-to-end propagation delay is high, such as 50 ms and 100 ms in Figures 6.2 and 6.3, RAP and IFTP can still can demonstrate good performance. Indeed, they are not as sensitive to the propagation delay as TCP NewReno and TCP WestwoodNR. When there are errors in the wireless link, however, their throughputs also decrease like other protocols discussed. Basically, RAP has a slightly higher throughout than IFTP.

### 6.1.2 Throughput with Multiple Flows

Since in practice the TCP traffic usually exists on all links, in the following section we will evaluate these protocols when they coexist with TCP NewReno flows.

We ran one connection from *sender 1* to *receiver 1*, and another connection from *sender 2* to *receiver 2*. They share the same link from the *router* to *BS* as shown in Figure 5.1. They also share the *11 Mbps* wireless link since the Mac layer protocol for wireless links is *IEEE802.11b*. The connection from *sender 1* to *receiver 1* runs on TCP NewReno protocols, while the connection from *sender 2* to *receiver 2* runs on one of the protocols we discussed above. We measure the throughput at the *receiver 2*.

Figures 6.4 and 6.5 show the results of throughput versus propagation delay. Figure 6.4 shows the results without any wireless error, while Figure 6.5 shows the results with 2% wireless error. The error model assumed here is still the uniform error model.

According to Figure 6.4, most protocols except IFTP have a higher throughput than TCP NewReno, especially when the propagation delay is high. The TFRC does not have a significant throughput improvement over TCP as it does when only TFRC traffic exists. That is the penalty of being TCP-friendly.

MULTFRC has higher throughputs when the propagation delay is 50 ms and 100 ms. That is because MULTFRC is very aggressive with the bandwidth. When the propagation delay is high, the bandwidth used by TCP NewReno decreases quickly. Then MULTFRC can share more of the available bandwidth. Afterward, when the propagation delay increases further, the throughput of MULTFRC starts to decrease as

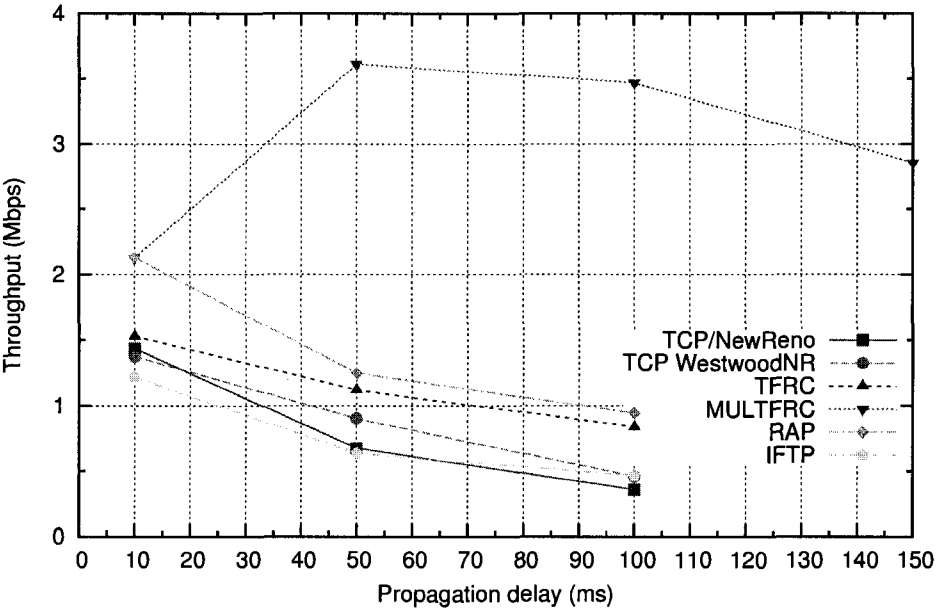


Figure 6.4: Throughput versus propagation delay without wireless error

the feedback information arrives too late to be helpful.

For situations when there are errors in the wireless link, the results are shown in Figure 6.5. The wireless errors have a significant effect on the throughputs for almost all protocols tested. It is thus very necessary to classify the loss type so that the protocols still work well in wireless environment.

### 6.1.3 High Fluctuation of MULTFRC

We noticed a high fluctuation in the throughput of MULTFRC. We set up three connections over three pairs of senders and receivers running on three different protocols: TCP NewReno, TFRC, and MULTFRC. The end-to-end propagation delay is set to 50 ms for

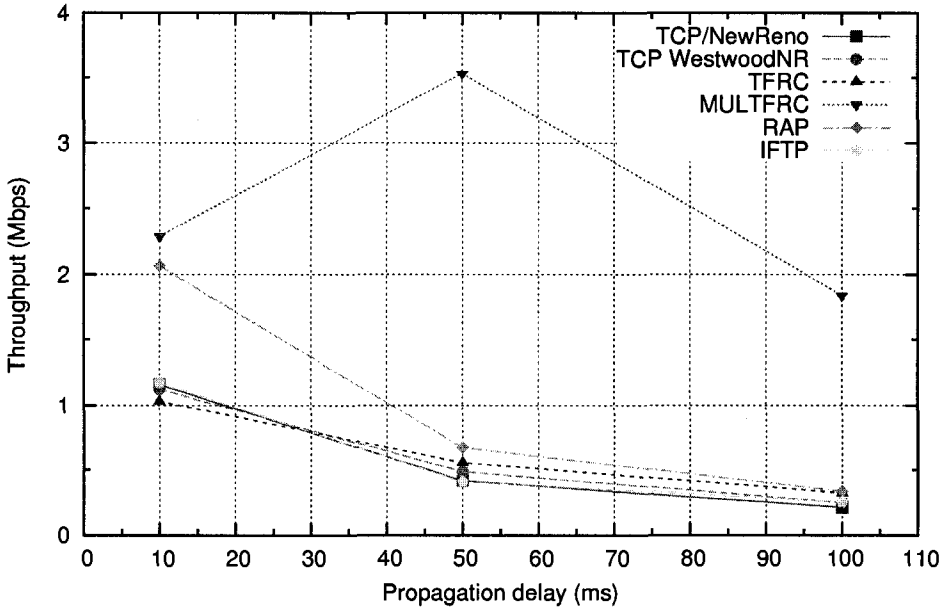


Figure 6.5: Throughput versus propagation delay with 2% wireless error

all of the connections. The wireless link is error free. We measure the throughputs on the receivers, the results of which are shown in Figure 6.6. Comparing the throughputs in the figure, we find that while MULTFRC achieved a very high throughput, its fluctuation is also very large. The high fluctuation of MULTFRC is caused by the frequent changes in the number of simultaneous connections.

## 6.2 TCP Friendliness

We use the definition of TCP-friendliness for Unicast in Ref. [68]: "A unicast-flow is considered TCP-friendly when it does not reduce the long-term throughput of any co-existent TCP flow more than another TCP flow on the same path would do under the

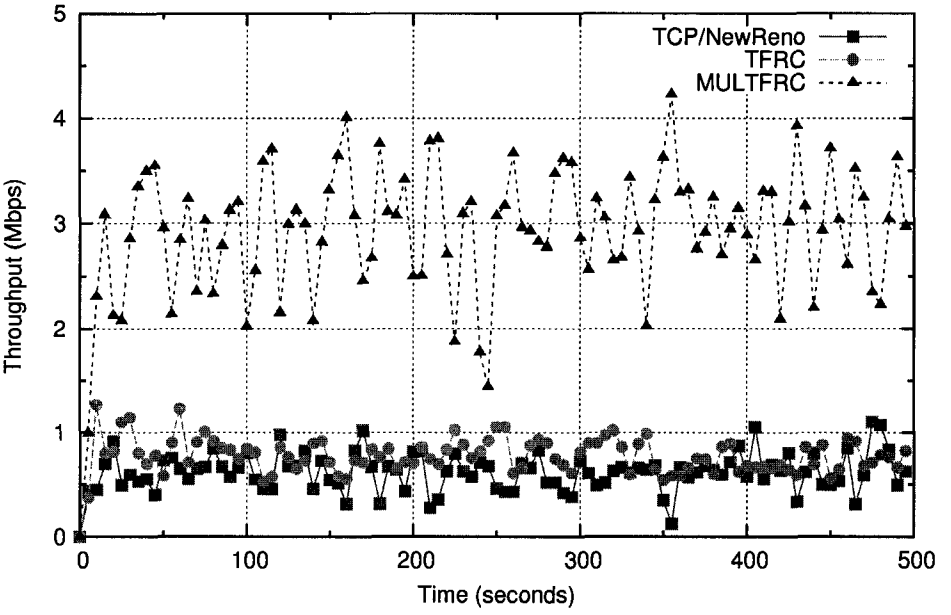


Figure 6.6: Throughputs of one TCP, one TFRC, and one MULTFRC coexisting

same network conditions".

We ran one TCP NewReno connection from *sender 1* to *receiver 1*, and another TCP NewReno connection from *sender 2* to *receiver 2*. They share the same link from *router* to *BS* as shown in Figure 5.1. They also share the *11 Mbps* wireless link since the Mac layer protocol for wireless links is *IEEE 802.11b*. We measured the throughput at the TCP NewReno *receiver 1*. We define it as $TP_{tcp-tcp}$. Then we change the connection of *sender 2* to *receiver 2* to run on one of the protocols we discussed above. We measure the throughput at the TCP NewReno *receiver 1* again. We define this throughput as $TP_{tcp-xxx}$, where $xxx$ is the protocol name that competes for the link with TCP NewReno. For example, $TP_{tcp-tfrc}$ means we are running TFRC on the link from sender

2 to receiver 2.

We compare the difference between $TP_{tcp-tcp}$ and $TP_{tcp-xxx}$ to obtain information about TCP-friendliness. If the throughput $TP_{tcp-xxx}$ is almost equal to $TP_{tcp-tcp}$ or larger than it, it means that this protocol has good TCP-friendliness. Otherwise, it is not TCP-friendly.

Figures 6.7 and 6.8 show the results for the friendliness evaluation. Figure 6.7 shows the results without any wireless error, while Figure 6.8 shows the results with 2% wireless error. The error model assumed here is the uniform error model.

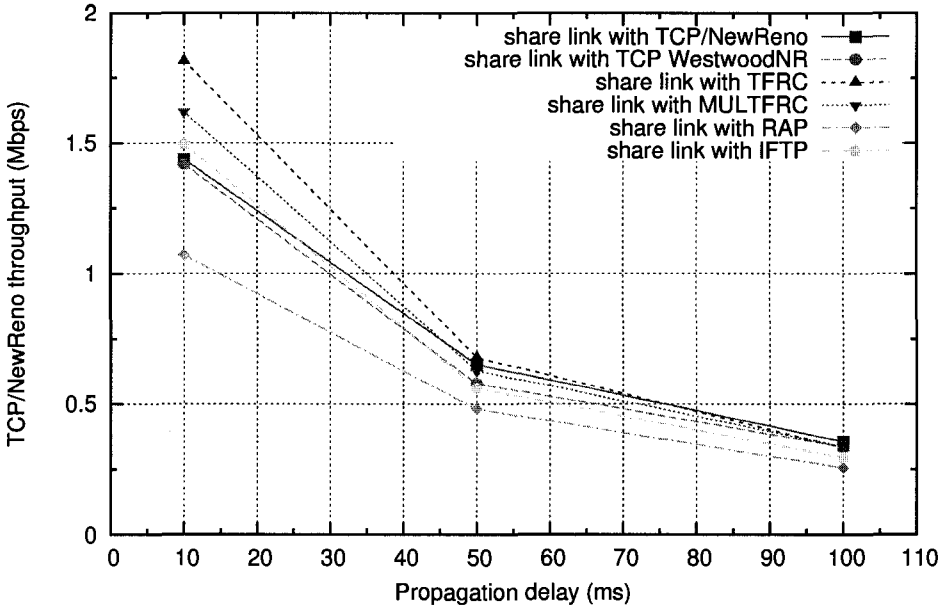


Figure 6.7: TCP-friendliness without wireless error

The simulation results in Figure 6.7 and 6.8 show that RAP is the worst protocol in terms of TCP-friendliness. In [54], it is shown that deploying RED queue management

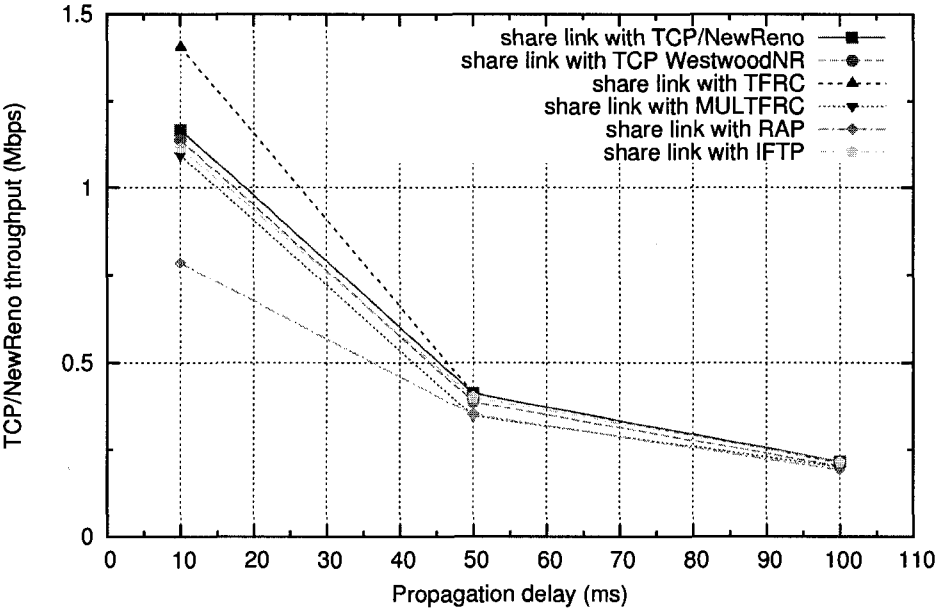


Figure 6.8: TCP-friendliness with 2% wireless error

can result in an ideal fairness between TCP and RAP traffic. However, in our simulation, DropTail queuing is adopted. In this scenario, RAP is not very TCP-friendly.

TFRC is the best protocol in terms of TCP-friendliness in all of the situations we evaluated, as shown in Figures 6.7 and 6.8. This is because TFRC responds to changes in available bandwidth more slowly than TCP while competing fairly for bandwidth [30].

For the MULTFRC protocol, TCP-friendliness in the presence of wireless errors is more difficult to establish than in error free situations. This is because the TCP NewReno assumes that all wireless losses are congestion losses and reduces its sending rate in response, while the MULTFRC flow is very aggressive with the available bandwidth.

## 6.3 Summary

TCP Westwood has some improvements over TCP Reno and NewReno. However, the throughput it can achieve is still lower than the other rate-based protocols we evaluated. The simulation results show that RAP is the worst protocol in terms of TCP-friendliness. As a consequence, it has a slightly higher throughput. IFTP has good TCP-friendliness, but the throughput it can achieve is lower than that of the others. TFRC is the best protocol for TCP-friendliness in all of the situations we evaluated, because *TFRC responds to changes in available bandwidth slower than TCP while competing fairly for bandwidth* [30]. However, TFRC's efficiency drops quickly in a lossy wireless network.

MULTFRC has the highest throughput in all of the evaluated situations. By appropriately choosing the number of connections and packet size, MULTFRC makes it possible to fully utilize the wireless bandwidth even in the presence of high wireless error rates. Furthermore, it is an end-to-end approach and only requires modifications at the application layer [17]. However, MULTFRC has two main drawbacks. First, because the number of connections must be an integer, the fluctuation of the throughput is very high, requiring a larger buffer to smooth the flow for practical multimedia applications. Second, operating multiple connections in one application could cause a large overhead and consume more system resources [17].

Wireless errors have a significant effect on the throughputs for almost all of the protocols. It is very necessary to classify the loss types properly so that the protocols still work well in wireless environments.

# Chapter 7

# Simulation Results and Analyzation of mSpike and other LDAs

## 7.1 mSpike and other LDAs over TFRC

We also use the ns2 [1] network simulator to evaluate the performance of mSpike and other LDA schemes we introduced. They run over the TFRC protocol. The basic network topology is the same as that shown in Figure 5.1. To simulate different degrees of congestion, we vary the number of connections from 1 to 6.

To evaluate the performance of all the loss differentiation algorithms, the following metrics are used: 1) Throughput: the average throughput of all the end-to-end connections; 2) $R_c$: the rate of congestion loss; and 3) $R_{mw}$: the rate of misclassifying wireless losses as congestion losses. The wireless error model we used is 5% uniform wireless error

| | TFRC | Biaz | mBiaz | SPLD | ZigZag | ZBS | Spike | PLC | TD | mSpike |
|---|---|---|---|---|---|---|---|---|---|---|
| Throughput (Mbps) | 0.681 | 1.465 | 1.196 | 1.215 | 0.903 | 1.450 | 1.444 | 1.445 | 1.451 | 1.457 |
| Rc (%) | 0.1 | 0.4 | 0.3 | 0.34 | 0.1 | 0.2 | 0.4 | 0.49 | 0.48 | 0.28 |
| Rmw (%) | 100 | 1.8 | 17.5 | 57 | 74.7 | 6.0 | 22.5 | 26.1 | 35.4 | 11.3 |

Table 7.1: Throughput of LDAs with one TFRC flow and 2Mbps wireless bandwidth

rate.

### 7.1.1 Evaluation Based on Throughput

Table 7.1 shows the results of running one TFRC flow with 2 Mbps wireless bandwidth. The wired links have 100 Mbps bandwidth. In this situation, the wireless link is the bottleneck for the connection, and most of the packets are sent back to back on the wireless link.

As shown in Table 7.1, TFRC without any LDA scheme has the lowest throughput because it treats all of the wireless losses as congestion losses and thus reduces its sending rate excessively. All the LDA schemes have improved this performance. Among them, the mSpike scheme has the same high throughput as the Biaz, ZBS, SPLD, Spike, PLC, and TD schemes in this situation. These schemes also doubled the throughput of the TFRC only situation. However, the improvement of the mBiaz and ZigZag schemes is very limited.

Figure 7.1 shows the results of running 1, 2, 4 and 6 TFRC flows with 11 Mbps wireless bandwidth. In this figure, the x-axis is the different LDA schemes which are

distributed as 1: No LDA 2: Biaz 3: mBiaz 4: SPLD 5: ZigZag 6: ZBS 7: Spike 8: PLC 9: TD 10: mSpike. This figure also includes the different results of 1, 2, 4, and 6 TFRC flows sharing the 11 Mbps wireless bandwidth.

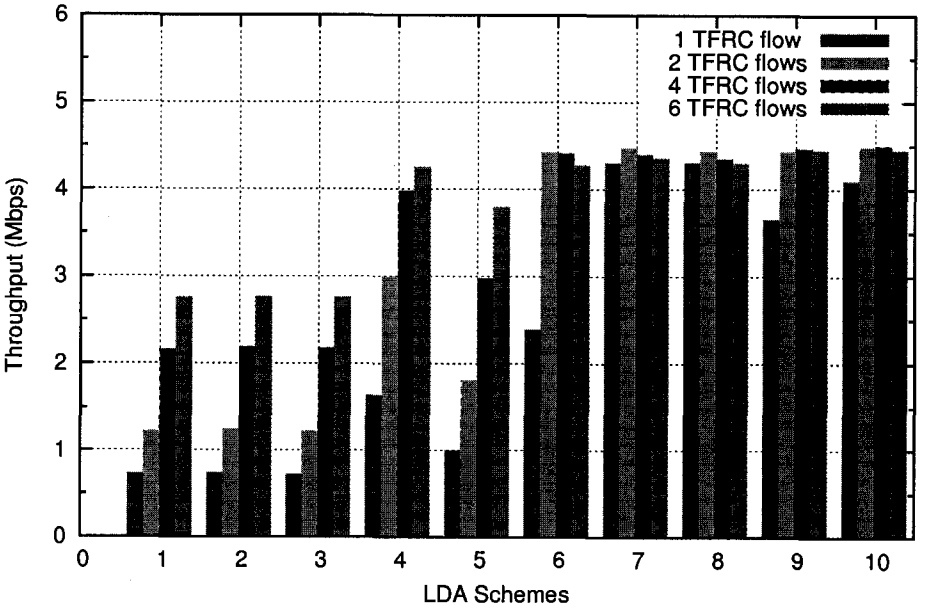


Figure 7.1: Throughput with different LDA schemes *(1: No LDA 2: Biaz 3: mBiaz 4: SPLD 5: ZigZag 6: ZBS 7: Spike 8: PLC 9: TD 10: mSpike)*

As shown in Figure 7.1, we can conclude that:

- The mSpike, Spike, PLC, and TD schemes perform the best and most consistently across all numbers of flows.

- The Biaz and mBiaz schemes cannot classify the packet loss type in the multi-streaming situation where several streams share the same wireless medium together.

- The SPLD is much better than the Biaz and mBiaz schemes since it uses the statistical data of inter-arrival time. It can successfully discriminate loss types in both single- and multi-streaming situations. However, the throughput of SPLD is still lower than the Spike, PLC, mSpike, and TD schemes.

- The ZigZag scheme uses the number of losses and the mean/deviation of ROTT to classify the loss type. It does not work well under most circumstances.

- The ZBS scheme selects different base schemes to be used according to the situation. However, its performance is even worse than only using the Spike scheme.

### 7.1.2 Evaluation Based on the Rate of Misclassification

Figure 7.2 shows the results of misclassifying the wireless loss rate. In this figure, the x-axis is the different LDA schemes, which are distributed as 1: No LDA 2: Biaz 3: mBiaz 4: SPLD 5: ZigZag 6: ZBS 7: Spike 8: PLC 9: TD 10: mSpike. This figure also includes the different results of 1, 2, 4, and 6 TFRC flows sharing the 11 Mbps wireless bandwidth.

As shown in Figure 7.2, a higher rate of misclassifying wireless losses as congestion losses usually causes lower throughput. The mSpike, Spike, PLC, and TD schemes have lower misclassification rates and thus can achieve higher throughput.

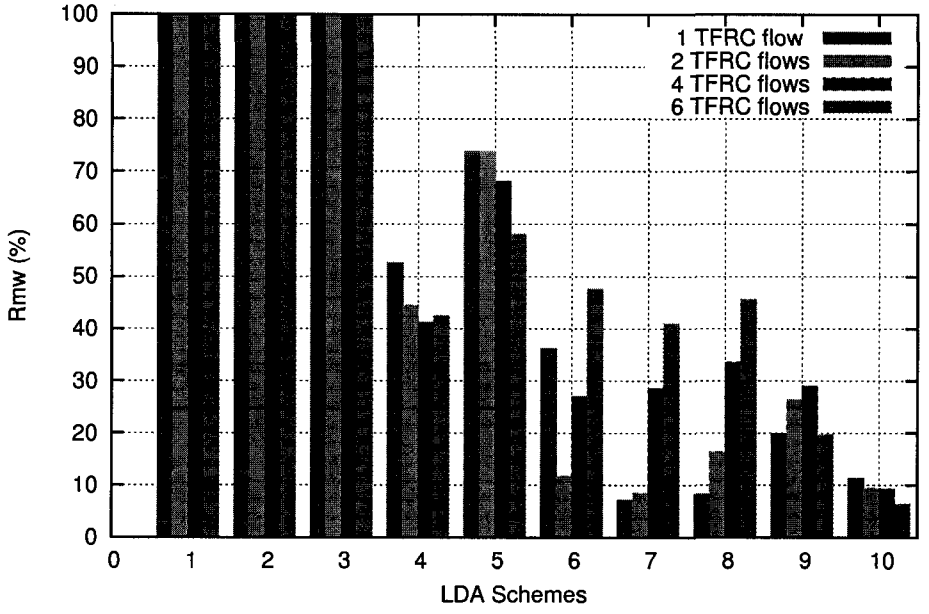


Figure 7.2: Misclassification rate with different LDA schemes *(1: No LDA 2: Biaz 3: mBiaz 4: SPLD 5: ZigZag 6: ZBS 7: Spike 8: PLC 9: TD 10: mSpike)*

### 7.1.3 Evaluation Based on Congestion Loss Rate

Figure 7.3 shows the results of the congestion loss rate. In this figure, the x-axis is the different LDA schemes, which are distributed as 1: No LDA 2: Biaz 3: mBiaz 4: SPLD 5: ZigZag 6: ZBS 7: Spike 8: PLC 9: TD 10: mSpike. It also includes the different results of 1, 2, 4, and 6 TFRC flows sharing the 11 Mbps wireless bandwidth.

In all of the simulated scenarios, these LDA schemes have the same low congestion loss rates as that of No LDA scheme. Most of congestion loss rates are below 2% which are acceptable for most multimedia applications. According to [61], "most real-time applications have a certain level of tolerance for packet loss".

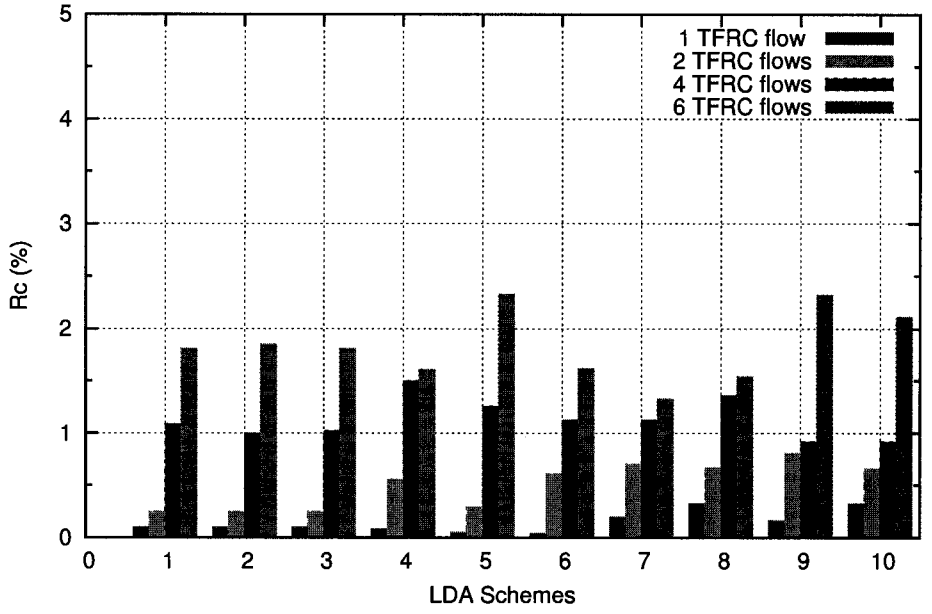


Figure 7.3: Congestion loss rate with different LDA schemes *(1: No LDA 2: Biaz 3: mBiaz 4: SPLD 5: ZigZag 6: ZBS 7: Spike 8: PLC 9: TD 10: mSpike)*

### 7.1.4 Comparing mSpike with Spike, PLC, and TD

The mSpike, Spike, PLC, and TD schemes are the four best LDA schemes according to the aforementioned simulation results and analysis. However, the Spike, PLC, and TD schemes have different drawbacks than mSpike.

- The Spike and PLC schemes have high throughput in most cases. However, they use the observed maximum and minimum ROTT values to classify the loss type, thus they cannot recover from an occasional extreme value of ROTT. Once a very large or very small ROTT value is measured due to some error, they will fail to work. The mSpike scheme performs well and consistently across all numbers of flows.

| Connection Number | 1 | 2 | 4 | 6 | 8 |
|---|---|---|---|---|---|
| mSpike's Throughput (Mbps) | 4.163 | 4.239 | 4.286 | 4.282 | 4.246 |
| Spike's Throughput (Mbps) | 4.286 | 4.332 | 4.282 | 4.207 | 4.141 |

Table 7.2: Performance of mSpike with Markov error model

The exponentially averaged ROTT used by mSpike is immune to the occasional extreme value of ROTT that would cause high misclassification rates in the Spike and PLC schemes.

- The TD scheme does not work as well as mSpike, Spike, or PLC in situations with light traffic, as shown in Table 7.1. Another drawback of the TD scheme is that its realization is more complicated than that of other LDA schemes. It uses the least quare method [46] to approximate the ROTT curve, and involves computation with matrices. However, the realization of mSpike is much simpler.

## 7.2 mSpike Scheme with Markov Error Model

Table 7.2 contains the simulation results for when the two-state Markov chain model is used to produce wireless losses in the wireless links. The parameters of the Markov wireless model are the same as those shown in Chapter 5.2. The wireless bandwidth is 11 Mbps. The connection number is changed from 1 to 6.

According to Table 7.2, we know that the mSpike scheme works as well as Spike under

the two-state Markov wireless error model. mSpike even shows better performance than Spike when the number of connections increases.

## 7.3 Fairness of mSpike

We use the ns2 [1] network simulator again to evaluate the fairness of mSpike. We run the mSpike scheme over TFRC. The basic network topology used in the simulation is the same as that shown in Figure 5.1. All parameters are the same as those in Chapter 5, except that the propagation delay is fixed to 10 ms.

We ran six TFRC plus mSpike connections on $sender1-6$ to $receiver1-6$ separately. We measured the throughputs on $receiver1$ to $receiver6$ under different wireless error rates of 2% and 5%. The error model assumed here is still the uniform error model. We divided each receiver's throughput by the average throughput. If this value is near 1, it indicates a good fairness.

Figure 7.4 shows the results of these simulations. It indicates that the mSpike scheme has very good fairness since all of the values for the six receivers are near 1.

## 7.4 mSpike over MULTFRC

From the simulation results of the congestion control protocols given above, we know that MULTFRC is a very efficient protocol for achieving a high throughput. It makes it possible to fully utilize the link bandwidth. However, its performance drops when

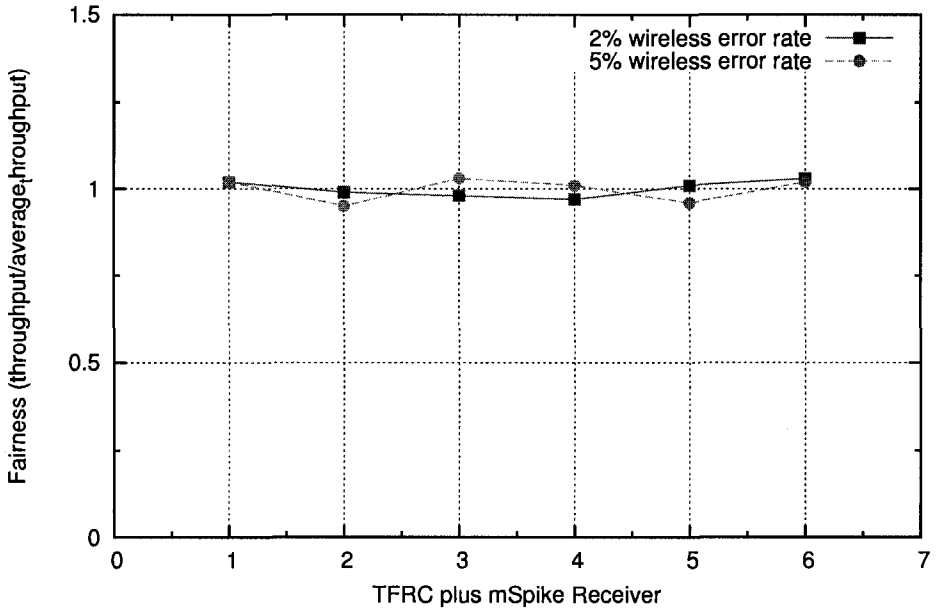


Figure 7.4: Fairness of mSpike

there are errors in the wireless link. The loss differentiation algorithms can classify the wireless losses and congestion losses, and thus can help the congestion control protocols work well in wireless environments. Among them, the mSpike scheme has been proven to be a good loss differentiation algorithm. If we combine the MULTFRC protocol and mSpike scheme, it is expected that they should work well over lossy wireless networks.

We use ns2 [1] network simulator to evaluate the performance of MULTFRC plus mSpike. The basic network topology used in the simulation is the same as that shown in Figure 5.1. All parameters are also the same as those given above.

We ran two TCP NewReno connections on *sender*1 to *receiver*1 and *sender*2 to *receiver*2, and one MULTFRC connection on *sender*3 to *receiver*3. We measured the

throughputs of TCP and MULTFRC on *receiver*1 and *receiver*3 under different wireless error rates. We varied the wireless link error rate between 0%, 2%, 4%, and 8%. The error model assumed here is still the uniform error model. Then we changed the set up to run MULTFRC plus mSpike over the link of *sender*3 to *receiver*3. We took the same measurements as discussed above.

Figure 7.5 shows the simulation results when the total propagation delay of the wired links is set to 50 ms.

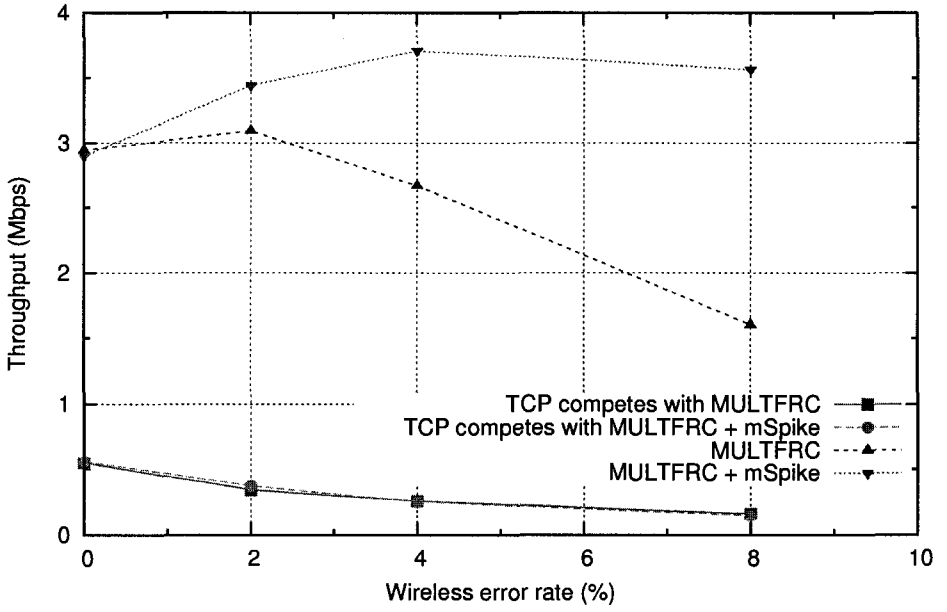


Figure 7.5: Throughput versus wireless error rate with 50 ms propagation delay

From the results shown in Figure 7.5, we observed that MULTFRC plus mSpike achieves higher throughput more efficiently than MULTFRC alone. The improvement is even better when the end-to-end propagation delay is longer and the wireless error rate

is higher. At the same time, MULTFRC plus mSpike has no effect on TCP-friendliness since the TCP throughputs are almost the same in these two figures.

Therefore, MULTFRC plus mSpike would be a good choice for applications with high bandwidth requirements.

## 7.5 Summary

In this chapter, we evaluated nine algorithms for loss differentiation.

The Biaz, mBiaz, and SPLD schemes are all based on inter-arrival time at the receiver side. They do not need the timestamp information contained in each packet, so they have less overhead. However, the Biaz and mBiaz schemes only perform well when the wireless link is the last and bottleneck link without any competing flows. The SPLD scheme works better with competing flows since it uses statistical information to avoid confusion caused by multiple streams. However, all of these scheme have lower throughput than other LDA schemes under most simulated scenarios.

The other schemes use the relative one-way trip time (ROTT) to classify the loss type. In most cases, the congestion loss often occurs around the peak of the ROTT that makes them working accurately. The ZigZag and mSpike schemes use the mean and deviation of ROTT, but ZigZag does not work as well as mSpike. ZBS is a hybrid algorithm, but it does not show much improvement in performance over using only the Spike algorithm.

The mSpike, Spike, PLC, and TD schemes have a consistently high throughput.

However, the realization of the TD scheme is more complicated than that of other LDA schemes. The Spike and PLC schemes cannot recover from an occasionally extreme value of ROTT. Therefore, the mSpike scheme is a better LDA scheme in most of the situations.

The mSpike scheme also has very good fairness, and MULTFRC plus mSpike would be a good choice for applications which have high bandwidth requirements.

# Chapter 8

# Conclusion and Future Work

## 8.1 Conclusion

We briefly surveyed several different TCP-friendly congestion control protocols. They are TCP Westwood, TFRC, MULTFRC, RAP, and IFTP. Compared to existing surveys, we pay more attention to the performance of these protocols over wireless lossy networks. Their performance is evaluated in various wireless scenarios with the *ns*2 Network Simulator.

The results show that TCP Westwood has limited improvements over TCP Reno and NewReno. RAP has a slightly higher throughput at the cost of decreased TCP-friendliness. IFTP has good TCP-friendliness, but the throughput it can achieve is even lower than that of TCP. TFRC is the best in TCP-friendliness in all the situations we evaluated. However, all of these protocols show very low throughputs in a lossy wireless

network since they treat all of the packet losses as congestion losses.

In contrast, MULTFRC exhibits greater efficiency and makes it possible to fully utilize the wireless bandwidth even in the presence of high rates of wireless errors. Furthermore, it is an end-to-end approach and only needs modifications at the application layer. The main problem with MULTFRC is its high fluctuation rate, which is caused by the frequent changes in the number of connections. It will need large buffering to smooth out the fluctuations for multimedia applications.

The loss differentiation algorithms can be used in the congestion control protocols to distinguish the wireless losses from the congestion losses. Then, the congestion control protocols will not need to react to the wireless losses, and they will thus work better in wireless lossy environments. Several loss differentiation algorithms are introduced in Chapter 3. They include the Biaz, mBiaz, SPLD, ZigZag, ZBS, Spike, PLC, and TD schemes.

The Biaz and mBiaz schemes only perform well when the wireless link is the last link and the bottleneck link without any competing flows. The SPLD scheme is also based on inter-arrival time, like Biaz and mBiaz, to classify the loss type. However, the SPLD scheme works well with competing flows since it uses statistical information to avoid any confusion caused by multiple streams. The other schemes use the relative one-way trip time (ROTT) to classify the loss type. However, the ZigZag scheme does not work well since it uses very conservative boundaries to discriminate the loss type. Also, although ZBS is a hybrid algorithm, it does not show much improvement in performance over

using even using the Spike algorithm alone. The Spike, PLC, and TD schemes have high throughputs most of the time. The TD scheme is the most complex scheme, since it uses the least square method to approximate the ROTT curve. The Spike and PLC schemes cannot recover from any occasional extreme value of ROTT.

Since these current LDA schemes have different problems, we propose a new LDA scheme: the mSpike scheme. The mSpike scheme is based on the exponential mean and deviation of ROTT to discriminate the loss type. The exponential values make it very easy to recover from any occasional error. Finally, we ran the mSpike scheme and other LDA schemes over TFRC to perform the evaluations with the ns2 network simulator. The experimental results show that mSpike is a very good LDA scheme in most of the situations we evaluated. It works well under both the uniform wireless error model and the Markov error model. The fairness of the mSpike scheme is also proven to be very good.

MULTFRC plus mSpike has also been tested and proven to be a good solution for applications requiring high throughput. We observe that MULTFRC plus mSpike can achieve higher throughputs than MULTFRC alone in wireless lossy environments. The improvement is even better when the error rate is increased. At the same time, MULTFRC plus mSpike is also TCP-friendly.

## 8.2 Future Work

TCP-friendly congestion control protocols combined with loss discrimination algorithms over wireless networks is an attractive area of research. For future work, an important issue that needs to be resolved in this MULTFRC plus mSpike solution is how to reduce the rate fluctuation while maintaining both good TCP-friendliness and high throughput. On the other hand, we only completed the simulation and evaluation in WLAN scenarios. In other wireless environments, especially those involving multihopping and mobility, the evaluation work will be more challenging. There are also many other issues still waiting to be investigated in the future.

# Appendix A

# List of Terminology

**TFRC**: TCP-Friendly Rate Control

**IFTP**: Internet Friendly Transport-level Protocol

**RAP**: Rate Adaptation Protocol

**MULTFRC**: Multiple TFRC

**LDA**: Loss Differential Algorithms

**ROTT**: Relative One-way Trip Time

**PLC**: Packet Loss Classification

**TD**: Trend-and-Loss-Density-based

**SPLD**: Statistical Packet Loss Discrimination

# Bibliography

[1] Network Simulator (Ver 2). http://www.isi.edu/nsnam/ns.

[2] Introduction about DCCP. http://www.opalsoft.net/qos/dccp-10.htm.

[3] G. Almes, S. Kalidindi, and M. Zekauskas. A one-way delay metric for ippm (rfc 2679), September 1999.

[4] Farooq Anjum and Leandros Tassiulas. Performance analysis of tcp-friendly aimd algorithms for multimedia applications. *IEEE Trans. on Multimedia*, 7(2):339–355, April 2005.

[5] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. Rem: Active queue management. *IEEE Network*, May 2001.

[6] D. Barman and I. Matta. Effectiveness of loss labeling in improving tcp performance in wired/wireless networks. *Boston University Technical Report*, 2002.

[7] D. Barman and I. Matta. A bayesian approach for tcp to distinguish congestion from wireless losses. *Technical Report, Department of Computer Science, Boston University*, 2003.

[8] S. Biaz and N. H. Vaidya. Distinguishing congestion losses from wireless transmission losses: a negative result. *Proc. of IEEE 7th Int. Conf. on Computer Communications and Networks*, October 1998.

[9] Saad Biaz and Nitin H. Vaidya. Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. *Proceedings of the 1999 IEEE Symposium on Application - Specific Systems and Software Engineering and Technology*, pages 10–17, March 24-27, 1999.

[10] L. Cai, X. Shen, J. W. Mark, and J. Pan. Performance modeling and analysis of window-controlled multimedia flows in wireless/wired networks. *IEEE Trans. on Wireless Communications*, 6(2), Feb. 2007.

[11] L. Cai, X. Shen, J. Pan, and J. W. Mark. Comparative study of various tcp versions over a wireless link with correlated losses. *IEEE/ACM Transactions on Networking*, 11(3):370–383, June 2003.

[12] Claudio Casetti, Mario Gerla, , S. Mascolo, M. Y. Sanadidi, and Ren Wang. Tcp westwood: End-to-end congestion control for wired/wireless networks. *Wireless Networks*, 8:467–479, 2002.

[13] Claudio Casetti, Mario Gerla, , S. Mascolo, M. Y. Sanadidi, and Ren Wang. Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. *In Proceedings of ACM Mobicom 2001*, pages 287–297, July 16-21 2001.

[14] Song Cen, Pamela C. Cosman, and Geoffrey M. Voelker. Endto-end differentiation of congestion and wireless losses. *IEEE/ACM Transactions on Networking (TON)*, 11(5):703–717, October 2003.

[15] Edward Chan and S. W. Ng. A rate-based streaming protocol for wireless networks. *IPDPS*, 2001.

[16] Jiwei Chen, Paganini F., Ren Wang, Sanadidi M.Y., and Gerla M. Fluid-flow analysis of tcp westwood with red. *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, 7:4064–4068, Dec. 2003.

[17] M. Chen and A. Zakhor. Rate control for streaming video over wireless. *IEEE Wireless Communications*, 12(4):32–41, August 2005.

[18] M. Chen and A. Zakhor. Aio-tfrc: A light-weighted rate control scheme for streaming over wireles. *Proceedings of the IEEE WirelessCom 2005: Symposium on Multimedia over Wireless*, June, 2005.

[19] M. Chen and A. Zakhor. Multiple tfrc connections based rate control for wireless networks. *IEEE Trans. Multimedia*, 8(5):1045–1062, October 2006.

[20] Cheng-Fu Chou, Ming-Wei Hsu, and Ching-Ju Lin. A trend-loss-density-based differential scheme in wired-cum-wireless networks. *International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2006.

[21] Jae Won Chung. *Congestion Control for Streaming Media.* PhD thesis, WORCESTER POLYTECHNIC INSTITUTE, October 2005.

[22] Hala ElAarag and Mostafa Bassiouni. An internet friendly transport protocol for continuous media over best effort networks. *International Journal of Communication Systems*, 15:881–898, 2002.

[23] Hala ElAarag and Andrew Moedinger. Performance evaluation of an internet friendly transport protocol over networks with lossy links. *Applied Telecommunication Symposium, Spring Simulation Multiconference*, 2:21–25, April 2005.

[24] Hala ElAarag and Andrew Moedinger. Iftp-w: A tcpfriendly protocol for multimedia applications over wireless networks. *ACM Southeast Conference*, 2:36–40, March 2005.

[25] Gilbert E.N. Capacity of a burst-noise channel. *Bell Systems Technical Journal*, pages 1253–1265, 1960.

[26] Elliott E.O. Estimation of error rates for codes on burst channels. *Bell Systems Technical Journal*, pages 1977–1997, 1963.

[27] Su Fang, Fan Yinglei, Li Yong, and Xu Huimin. A rod based fuzzy packet loss differentiating algorithm for tcp in the hybrid wired/wireless network. *International Conference on Systems and Networks Communication (ICSNC'06)*, 2006.

[28] S. Floyd, M. Handley, and J. Padhye. A comparison of equation-based and aimd congestion control, May 2000.

[29] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. *Proc. ACM SIGCOMM*, page 4356, August 2000.

[30] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Tcp friendly rate control (tfrc): Protocol specification. *RFC 3448, Proposed Standard*, January 2003.

[31] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Tcp friendly rate control (tfrc): Protocol specification. *a revision of RFC 3448*, March 2007.

[32] Cheng Peng Fu and Soung C. Liew. Tcp veno: Tcp enhancements for transmission over wireless access networks. *IEEE Journal on Selected Areas in Communication*, 21:216–227, February 2003.

[33] Mario Gerla, Bryan K. F. Ng, M. Y. Sanadidi, Massimo Valla, and Ren Wang. Tcp westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs. *Computer Communications*, 27(1):41–58, January 2004.

[34] L. A. Grieco and S. Mascolo. Performance evaluation and comparison of westwood+, new reno and vegas tcp congestion control. *ACM Computer Communication Review*, 34(2), April 2004.

[35] ElAarag H. and Hogg C. Enhancement of iftp for transmission over wireless access networks. *IEEE Southeast Conference 2007*, pages 651–656, March 2007.

[36] H.-F. Hsiao, A. Chindapol, J. Ritcey, Y.-C. Chen, and J.-N. Hwang. A new multimedia packet loss classification algorithm for congestion control over wired/wireless channels. *IEEE ICASSP*, March 2005.

[37] Padhye J., Firoiu V., Towsley D., and J. Kurose. Modeling tcp throughput: A simple model and its empirical validation. *Proc. ACM SIGCOMM*, 1998.

[38] Hoe JC. Improving the start-up behaviour of a congestion control scheme for tcp. *ACM SIGCOMM Computer Communication Review, Conference Proceedings on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 26, August 1996.

[39] L. Larzon, M. Degermark, and S. Pink. Efficient use of wireless bandwidth for multimedia applications. *IEEE MoMUC*, November 1999.

[40] E. Lengliz, H. Touati, F. Kamoun, and M. Y. Sanadidi. Experimentations towards tcp westwood application. *Adhoc Mobile Networks Med-Hoc Net 2003*, June 2003.

[41] Stephane Lohier, Yacine Ghamri Doudane, and Guy Pujolle. Cross-layer design to improve elastic traffic performance in wlans. *International Journal of Network Management*, 2007.

[42] Stphane Lohier and Yacine Ghamri-Doudane. Efficiency of loss differentiation algorithms in 802.11 wireless networks. *MMNS 2006*, pages 141–144, 2006.

[43] Allman M., Paxson V., and W. Stevens. Tcp congestion control. *http://www.ietf.org/rfc/rfc2581.txt*, April 1999.

[44] J. Mahdavi and S. Floyd. Tcp-friendly unicast rate-based flow control. *Technical note sent to the end2end-interest mailing list*, January 8, 1997.

[45] S. Mascolo and G. Racanelli. Testing tcp westwood+ over transatlantic links at 10 gigabit/second rate. *Third International Workshop on Protocols for Fast Long-Distance Networks (PFLDNET05)*, February 3,4 2005.

[46] Least Square Method. http://www.bsu.edu/web/jkshim/mathandstat/lsm/leastsquare.htm.

[47] T. Nguyen and Sen-Ching S. Cheung. Multimedia streaming using multiple tcp connections. *Performance, Computing, and Communications Conference, 2005. 24th IEEE International*, pages 215–223, 2005.

[48] The ns Manual. http://www.isi.edu/nsnam/ns/ns-documentation.html, November 20, 2006.

[49] TCP WESTWOOD Home Page. http://www.cs.ucla.edu/nrl/hpi/tcpw/.

[50] Min Kyu Park, Kue-Hwan Sihn, and Jun Ho Jeong. A statistical method of packet loss type discrimination in wired-wireless networks. *IEEE CCNC*, 2006.

[51] J. Postel. Rfc 768: User datagram protocol. *http://www.faqs.org/rfcs/rfc768.html*, 28 August 1980.

[52] Jon Postel. Transmission control protocol. *http://www.ibiblio.org/pub/docs/rfc/rfc793.txt*, September 1981.

[53] Yang Y. R. and Lam S. S. General aimd congestion control. *in Proceedings of the 8th International Conference on Network Protocols (ICNP)*, November 2000.

[54] R. Rejaie, M. Handley, and D. Estrin. Rap: An end-to-end rate-based congestion control mechanism for realtime stream in the internet. *Proceedings of IEEE Infocom*, 1999.

[55] Bregni S., Caratti D., and Martignon F. Enhanced loss differentiation algorithms for use in tcp sources over heterogeneous wireless networks. *IEEE Global Communications Conference*, December 2003.

[56] Floyd S. Rfc 2914: Congestion control principles. *RFC 2914, Best Current Practice*, September 2000.

[57] Floyd S. and Fall K. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, August 1999.

[58] Floyd S. and Jacobson V. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking, V.1 N.4*, pages 397–413, August 1993.

[59] N.K.G. Samaraweera. Non-congestion packet loss detection for tcp error recovery using wireless links. *IEEE Proceedings Communications*, 146(4):222–230, August 1999.

[60] H. Shen, L. Cai, and X. Shen. Performance analysis of tfrc over wireless links with truncated link level arq. *IEEE Trans. on Wireless Communications*, 5(6):1479–1487, June 2006.

[61] Jitae Shin, Daniel C. Lee, and C.-C. Jay Kuo. *Quality of Service for Internet Multimedia.* Prentice Hall, 2003.

[62] TCP Friendly Rate Control (TFRC): Protocol Specification. http://www.icir.org/tfrc/.

[63] S. Sural and P. Das. A two-state markov chain model of degraded document images. *In Proc. of International Conference on Document Analysis and Recognition*, pages 463–466, September 1999.

[64] Andrew S. Tanenbaum. *Computer Networks.* Prentice Hall, 2003.

[65] Jacobson V. Modified tcp congestion avoidance algorithm. *end2end-interest mailing list, ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail*, April 30, 1990.

[66] Jacobson V. Congestion avoidance and control. *Computer Communication Review*, 18(4):314–329, Aug. 1988.

[67] Stevens W. Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. *RFC 2001*, January 1997.

[68] J. Widmer, R. Denda, and M. Mauve. A survey on tcp-friendly congestion control. *IEEE Network*, 15:28–37, May 2001.

[69] Liu Y. and Claypool M. Using redundancy to repair video damaged by network data loss. *in Proceedings of IST/SPIE/ACM Multimedia Computing and Networking (MMCN)*, January 2000.

[70] Guang Yang, Ling-Jyh Chen, Tony Sun, Mario Gerla, and M. Y. Sanadidi. Real-time streaming over wireless links: A comparative study. *ISCC*, pages 249–254, 2005.

[71] Guang Yang, Tony Sun, Mario Gerla, M. Y. Sanadidi, and Ling-Jyh Chen. Smooth and efficient real-time video transport in the presence of wireless errors. *TOMCCAP*, 2(2):109–126, 2006.

[72] Bin Zhou and Cheng Peng Fu. An enhancement of tcp veno over light-load wireless networks. *IEEE Communications Letters*, 10, June 2006.