BALANCING LOSS DISCRIMINATION AND CONGESTION CONTROL IN A RATE-BASED PROTOCOL

BY

SHRAVAN GAONKAR

B.E., Mangalore Univeristy, 2001

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois

To my father and the memory of my mother,
whom I am forever indebted

ABSTRACT

The stability of the Internet depends on the presence of well-behaved flows. While TCP forms the majority of the traffic over the Internet, the Internet user community is also dedicated to supporting mobile devices over wireless networks. However, TCP-like behavior is not well suited to support applications over lossy wireless networks. Ideally, these applications require a protocol that adapts to network conditions with the ability to discriminate congestion-based and transmission-based losses.

In this thesis, we present a rate-based transport protocol that is designed to support Internet-based communication for mobile nodes using infrastructure-based wireless networks. We propose an end-to-end approach to loss discrimination based on network state estimation at the receiver. Discrimination is achieved by correlating short-term history of packet inter-arrival times with the loss. Simulations of our protocol prototype show that rate-based protocols can provide better correlation to network conditions than ACK-based protocols. We present an analytical model of XRTP's congestion control that enabled us to detect the reasons for XRTP's aggresive behavior. We provide extensions to XRTP's congestion control and by extensive evaluation through simulations, we demonstrate that our protocol is well-behaved over a wide range of scenarios.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Robin Kravets, for her support and dedication to her research group. I thank her for the academic guidance and the opportunity to be a member of her research group. I would like to thank my friends Arshad Ahmed, Lee White Baugh, Pradeep Kyasanur, Romit Roy Choudhury, ... with whom I spend most of my free time. In particular, I thank Pradeep Kyasanur, who always has time and patience to listen to me.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

ACK	Acknowledgment.
CBR	Constant Bit Rate.
ECN	Explicit Congestion Notification.
ELN	Explicit Loss Notification.
EWMA	Exponential Weighted Moving Average.
FCFS	First Come First Serve.
IEEE	Institute of Electrical and Electronics Engineers.
IP	Internet Protocol.
IR	Infrared.
KDE	Kernel Density Estimation.
RAP	Rate Adaptive Protocol.
RED	Random Early Detection.
RTT	Round Trip Time.
SMCC	Streaming Media Congestion Control.
TCP	Transmission Control Protocol.
TFRC	TCP Friendly Rate Control.
UDP	User Datagram Protocol.
WTCP	Wireless Transmission Control Protocol.
XRTP	eXtended Rate-based Transport Protocol.

CHAPTER 1

INTRODUCTION

Advances in communication technology have increased the reliability of wired networks to the point where transmission losses are rare. As a result, losses in the Internet are generally due to congestion caused by path overuse. Such specific knowledge about network reliability has enabled TCP to be tuned to assume that all losses are caused by congestion. The introduction of technologies such as radio networks with higher loss rates breaks this assumption. Some losses can be handled by making the lossy link appear reliable. For example, wireless technologies like IEEE 802.11 [1], provide link-layer error correction to mask packet loss due to transmission errors. Such an approach assumes that the transmission losses can be recovered locally in a fashion that will not adversely affect the end-to-end transmission of data. These approaches do not, however, guarantee that no transmission losses will occur. Therefore, it is necessary to adapt transport protocols to handle transmission losses while retaining important transport protocol behavior such as congestion control and fairness to other flows. The challenge to designing such a protocol lies with the fact that a communication channel may span both wired and wireless links, introducing both congestion and transmission losses into the channel.

In general, transport protocols should react to congestion losses in a fashion that helps alleviate congestion in the network. If congestion losses are treated as transmission losses, the sender will not decrease its offered load and more congestion will build up in the network. On the other hand, losses due to transmission do not indicate congestion.

Therefore, if transmission losses are treated as congestion losses, the sender will unnecessarily reduce its offered load, reducing the throughput of the stream. Either type of misclassification of losses can have detrimental effects on the communication flow and on the network. These limitations demand the design of a protocol that enables distinguishing congestion losses from transmission losses.

Recently, there have been several proposals to optimize TCP for wireless networks [2, 3, 4, 5]. Some of these solutions [2, 3] maintain the semantics of TCP congestion control, hiding transmission losses from the end hosts by modifying the underlying infrastructure. Techniques like Explicit Loss Notification (ELN) [6] and Explicit Congestion Notification (ECN) [7] can be integrated into an end-to-end approach to provide almost perfect knowledge of the cause of a loss. However, such an approach requires the expensive replacement of routers and other support infrastructure making deployment impractical. Therefore, a precise end-to-end loss discrimination scheme is necessary to maximize the performance of a transport protocol in dynamic wired-wireless networks without support from the underlying infrastructure.

In this thesis, we propose and analyze a rate-based protocol called eXtendend Rate-based Transport Protocol (XRTP). The rate-based nature of XRTP provides regularity of packet transmission that enables the tracking of congestion in the network, allowing XRTP to discriminate packet losses effectively and support proactive congestion avoidance. The functionality of proactive congestion avoidance is achieved through estimation of available bandwidth using a combination of packet-pair [8] and jitter (i.e., the difference between inter-arrival times of packets at the receiver and inter-sending times at the sender). Given the potential inaccuracies in bandwidth estimation using packet-pair, XRTP integrates non-parametric density estimation [9] to filter out irrelevant measurements. These mechanisms allow XRTP to achieve good throughput in combined wired/wireless environments while maintaining fairness in the presence of competing traffic. The XRTP design supports both bulk data transfer and multimedia traffic. How-

ever, in this thesis, we present a prototype targeted at bulk data transfer. An extension to multimedia traffic would not impact the flow control or congestion control mechanisms, but simply require different reliability support with knowledge of priorities and timeliness. In addition, the rate-based nature of XRTP naturally supports multimedia traffic.

The rest of this thesis is organized as follows. Chapter 2 discusses transport protocol design in the context of current research in the area. Chapter 3 presents the design of the XRTP. Chapter 4 presents the performance evaluation of the protocol through simulations. Chapter 5 analyzes XRTP's congestion control mechanism and provides extensions to make it network friendly. Finally, in Chapter 6, we present conclusions and directions for future research.

CHAPTER 2

TRANSPORT PROTOCOLS FOR WIRED/WIRELESS NETWORKS

Without explicit loss or congestion notification from the network, successful loss discrimination is dependent on implicit mechanisms available at the transport layer to classify losses. Since transport protocols implicitly monitor the transmission stream for indications of congestion, similar techniques can be used to determine if there is congestion when a loss occurs. This integration of loss discrimination and congestion control must be designed into a protocol to insure correct behavior.

Traditional protocols designed for low loss wired links do not incorporate appropriate recovery mechanisms for transmission losses. Simply adding loss discrimination techniques to existing transport protocols is ineffective since the protocol's flow and congestion control mechanisms would not work synergistically with loss discrimination. Without support from the underlying infrastructure, designing an end-to-end transport protocol for wireless networks that enables effective loss discrimination requires three separate components: flow control, bandwidth estimation and loss discrimination. This section describes various issues involved with the design of these mechanisms, providing insight into the design decisions of XRTP.

2.1 Flow Control

Current transport protocols use either ack-based or rate-based mechanisms to control the transmission flow of a stream. Ack-based protocols have dominated in wired environments due to their self-clocking nature and ease of implementation. However, rate-based protocols, such as RAP [10] and TFRC [11] have become popular due to their smoother transmission when compared to the burstiness of ack-based protocols. Essentially, rate-based flow control provides a regularity in transmission that supports improved monitoring of network conditions. Additionally, packet losses due to congestion will be reduced since router queues will be uniformly filled by regular packet transmissions instead of overflowing from bursts. While rate-based protocols have been proposed for wired networks, current protocols are not directly applicable to wireless networks. RAP mimics the additive increase/multiplicative decrease congestion control scheme of TCP. However, this approach inhibits RAP from reacting to rapidly changing bandwidth as expected in wireless networks. TFRC is a TCP throughput equation-based rate-control protocol that includes packet loss rate and round trip time in the congestion control algorithm. The losses experienced by a flow shapes the throughput of TFRC. Without explicit loss discrimination, high loss rates degrade the performance of TFRC.

There have also been several protocols proposed to enhance TCP with packet-spacing to exploit the advantages obtained by transmitting packets at regular intervals while maintaining the semantics of TCP [12, 13]. Even though packet-spacing alleviates the bursty nature of TCP, it does not de-couple the protocol's reliability mechanisms from congestion control. Retransmissions in TCP for reliability could potentially stop the transmission of packets limiting the benefits gained by packet spacing. Wireless-TCP [14] is a reliable end-to-end rate-based transport protocol designed to operate on wireless wide-area networks. WTCP tracks the ratio of inter-packet separation at the receiver to inter-packet separation at the sender. This ratio corresponds to the ratio of available bandwidth and actual bandwidth used by the flow. WTCP tries to achieve the target available

bandwidth by keeping this ratio close to one. WTCP also uses long-term jitter to track congestion and damp the rate of protocol. However, long-term jitter is not the most accurate metric since channel conditions in wireless networks change too rapidly to correlate channel conditions with long-term jitter. In XRTP, we extend the approach used by WTCP by using observations about short-term jitter to monitor congestion in the network.

2.2 Bandwidth Estimation

The goal of a well-behaved transport protocol is to target its rate at a level close to the available bandwidth in the network. In general, the available bandwidth, and so the target rate, to a flow is a dynamic value, constantly changing with channel usage. Transport protocols can monitor the transmission stream to infer information about bandwidth in the network. However such information may track two very different parameters: bottleneck bandwidth and available bandwidth. Bottleneck bandwidth can be defined as the maximum throughput that can be obtained between two hosts in the absence of any cross traffic. Available bandwidth is a flows fair share of the bandwidth in the presence of other flows.

Ideally, transport protocols use estimates of available bandwidth to prevent unfair use. However, measuring available bandwidth or fair share is a non-trivial task in the current Internet where most routers use a first-come first-serve (FCFS) scheduling policy. Keshav [15] observed the intrinsic difficulty in measuring available bandwidth in FCFS routers when looking at the use of packet-pair in congestion control. Essentially, packet-pair always measures bottleneck bandwidth instead of available bandwidth. Protocols like Streaming Media Congestion Control (SMCC) [16] use packet-pair to determine an estimate of available bandwidth. However, their approach assumes the use of RED [17] enabled routers that are not common in today's Internet. Dovrolis [18] showed that

available bandwidth could be estimated for FCFS routers in a one hop network using packet-pair along with an estimate of cross-traffic utilization of the bottleneck link. By extending this to an infrastructure-based wireless network with the last hop to the mobile node as the bottleneck link, XRTP estimates the available bandwidth using packet-pair in conjunction with jitter measurements (which indicate cross-traffic utilization) and packet loss. This approach is limited by the number of jitter measurements observed in a round-trip time. Thus, XRTP can use packet-pair to estimate available bandwidth, provided it is able to maintain sufficient packets in flight to gage network conditions.

The first step to estimating available bandwidth is to accurately determine the bottleneck bandwidth. Techniques like packet-pair have been widely used by off-line tools that measure bandwidth (e.g. *tcpanaly* [19], *bprobe* [20]). However, measurements from packet-pair often include erroneous values and the data filtering techniques used by these tools lack statistical robustness. Lai et. al. [21] suggest the use of kernel density estimation (KDE) to filter data for estimating the bandwidth of all the hops from the source to the destination. The disadvantage of directly incorporating such estimation algorithms from these bandwidth measurement tools into a transport protocol lies in the computational complexity of the algorithms. Essentially, offline measurement tools can afford time and resource intensive computations to converge onto accurate estimates, while transport protocols need to be fast and efficient to maximize performance. XRTP uses an optimized KDE algorithm that provides efficient filtering without loss of accuracy using techniques like finite differencing.

2.3 Loss Discrimination

Solutions for adapting transport protocols for networks with heterogeneous loss characteristics can be classified into three approaches: transparent or infrastructure-based, hybrid or infrastructure-assisted and end-to-end. Infrastructure-based approaches try to

hide losses from the transport protocol, which is most often TCP, by adding changes along the path, either at the link layer [22] (i.e., TCP-SNOOP [2]) or at the transport-layer (i.e., indirect-TCP [3]). Such solutions either blindly recover all losses, which may negatively impact some protocols, or require explicit knowledge of the transport protocol, which may not be extensible to new transport protocols. Additionally, infrastructure-based approaches like I-TCP violate the end-to-end semantics of TCP. In infrastructure-assisted or hybrid approaches, the transport protocol is modified along with support from the underlying network to indicate congestion losses (explicit congestion notification (ECN)) [7] or transmission losses (explicit loss notification (ELN)) [6]. Krishnan et. al. [5] have extensively analyzed the potential performance benefits of using ELN. While such protocols approach ideal behavior due to the accurate classification of losses, deployment of such hybrid approaches is limited by the necessity to change routers in the network.

The final approach tries to classify losses using purely end-to-end mechanisms [5, 23, 24, 25]. TCP Westwood [4] does not have an explicit scheme to discriminate losses, but the authors claim that rate estimation can account for wireless losses. Bandwidth estimates are based on information about the rate at which ACKs are received. After a packet loss indication (i.e., three duplicate ACKs), which could be due to either congestion or link errors, the sender uses the estimated bandwidth to properly set the congestion window and the slow-start threshold. However, this scheme will not accurately account for all types of losses making it too conservative to discriminate all transmission losses. A number of protocols have been proposed that enhance TCP to achieve loss discrimination. Biaz et. al. [23] distinguish between congestion and wireless loss using packet inter-arrival times at the receiver. Barman et. al. [24] assume that the variation in the round trip time and the nature of the loss are correlated. Essentially, congestion losses cause the RTT to vary over the standard deviation but random losses do not. Samaraweera [25] correlates the round trip time with the throughput-load graph of the flow. They assume that a packet loss about a certain threshold is usually due to congestion, otherwise it is due to wireless or random loss. The limitations of all these end-to-end approaches is due

to the fact that they propose their heuristics for TCP-like ack-based protocols that are bursty in nature. However, these heuristics are better suited for rate-based protocols that transmit at regular intervals, [13], providing better correlation of actual network conditions with heuristic parameters like round-trip time, variance of round-trip time or the throughput-load graph of the flow. In the context of rate-based flow control, XRTP can better correlate inter-packet spacing or measured jitter with network conditions to discriminate congestion-based losses from transmission-based losses. To be conservative in discrimination and reduce misclassification of losses, XRTP also incorporates the relative one-way trip time and its variance into the heuristic to better gage network conditions.

CHAPTER 3

EXTENDED RATE-BASED TRANSPORT PROTOCOL

To support data transmission in combined wired / wireless environments, we present XRTP, a rate-based protocol that supports discrimination between congestion-based and transmission-based losses. The rate-based component of XRTP provides smooth transmission of data, while supporting effective estimation of network conditions. The XRTP receiver closely monitors available bandwidth through a combination of packet-pair and measured jitter. This monitoring allows the receiver to react to pending congestion (i.e., congestion avoidance) and classify losses. The bandwidth estimation techniques are enhanced with a non-parametric density estimator to filter out anomalies. In this section, we present the protocol architecture followed by the main mechanisms of XRTP: bandwidth estimation, congestion control and loss discrimination.

3.1 Protocol Architecture

XRTP is targeted at environments where losses caused by transmission errors are orders of magnitude greater than for current wired links. XRTP estimates the available bandwidth between end hosts using a combination of packet-pair and measured jitter at the receiver side to operate at a level that is fair to other flows using the same intermediate

links in the network. XRTP reduces the transmission rate based in reaction to measured positive jitters to prevent network congestion proactively.

In a lossy environment, XRTP stabilizes throughput with a loss discriminating heuristic by breaking the fundamental assumption that all losses of packets are due to congestion. The protocol relies on the fact that loss of any type would create an anomaly in the jitter. By classifying the anomaly accurately, XRTP categorizes the packet loss, achieving better throughput compared to protocols without loss discrimination.

The design of XRTP decouples reliability from flow control. By combining rate-controlled acknowledgments with a monotonically increasing sequence number for all packets, XRTP eliminates the need for a retransmission timer. Essentially, sequence numbers and gap detection are used to detect losses and window-based selective acknowledgments are used to inform the sender about any losses. While loss discrimination is used to determine the impact of a loss on the transmission rate, the reliability mechanisms (i.e. retransmission) can be based on more application-specific knowledge. For bulk data traffic, XRTP retransmits all lost packets. To support multimedia traffic, XRTP uses information like packet deadlines and priorities to determine which packets should be retransmitted and which should be dropped. Since the focus of this thesis is on the congestion control and loss discrimination mechanisms of XRTP, exact details of the reliability mechanisms have been omitted.

3.2 Bandwidth Estimation

XRTP estimates available bandwidth using packet-pair and jitter measurements at the receiver to set a target transmission rate. Bottleneck bandwidth must be estimated constantly since it could vary dynamically due to changes in channel conditions. The advantage of measuring bottleneck bandwidth using packet-pair is due to the fact that

it can be done with two data packets and it is very simple to implement. This section describes the packet-pair technique and the statistical method used to filter the observed data to obtain accurate estimates.

3.2.1 Packet Pair

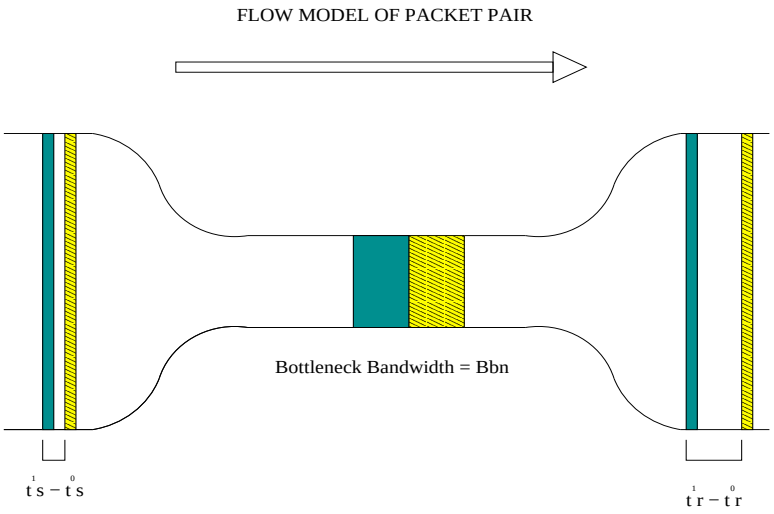

Figure 3.1 Fluid model of packet pair with two packets.

The fluid flow model in Figure 3.1 depicts two packets of the same size traveling from a source to a destination. The narrow part represents the bottleneck link. Once the packet-pair leaves the bottleneck link, a gap is inserted between the two packets due to queuing at the bottleneck link. Let B_{bn} be the bottleneck bandwidth, S be the size of the packet, t_0^s and t_1^s be the time when the first and second packets are sent back-to-back from the sender and t_0^r and t_1^r be the time when they are received at the receiver. To maintain conservation of flow, the equilibrium equation for the described model is given by

$$t_1^r - t_0^r = max\left(\frac{S}{B_{bn}}, t_1^s - t_0^s\right). \tag{3.1}$$

The equation reveals that if the two packets are sent close enough in time forcing them to queue at the bottleneck link back-to-back, the packets will arrive at the receiver with

the same spacing, $t_1^r - t_0^r$, as the spacing introduced by the bottleneck link's bandwidth, $\frac{S}{B_{bn}}$. Rearranging Equation (3.1), the bottleneck bandwidth can now be computed as

$$B_{bn} = \frac{S}{t_1^r - t_0^r}. \tag{3.2}$$

However, Equation (3.2) only holds provided that the two packets are queued only at the bottleneck link and at no later link downstream, the routers implement first-come first-serve (FCFS) queuing with simple store and forward and the transmission delay is linear with respect to packet size [21]. Among these assumptions, it is extremely difficult to satisfy the first since packets travel through multiple hops and each hop carries multiple flows. Due to this dynamic nature of the network, packets from other flows can get inserted between the packet pairs increasing the gap between the packets and causing an overestimation of the bottleneck bandwidth (time-expansion). The packet-pair may also get queued at a later router decreasing the gap between packets causing an underestimation of bottleneck bandwidth (time-compression). To satisfy the first assumption and enable the use of packet-pair for bottleneck bandwidth estimation, a statistically robust filtering algorithm is necessary to eliminate anomalies like time-compression and time-expansion. XRTP uses a modified version of a kernel density estimation (KDE) algorithm [9] particularly adapted to the rate-based protocol.

3.2.2 Kernel Density Estimation

Probability density functions are the basis for estimation of any random quantity. Consider the set of observed data points of an unknown probability density function. To avoid making assumptions about the distribution of the observed data, a non-parametric approach such as KDE [9] can be used to filter the observations. The best estimate of the observed data would be the mean of this random variable.

The probability density function for a kernel estimator with kernel K is defined by

$$f(x) = \frac{1}{nh(x)} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h(x)}\right). \tag{3.3}$$

Here, *h(x)* is the variable window width, also called the smoothing parameter or bandwidth, n is the number of observations collected and K is the kernel function that satisfies the following condition

$$\int_{-\infty}^{\infty} K(x)dx = 1.$$

The time complexity of the KDE algorithm is of order $O(n^2)$. Unlike Lai et. al. [21] who used this estimation technique for measuring bottleneck bandwidth in their off-line tool *pathchar*, estimating bandwidth on-line in a transport protocol requires that the estimator algorithm be optimized to work as efficiently as possible. Looking at the estimator function as described in Equation (3.3), for a fixed width h, a standard finite differencing technique can be used to reduce the algorithm to linear time (see Figure 3.2). Specifically, each time a new observation or measurement is added into the estimator, the density of the observation being removed (oldest observed measurement) can be subtracted from all observations and the density of the new observation can be added to all observations in linear time. In an adaptive estimation algorithm where h varies, there is a possibility of the introduction of an error term into the estimation in the linearized KDE algorithm due to finite differencing. However, this error is not cumulative and is orders of magnitude smaller than the bandwidth estimates, making the error negligible and limiting the impact on the accuracy of estimation. Therefore, XRTP uses this linearized KDE algorithm to filter out irrelevant observations (i.e., effects of time-compression and time-expansion) to accurately estimate bottleneck bandwidth.

3.3 Rate and Congestion Control

XRTP sends packet-pairs once every four packets to estimate bottleneck bandwidth. Each time the receiver estimates the new bandwidth, it uses a weighted average of the current

array obsrv[n] // last n observations
array pdf[n] // density estimations
est // current estimate
function bandwidth_estimation(new)
  // subtract the density of the oldest observation
  for i = 2 *to* n
    $pdf[i] - = \frac{K(\frac{est-obsrv[i]}{h})}{n.h}$
  next i
  // delete the oldest observation along
  // with addition of new density estimate
  for i = 1 *to* n-1
    $pdf[i] = pdf[i+1] + \frac{K(\frac{est-new}{h})}{n.h}$
    obsrv[i] = obsrv[i+1]
  next i
  // compute the density of new observation
  obsrv[n] = new
  pdf[n] = $\frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{est-obsrv[i]}{h}\right)$
   // compute new width h and estimate i.e., mean
  // return new estimate
end function

Figure 3.2 Linear kernel density estimation for bandwidth filtering.

rate and the new estimated bandwidth to update the rate of the protocol. Exponentially weighted moving average (EWMA) is a poor estimator to measure bottleneck bandwidth accurately since it cannot eliminate spurious observations which the KDE does elegantly. However, EWMA is extremely efficient in maintaining moving averages provided the observations are accurate. Therefore, XRTP updates the new rate using the standard EWMA equation with smoothing parameter α (see Equation 3.4). The values of α can range from 0 to 1 and these values determine the reactiveness of XRTP. XRTP sets the value of α to 0.8 or below to be reactive to network conditions. The receiver sends the updated rate to the sender in the acknowledgments.

$$new\_rate = old\_rate * \alpha + new\_est\_rate * (1 - \alpha) \tag{3.4}$$

When there are multiple flows sharing a link, running XRTP at the bottleneck rate would cause congestion since the bottleneck rate is not the fair share. To prevent the potential of congestion by transmitting data at a rate greater than the fair share, XRTP provides a congestion avoidance mechanism. Since XRTP sends packets at regular intervals, they are expected to be received at regular intervals; in other words, the inter-arrival time between packets at the receiver should be equal to the inter-sending time at the sender. In reality, the inter-sending time may not equal the inter-arrival time at the receiver due to queuing delays and the presence of other flows. XRTP keeps track of this jitter, the difference between the inter-arrival time and the inter-sending time, to damp the rate and operate close to its fair share of the bandwidth.

To estimate fair share at the receiver, XRTP observes the arrival time of packets. Let the arrival time of the n^{th} packet, t_r^n, be the sum of the time it was sent, t_n^s, and the transmission time, tt_n (See Equation (3.5)).

$$t_n^r = t_n^s + tt_n \tag{3.5}$$

Transmission time can be divided into two components, propagation time, pt_n and queue time, q_n (at the routers), as shown in Equation (3.6). If the routes are not changing,

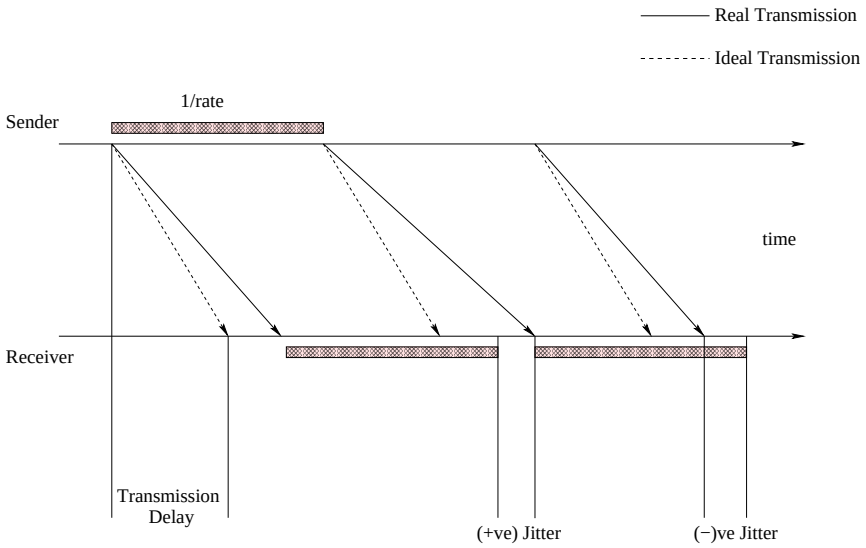

Figure 3.3 Jitter caused by variation in queuing delay.

propagation time, pt_n, is constant since it only depends on the transmission speed of the media. Queue time, q_n, mainly includes the time the packets wait to be processed and negligible processing time.

$$tt_n = pt_n + q_n \tag{3.6}$$

The number of packets along the path of communication represents the load in the network. As this number increases, a packet in transmission experiences longer wait times at the router, and so longer end-to-end transmission times. On the other hand, as network load decreases, end-to-end transmission time decreases to the limits of the propagation delay on the transmission medium. The inter-arrival time, iat_n, between two successive packets, $n-1$ and n, can be used to determine the presence of congestion in the network. Substituting Equation (3.5) into Equation (3.7), we get Equation (3.8) for inter-arrival time.

$$iat_n = t_n^r - t_{n-1}^r \tag{3.7}$$

$$iat_n = (tt_n - tt_{n-1}) + t_n^s - t_{n-1}^s \tag{3.8}$$

Since the sending time of the packets is known, it can be subtracted from Equation (3.8) to get a measure of load in the network, as shown in Equation (3.9). Thus, jitter is governed

by the difference in packet travel time. Along with the assumption that propagation time is constant, the only variant is waiting time in the queue as shown in Equation (3.11). Therefore, the main component of jitter is the time the packet spent waiting in the router queues, which in turn indicates the amount of congestion in the network.

$$jitter_n = (tt_n - tt_{n-1}) \tag{3.9}$$

$$jitter_n = (pt_n - pt_{n-1}) + (q_n - q_{n-1}) \tag{3.10}$$

$$jitter_n = q_n - q_{n-1} \tag{3.11}$$

XRTP uses time-stamps to determine the difference in send time. Therefore, the jitter for the n^{th} packet can be computed as

$$jitter = (t^r_n - t^r_{n-1}) - (t^s_n - t^s_{n-1}). \tag{3.12}$$

In general, jitter may be positive or negative since delays from the previous or current packet may cause the current packet to arrive early or late. Ideal or zero jitter indicates that the queue levels at the routers are maintained at a constant level. Positive jitter indicates increasing queue lengths and occurs when the cumulative rate of all flows is greater than the capacity of some link along the path. Negative jitters indicates decreasing queue length and occurs when the network is unloading or when packets are lost due to congestion. In all cases, positive jitter implies that the current rate could cause unfair use of bandwidth. Thus, positive jitter triggers congestion avoidance and cause XRTP to reduce its rate. XRTP tries to maintain zero jitter to prevent queue buildup. Intuitively, positive jitter indicates the amount by which XRTP overshot the available bandwidth. To be conservative, XRTP cuts downs its rate based on the average of the history of the past three positive jitters.

$$new\_rate\ -= \frac{3}{\sum of\ past\ 3\ positive\ jitters} \tag{3.13}$$

XRTP always reacts to congestion losses by cutting its rate in half, essentially using multiplicative decrease. After a change in rate, it would take at least half the round-trip time for the sender to react and another half for the receiver to notice a change. Therefore,

Table 3.1 Content of DATA and ACK packet in XRTP.

DATA packet	$packet_{seqnumber}, reliability_{seqnumber}$, $timestamp, datasize$,DATA
ACK packet	$packet_{seqnumber\_echo}, CACK, SACK$, $updated\_rate, echo\_timestamp$

similar to other protocols, XRTP cuts its rate by half once per round-trip time (RTT) for any packet loss due to congestion.

3.4 Reliability

Estimating accurate round trip time (RTT) or precise retransmission timeout (RTO) has always been a difficult problem [26, 27]. Allman and Paxson showed that there is an intrinsic difficulty in finding optimal parameters to make an accurate estimation for retransmission time [26]. Given the difficulties and inaccuracies associated with any scheme of achieving reliability using retransmission timers, XRTP uses selective acknowledgments (SACK) and monotonically increasing sequence numbers to achieve reliability. The receiver sends an ACK for every data packet received. SACK provides advantages in situations where the ACKs are lost in the channel. SACK enables the accurate reflection of the state of the receiver in terms of the packets received, thereby reducing the number of unnecessary re-transmission since each ACK will carry information about all the packets that have be logged in the receive window of the receiver.

The reliability mechanism in XRTP relies on a monotonically increasing sequence number, denoted as the *packet_sequence_number*. This scheme has evolved from a similar scheme in RMTP[28] where the sender keeps a log or window of unacknowledged packets. The size of this window is determined by the delay-bandwidth product and is generally set to twice the size of the delay-bandwidth product. Each entry in the window has the following contents: *reliability_sequence_number (RSN), packet_sequence_number (PSN)* and

```
function process_packet(Acknowledgment ack)
        // process the ack.SACK and update the
        // boolean 'received at receiver'
        // to reflect if the packet has already
        // been received by the receiver
        if (ack.CACK ≥ Sender.expectedACK) then
                // no packet loss
                // delete all the window entries
                // whose RSN ≤ ack.CACK
        else
                // find all packets that have not been
                // yet received by the receiver and
                // whose Sender.PSN ≤ ack.PSN_echo
                // and schedule for re-transmission.
        end if

 end function
```

Figure 3.4 Algorithm to process the acknowledgments received at the Sender

a boolean *received_at_receiver*. Each time an ACK is received, the sender tries to determine if there is a gap (i.e., packet loss in the byte sequence sent). Table 3.1 describes the header format of the XRTP data and acknowledgment packet that is used to maintain the state between the sender and the receiver. If XRTP's sender finds that there are no gaps, entries with *reliability_sequence_number* less than the cumulative ACK are deleted. In all cases, the SACK bitmap is used to update the window of unACKed packet by setting the flag *received_at_receiver* to *true*. All packets with *packet_sequence_number* less that the echoed PSN are scheduled for retransmission. The algorithm for processing acknowledgments for reliability at the sender's side is described in Figure 3.4.

Thus, XRTP sender's transmission can be limited by the window size. Once the window gets filled with unacknowledged packets and the sender receives no ACKs, the sender assumes that the channel has blacked out and sends one packet every round trip time. Similar to the sender, the receiver goes into a blackout mode and sends one ACK every RTT if the receiver does not receive data packets for a certain threshold amount of time.

When channel conditions improve, the protocol resumes normal transmission.

3.5 Loss Discrimination

The main challenge in loss discrimination is that the scheme must be very accurate with a very low rate of misclassification, in particular, classifying a congestion loss as a transmission loss. Each time XRTP discriminates a loss as a congestion loss, it goes into the congestion avoidance phase as described in the previous section. Since XRTP only cuts down its rate once per RTT regardless of the number of losses during that RTT, loss discrimination during the congestion avoidance phase has no effect on congestion control. Essentially, a transmission loss classified as a congestion loss will put XRTP into the congestion avoidance phase and render the outcome of loss discrimination irrelevant for one RTT. While this makes it very important to achieve accurate discrimination, it also limits the negative impact of any misclassification's. Therefore, XRTP uses conservative loss discrimination to reduce misclassification's as much as possible while maintaining good congestion control behavior.

XRTP uses the history of jitter measurements to classify losses. As discussed in the previous section, positive jitter indicates congestion in the network. If congestion is not alleviated, the queues overflow and cause a congestion loss. This loss causes packets adjacent to the lost packet in the queue to become closer to each other. Thus, XRTP's receiver notices negative jitter for the packet received after the congestion loss. To differentiate between negative jitter due to a congestion loss and negative jitter due to unloading of the network, XRTP tracks the jitter preceding the loss. Thus, positive jitter followed by negative jitter indicates a congestion loss. This characterization is conservative since XRTP reacts to a transmission loss during the reduction of network load as if it were a congestion loss, reducing the sending rate.

```
function loss_discrimination(packet p)
  if history of positive jitters followed
  by packet loss and negative jitter then
    LOSS = CONGESTION
  else
    // estimate the network congestion
    p = curROTT−minROTT / maxROTT−minROTT
    if p ≥ threshold and l1 out of last
    n ROTT was over deviation then
      LOSS = CONGESTION
    else
      LOSS = TRANSMISSION
    end if
  end if
end function
```

Figure 3.5 Hybrid Loss Discrimination Heuristics.

In last hop wireless networks, packets lost due to link errors during propagation take the same amount of transmission time as packets successfully transmitted. Therefore, the packet following a transmission loss does not experience any significant change in jitter. Hence, a packet loss with no change in the trend of jitters indicates a transmission loss. Similar to other end-to-end loss discrimination schemes, this heuristic with only jitter as a parameter has limitations. Jitter noticed due to congestion loss of a packet could be masked by the presence of cross traffic flows. The empty slot of a lost packet could be occupied by packets from a competing flow instead of the packet following the lost packet from the same flow. This could drastically reduce the noticeable negative jitter at the receiver. Additionally, several lower layer protocols like IEEE 802.11 implement retransmission for packets lost due to transmission error. Retransmission schemes in lower layers severely skew the jitter metric making it difficult for the heuristic to classify losses accurately. Classifying congestion losses as transmission losses is detrimental to the network and other flows. To compensate for such misclassification's and reduce their occurrence, XRTP's loss discrimination incorporates the relative one-way trip times (ROTT) [29] and the deviation of ROTT[24] into the heuristic (See Figure 3.5). XRTP determines the ratio of the difference between current and minimum ROTT to the dif-

ference between maximum and minimum ROTT. The congestion in the network directly correlates to this ratio. XRTP compares this ratio with a threshold to determine the condition of the network when the packet loss occurred. Whenever the heuristic based on jitter classifies a loss as transmission loss, XRTP uses additional heuristic parameters (i.e., ROTT and its deviation) to confirm the transmission loss.

CHAPTER 4

PERFORMANCE EVALUATION AND ANALYSIS

In terms of performance metrics, the most important goal of any transport protocol is high throughput and fair use of available bandwidth. The evaluations in this section are designed to demonstrate the effectiveness of (1) XRTP's accurate bandwidth estimation, (2) the operation of XRTP in lossy environments, and (3) the loss discrimination heuristics of XRTP. To evaluate XRTP, we use experimental results and simulations in ns-2 [30] (version 2.26). The experiments were conducted over a local wireless area network (IEEE 802.11 between two hosts with available bandwidth of 2Mbps to 11Mbps) to evaluate the KDE algorithm, while the protocol was evaluated through simulations.

Figure 4.1 represents the network topology used for the simulation. Router *R1* has traffic sources using XRTP, TCP, TFRC and TCP-Westwood. Link *l2*, from router *R2* to the sink, has a bandwidth of 2Mbps with a 0.01ms propagation delay. This link represents both the wireless link with transmission errors and the shared bottleneck link for all flows in the network. The packet loss rate varies from 0% (ideal case) to 5% to represent a variety of wireless technologies. Link *l1* is set with a propagation delay of 40ms and a bandwidth of 10Mbps. All other links are set with a bandwidth of 10Mbps with propagation delay of 1ms. The queue length at router *R2* is greater than the bandwidth delay product of link *l2* so that it does not become the rate limiting factor for protocols using *l2*. Each simulation is run for 200 seconds. The cross-traffic sources and sinks are

either CBR flows or TCP flows that switch on and off during each simulation run based on a uniform distribution.

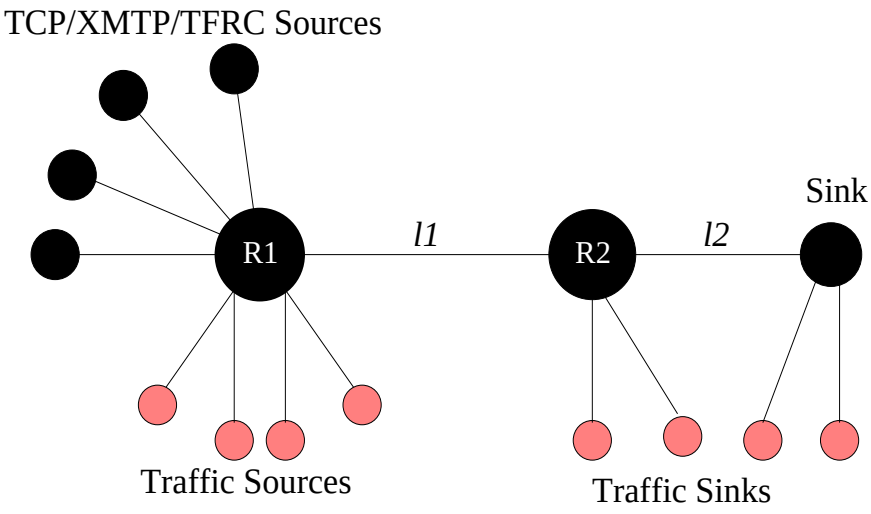

Figure 4.1 Infrastructure-based wireless network topology setup in ns2.

4.1 Comparison of EWMA and KDE

In this section, we demonstrate the effectiveness of an optimized KDE algorithm to estimate bottleneck bandwidth. Since bandwidth estimation in ns2 is unrealistically accurate due to the use of a virtual clock, these traces were generated by running the packet-pair algorithm on a local wireless area network sending packet-pairs every 30 ms. This represents XRTP running on a network with the bottleneck bandwidth ranging from 2 to 11 Mbps. The traces were used as input to compare EWMA with α set to 0.6 and 0.8 and KDE with time-complexity $O(n^2)$ and $O(n)$, where n is the number of sampled data-points stored to make an estimation.

Figure 4.2 represents estimates obtained for these algorithms for one set of generated traces. In this scenario, the bandwidth is 11Mbps from 0 to 5 seconds and falls to 2Mbps for 2 seconds at time 2 seconds. The n value is set to 8 for KDE, which provides good accuracy and agility for most general scenarios. As expected, the algorithms have variance less than the actual measured value (see Figure 4.2) (i.e., they are less noisy) than

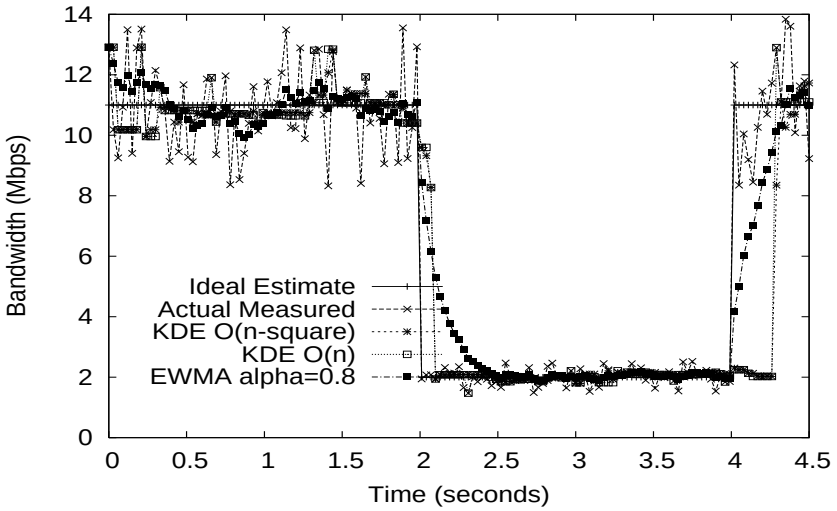

Figure 4.2 Bandwidth estimation using KDE and EWMA.

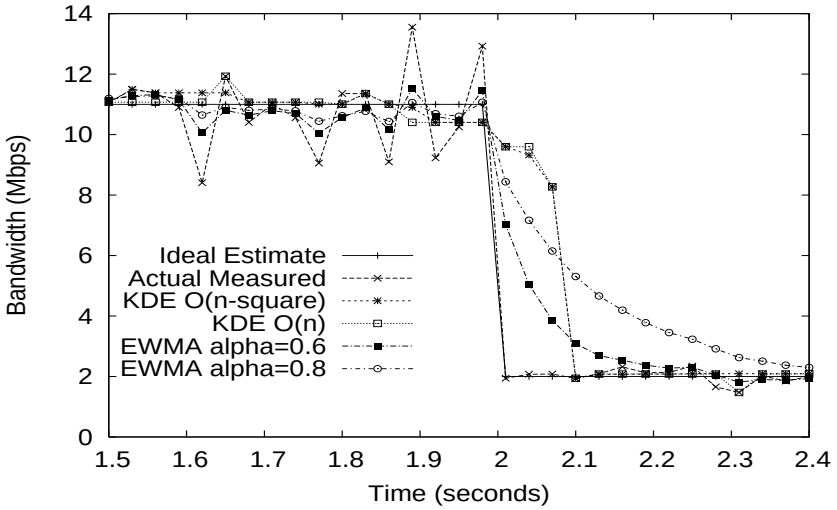

Figure 4.3 Snapshot of bandwidth estimation in wireless network.

the actual observations. The EWMA based estimator takes a longer time to change its estimate from 11Mbps to 2Mbps when the channel bandwidth falls to 2Mbps. Both KDE algorithms are much quicker to adapt to the new 2Mbps bandwidth. To look specifically at the estimated bandwidth traced by the algorithms, a snapshot of (Figure 4.2) from time 1.5 seconds to 2.4 seconds is shown in Figure 4.3. Notice that EWMA overshoots the channel bandwidth at 1.92 and at 2.01 seconds for α set to 0.6. Both KDE algorithms filter out such spurious estimates and stay close to the channel capacity.

The evaluation shows that KDE performs better than EWMA with agility and efficiency providing accurate estimates of the bottleneck bandwidth by eliminating spurious observations. The KDE $O(n)$ algorithm shows similar estimates compared to the KDE $O(n^2)$ algorithm in terms of accuracy. Selecting larger values of n provides more accurate estimates of the bandwidth, but at the expense of agility to adapt to fluctuations in channel bandwidth.

4.2 Evaluation of XRTP's Loss Discrimination Heuristics

The effectiveness of XRTP's loss discrimination techniques is based on the accuracy of the discrimination heuristics. The misclassification of a loss can adversely affect the throughput of the stream or increase congestion in the network. Therefore, we evaluate the probability of each type of misclassification. The goal of the simulations in this subsection is to determine the upper bounds on throughput and the probability of misclassification.

To compare the actual performance gains of using a heuristic to discriminate loss, XRTP was simulated under three different configurations as a single isolated flow on the network

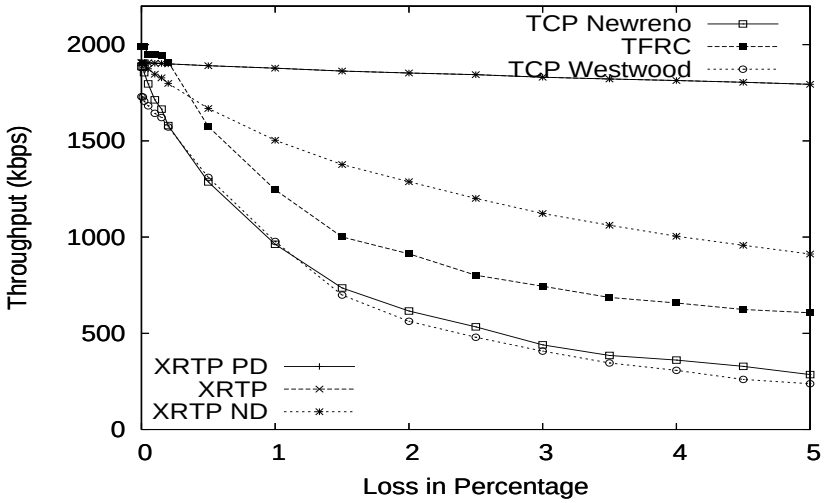

Figure 4.4 Performance of XRTP and other flows in isolation.

topology (see Figure 4.1). In the first configuration, XRTP was run with no discrimination to gage the degree of decrease in throughput with the increase in the loss rate over the wireless link (XRTP-ND). In the second configuration, XRTP was run with the perfect discrimination knowledge (i.e., similar to explicit loss notification) to determine the ideal throughput XRTP can provide in a lossy environment (XRTP-PD). In the last configuration, XRTP was run with the loss discrimination heuristic described in Section 3. Figure 4.4 shows a comparison of the performance of XRTP with each configuration along with TCP Newreno, TFRC and TCP Westwood, each run separately, as the loss rate in the wireless channel is increased from 0 to 5%. As expected, the throughput of TCP Newreno, TFRC, TCP Westwood and XRTP-ND falls rapidly with increasing loss rate, since every transmission loss is considered a congestion loss causing the protocols to back off. XRTP performs exactly like the configuration with perfect knowledge of the type of loss (XRTP-PD). Note that the two lines for XRTP-PD and XRTP overlap each other in Figure 4.4. This implies that every transmission loss was accurately discriminated by XRTP.

In the next experiment (see Figure 4.5), XRTP is again run with the same three configurations, but with cross traffic of CBR flows that cumulatively use 1Mbps of the

Table 4.1 Loss Discrimination Heuristics. XRTP with CBR

Loss	TOTAL	C|C	C|T	T|C	T|T	C|C	C|T	T|C	T|T
0	806	82.63	0	0	0	17.36	0	0	0
0.01	788	82.23	0	0	0	17.63	0	0	0.12
0.02	778	80.97	0.12	0	0	17.99	0.25	0	0.64
0.05	795	81.38	0.62	0	0	16.1	0.75	0	1.13
0.1	785	78.98	0.63	0	0.12	17.83	0.63	0	1.78
0.2	761	75.29	1.83	0	0.13	17.87	1.83	0	3.02
0.5	828	70.41	4.22	0	0.72	14.13	4.1	0	6.4
1.0	864	59.83	7.06	0	1.38	10.41	7.63	0	13.65
2.0	1023	36.65	14.95	0	3.51	5.57	10.36	0	28.93
5.0	1346	12.18	17.53	0	6.53	1.41	9.5	0	52.82

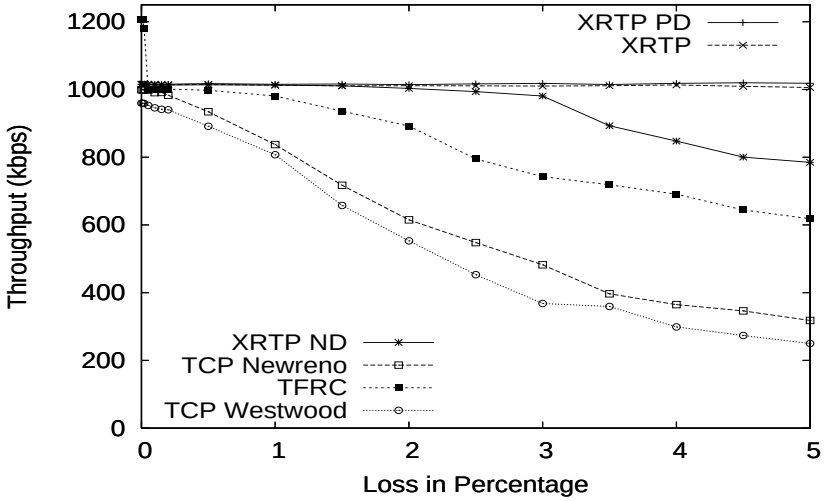

Figure 4.5 XRTP and other flows with CBR cross traffic of 1 Mbps.

bandwidth on Link *l2*. As in the previous simulation, TCP Newreno, TFRC, TCP Westwood and XRTP-ND are hit hard by losses due to transmission while XRTP keeps up with XRTP-PD in terms of throughput. To clearly understand how XRTP is always able to maintain bandwidth close to 1Mbps even with increasing channel loss, it is necessary to look at the losses and understand how the protocol discriminated. Table 4.1 depicts the performance of the loss discriminator used in XRTP. The first column represents the loss rate on wireless Link *l2* in percentage. Column 2 represents the average of the total number of packet losses across the simulations. Columns 3 through 10 represent the loss discrimination done by XRTP in percentages against the actual cause of packet loss. For example, column 4 is denoted by $C \mid T$ and is read as "the percentage of total losses that the heuristic discriminated as congestion loss given the loss was actually a transmission loss". Columns 3 through 6 are loss discriminations that did not affect XRTP's congestion control (i.e., occurred during the congestion avoidance phase). Columns 7 through 10 are loss discriminations that caused XRTP to cut its rate by half.

Notice that the number of congestion losses that caused XRTP to reduce its rate is almost constant (the sum of columns 7 and 8 times column 2). Thus, the throughput of XRTP remains fairly constant even with increasing transmission losses. Also, as transmission loss rates increase, XRTP misclassifies a larger percentage of transmission losses as congestion losses. However, there were no congestion losses that were classified as transmission losses. The reasons can be attributed to the ideal simulation environment presented by ns2. The bottleneck router *R2* is set with a buffer which is a multiple of twice the delay-bandwidth product of link *l2*. Thus, the CBR flows in the worst case fills only half of the queue since they are using half of the bandwidth. Also, XRTP gets the remaining queue space in the router and all the characteristic required for the heuristic are maintained causing no misclassification of congestion loss. In other scenarios (as shown later), there could be misclassification of congestion losses as transmission losses, but it would be very small compared misclassification of transmission losses as congestion

losses since the parameters of the heuristic are set to conservative values.

Table 4.2 Loss Discrimination Heuristics. XRTP with TCP and CBR

Loss	TOTAL	C|C	C|T	T|C	T|T	C|C	C|T	T|C	T|T
0	128	54.68	0	0	0	44.53	0	0.78	0
0.01	127	51.96	0	0	0	44.09	0	3.14	0.78
0.02	144	60.41	0	0	0	36.11	0.69	2.08	0.69
0.05	100	48	0	0	0	47	2	1	2
0.1	108	42.59	0	0	0	42.59	7.4	2.77	4.62
0.2	93	40.86	1.07	0	0	35.48	7.52	1.07	13.97
0.5	106	21.69	6.6	0	0.94	19.81	16.03	0	34.9
1	152	2.63	3.28	0	1.97	3.94	19.07	0	69.07
2	255	0	0.78	0	1.17	0	5.88	0	92.15
5	719	0	0.69	0	0.13	0	1.52	0	97.63

4.3 Evaluation of XRTP in presence of competing flows and cross traffic

To evaluate the performance of XRTP in the presence of cross traffic, the following simulations were run with the same network topology as described in Figure 4.1.

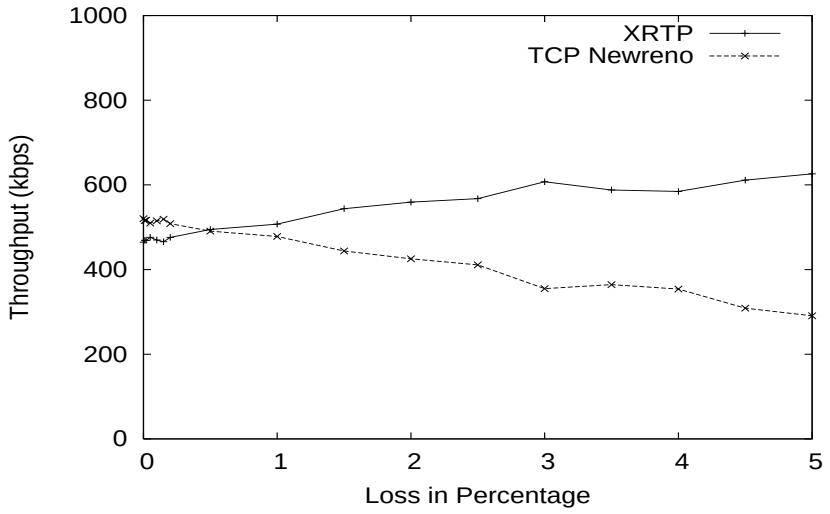

Figure 4.6 XRTP versus TCP Newreno with CBR traffic of 1 Mbps.

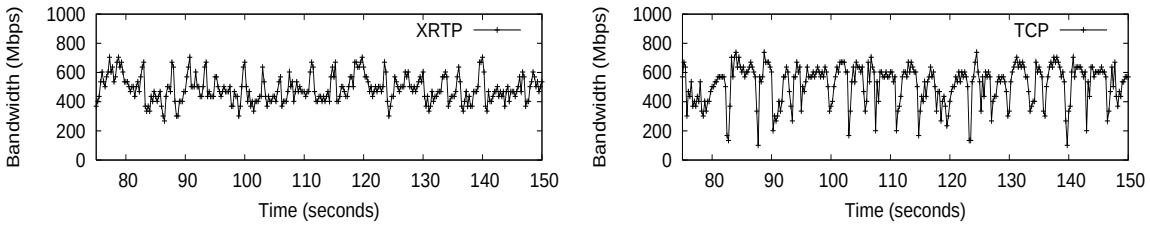

Figure 4.7 Throughput of XRTP and TCP Newreno with CBR flows in time

Figure 4.6 shows the throughput for a single XRTP flow competing with a single TCP Newreno flow with background traffic of CBR flows that cumulatively use 1Mbps of Link *l2*. The graph depicts the throughput of each flow as the loss rate on the congested link increases. Figure 4.7 describes each flows throughput on the congested link averaged over a 0.25 second interval (greater than the round-trip time of nodes in the simulation) for a link loss rate of 0.5%. The throughput of XRTP averaged at 488.5Kbps with standard deviation of 90.54Kbps while TCP averaged at 522Kbps with standard deviation of 135.4Kbps. Complementing the graph, Table 4.2 depicts the loss discrimination of XRTP.

Figure 4.6 clearly shows improved performance due to the use of loss discrimination. Discriminating congestion losses as transmission losses is negligible. It is important to notice that even though the throughput of XRTP is increasing with an increase in loss rate over the wireless link, XRTP is utilizing only that extra bandwidth that is made available by TCP since TCP is reacting to transmission losses as congestion losses. This insight can be deduced by looking at Table 4.2. With increasing loss rates, the percentage of congestion losses (sum of columns 2 and 7) is decreasing, implying that TCP is reacting to transmission losses while XRTP is able to discriminate them as transmission loss. Table 4.2 also shows how XRTP actively uses congestion avoidance to prevent packet losses. The total number of packets lost due to congestion rapidly falls with increasing loss rates, implying that XRTP is aware of the other flows in the network. Consider the simulation with loss rate of 2%. The total number of losses classified as congestion that affected XRTP is about 15 packets (6% of 255) for the whole 200 second simulation although the throughput averaged around 650Kbps. Comparing it with Table 4.1 for the same loss

rate of 2%, the total number of losses classified as congestion losses that affected XRTP is about 92 packets (9% of 1023) while the throughput is about 1000Kbps. This confirms that XRTP's proactive congestion avoidance plays an important role in adapting the protocol in the presence of competing flows.

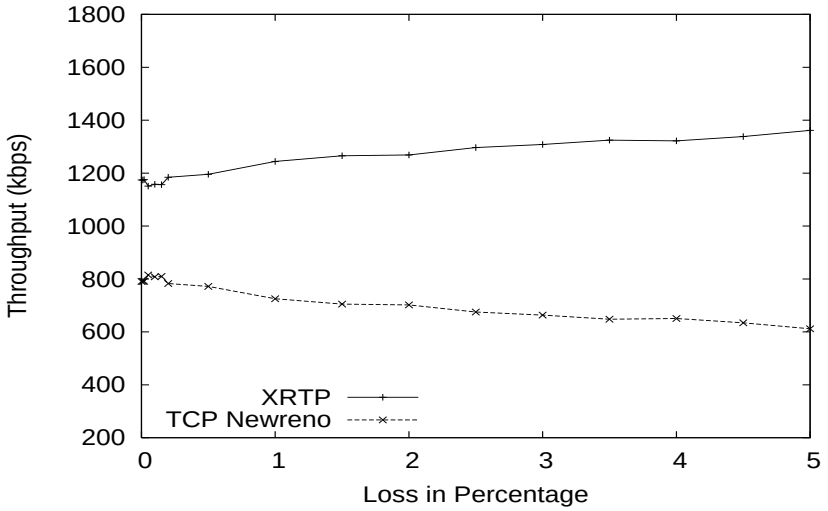

Figure 4.8 Four flows of XRTP against four flows of TCP Newreno.

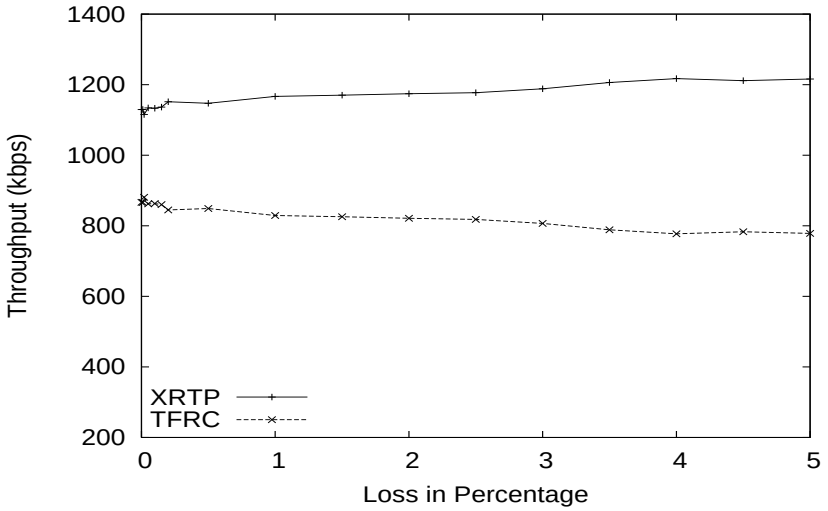

Figure 4.9 Four flows of XRTP against four flows of TFRC.

To complete the evaluation of XRTP in the presence of cross-traffic, Figures 4.8, 4.9 and 4.10 show the results from simulation scenarios with four flows of XRTP compet-

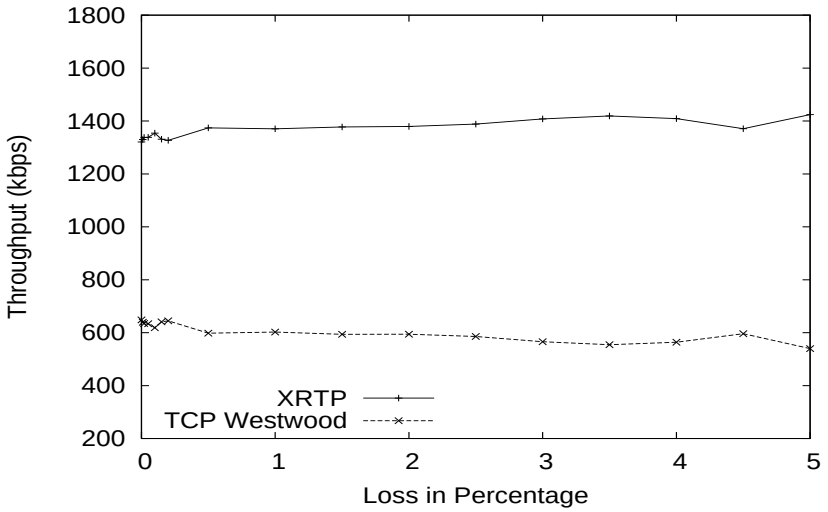

Figure 4.10 Four flows of XRTP against four flows of TCP Westwood.

ing with four flows of TCP Newreno, TRFC or TCP Westwood. Even with increasing loss rates, XRTP flows only use the unused bandwidth made available by the competing flows. Hence, the figures show that the graphs are smooth lines with XRTP's cumulative throughput only increasing by a fraction compared to the throughput without loss. The throughput of the competing flows fall due to the absence of loss discrimination. This also implies that loss discrimination is working accurately with negligible misclassification even in the presence of multiple flows. As discussed earlier, XRTP depends on continuous feedback about network conditions through jitter to be fair to other flows (i.e., XRTP needs to send sufficient packets in each RTT to determine the network condition). Thus, in scenarios, where XRTP is not able to send sufficient packets per RTT, it could behave aggressively to TCP or other competing flows. Chapter 5 analyzes the conditions that lead to aggressiveness and provides solution to alleviate the problem.

CHAPTER 5

ANALYSIS AND EXTENSION OF XRTP

5.1 Effect of Clock Granularity

XRTP, like other rate-based protocols, is very sensitive to the precision of the operating system's internal clock. For example, on a Linux system running on an Intel platform, the best timer resolution is about 10 milliseconds (called a jiffy). Assuming packet size of 1000 bytes and timer precision of 10 milliseconds, a rate-based protocol could perform at its optimum only on those network paths whose bottleneck bandwidth is less than 8Mbps. This problem does not arise in simulators like ns2 because of the use of virtual clocks that allow simulators to send packets at the precise virtual time. To handle this situation, a two pronged strategy can be followed: either provide a mechanism in the protocol itself to handle coarse granularity or devise a method to obtain finer granularity in the operating system like KURT Linux [31].

For the first strategy, it is obvious that there is a limit to the granularity of the timer beyond which rate-based protocols would start behaving like a TCP protocol. Consider a scenario where a single XRTP flow is running on a link which is 100Mbps and the granularity of the clock is 10 millisecond. To obtain optimum throughput, the protocol must send at least a burst of 125 packets in a jiffy with the packet size of 1000 bytes each (as shown in the Figure 5.1). This would greatly alter the scheme for the computation of jitter and other congestion control parameters of the protocol. Since the time difference

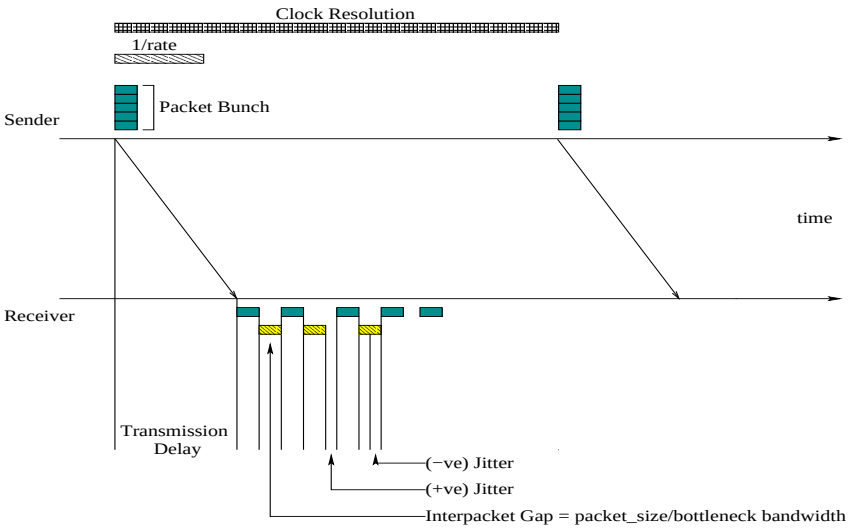

Figure 5.1 Jitters caused by variance in propagation delay using coarse grained timer

between sending two packets could be less that of the transmission time through the bottleneck link, the packets behave similar to packet-pairs, as described in Figure 3.1. To accommodate for this factor in the computation of jitter, it is modified as follows:

$$jitter = (t_r^{n+1} - t_r^n) - max\left((t_s^{n+1} - t_s^n), \frac{S}{B_{bn}}\right) \tag{5.1}$$

With this modification to the computation of jitter, the simulation results of using a coarse-grained rate-controlled protocol has shown results similar to the performance of XRTP without the loss discrimination heuristic. However, the modification to support coarse grained timers with bursty transmission totally undermines the loss discrimination heuristics, pushing us towards the direction of devising methods to provide finer granularity in the operating systems for implementing rate-based protocols.

5.2 Throughput of XRTP

The quantitative analysis of XRTP's average throughput is necessary to clearly understand the congestion control and congestion avoidance mechanisms. The modeling also allows us to determine how friendly XRTP is with the other flows in the network. Since the average rate of transmission directly maps to the throughput of the protocol, it is

sufficient to model the average rate of XRTP. Using this average rate, the throughput can be calculated.

Table 5.1 describes the list of variables and the notation used to represent the model. The set of assumptions that enable modeling XRTP's congestion control are described below. These assumptions are necessary to make the analysis feasible. The model provides an bound on the behavior of the XRTP since the essence of the protocol is maintained in an ideal condition even with these assumptions.

- For any given flow, the fair share bandwidth on the bottleneck link is equal to the fair share of bandwidth for the flow.
- Packet-pair's used to estimate bottleneck bandwidth are never lost.
- The channel capacity and available bandwidth on the bottleneck link does not vary with time.
- There are no transmission errors or multiple packet losses in a single RTT or losses of acknowledgment packets.

Table 5.1 Variable notations used to model XRTP.

Notation	Details
α	Smoothing parameter ($0 \leq \alpha \leq 1$)
B	Available bandwidth per flow
C	Estimated bottleneck Channel Capacity
γ	Overestimate parameter (i.e., number of flows in the network) $\gamma = \frac{C}{B}$ ($\gamma \geq 1$)
k	Packet-pair frequency i.e., once every k packets
J	Average jitter during steady state
R_{avg}	Steady state average rate of the protocol
RTT	Round-trip time
S	Packet size in bits

Packet-pair in XRTP tries to measure the channel capacity, C. XRTP uses the jitter measurements and packet loss to determine the available bandwidth (as discussed in

Chapter 3). XRTP's congestion avoidance is modeled in terms of *epoch's*. Each *epoch* is defined as the time interval between two packet losses (i.e., each packet loss defines the end of an *epoch* and the beginning of a new *epoch*). The following sections describe a steady state analysis of XRTP under various scenarios.

5.2.1 Case $\gamma = 1$

In situations where the available bandwidth, B, is equal to the channel capacity, C, XRTP never has a packet loss, thus the throughput is equal to the channel capacity, C. Such a situation arises when XRTP is an isolated flow in the network.

$$throughput = C \tag{5.2}$$

5.2.2 Case $\gamma > 1$ and $J = 0$

When the available bandwidth, B, is less than the channel capacity, C, XRTP suffers packet losses every time it overshoots its available bandwidth, B. To get a better picture of XRTP's congestion avoidance, consider the start of an *epoch*, where XRTP just overshot its available bandwidth resulting in a packet loss. XRTP cuts the rate by half to account for congestion in the network. For simplicity, let the rate at which XRTP begins to transmit at the start of each *epoch* be r_0, where this rate uses half the available bandwidth.

$$r_0 = \frac{B}{2S} \tag{5.3}$$

After cutting down the rate to r_0, XRTP ramps up its rate each time it estimates the bandwidth (i.e., every time it receives a packet-pair probe) until it overshoots the available bandwidth B again. The protocol loses a packet causing it to cut its rate by half as soon as it overshoots the available bandwidth.

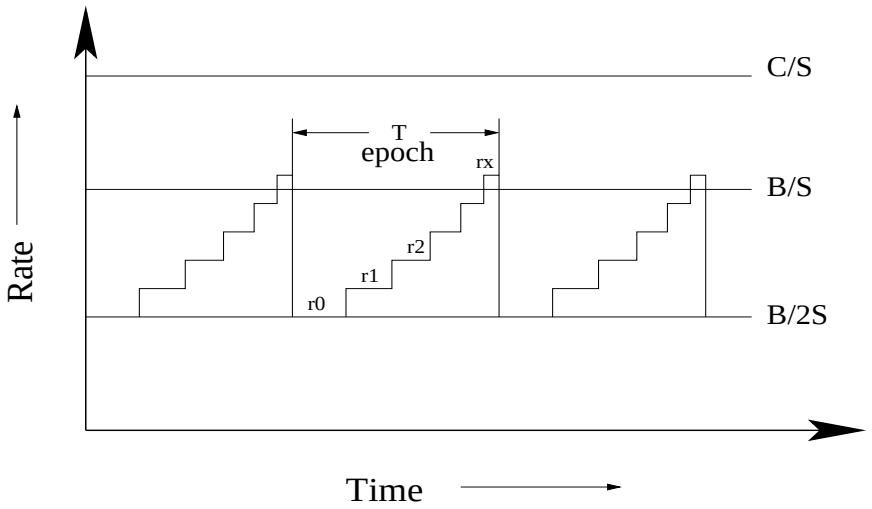

Figure 5.2 Evolution of rate over time in XRTP's congestion avoidance phase.

It is important to know the difference between an *epoch* and RTT to accurately estimate the throughput. In the congestion avoidance phase, XRTP uses RTT as the measure to cut its rate by half for packet losses. However, an *epoch* is defined as the time between the instant XRTP cuts its rate by half and an event of first packet loss after cutting the rate (i.e., when XRTP overshoots its available bandwidth). Figure 5.2 depicts an ideal scenario where the RTT is equal to an *epoch*. In this case, XRTP cuts its rate every time it overshoots its available bandwidth.

The rate increases in a saw-tooth pattern. However, the width of each tooth progressively decreases. Each tooth represents the time XRTP took to send k packets (packet-pair is sent once every k packets). Since XRTP ramps its rate up every k packets, the time to send k packets decreases progressively. Consider the snapshot of the protocol behavior between two loss events. At the start of the first loss event, XMTP cuts the rate in half. Now for every k ACKS received, XRTP's ramps up its rate by a fraction $(1 - \alpha)$ of the new estimate (i.e., $\frac{S}{C}$) using the EWMA (See Equation (3.4)).

$$r_1 = \alpha r_0 + (1 - \alpha)\frac{C}{S} \tag{5.4}$$

Substituting r_0 from Equation (5.3) into Equation (5.4), we obtain:

$$r_1 = \alpha\frac{B}{2S} + (1-\alpha)\frac{C}{S}. \tag{5.5}$$

Simplifying r_1 and substituting γ (γ denotes the number of flows in the network), we obtain:

$$r_1 = \frac{C}{S}\left(1 - \frac{(2\gamma-1)}{2\gamma}\alpha\right). \tag{5.6}$$

As shown in Figure 5.2, let r_x be the rate at which XRTP overshoots the available bandwidth, where the value of x needs to be determined. Here x is defined as the number of packet-probes that need to be sent by the sender before XRTP overshoots its available bandwidth causing a packet loss. Continuing to compute the recurrence relation, after receiving xk packets, the rate of the protocol is

$$r_x = \frac{C}{S}\left(1 - \frac{(2\gamma-1)}{2\gamma}\alpha^x\right). \tag{5.7}$$

In the ideal case, the XRTP's rate, r_x, must always be less than or equal to its available bandwidth:

$$r_x \leq \frac{B}{S} \tag{5.8}$$

Substituting r_x from Equation (5.7) into Equation (5.8), we obtain:

$$\frac{C}{S}\left(1 - \frac{(2\gamma-1)}{2\gamma}\alpha^x\right) \leq \frac{B}{S}. \tag{5.9}$$

Substituting $\frac{C}{B}$ with γ and then simplifying, we get a bound on x. x essentially represents the number of packet probes XRTP requires to overshoot its available bandwidth.

$$x \leq \frac{\ln\left(\frac{2(\gamma-1)}{2\gamma-1}\right)}{ln(\alpha)}. \tag{5.10}$$

The throughput of XRTP can be computed as the total number of packets (in bits) sent in one RTT. Let T be the total time between the two loss events (i.e., *epoch*). Now, T also represents the time XRTP took to begin transmitting at rate, r_x. Since XRTP starts transmitting at rate r_0 at the beginning of each *epoch*, T can be computed as the sum of

the time spent by the sender to send k packets at rate r_i where i varies from 0 to x as shown in Equation (5.11). In total, the sender sends $(k+1)x$ packets.

$$T = \sum_{i=0}^{i \leq x} \left(\frac{k}{r_i} \right) \tag{5.11}$$

$$= \frac{kS}{C} \sum_{i=0}^{i \leq x} \left(1 - \frac{(2\gamma - 1)}{2\gamma} \alpha^i \right)^{-1} \tag{5.12}$$

The total number of bits, P, sent between the two loss events is given by

$$P = (x+1)kS \tag{5.13}$$

Hence, the throughput is obtained by combining Equation (5.11) and Equation (5.13).

$$throughput = \frac{(x+1)C}{\sum_{i=0}^{i \leq x} \left(1 - \frac{(2\gamma-1)}{2\gamma} \alpha^i \right)^{-1}} \tag{5.14}$$

The throughput defined in Equation (5.14) holds, provided that the time T to reach the bottleneck bandwidth (i.e., *epoch*) is greater than round-trip time RTT:

$$T \geq RTT. \tag{5.15}$$

If XRTP reaches its fair-share rate well before the RTT, it overshoots its fair-share of available bandwidth. In situations where the RTT is greater than *epoch* time, ($RTT \geq T$), XRTP continues to ramp its rate up above its fair share. In such cases, the throughput can be computed by evaluating the ratio of the total number of packets sent in one RTT to the RTT (Set $T = RTT$ in Equation (5.11) and determine the value of x that satisfies the equality). The next section analyzes the implication and effect of XRTP's congestion avoidance on other flows using the model derived here. To complete the analysis, Appendix B shows the equivalent derivation of Equation 5.10 incorporating measurable jitter into the model. XRTP without jitter to damp its rate is more aggressive, thus giving an upper bound on its aggressiveness. Hence, the following section analyzes XRTP without incorporating jitter into the model.

5.3 Analysis of the model

The following analysis assumes scenarios with equal number of XRTP and competing flows using a bottleneck link. The competing flows are well-behaved and aggregately use half of the bottleneck capacity. It is also assumes that each XRTP flow starts with a rate equal to half of its fair-share rate.

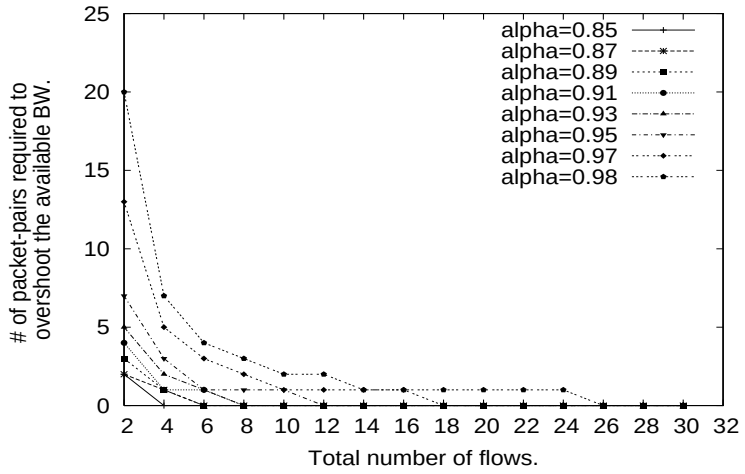

Figure 5.3 Number of packet-pair probes required to overshoot available bandwidth for a given α.

Figure 5.3 represents the number of packet pair probes XRTP requires to overshoot its available bandwidth (See variable x in Equation (5.10)) for various values of *alpha* as the number of flows in the network increases. x times the packet-pair probe interval (i.e., k) represents the number of packets in flight after which XRTP overshoots its fair-share. The graphs show that x is very sensitive to α. As the number of flows in the network increases, the number of packets that can be in flight before XRTP overshoots its available bandwidth falls rapidly. This implies that XRTP needs to set the value of α close to 1 to be friendly to other flows in the network. However, setting α close to 1 causes XRTP to behave conservatively in presence of fewer flows. This necessitates the need to vary α based on the number of flows.

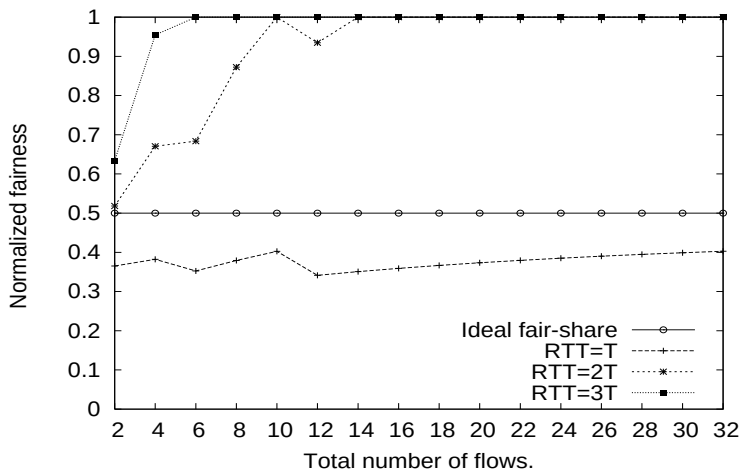

Figure 5.4 Fairness index of XRTP with $\alpha = 0.95$ for different RTT

Figure 5.4 represents the effect of the overestimation on the fairness-index, which is computed as *throughput* (From Equation (5.14)) times $\frac{\gamma}{2}$, since it is assumed that a equal number of XRTP and competing flows are using the network. An ideal fairness-index value is 0.5. Values below 0.5 indicate that the flow is unfair to itself while values greater than 0.5 indicate that it is unfair to other flows. As shown in the Figure 5.4, XRTP is fair to competing flows when the *epoch* time, T, is equal to the RTT. In scenarios where $T < RTT$, XRTP always overshoots its fair-share. This causes other flows to cut their transmission rate, effectively allowing XRTP to use more than its fair-share. The graphs for $RTT = 2T$ and $RTT = 3T$ projects this effect in terms of fairness-index as the number of flows increase in the network.

Even though this chapter provides an approximate model of XRTP's congestion control, simulations from Chapter 4 and analysis here have shown that XRTP is unfair to other flows in the network. Figure 5.3 and Figure 5.4 showed that XRTP tends to be unfair as the number of XRTP flows increase in the network. The main cause of this unfairness can be attributed to aggressive bandwidth estimation. Additionally, XRTP does not account for multiple packet losses in its congestion avoidance mechanism.

5.4 Improving fairness-index of XRTP

The analytical model of XRTP showed that XRTP causes unfairness due to the following reasons: unresponsive behavior towards multiple packet losses per RTT, high sensitivity to the smoothing parameter α and aggressive estimation of bottleneck channel capacity instead of available bandwidth. Based on these insights, this section evaluates two techniques that modify XRTP's congestion control algorithm. Both of these mechanisms attempt to alleviate the unfairness exhibited by XRTP by estimating the available bandwidth instead of the bottleneck bandwidth.

5.4.1 Techniques to improve fairness of XRTP

```
declare a, b as real (a ≤ α ≤ b)
function incorporate_fairness_using_α
    Track the number of packet-losses per RTT
    if packet-losses > 1 then
        α = (α + b)/2
    else if packet-losses < 1 then
        α = (α + a)/2
    else // 1 packet loss per RTT
        // do nothing to α
    end if
end function
```

Figure 5.5 Algorithm to incorporate fairness by varying smoothing parameter, α.

In the first technique, the indication of multiple packet losses is combined with the smoothing parameter, α. Multiple packet losses in a given RTT indicates that XRTP is flowing at a rate that is larger than its fair share. As noticed in Figure 5.3, XRTP flows take a longer time to overshoot the available bandwidth when α is closer to 1. Hence, increasing the value of α closer to 1 damps the rate at which XRTP ramps its transmission rate. Thus, varying the value of smoothing parameter, α, based on packet losses enables XRTP to adjust to other competing flows in the network. Figure 5.5 describes the

extension to XRTP's congestion control by varying α to control the unfairness exhibited by XRTP's original algorithm.

```
declare B, C as real
B = C // set B equal to channel capacity
function incorporate_fairness_using_bw_estimation
    Track the number of packet-losses per RTT
    if packet-losses > 1 then
       B = B/2
    else if packet-losses < 1 then
       B = min(C, B * 2)
    else // 1 packet loss per RTT
       // do nothing
    end if
end function
```

Figure 5.6 Algorithm to incorporate fairness by varying estimated available bandwidth.

In the second technique, the indication of multiple packet losses is combined with an estimation of available bandwidth. In the original XRTP specification (See Chapter 3 and the section of analysis in this chapter), XRTP uses a measurement of the channel capacity, C, using packet-pair to ramp its rate. This could result in an over-estimation of the available bandwidth. To compensate for over-estimation, a new parameter, called the *estimated available bandwidth*, B, is introduced into the congestion control algorithm of XRTP. The estimation of available bandwidth follows the binomial congestion control algorithm, (i.e., each time XRTP notices multiple packet losses in a single RTT, it cuts its estimate of available bandwidth by half). Likewise, when XRTP notices no packet loss, it doubles its estimate of available bandwidth. The estimate of available bandwidth is limited by the channel capacity of the bottleneck link. The variable *new_est_rate* in Equation (3.4) is modified to incorporate *estimated available bandwidth* (B) instead of the channel capacity (C). Using this algorithm, B is maintained such that ($A/2 \leq B \leq 2A$), where A is the actual available bandwidth to a flow. Figure 5.6 describes the working of the second technique.

5.4.2 Evaluation of the techniques

To evaluate the improvements obtained by incorporating extensions in XRTP's congestion control, the network is setup as described in Figure 4.1 with equivalent parameters. For simplicity, XRTP with no modification is labeled as XRTP-N, XRTP with varying smoothness parameter, α, is called as XRTP-A, XRTP with varying *estimated available bandwidth* is denoted by XRTP-C and XRTP with both techniques incorporated into its congestion control is denoted by XRTP-B. In the first set of evaluations, each flavor of XRTP is run against an equivalent number of either TCP or TFRC flows. The number of flows in the network are increased in multiples of two and the average aggregate throughput is measured for each set of flows.

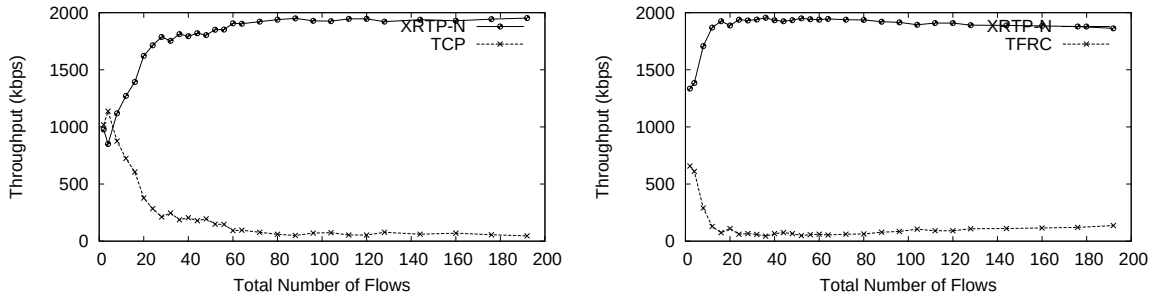

Figure 5.7 Evaluation of fairness XRTP-N (without modification) against competing flows.

Figure 5.7 depicts XRTP-N running against TCP and TFRC flows. As described by the analytical model, XRTP-N is aggressive when the *epoch* time, T, is greater than RTT. Hence, the graphs show that XRTP-N collectively grabs almost all of the bandwidth available on link *l2*.

Figure 5.8 describes the behavior of XRTP-A against TCP/TFRC flows. It is noticeable that XRTP-A and TCP are fair to each other. XRTP-A and TCP roughly use half of the bottleneck bandwidth. The evaluations show that varying α tracks the available bandwidth in the network making XRTP friendly to other competing flows. However, in Figure 5.8, the aggressiveness of TFRC against XRTP-A is due to the fact that XRTP-

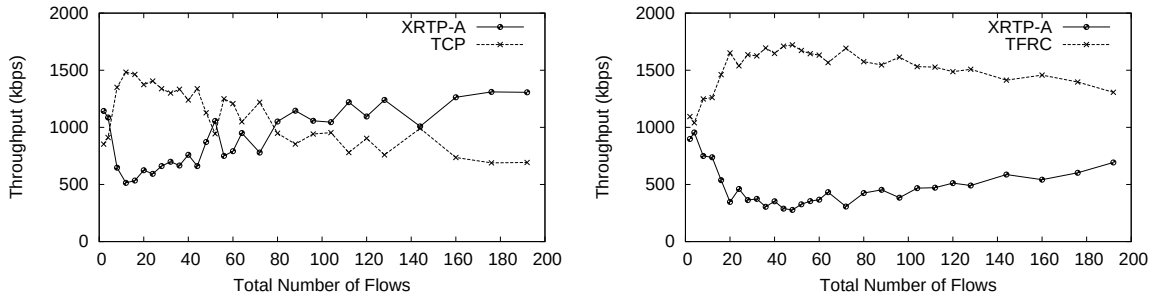

Figure 5.8 Evaluation of fairness XRTP-A (by varying α) against competing flows.

A's parameters are set to make it TCP friendly. Since TFRC is aggressive to TCP [32], it is aggressive towards XRTP-A.

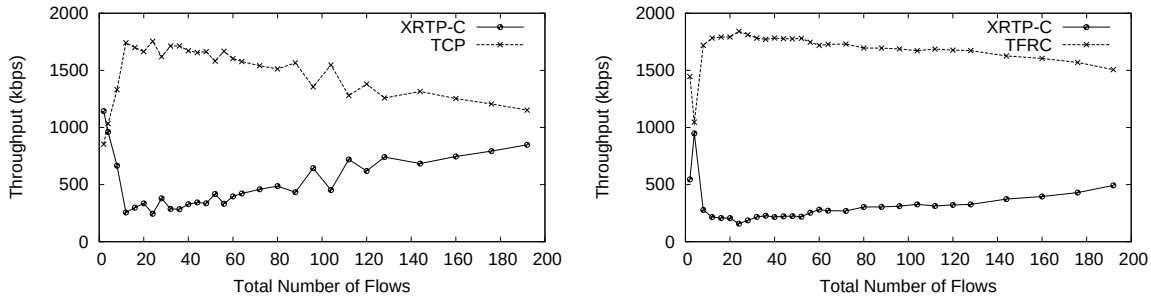

Figure 5.9 Evaluation of fairness XRTP-C (by varying estimated bandwidth) against competing flows.

Figure 5.9 shows the effect of varying the estimated available bandwidth on the competing flows. Notice that XRTP-C is very conservative estimating the actual available bandwidth, causing it to have a low aggregate throughput. In situations where XRTP-C has a large number of competing flows, XRTP-C experiences multiple packet losses for more than one RTT causing it to cut its estimate to a value lower than the actual bandwidth. Since either TCP or TFRC are also competing for available bandwidth, they grab the bandwidth made available by the conservative approach of XRTP-C. Hence, XRTP-C takes all the slack resulting in a drop of aggregate bandwidth.

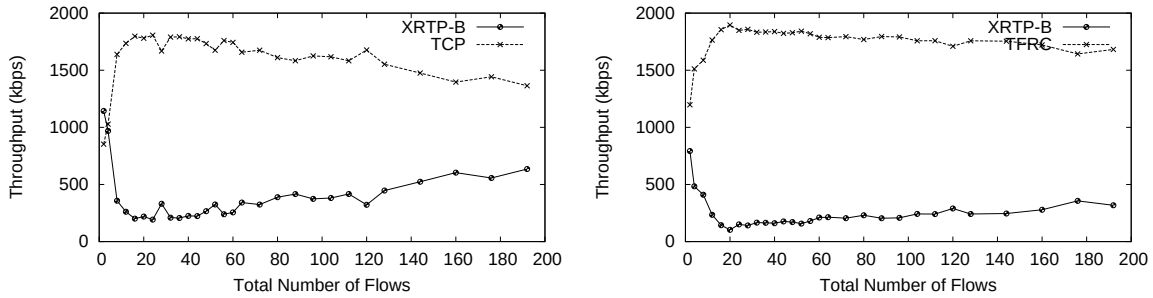

Figure 5.10 Evaluation of fairness XRTP-B (by varying both: estimated bandwidth and *alpha*) against competing flows.

Figure 5.10 shows XRTP-B's behavior in the presence of multiple flows. Due to the combination of both techniques in XRTP-B, the conservative estimated available bandwidth overshadows the effect of varying α. Thus, Figure 5.10 has characteristics similar to that of Figure 5.9.

When XRTP and TCP or TFRC flows use the same network paths, there is a constant competition for capturing available bandwidth. To clearly understand how XRTP tracks available bandwidth, it is necessary to evaluate how XRTP competes against background traffic that is not affected by XRTP's congestion control behavior. To evaluate such scenarios, CBR traffic was setup that constantly uses half of the channel capacity of *l2* (See Figure 4.1). The number of XRTP flows in the network is increased in multiples of two and the aggregate bandwidth used by XRTP is measured.

Figure 5.11 describes the behavior of all flavors of XRTP flows against the CBR traffic. In an ideal scenario, the graphs of XRTP and CBR throughput should overlap. Overlapping graphs show that each individual flow has been able to accurately estimate its available bandwidth. As seen in the figure, increasing the number of XRTP-N flows in the network increases the gap between XRTP-N and CBR throughput. This implies that XRTP-N is not able to estimate the available bandwidth. The effect of over estimation causes the CBR flow to loose packets at greater rates as the number of XRTP-N flows increases. Even though XRTP-N collectively shows good throughput, it is necessary to

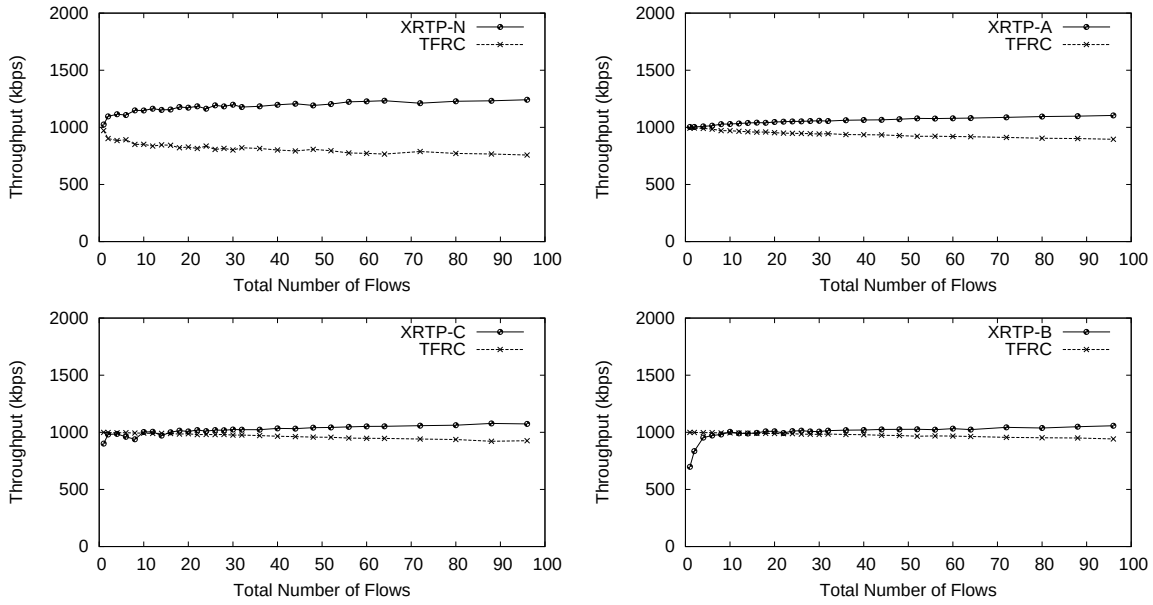

Figure 5.11 Evaluation of all flavors of XRTP against CBR flow.

remember that this throughput is achieved at the expense of high losses experience by each individual flow. With the extensions to XRTP, the gap between the throughputs of XRTP-A or XRTP-C or XRTP-B and CBR is minimal compared to the throughput of XRTP-N and CBR. These evaluations conclude that the extensions provided to XRTP can efficiently track the available bandwidth.

This chapter analyzed the original XRTP congestion control algorithm, which was determined to be unfriendly to other transport protocols due to its aggressive bandwidth estimation. To alleviate this problem, two techniques were presented using the knowledge of multiple packet losses with congestion control parameters of XRTP like the *smoothing parameter*, α, or the *estimated available bandwidth*. Evaluations showed that varying the *smoothing parameter* provides XRTP with better control over the varying network conditions while varying the *estimated available bandwidth* causes XRTP to take a very conservative approach towards competing flows. In conclusion, these extensions demonstrate that these techniques are successful in making XRTP network friendly.

CHAPTER 6

CONCLUSION

In this thesis, we propose a rate-based transport protocol for lossy wireless networks that uses bandwidth estimation and jitter measurement to share the medium fairly and accurate loss discrimination to distinguish congestion and transmission losses. Through evaluation, we show how non-parametric density-based filters can be used efficiently in protocols when compared to noisy EWMA filters. Our simulations show that the accuracy of the heuristics are directly tied to the regularity of the flow. Using the same loss discrimination scheme, rate-based transport protocols can react wisely to loss, by reducing the sending rate in the presence of congestion and maintaining the current rate in case of transmission loss. We analyzed the behavior of XRTP's flow by developing an analytical model of its congestion avoidance and control. Using the model, we were able to propose changes to the congestion control mechanism to alleviate XRTP's unfair behavior towards other flows.

While this thesis has demonstrated the effectiveness of XRTP, there are still a number of issues that need to be researched. A real implementation of XRTP in the Linux operating system is necessary to evaluate the impact of the protocol on the Internet. Since rate-based protocols are sensitive to timers, the impact of Linux's coarse grain timers on XRTP must be accurately measured. Finally, given the implementations, XRTP should be evaluated in realistic scenarios.

REFERENCES

[1] "IEEE standard for wireless LAN-medium access control and physical layer specification, P802.11,",", 1999.

[2] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proceedings of the first annual international conference on Mobile computing and networking, MOBICOM*, ACM Press, 1995, pp. 2–11.

[3] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proceedings of the 15th International Conference in Distributed Computing Systems -ICDCS*, 1995, pp. 136–143.

[4] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP westwood: Bandwidth estimation for enhanced transport over wireless links," in *Mobile Computing and Networking - SIGMOBILE*, 2001, pp. 287–297.

[5] R. Krishnan, M. Allman, C. Partridge, J. P. Sterbenz, and W. Ivancic, "Explicit Transport Error Notification (ETEN) for error-prone wireless and satellite networks - summary," in *Earth Science Technology Conference*, june 2002.

[6] H. Balakrishnan and R. H. Katz, "Explicit Loss Notification and wireless web performance," *GLOBECOM*, 1998.

[7] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 8–23, 1994.

[8] V. Jacobson, "Congestion avoidance and control," in *Symposium proceedings on Communications architectures and protocols*, ACM Press, 1988, pp. 314–329.

[9] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[10] R. Rejaie, M. Handley, and D. Estrin, "Rap : An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, vol. 3, 1999, pp. 1337–1345.

[11] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based TCP-friendly rate control protocol," in *Network and Operating System Support for Digital Audio and Video - (NOSSDAV)*, 1999.

[12] A. C. Feng, A. C. Kapadia, W. C. Feng, and G. G. Belford, "Packet spacing: An enabling mechanism for delivering multimedia content in computational grids," *The Journal of Supercomputing*, vol. 23, pp. 51–66, 2002.

[13] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, vol. 3, 2000, pp. 1157–1165.

[14] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," *Wireless Networks*, vol. 8, no. 2-3, pp. 301–316, 2002.

[15] S. Keshav, "A control-theoretic approach to flow control," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 188–201, 1995.

[16] N. Aboobaker, D. Chanady, M. Gerla, and M. Y. Sanadidi, "Streaming media congestion control using bandwidth estimation," *IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, 2002.

[17] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking (TON)*, vol. 1, no. 4, pp. 397–413, 1993.

[18] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?," in *Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, vol. 2, 2001, pp. 905–914.

[19] V. Paxson, "Automated packet trace analysis of TCP implementations," in *Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, ACM Press, 1997, pp. 167–179.

[20] R. L. Carter and M. Crovella, "Measuring bottleneck link speed in packet-switched networks.," *Performance Evaluation*, vol. 27/28, pp. 297–318, October 1996.

[21] K. I.-S. Lai, "Measuring the bandwidth of packet switched network," Ph.D. dissertation, Standford University, September 2002.

[22] K. Pentikousis, "TCP in wired-cum-wireless environments," *IEEE Communications Surveys and Tutorials.*, vol. 24, no. 5, 2000.

[23] S. Biaz and N. Vaidya, "Discriminating congestion losses from wireless losses using inter-arrival times at the receiver," in *IEEE Application-Specific Systems and Software Engineering and Technology ASSET*, 1999, pp. 10–17.

[24] D. Barman and I. Matta, "Effectiveness of loss labeling in improving TCP performance in wired/wireless networks," in *Tenth International Conference on Network Protocols, ICNP*, 2002, pp. 2–11.

[25] N. Samaraweera, "Non-congestion packet loss detection for TCP error recovery using wireless links," in *IEEE Proceedings of Communications*, vol. 146, August 1999, pp. 222–230.

[26] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, ACM Press, 1999, pp. 263–274.

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

[28] L. Magalhaes and R. Kravets, "Transport level mechanisms for bandwidth aggregation on mobile hosts," in *Ninth International Conference on Network Protocols, ICNP*, 2001, pp. 165–171.

[29] N. Samaraweera and G. Fairhurst, "Explicit loss indication and accurate RTO estimation for TCP error recovery using satellite links," in *In IEE Proceedings - Communications*, vol. 144, February 1997, pp. 47–53.

[30] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," *IEEE Computer*, vol. 33, pp. 59–67, May 2000.

[31] B. Srinivasan, S. P. Robert, H. F. Ansari, and D. Niehaus, "A firm real-time system implementation using commercial off-the-shelf hardware and free software," *4th IEEE Symposium on Real-time Technology and Applications,*, pp. 112–119, June 1998.

[32] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic behavior of slowly-responsive congestion control algorithms," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications-SIGCOMM*, ACM Press, 2001, pp. 263–274.

APPENDIX A

COMPUTING MEAN AND VARIANCE OF RTT IN LINEAR TIME

A good estimation of RTT is necessary to classify the transmission-based losses and congestion-based losses. The estimate of the RTT is accurate due to use of timestamp option in the XRTP's packet header. However, the average and deviation of previous n RTT observations are necessary to classify the losses. With the assumption that RTT values are uniformly distributed, the mean μ and variance σ^2 of n observed RTT can be evaluated as

$$\mu = \frac{\sum_{i=1}^{n} RTT(i)}{n} \tag{A.1}$$

$$\sigma^2 = \frac{\sum_{i=1}^{n} RTT(i)^2}{n} - \mu^2 \tag{A.2}$$

The computationally expensive expression can be transformed into less expensive incremental computations using finite differencing. In particular, μ and σ^2 can be incrementally maintained each time a new RTT is observed. Consider a scenario where XRTP has computed the mean of RTT until time t_{i-1}. The RTT mean is defined as given below.

$$\mu_{i-1} = \frac{\sum_{i=1}^{n} RTT(i)}{n} \tag{A.3}$$

Consider that RTT_{new} arrives at time t_i. The new RTT mean is computed as follows.

$$\mu_i = \frac{\sum_{i=2}^{n} RTT(i) + RTT_{new}}{n} \tag{A.4}$$

```
Declare RTT:{0..n-1} as Integer
Declare curRTTptr as Integer = 0 // initialize to 0

function compute_RTT_μ_and_σ² (RTT_new)
        if not (observed at least n samples) then
                // computing mean and variance
                RTT[curRTTptr] = RTT_new
                σ² = (σ² + μ²) * curRTTptr
                μ = μ * curRTTptr + RTT_new
                curRTTptr = (curRTTptr + 1) % n
                μ = μ/curRTTptr
                σ² = (σ²+(RTT_new)²) / curRTTptr − μ²
        else
                // computing mean and variance
                σ² = σ² + μ + ((RTT²_new − RTT[curRTTptr]²) / n)
                μ = μ + ((RTT_new − RTT[curRTTptr]) / n)
                RTT[curRTTptr] = RTT_new
                curRTTptr = (curRTTptr + 1) % n
        end if
 end function
```

Figure A.1 Algorithm to compute RTT mean and variance using finite differencing

Subtracting Equation (A.3) from Equation (A.4), we obtain Equation (A.6)

$$\mu_i - \mu_{i-1} = \frac{RTT_{new} - RTT(1)}{n} \tag{A.5}$$

$$\mu_i = \mu_{i-1} + \frac{RTT_{new} - RTT(1)}{n} \tag{A.6}$$

Looking at Equation (A.6), it is easy to deduce that each time a new RTT arrives, the oldest observed RTT can be replace by RTT_{new}, maintaining the last n observed RTT's in a data-structure like circular-queue. Further optimizing the computation, the value of n can be set to the powers of 2, so that division in Equation (A.6) can be replaced by

integer right shift operation.

Similarly, the variance can be incrementally computed using the following equation.

$$\sigma_i^2 = \sigma_{i-1}^2 + \frac{RTT_{new}^2 - RTT(1)^2}{n} + \left(\mu_i^2 - \mu_{i-1}^2\right) \tag{A.7}$$

Figure A.1 describes the algorithm used in XRTP to compute mean and variance.

APPENDIX B

MODELING XRTP'S CONGESTION CONTROL IN PRESENCE OF JITTERS

Positive jitter damp XRTP's transmission rate to prevent overuse of its fair-share bandwidth. Since jitter must follow the conservation of time i.e, sum of all the jitters for a given flow must equal zero. Hence, on an average, for every k packets sent, XRTP would encounter $\frac{k}{2}$ positive jitters and $\frac{k}{2}$ negative jitters. Following steps similar to Chapter 5, XRTP's congestion avoidance model with jitter can be derived as shown below.

$$r_0 = \frac{B}{2S} - \frac{k}{2J} \tag{B.1}$$

XRTP'S rate is ramped up a fraction $1-\alpha$ of new estimate i.e. $\frac{C}{S}$.

$$r_1 = \alpha r_0 + (1-\alpha)\frac{C}{S} - \frac{k}{2J} \tag{B.2}$$

Substituting r_0.

$$r_1 = \alpha\left(\frac{B}{2S} - \frac{k}{2J}\right) + (1-\alpha)\frac{C}{S} - \frac{k}{2J} \tag{B.3}$$

Simplifying r_1,

$$r_1 = \frac{C}{S}\left(1 - \frac{(2\gamma-1)}{2\gamma}\alpha\right) - \frac{k}{2J}\left[1+\alpha\right] \tag{B.4}$$

Continuing the evaluate the recurrance relation, after receiving xk packets, r_x, the rate of the protocol would be

$$r_x = \frac{C}{S}\left(1 - \frac{(2\gamma-1)}{2\gamma}\alpha^x\right) - \frac{k}{2J}\left[\frac{1-\alpha^x}{1-\alpha}\right] \tag{B.5}$$

We can now solve the equation for x to get the number of packets lost before reaching the bottleneck rate as shown below.

$$x = \frac{\ln\left(\frac{CJ(1-\alpha)2(\gamma-1)-kS\gamma}{CJ(1-\alpha)(2\gamma-1)-kS\gamma}\right)}{\ln\alpha} \tag{B.6}$$