

Designing a Rate-based Transport Protocol for Wired-Wireless Networks

Shravan Gaonkar¹, Romit Roy Choudhury², Luiz Magalhaes³, and Robin Kravets⁴

Computer Science,
University of Illinois,
Urbana Champaign, IL
{gaonkar¹, rhk⁴}@uiuc.edu

Electrical and Computer Engineering,
Duke University,
Durham, NC
romit@ee.duke.edu²

Engenharia de Telecomunicaes,
Universidade Federal Fluminense,
Brazil
schara@midia.com.uff.br³

Abstract—A large majority of the Internet traffic relies on TCP as its transport protocol. In future, as the edge of the Internet continues to extend over the wireless medium, TCP (or its close variants) may not prove to be appropriate. The key reason is in TCP’s inability to discriminate congestion losses from transmission losses. Since transmission losses are frequent in wireless networks, the penalty from loss misclassification can become high, leading to performance degradation.

This paper presents an eXtended Rate-based Transport Protocol (XRTP), designed to support communication over lossy wireless media. We depart from the ack-based rate control paradigm. Instead, we try to estimate the network conditions by injecting probe packets at the sender, and then observing the spacing between packets that arrive at the receiver. We show that these observations can be useful indicators of available bandwidth, network congestion, and even the cause of packet loss. The inferences from the observations are utilized to regulate the transmission rate at the sender, leading to desirable properties of congestion control and loss discrimination. Simulation results show the efficacy of our proposed rate-based protocol in lossy wireless environments.

I. INTRODUCTION

Advances in communication technology have increased the reliability of wired links to the point where transmission losses are rare. Building on these advancements, TCP has been tuned to assume that all losses are caused solely by congestion. With the internet rapidly extending over the wireless medium, this assumption does not hold for combined wired-wireless networks (such as wireless LANs). In these hybrid networks, packets are lost due to transmission errors over wireless links, as well as due to congestion within the wired infrastructure. This necessitates a transport protocol that can respond appropriately to the type of loss, while retaining important TCP-like properties such as congestion control and fairness. Choosing an appropriate response is non-trivial because the transport layer has no explicit knowledge about the cause of packet loss. Furthermore, the appropriate responses are in conflict with each other, hence, an incorrect response can incur substantial performance penalties, as discussed next.

Consider a case in which the transport layer is required

to react to a packet loss. If the loss is due to congestion, the transport protocol should react in a fashion that alleviates congestion in the network. However, if the loss is due to transmission (hence not indicative of congestion), the transport protocol can perhaps ignore the loss, and continue with its operations. Now, if a congestion loss is incorrectly classified as a transmission loss, the sender will not decrease its offered load and more congestion will build up in the network. Conversely, if transmission loss is treated as a congestion loss, the sender will unnecessarily reduce its offered load, reducing the throughput of the stream. When several losses occur over time, the impact of such misclassification can be significant.

Several proposals have aimed to perform loss discrimination. However, many of them either require changes inside the network infrastructure, or deviate from the end-to-end semantics. This paper explores the possibility of a rate-based protocol (XRTP) that conforms to the desired characteristics of a transport protocol, while addressing the specific issues in wired-wireless environments. Our main idea in XRTP can be briefly described as follows.

An XRTP sender sends streams of data packets using a suitably chosen rate¹. Periodically interspersed with these data packets, the sender also sends back-to-back packet-pairs. Using well-known statistical techniques [1], the separation between received packet-pairs are used to estimate the available bandwidth in the network. This estimate is in turn used to regulate the transmission rate in small increments, resulting in a smooth stream of packets. Now, XRTP takes advantage of this smoothness to facilitate loss discrimination. Specifically, we show that *congestion and transmission losses typically produce distinguishable anomalies on the timings at which packets are received*. By observing these anomalies, loss discrimination can be achieved with reasonable accuracy. While some false alarms are possible, we utilize additional mechanisms to reduce its probability. Simulation results show

¹This is in contrast to the ACK-based transmission in TCP, which can often be bursty in nature.

that XRTP is capable of improving network utilization, while maintaining congestion control and fairness in combined wired-wireless environments.

The remainder of this paper is organized as follows. Section II discusses related work in the area of transport protocols for combined wired and wireless networks. In Section III we describe the architecture of XRTP, and include detailed discussions on its three major components – (i) bandwidth estimation, (ii) rate and congestion control, and (iii) loss discrimination. We evaluate XRTP in Section IV, and discuss some issues in Section V. We conclude the paper with a brief summary in Section VI.

II. RELATED WORK

There have been several proposals to optimize TCP for wireless networks [2]–[9]. Some of these solutions [2], [3] try to maintain the semantics of TCP congestion control by hiding transmission losses from the end hosts by modifying the underlying infrastructure. Techniques like Explicit Loss Notification (ELN) [10] and Explicit Congestion Notification (ECN) [11] 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 wide-scale updation of routers and other support infrastructure, making deployment difficult. An end-to-end approach is necessary that does not rely on additional support from the underlying wired-wireless network.

Recently, rate-based protocols such as RAP [12] and TFRC [13] have become popular due to their smoother transmission rates in comparison to ack-based protocols². While effective for wired networks, the existing rate-based protocols are not directly applicable to wireless networks because of dynamically changing data rates. Another proposal, named Wireless-TCP [14], strives to cope with the changing bandwidth by observing the ratio of inter-packet separation at the receiver to inter-packet separation at the sender. By exploiting long-term jitter, WTCP carefully tries to achieve the available bandwidth. However, long-term jitter is not the most accurate metric since channel conditions in wireless networks change too rapidly. In one component of XRTP, we partially adopt the approach used by WTCP by using observations about short-term jitter to monitor congestion in the network. When congestion is detected, an XRTP transmitter is instructed to reduce its transmission rate. The new transmission rate is guided by the available bandwidth in the network.

XRTP uses back-to-back packet-pairs (also called probes) to estimate available bandwidth. Techniques like packet-pair have been widely used by off-line tools that measure bandwidth (e.g. *tcpanaly* [15], *bprobe* [16]). However, measurements from packet-pair often include erroneous

values and the data filtering techniques used by these tools lack statistical robustness. Lai et. al. [17] 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. XRTP uses an optimized KDE algorithm that provides efficient filtering without loss of accuracy, using the technique of finite differencing. Sundaresan et.al. [18] approach the problem of a reliable transport protocol for ad-hoc networks using packet probes and support from the intermediate nodes to estimate available bandwidth. TCP friendliness was not a constraint for their ATP (ad-hoc transport protocol) protocol, since ATP was targeted for ad-hoc nodes which implement a dedicated protocol stack. XRTP, on the otherhand, does not expect any network support and it is expected to exhibit fairness to both XRTP and TCP flows.

A number of protocols have been proposed that enhance TCP to achieve loss discrimination. Biaz et. al. [19] distinguish between congestion and wireless loss using packet inter-arrival times at the receiver. Barman et. al. [20] 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 [21] correlates the round trip time with the throughput-load graph of the flow. Although interesting, the problem with these approaches is that their heuristics are proposed for TCP-like ack-based protocols that are bursty in nature. However, these heuristics are better suited for rate-based protocols [22], providing better correlation of actual network conditions with parameters like round-trip time (RTT), RTT-variance, or the throughput-load graph of the flow. These possibilities with rate-based approaches partly motivate the design of XRTP.

III. EXTENDED RATE-BASED TRANSPORT PROTOCOL

Traditional transport protocols, designed for wired networks, do not incorporate appropriate recovery mechanisms for transmission losses. While designing solutions for wired-wireless environments, an intuitive approach may be to extend the traditional protocols with a loss discrimination module. We argue that this is not sufficient because the protocol’s flow and congestion control mechanisms are synergistically related to loss discrimination. Modifications to any one component will need synergistic modifications to the others. In view of this, the proposed XRTP framework is centered around three main components, namely (i) flow control, (ii) bandwidth estimation, and (iii) loss discrimination. We begin this section by describing the basic structure of XRTP to offer a high level intuition. Subsequently, in subsections III B, C, and D, we present the details of each of the components.

A. Protocol Structure

XRTP is a protocol that regulates the transmission rate of the sender based on available bandwidth and network congestion. To estimate available bandwidth, XRTP sources

²Ack-based protocols are based on self-clocked behavior, and have dominated the Internet for their ease of implementation.

periodically send back-to-back probe packets interspersed with its regular rate of data packets. The XRTP receiver observes the spacing between the probes, as well as the spacing between data packets, to derive the available bandwidth in the network. The spacing between packets, also called jitter, can be an indicator of network health. In the event of increasing jitter – an indication of growing network congestion – XRTP reduces the transmission rate proactively. Since the reduction is proactive, it can be small, which in turn makes the transmission rates smooth. This also helps loss discrimination – congestion and transmission losses introduce distinct irregularities in the smoothness, making it possible to tell them apart. When the results of this discrimination is fed into the congestion control mechanism, XRTP’s transmission rate copes better with the dynamic network conditions. The penalty from incorrect loss discrimination reduces, which in turn reduces the burstiness in traffic. The benefits can be substantial in infrastructure-based wireless networks, such as WLANs.

The applicability of back-to-back packet probes (for estimating bottleneck bandwidth) is not our contribution. It had been proposed in literature (e.g. *tcpanaly* [15], *bprobe* [16]). Since measurements from such probes often include erroneous values (primarily due to cross-traffic), statistical techniques have also been proposed for data filtering. Lai et. al. [17] 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 computational complexity of incorporating Lai’s technique in an actual network can be high. In view of this, we optimize the KDE algorithm to enable online execution, with marginal loss of accuracy in the estimation process. The details of the optimized KDE algorithm are presented later in this section. We first describe the three main components of XRTP.

B. Bandwidth Estimation

XRTP aims to estimate the *available bandwidth* in the network, and regulate its transmission rate based on this availability. However, the available bandwidth is a dynamic value, and changes constantly with varying traffic patterns in the network. As shown by Keshav [23], measuring available bandwidth is a difficult task in FCFS routers, and packet-pair techniques do not work. This is because packet pairs (i.e., back-to-back probes) measure the *bottleneck bandwidth*, defined as the *maximum throughput that can be obtained between two hosts in the absence of any cross traffic*. Now, using bottleneck bandwidth measurements (from packet-pairs), and combining it with short-term history of jitter (from regular data packets), XRTP determines the trend in network congestion, and in turn prescribes how the transmission rate should be regulated. *One may view this as an indirect way of estimating available bandwidth*. However, unless handled carefully, specific cases can inject error into the estimation. We discuss these cases in detail and apply statistical methods to address the potential causes of inaccuracy. We begin with

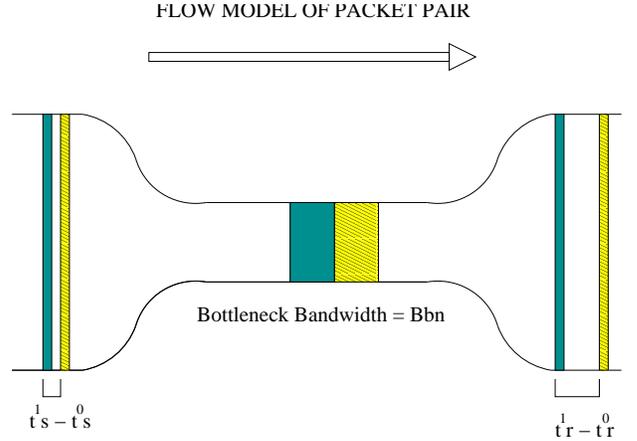


Fig. 1. Fluid model of packet pair with two packets. The separation between packets is ideally dictated by the bottleneck link.

a closer look into the packet-pair technique.

1) **Packet Pair:** The fluid flow model in Figure 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 different transmission times of the bottleneck link and the subsequent (high-bandwidth) link. Let B_{bn} be the bottleneck bandwidth, S be the size of the packet, t_0^s and t_1^s be the times when the first and second packets are sent back-to-back from the sender, and t_0^r and t_1^r be the times when the first and second packets are received at the final receiver. To maintain conservation of flow, the equilibrium equation for the described model is given by

$$t_1^r - t_0^r = \max\left(\frac{S}{B_{bn}}, t_1^s - t_0^s\right). \quad (1)$$

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 a spacing, $t_1^r - t_0^r$, same as that introduced by the bottleneck link’s bandwidth, $\frac{S}{B_{bn}}$. Rearranging Equation (1), the bottleneck bandwidth can now be computed as

$$B_{bn} = \frac{S}{t_1^r - t_0^r}. \quad (2)$$

However, Equation (2) only holds provided that the two packets are queued only at the bottleneck link and at no later link downstream. This assumption is difficult to satisfy 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 underestimation of the bottleneck bandwidth (time-expansion). The packet-pair may also get queued at a later router decreasing the gap between packets, and causing an overestimation of bottleneck bandwidth (time-compression). To 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 kernel density estimation (KDE) algorithm [24] specifically optimized for the rate-based protocol.

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 [24] 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

$$f(x) = \frac{1}{nh(x)} \sum_{i=1}^n K\left(\frac{x - X_i}{h(x)}\right). \quad (3)$$

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 $O(n^2)$, where n is the number of consecutive jitter samples stored in the XRTP cache to make an estimation. Lai et. al. [17] used this estimation technique offline for measuring bottleneck bandwidth in their tool *pathchar*. However, for XRTP, the KDE algorithm needs to execute online, and should ideally have a low running time. This motivates optimizing the KDE algorithm. Looking at the estimator function as described in Equation (3), for a fixed width h , a standard finite differencing technique can be used to reduce the algorithm to linear time (see Figure 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) and accurately estimates bottleneck bandwidth. The pseudo code for the algorithm is presented in Figure 2.

C. Rate and Congestion Control

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

```

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] -= K((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] + K((est-new)/h) / (n*h)
        obsrv[i] = obsrv[i+1]
    next i
    // compute the density of new observation
    obsrv[n] = new
    pdf[n] = 1/nh * sum_{i=1}^n K((est-obsrv[i])/h)
    // compute new width h and estimate i.e., mean
    // return new estimate
end function

```

Fig. 2. Linear kernel density estimation for BW filtering.

the current rate and the new estimated bandwidth to update the rate of the protocol. XRTP updates the new rate using the standard EWMA equation with smoothing parameter α (see Equation 4). The receiver sends the updated rate to the sender in the acknowledgments.

$$new_rate = old_rate * \alpha + new_est_rate * (1 - \alpha) \quad (4)$$

As mentioned earlier, packet-pairs measure the bottleneck bandwidth, which typically is higher than the available bandwidth of the network. As a result, the rates chosen by XRTP in (4) can be greater than its fair share, and may lead to congestion. To prevent this, XRTP incorporates a congestion avoidance mechanism. Since XRTP sources send data 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 ideally be equal to the inter-sending time of the corresponding packets at the sender. In reality, the inter-sending time may not equal the inter-reception time due to queuing delays and the presence of other flows. Observe that the difference between inter-reception time and inter-sending time, also called *jitter*, is an indication of the congestion trend in the network. XRTP keeps track of this jitter, and suitably regulates the transmission rate to remain close to the flow's fair share of bandwidth. The rate control mechanism is discussed next.

Quantitatively, let the reception time of the n^{th} packet, t_n^r , be the sum of the time it was sent, t_n^s , and the transmission time, tt_n (i.e., $t_n^r = t_n^s + tt_n$). Transmission time can be divided into two components, propagation time, pt_n , and queue time, q_n (at the routers), i.e., ($tt_n = pt_n + q_n$). If the routes are not changing, propagation time, pt_n , is almost constant since it

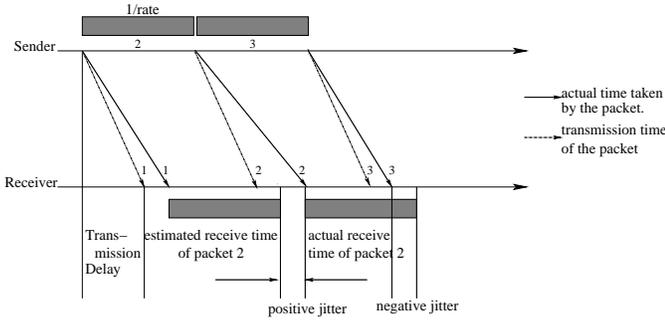


Fig. 3. Jitter caused by varying queuing delay. Positive jitter arises when the latter packet gets more delayed than the earlier. Negative jitter arises when it is the vice versa.

only depends on the transmission speed of the media. Queue time, q_n , mainly includes (i) the time the packets wait to be processed and (ii) the negligible processing time.

As the network load increases, a packet in transmission will experience longer wait times at the router, resulting in longer end-to-end transmission times. On the other hand, as network load decreases, end-to-end transmission time will decrease to the limits of the propagation delay on the transmission medium. The inter-reception time between two consecutive packets $n - 1$ and n , denoted by irt_n is given by Equation (5). Substituting each of the right hand side terms using ($t_n^r = t_n^s + tt_n$), we get Equation (6) for inter-reception time.

$$irt_n = t_n^r - t_{n-1}^r \quad (5)$$

$$irt_n = (tt_n - tt_{n-1}) + t_n^s - t_{n-1}^s \quad (6)$$

Since the time the packets were sent is known, the term ($t_n^s - t_{n-1}^s$) can be subtracted from Equation (6) giving us a measure of network load as shown in Equation (7). Thus, jitter is governed by the difference in packet travel time. Now assuming that propagation time is constant for all packets, the only variant is the queuing time as shown in Equation (9). A varying queuing time (measured as jitter at the destination) can be used as an indicator of network congestion.

$$jitter_n = (tt_n - tt_{n-1}) \quad (7)$$

$$jitter_n = (pt_n - pt_{n-1}) + (q_n - q_{n-1}) \quad (8)$$

$$jitter_n = q_n - q_{n-1} \quad (9)$$

XRTP uses timestamps to determine the sending time. Therefore, the jitter for the n^{th} packet can be computed as $jitter = (t_n^r - t_{n-1}^r) - (t_n^s - t_{n-1}^s)$.

In general, jitter can be negative, positive, or zero. Figure 3 pictorially illustrates the computation of jitter. Negative jitter can arise when the (n)th packet gets less delayed than the ($n-1$)th packet. Positive jitter happens for the vice versa. Ideal or zero jitter indicates that the network congestion remains unchanged over the time scale of consecutive packets. Positive jitter implies increasing queue lengths and occurs when the cumulative rate of all flows is greater than the capacity of

```

function loss_discrimination(packet p)
  if history of positive jitters followed by
  missing pkt followed by negative jitter then
    LOSS = CONGESTION
  else
    // estimate the network congestion
     $p = \frac{curROTT - minROTT}{maxROTT - minROTT}$ 
    if  $p \geq threshold$  and  $l1$  out of last
     $n$  ROTT was over deviation then
      LOSS = CONGESTION
    else
      LOSS = TRANSMISSION
    end if
  end if
end function

```

Fig. 4. Loss Discrimination algorithm – when jitter sequences appear to imply a transmission loss, XRTP resorts to precautionary checks.

some link along the path – an early indication of unfair use of bandwidth. Therefore, positive jitters trigger congestion avoidance and cause XRTP to reduce its transmission rate at the source. To be conservative, XRTP cuts downs its rate based on the average of the history of the past three positive jitters.

$$new_rate = \frac{3}{\sum \text{of past 3 positive jitters}} \quad (10)$$

XRTP reacts to congestion losses by cutting its rate in half.

D. Loss Discrimination

The above discussion presents how the spacing between explicit packet-pair probes, and the jitter between data packets, can be jointly used to perform rate and congestion control in XRTP. However, the performance of congestion control will also depend on the accuracy of loss discrimination. For this, whenever the XRTP receiver discovers a missing packet, it attempts to classify its cause by looking into the history of jitter measurements. Recall that positive jitter indicates growing congestion in the network. If a congested buffer overflows, a packet will be dropped, causing the preceding and following packets to get closer in the buffer. Thus, the XRTP receiver will notice negative jitter for the packet received after the congestion loss. However, a negative jitter may also indicate an unloading network (that happens when XRTP sources reduce their transmission rates). To differentiate between negative jitter due to a congestion loss and negative jitter due to network unloading, XRTP analyses the sequence of jitters before and after the missing packet. Thus, positive jitter (before the missing packet) followed by negative jitter (after the missing packet) is interpreted as a congestion loss. False alarms are possible because a transmission loss will also cause two (preceding and following) packets to come closer during transit. Thus, XRTP may react to a transmission loss as if it were congestion loss, reducing the sending rate.

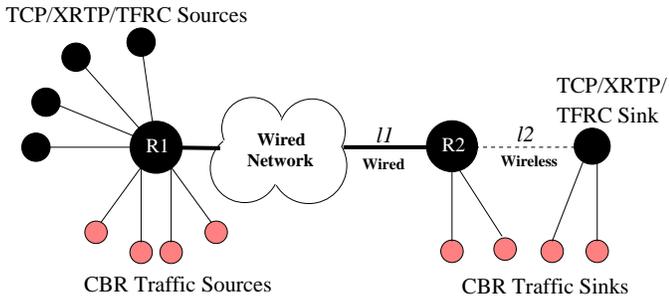


Fig. 5. The structure of the topology used for simulation. Link $l1$ represents a wired network, with $R1$ as the traffic sender. Link $l2$ represents a wireless link with the sink as the wireless receiver. The gray traffic sources initiate CBR cross traffic that terminate at the gray sinks.

However, since MAC protocols like IEEE 802.11 typically implement retransmission schemes for transmission losses, the time consumed for multiple retransmissions can often reduce (and even eliminate) negative jitter³. This reduces the frequency of false alarms.

Although useful, such a technique may not offer sufficient loss discrimination accuracy. This is because the empty slot from a congestion loss can get occupied by packets from cross traffic (instead of the packet following the congestion loss). In other words, the time compression caused by a dropped packet may not translate into a negative jitter. As a result, what seems like a transmission loss may well be a congestion loss. This misclassification can be serious, because XRTP will not reduce its offered load even though the network is congested. To handle such cases, we favor a conservative approach. When the analysis of jitter sequences indicate that the loss in question is due to transmission, XRTP resorts to additional precautionary mechanism as follows. XRTP incorporates the relative one-way trip times (ROTT) [25] and the deviation of ROTT [20] into the heuristic. XRTP determines the ratio of the difference between current and minimum ROTT to the difference between maximum and minimum ROTT. XRTP compares this ratio (i.e., $\frac{currROTT - minROTT}{maxROTT - minROTT}$) with a threshold to determine the condition of the network when the packet loss occurred [20]. The congestion in the network directly correlates to this ratio, and XRTP uses it to improve the confidence of loss estimation. The sketch of the loss discrimination heuristic is presented in Figure 4.

IV. PERFORMANCE EVALUATION

We simulate XRTP over a wired-wireless network and evaluate (1) bandwidth estimation, (2) rate/congestion control, and (3) loss discrimination. The simulations are performed in the ns-2 simulator (version 2.26). However, we also generate traces from a real wireless LAN testbed to feed the simulation.

³Observe that CSMA/CA protocols like IEEE 802.11 will retransmit after allowing other transmitters in the vicinity to send their own packets. In addition, each re-transmission will be preceded with a random backoff, RTS, CTS, and followed by an ACK packet. The sum of all these durations is fairly long.

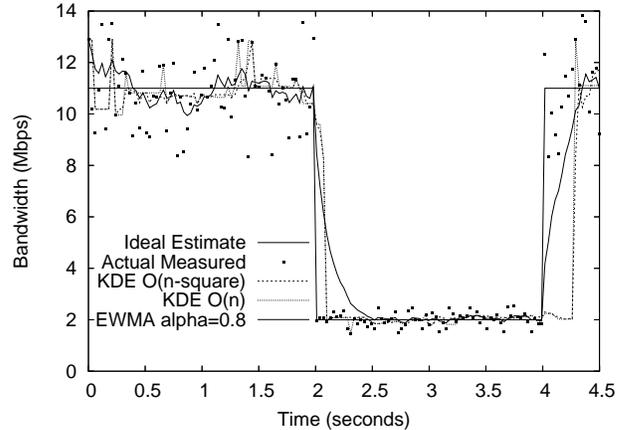


Fig. 6. Bandwidth estimation using KDE and EWMA.

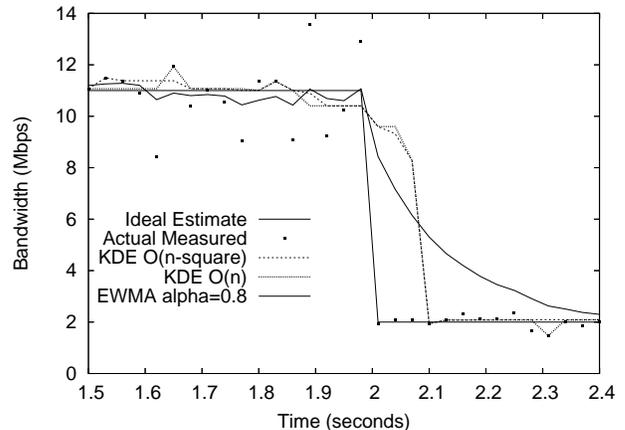


Fig. 7. Snapshot of bandwidth estimation in wireless network.

Figure 5 shows the network topology used – router $R1$ has traffic sources using XRTP, TCP, TFRC and TCP-Westwood. The last hop link, $l2$, is the wireless link with a bandwidth of 2 Mbps. The wireless MAC protocol used is IEEE 802.11. The wired link, $l1$, connects the internal network infrastructure to the wireless router, $R2$. Link $l1$ and all other links connecting the sources to $R1$ have a bandwidth of 10Mbps. The packet loss rate varies from 0% to 5% on the wireless link. The cross-traffic sources and sinks are either CBR or TCP flows, and are randomly switched on and off during the simulations, creating a variety of congestion scenarios. Each simulation is run for 200 seconds. The results are averaged over 500 simulation runs each.

A. Comparing bandwidth estimation using EWMA and KDE

We demonstrate the efficacy of an optimized KDE algorithm to estimate bottleneck bandwidth. Since bandwidth estimation in ns2 is unrealistically accurate due to the use of a virtual clock⁴, we generated traces from running the packet-pair algorithm on a real wireless LAN by sending packet-pairs every 30 ms. The traces were used as input to

⁴We will discuss this in more detail in Section V.

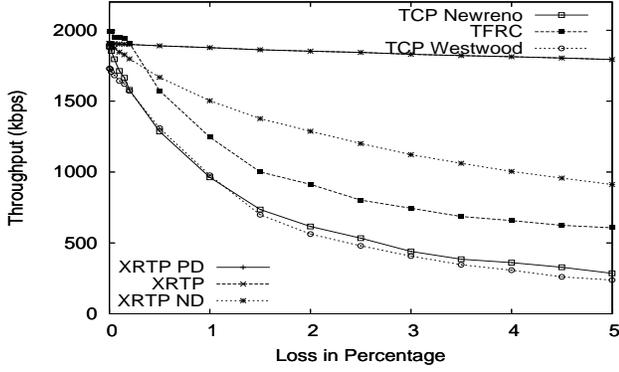


Fig. 8. Performance of XRTP and other flows in isolation.

compare EWMA (with α set to 0.6 and 0.8) and KDE (with $n = 8$).

Figure 6 represents estimates obtained for these algorithms for one set of generated traces. In this particular scenario, the bandwidth is 11Mbps from 0 to 2 seconds, falls to 2Mbps for 2 to 4 seconds, and increases back to 11Mbps till 5 seconds. As expected, the algorithms have less variance than the actual measured value (see Figure 6). In comparison to EWMA, both the KDE algorithms are quicker to adapt to the fall in data-rate. A zoom-in view of Figure 6 is presented in Figure 7. Observe that EWMA overshoots the channel bandwidth at 1.92 and at 2.01 seconds for α set to 0.6. However, both KDE algorithms filter out such spurious estimates and stay close to the channel capacity, proving better agility in bottleneck bandwidth estimation. Moreover, the KDE $O(n)$ algorithm performs comparably to the KDE $O(n^2)$.

B. 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.

We compare XRTP's loss discrimination heuristic with the best and worst cases in loss discrimination. For the best case, we run XRTP with *perfect discrimination* knowledge (XRTP-PD); for the worst case, we run XRTP with *no discrimination* at all (XRTP-ND). We also contrast these results with TCP Newreno, TFRC and TCP Westwood, each run separately for increasing loss rates, varying between 0% to 5%. Results are presented in Figure 8. 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, however, performs very close to XRTP-PD; the two lines for XRTP-PD and XRTP are

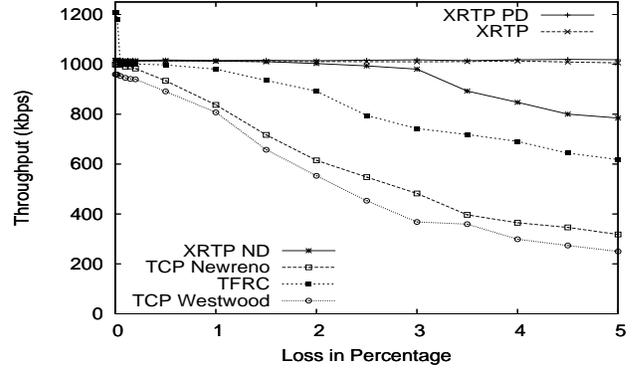


Fig. 9. Performance of XRTP in the presence of CBR cross traffic.

indistinguishable.

In the next experiment (see Figure 9), XRTP is again run with the same three configurations, but with cross traffic of CBR flows that cumulatively use 1Mbps of the bandwidth on Link *l2*. As in the previous simulation, TCP Newreno, TFRC, TCP Westwood and XRTP-ND are seriously impaired 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 them. Table I depicts the performance of the loss discriminator used in XRTP.

- Column 1 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 $C | T$ 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

TABLE I

LOSS DISCRIMINATION HEURISTICS. XRTP WITH CBR. (C DENOTES CONGESTION, AND T DENOTES TRANSMISSION). FOR EXAMPLE, C|T DENOTES PERCENTAGE OF TRANSMISSION LOSSES CLASSIFIED AS CONGESTION LOSSES.

1	2	3	4	5	6	7	8	9	10
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

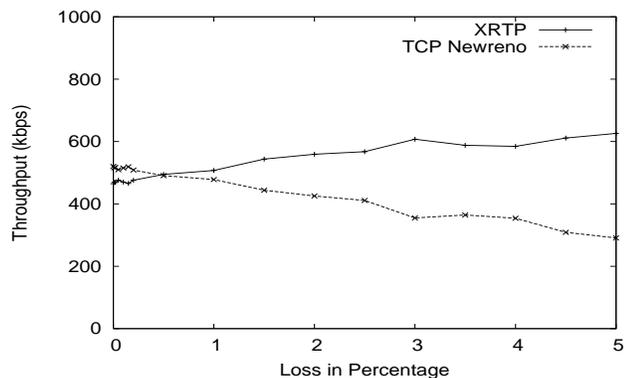


Fig. 10. XRTP versus TCP Newreno with CBR traffic of 1 Mbps.

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 characteristics 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 to misclassification of transmission losses as congestion losses since the parameters of the heuristic are set to conservative values.

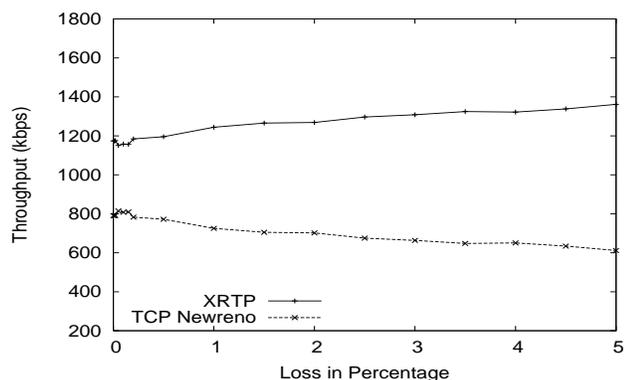


Fig. 11. Four flows of XRTP against four flows of TCP Newreno.

C. Evaluating XRTP in presence of competing cross traffic

Figure 10 shows the throughput for a single XRTP flow competing with a single TCP Newreno flow with background CBR flows that cumulatively use 1Mbps of link $l2$. The graph depicts the throughput of each flow as the loss rate on the wireless link increases. Figure 10 clearly shows improved performance of XRTP, mainly due to accurate classification of transmission losses, and infrequent errors in classifying congestion losses. *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 actually utilizing only that extra bandwidth that is made available by TCP-NewReno since TCP-NewReno is reacting to transmission losses as congestion losses.* This insight can be deduced from Table II.

With increasing loss rates, the percentage of congestion losses (sum of columns 3 and 7) is decreasing, implying that TCP is reacting to transmission losses, while XRTP is able to discriminate them as transmission loss. Table II 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 I 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 suggests that XRTP's proactive congestion avoidance plays an important role in adapting the protocol in the presence of competing flows.

As an evaluation of XRTP in the presence of cross-traffic, Figures 11, 12 and 13 show the results from simulation scenarios with four flows of XRTP competing with four flows of TCP Newreno, TFRC or TCP Westwood, respectively. 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

TABLE II
LOSS DISCRIMINATION STATISTICS OF XRTP AND TCP RESPECTIVELY (C DENOTES CONGESTION, AND T DENOTES TRANSMISSION). FOR EXAMPLE, C|T DENOTES PERCENTAGE OF TRANSMISSION LOSSES CLASSIFIED AS CONGESTION LOSSES.

1	2	3	4	5	6	7	8	9	10
Loss Rate	Total Losses	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

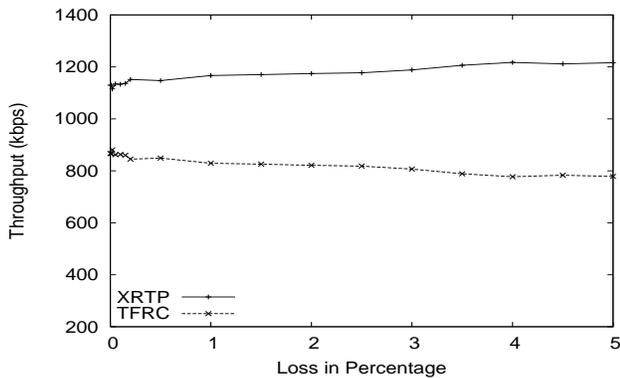


Fig. 12. Four flows of XRTP against four flows of TFRC.

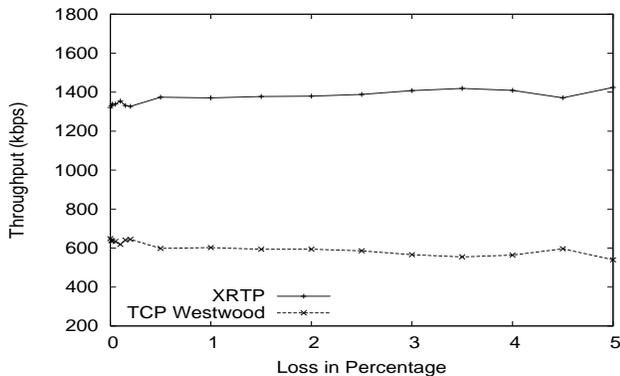


Fig. 13. Four flows of XRTP against four flows of TCP Westwood.

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.

V. DISCUSSION

Effect of Clock Granularity

XRTP, like other rate-based protocols, is sensitive to the precision of the operating system's internal clock. The timer

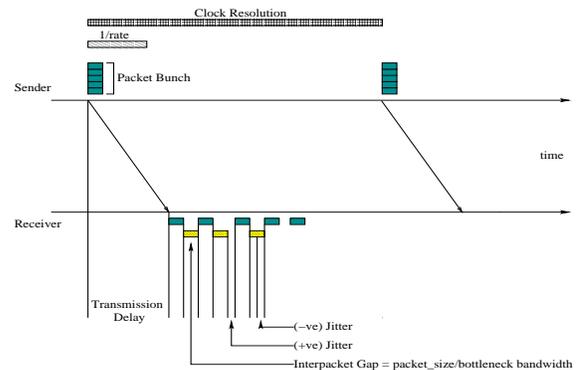


Fig. 14. Jitters caused by variance in propagation delay using coarse grained timer

resolution is about 10 milliseconds (called a jiffy) on a stable Linux system. Ideally, we would like one packet to be sent or received at each jiffy, else packets would bunch at the sender or receiver causing erroneous estimation of jitter and bandwidth. Under the constraint of a timer with 10 milliseconds timeout, XRTP would perform optimally when bottleneck bandwidth is less than 20 Mbps based on the largest packet size. Finer granularity in the timer is available in certain flavors of Linux (such as KURT Linux [26]). With 1 millisecond jiffies, the bottleneck bandwidth can be in the order of 200 Mbps. While we believe that XRTP will operate well under such conditions, higher clock granularity causes systems to expend more energy, which is not desirable. Figure 14 explains the effect of coarse-grained timers on jitter. As a part of our future work, we plan to investigate mechanisms to enhance XRTP for high-granularity clocks.

Choice of parameters in XRTP

The choice of α in the EWMA equation for rate control was chosen to be 0.8. This value was chosen empirically, after simulating the network under various scenarios, and observing the sensitivity to α . As a part of our future work, we intend to choose the value of α through an analytical framework.

The congestion control mechanism, based on the history of 3 jitters, is also empirical. From our simulations with

different number of jitter observations, we noticed that 3 jitters offer consistently higher performance. Nevertheless, these parameters need to be chosen more formally. Our initial attempts toward this direction shows that modeling the impact of different number of jitters on XRTP is mathematically very complicated (like several other functions at the transport layer). We are currently investigating the choice of these parameters in a setting with simplified assumptions.

Potential deployment scenarios

While we designed and developed a transport protocol that works well with TCP flows, as well as smoothen the transmission rate in a lossy wireless environment, it is obvious that it is only suitable in environments, where both the servers and the clients support the XRTP protocol. An example of an emerging application that can use the XRTP protocol is the *mobile phone browser* where service providers provide proxy servers that allows users to access the Web on mobile phones that would normally be incapable of running a Web browser. Other streaming applications such as internet radio stations or internet TV stations are potential candidates that are trying to capture the mobile market. Many of the newer mobile handsets support 802.11bg connectivity and WiFi hotspots are on an upswing. We believe that XRTP type rate-based protocols have a potential in such markets. The service providers would be unburdened from the task of developing UDP streaming protocols to support their services. The mobile phones would be unburdened from installing large number of applications to obtain services as they are limited in their processing and memory capabilities. The underlying networks would be unburdened by rogue (UDP) flows that are unfriendly to other data flows.

VI. CONCLUSION

In this paper, we propose a rate-based transport protocol (XRTP) for lossy wireless networks. The protocol suggests a mechanism that discriminates packet losses based on the spacing between packets that arrive at the destination. The sequence of observed spacings and efficient statistical filtering are also shown to be useful for bandwidth estimation. Bandwidth estimation and loss discrimination are in turn used for continuous congestion control, leading to a constructive synergy between the transport layer components. Simulation results show that XRTP copes well with the dynamic changes in network bandwidth, and achieves fairly accurate loss discrimination for a lossy wireless channel. Also, XRTP is TCP friendly, and does not use up an unfair share of the network bandwidth. While the results are encouraging, we believe that further evaluation is necessary to prove the efficacy of XRTP under complex internet-like topologies. We are also implementing XRTP in the Linux operating system to understand its performance in real wired-wireless environments.

REFERENCES

- [1] V. Jacobson, "Congestion avoidance and control," in *Symposium proceedings on Communications architectures and protocols -SIGCOMM*, 1988.
- [2] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *MOBICOM*, 1995.
- [3] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *ICDCS*, 1995.
- [4] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP westwood: Bandwidth estimation for enhanced transport over wireless links," in *MOBICOM*, 2001.
- [5] R. Krishnan, M. Allman, C. Partridge, J. P.G. Sterbenz, and W. Ivancic, "Explicit transport error notification (ETEN) for error-prone wireless and satellite networks - summary," in *Earth Science Technology Conference*, 2002.
- [6] S. Alfredsson and A. Brunstrom, "TCP-L: Allowing bit errors in wireless TCP," *Proceedings of IST Mobile and Wireless Communications Summit 2003*, June 2003.
- [7] J. Eklund and A. Brunstrom, "Impact of sack delay and link delay on failover performance in sctp," in *Proceedings of 3rd International Conference on Communications and Computer Networks (IC3N06)*, October 2006.
- [8] A. Chockalingam and Michele Zorzi, "Wireless tcp performance with link layer fec/arq," .
- [9] H. Hsieh, K. Kim, Y. Zhu, and R. Sivakumar, "A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces," 2003.
- [10] H. Balakrishnan and R. H. Katz, "Explicit loss notification and wireless web performance," *GLOBECOM*, 1998.
- [11] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, 1994.
- [12] R. Rejaie, M. Handley, and D. Estrin, "Rap : An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *INFOCOM*, 1999.
- [13] 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.
- [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, 2002.
- [15] V. Paxson, "Automated packet trace analysis of TCP implementations," in *SIGCOMM*, 1997.
- [16] R. L. Carter and M. Crovella, "Measuring bottleneck link speed in packet-switched networks.." *Performance Evaluation*, vol. 27/28, 1996.
- [17] K. I. Lai, *Measuring the Bandwidth of Packet Switched Network*, Ph.D. thesis, Stanford University, 2002.
- [18] Karthikeyan Sundaresan, Vaidyanathan Anantharaman, Hung-Yun Hsieh, and Raghupathy Sivakumar, "Atp: a reliable transport protocol for ad-hoc networks," in *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, New York, NY, USA, 2003, pp. 64–75, ACM Press.
- [19] S. Biaz and N. Vaidya, "Discriminating congestion losses from wireless losses using inter-arrival times at the receiver," in *ASSET*, 1999.
- [20] D. Barman and I. Matta, "Effectiveness of loss labeling in improving TCP performance in wired/wireless networks," in *ICNP*, 2002.
- [21] N. Samaraweera, "Non-congestion packet loss detection for TCP error recovery using wireless links," in *IEEE Proceedings of Communications*, 1999, vol. 146.
- [22] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *INFOCOM*, 2000.
- [23] S. Keshav, "A control-theoretic approach to flow control," *SIGCOMM*, vol. 25, 1995.
- [24] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, 1986.
- [25] N. Samaraweera and G. Fairhurst, "Explicit loss indication and accurate RTO estimation for TCP error recovery using satellite links," in *In IEE Proceedings - Communications*, 1997.
- [26] Balaji Srinivasan, Shyamalan Pather Robert, Hill Furguan Ansari, and Douglas 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*, 1998.