Multiple TFRC Connections Based Rate Control for Wireless Networks

Minghua Chen, *Student Member, IEEE,* and Avideh Zakhor *Fellow, IEEE*

***Abstract*— Rate control is an important issue in video streaming applications for both wired and wireless networks. A widely accepted rate control method in wired networks is equation based rate control [1], in which the TCP Friendly rate is determined as a function of packet loss rate, round trip time and packet size. This approach, also known as TFRC, assumes that packet loss in wired networks is primarily due to congestion, and as such is not applicable to wireless networks in which the bulk of packet loss is due to error at the physical layer. In this paper, we propose multiple TFRC connections as an end-to-end rate control solution for wireless video streaming. We show that this approach not only avoids modifications to the network infrastructure or network protocol, but also results in full utilization of the wireless channel. NS-2 simulations, actual experiments over 1xRTT CDMA wireless data network, and and video streaming simulations using traces from the actual experiments, are carried out to validate, and characterize the performance of our proposed approach.**

I. Introduction

Rate control is an important issue in both wired and wireless streaming applications. A widely popular rate control scheme over wired networks is equation based rate control [1] [2], also known as TCP Friendly Rate Control (TFRC). There are basically three advantages to rate control using TFRC: first, it does not cause network instability, thus avoiding congestion collapse. Second, it is fair to TCP flows, which is the dominant source of traffic on the Internet. Third, the TFRC's rate fluctuation is lower than TCP, making it more appropriate for streaming applications which require constant video quality. The *key* assumption behind TCP and TFRC is that packet loss is a sign of congestion. In wireless networks however, packet loss can also be caused by physical channel errors, thus violating this assumption. Neither TFRC nor TCP can distinguish between packet loss due to buffer overflow and that due to physical channel errors, resulting in underutilization of the wireless bandwidth. Hence streaming rate control and congestion control over wireless are still open issues.

Consequently, There have been a number of efforts to improve the performance of TCP or TFRC over wireless [3]–[24]. These approaches either hide end-hosts from packet loss caused by wireless channel error, or provide end-hosts the ability to distinguish between packet loss caused by congestion, and that caused by wireless channel error. To gain a better understanding of the spectrum of approaches to rate control over wireless, we briefly review TCP and TFRC solutions over wireless.

Snoop, a well-known solution, is a TCP-AWARE local retransmission link layer approach [3]. A Snoop module resides on router or base station on the last hop, i.e. the wireless link, and records a copy of every forwarded packets. Assuming snoop module can access TCP acknowledgement packets (ACK) from the TCP receiver, it looks into the ACK packets and carries out local retransmissions when a packet is corrupted by wireless channel errors. While doing the local retransmission, the ACK packet is suppressed and not forwarded to the TCP sender. Other similar approaches based on local link layer retransmission include [9], [12]–[16]. These schemes can potentially be extended to TFRC in order to improve performance, by using more complicated treatment of the ACK packets from the TFRC receiver.

Explicit Loss Notification (ELN) can also be applied to notify TCP/TFRC sender when a packet loss is caused by wireless channel errors rather than congestion [4], [17]. In this case, TFRC can take into account only the packet loss caused by congestion when adjusting the streaming rate.

End-to-end statistics can be used to help detect congestion when a packet is lost [5]–[8], [10], [11], [18]–[22], [24], [25]. For example, by examining trends in the one-way delay variation, Parsa and Garcia-Luna-Aceves [21] interpret loss as a sign of congestion if one-way delays are increasing, and a sign of wireless channel error otherwise. One-way delay can be associated with congestion in the sense that it monotonically increases if congestion occurs as a result of increased queueing delay, and remains constant otherwise. Similarly, Barman and Matta [5] proposed a loss differentiation scheme based on the assumption that the variance of round trip time is high when congestion occurs, and is low otherwise.

Cen et. al. present an end-to-end based approach to facilitate streaming over wireless [19]. They combine packet inter-arrival times and relative one way delay to differentiate between packet loss caused by congestion, and that due to wireless channel errors. There are two key observations behind their approach; first, relative one way delay increases monotonically if there is congestion; second, inter-arrival time is expected to increase if there is packet loss caused by wireless channel errors. Therefore, these two statistics can help differentiate between congestion and wireless errors. However, the high wireless error misclassification rate may result in under-utilizing the wireless bandwidth, as shown in [19]. Yang et. al. [24] also propose a similar approach to improve video streaming performance in presence of wireless error, under the assumption that wireless link is the bottleneck.

This work was supported by AFOSR contract F49620-00-1-0327.

M. Chen and A. Zakhor are with Video and Image Processing Lab, Department of Electrical Engineering and Computer Science at University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: {minghua, avz}@eecs.berkeley.edu).

Other schemes such as [6]–[8], [10], [11], [18] that use end-to-end statistics to detect congestion, can also be combined with TFRC for rate control. The congestion detection scheme can be used to determine whether or not an observed packet loss is caused by congestion; TFRC can then take into account only those packet losses caused by congestion when adjusting streaming rate.

Tang et. al. proposed the idea of using small dummy packets to actively probe whether the network is congested in case of packet loss, so as to differentiate between packet loss due to congestion and that due to channel error [23]. Yang et. al. [25] propose a cross-layer scheme that uses link layer information to determine whether a packet loss is caused by channel error or congestion, assuming that only the last link is wireless. In this approach, when a packet is lost, TFRC goes beyond layering abstraction and enquires the link layer about the recent signal strength. The packet loss is recognized to be due to wireless channel error if recent signal strength is low, and due to congestion otherwise. A similar assumption is made by Akan and Akyildiz in [26] to derive a wireless TFRC-like equation based protocol to facilitate video streaming.

The disadvantage of end-to-end statistics based approaches is that congestion detection schemes based on statistics are not sufficiently accurate, and they either require cross layer information or modifications to the transport protocol stack.

Another alternative is to use non-loss based rate control schemes. For instance, TCP Vegas [27], in its congestion avoidance stage, uses queueing delay as a measure of congestion, and hence could be designed not to be sensitive to any kind of packet loss, including that due to wireless channel error. It is also possible to enable the routers with ECN markings capability to do rate control using ECN as the measure of congestion [28]. As packet loss no longer corresponds to congestion, ECN based rate control does not adjust sending rate upon observing a packet loss.

In this paper, we show that using one TFRC connection in wireless streaming applications results in underutilization of the wireless bandwidth. We then propose the use of multiple simultaneous TFRC connections for a given wireless streaming application. The advantages of our approach are as follows: first, it is an end-to-end approach and does not require any modifications to network infrastructure and protocols, except at the application layer. Second, as will be pointed out later, it has the potential to fully utilize the wireless bandwidth provided the number of connections and packet size are selected appropriately. The disadvantages are, more complex control procedures, and more system resources, e.g. memory, for opening more connections on end-hosts.

Other similar work, but not related to our approach include MULTCP [29] and NetAnts [30]. They both open multiple connections to increase throughput. MULTCP was originally used to provide differential service, and was later used to improve the performance in high bandwidth-round-trip-time product networks. NetAnts achieves higher throughput by opening multiple connections to compete for bandwidth against others. Since fairness of TCP is at the connection level rather than application level, using more connections than other applications can result in higher individual throughput. The difference between NetAnts and our approach are as follows. First, opening more connections than needed in wired networks increases the end-to-end packet loss rate experienced by end-host. Second, unlike our approach, there is no mechanism to control the number of connections in NetAnts.

The rest of the paper is structured as follows. In Section II, we present the problem formulation together with an optimal strategy based on multiple TFRC connections. NS-2 simulations and actual experiments are carried out to validate the basic idea. In Section III, we propose a practical system called MULTFRC to implement the approach discussed in Section II. NS-2 simulations, actual experimental results, and video streaming simulations using traces from the actual experiments are included in Section IV to show the efficiency of MULTFRC. Conclusions and future works are in Section V.

II. Problem Formulation

In this section, we begin by analyzing the performance of one TFRC for streaming over wireless. We then propose a rate control strategy, based on multiple TFRC connections, that has the potential to achieve optimal performance, i.e. maximum throughput, and minimum end-to-end packet loss rate.

A. Setup and Assumptions

The typical scenario for streaming over wireless is shown in Figure 1 where the sender is denoted by s, and the receiver by r. As shown, a video server in the wired network is streaming video to a receiver in the wireless network. The wireless link is assumed to be the bottleneck, and is associated with available bandwidth B_w, and packet loss rate p_w, caused by wireless channel error. This implies that the maximum throughput over the wireless link is $B_w(1-p_w)$. There could also be packet loss caused by congestion at node 2, denoted by p_c. The end-to-end packet loss rate observed by receiver is denoted by p. The streaming rate is denoted by T. This implies that the streaming throughput is $T(1-p)$. We refer to the wireless channel as underutilized if $T(1-p) < B_w(1-p_w)$.

The reasons for choosing this scenario to analyze are that first, it is a simplified version of the popular cellular wireless data transmission scenario in which we are interested. Second, it captures the fundamental problem that we wish to analyze. Third, it makes our analysis easy to understand and evaluate. To evaluate performance of the proposed scheme in a more realistic environment, we rely on NS-2 simulations and actual experiments over Verizon 1xRTT wireless data network in Section IV.

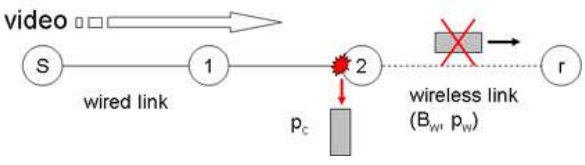


Fig. 1. Typical scenario for streaming over wireless.

Given this scenario, we assume the following:

1) The wireless link is assumed to be the long-term bottleneck. By this, we mean there is no congestion at node 1. Hence the end-to-end packet loss rate depends only on p_c and p_w.
2) There is no congestion and queuing delay at node 2, if and only if the wireless bandwidth is underutilized, i.e. we achieve $p_c = 0$ and minimum round trip time, denoted by RTT_{min} if and only if $T \leq B_w$. This in turn implies that if $T > B_w$ then $rtt \geq RTT_{min}$, where rtt is the end-to-end round trip time.
3) B_w and p_w are assumed to be constant.
4) The packet loss caused by wireless channel error is assumed to be random and stationary.
5) Our objective is to optimize long-term streaming throughput and packet loss rate performance rather than short term behavior.
6) Packet size S for all connections of one application are the same, unless otherwise stated.
7) We assume one TFRC connection not to fully utilize B_w, otherwise it already achieves optimal performance, and no improvement is to be expected.
8) For simplicity, the backward route is assumed to be error-free and congestion-free; otherwise one can always carry out sufficient amount of retransmissions or use reliable protocol (e.g. TCP) to transmit the limited amount of information back to sender reliably.

Based on this scenario, the two goals of our rate control can be stated as follows. First, the streaming rate should not cause any network instability, i.e. congestion collapse. Second, it should lead to the optimal performance, i.e. it should result in highest possible throughput and lowest possible packet loss rate.

TFRC can clearly meet the first goal, because it has been shown (a) to be TCP-friendly, and (b) not to cause network instability. In the remainder of this paper, we propose ways of achieving the second objective listed above, using a TFRC-based solution, without modifying the network infrastructure and protocols.

B. A Sufficient and Necessary Condition for Underutilization

We use the following model for TFRC to analyze the problem [2]:

$$T = \frac{kS}{rtt\sqrt{p}} \tag{1}$$

T represents the sending rate, S is the packet size, rtt is the end-to-end round trip time, p is the end-to-end packet loss rate, and k is a constant factor between 0.7 [31] and 1.3 [32], depending on the particular derivation of Equation (1). Although this model has been refined to improve accuracy [1] [33], it is simple, easy to analyze, and more importantly, it captures all the fundamental factors that affect the sending rate. Furthermore, the results we derive based on this simple model can be extended to other more sophisticated models, such as the one used in [1], which we have empirically verified not to affect our results and conclusions.

Given this model, the average throughput measured at the receiver is $T(1-p)$, when streaming rate is T, and overall packet loss rate is p. End-to-end packet loss rate p is a combination of p_w and p_c, as follows:

$$p = p_w + (1 - p_w)p_c \tag{2}$$

Equation (2) shows that p_w is a lower bound for p, and that the bound is reached if and only if there is no congestion, i.e. $p_c = 0$. Combining Equations (1) and (2), an upper bound, T_b, on the streaming rate of one TFRC connection can be derived as follows:

$$T \leq \frac{kS}{RTT_{min}\sqrt{p_w}} \equiv T_b \tag{3}$$

If there is no congestion, i.e. $p_c = 0$, and hence no queuing delay caused by congestion, we get $rtt = RTT_{min}$, $p = p_w$, and therefore $T = T_b$ in Equation (3). In this case, the throughput is $T_b(1 - p_w)$, which is the upper bound of throughput given one TFRC connection for the scenario shown in Figure 1. We define the wireless link to be underutilized if the overall end-to-end throughput is less than $B_w(1 - p_w)$. Based on these, we can state the following:

Theorem 1: Given the assumptions in Section II.A, sufficient and necessary condition for one TFRC connection to underutilize wireless link is

$$T_b < B_w. \tag{4}$$

Proof: Since $T_b(1 - p_w)$ is the upper bound of one TFRC's throughput, clearly Equation (4) implies underutilization of the wireless channel, and hence the "sufficient" part of the Theorem is obvious. To see the necessary part, note that if underutilization happens, i.e. $T(1-p) < B_w(1-p_w)$, then invoking assumption 2 in Section II.A, no congestion happens, thus $rtt = RTT_{min}$, $p = p_w$ and $T = T_b$, resulting in $T_b(1 - p_w) < B_w(1 - p_w)$. ■

If the condition in (4) is satisfied, then direct application of TFRC or TCP to wireless scenario results in underutilization. In essence, the approaches taken in [3], [4], [6]–[19], [34] ensure the condition in (4) is not satisfied, through modifications to network infrastructure or protocols.

For example in the TFRC-AWARE Snoop-like solution, p_w becomes effectively zero through local retransmissions. This makes $T_b \to \infty$ and thus ensures the condition in (4) is avoided. Basically by effectively setting $p_w = 0$, Snoop-like module translates the new problem, i.e. rate control for streaming over wireless, into an old one, i.e. rate control for streaming over wired network, for which a known solution exists. Similarly, ELN and end-to-end statistics based approaches make TFRC not respond to packet loss caused by wireless channel errors, thus not taking p_w into account when adjusting streaming rate. This is effectively the same as setting $p_w = 0$, thus improving the performance of the TFRC connection.

C. A Strategy to Reach the Optimal Performance

It is not necessary to avoid the condition in (4) in order to achieve good performance for one *application*. This is because it is conceivable to use multiple simultaneous connections for a given streaming application. The total throughput of the application is expected to increase with the number of connections until it reaches the hard limit of $B_w(1 - p_w)$.

1) Analysis on the Optimal Number of Connections: Given the scenario shown in Figure 1, and the assumptions stated in Section II.A, we now argue that multiple connections can be used to achieve optimal performance, i.e. throughput of $B_w(1-p_w)$, and packet loss rate of p_w. To see this, let us consider a simple example in which

$$B_w(1-p_w) = \frac{2.5kS}{RTT_{min}\sqrt{p_w}}(1-p_w) = 2.5T_b(1-p_w)$$

By opening one TFRC connection with packet size S, the application achieves a throughput of $\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w) = T_b(1-p_w)$ and packet loss rate of p_w. This is because according to Theorem 1, underutilization implies $rtt = RTT_{min}$, $p = p_w$ and $T = \frac{kS}{RTT_{min}\sqrt{p_w}} = T_b$.

Let us now consider the case with two TFRC connections from sender s to receiver r in Figure 1. Following the assumptions and analysis in Sections II.A and II.B, it is easy to see that p_w for each of the two TFRC connections remain unchanged from the case with one TFRC connection. This is because the throughput upper bound for each of the two TFRC connections is still $T_b(1-p_w)$, and the aggregate throughput upper bound for both of them is $2T_b(1-p_w)$, which is smaller than $B_w(1-p_w)$, implying channel underutilization. Invoking assumption 2, we conclude that there is no congestion and hence $rtt = RTT_{min}$ and $p_c = 0$, and thus $p = p_w$. The throughput for each connections is then $\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w)$. Consequently, the total throughput for both connections is $2\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w)$ with packet loss rate at p_w.

A similar argument can be made with three TFRC connections, except that the wireless channel is no longer underutilized and $rtt > RTT_{min}$. Furthermore, if the buffer on node 2 overflows then p_c will no longer be zero and hence using Equation (2) we get $p > p_w$. In this case the wireless link is still fully utilized at $T(1-p) = B_w(1-p_w)$, but round trip time is no longer at the minimum value RTT_{min}; furthermore overall packet loss rate p could exceed p_w, i.e. the overall packet loss rate in the two connections case.

In general, given B_w, p_w, and the packet size S for each connection, it can be shown that when full wireless channel utilization occurs, the optimal number of connections, n_{opt}, satisfies:

$$\begin{aligned} B_w(1-p_w) &= n_{opt}\,\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w) \\ \Rightarrow\ n_{opt}S &= B_w\,\frac{RTT_{min}\sqrt{p_w}}{k} \end{aligned} \tag{5}$$

Thus what really matters is the product of n_{opt} and S, and it is always possible to achieve full wireless channel utilization by choosing n_{opt} to be an integer, and by selecting S accordingly[1]. It is also possible to analyze the case with different packet sizes for different connections, but this is harder to analyze, and it is not fundamentally different from the case with the same packet size for all connections. For the case with the packet size fixed at S, the optimal number of connections is given by

$$\left\lfloor B_w\,\frac{RTT_{min}\sqrt{p_w}}{kS}\right\rfloor \equiv \hat{n}_{opt} \tag{6}$$

resulting in throughput of $\hat{n}_{opt}\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w)$ and packet loss rate of p_w.

To show that opening more than n_{opt} connections results in larger rtt, or possibly higher end-to-end packet loss rate, assume n_{opt} and S lead to the optimal performance, and consider opening $n_{opt}+\delta n$ connections, where δn is a positive integer. Denoting the end-to-end packet loss rate as p' for this case, the overall throughput is given by $(n_{opt}+\delta n)\frac{kS}{rtt\sqrt{p'}}(1-p') = B_w(1-p_w)$ and hence

$$(n_{opt}+\delta n)S = B_w\,\frac{1-p_w}{1-p'}\,\frac{rtt\sqrt{p'}}{k} \tag{7}$$

Comparing the above equation with Equation (5), and taking into account that the right hand sides of Equations (5) and (7) are monotonically increasing functions with respect to overall packet loss rate and round trip time, we conclude that either $rtt > RTT_{min}$ and/or $p' > p_w$.

The intuition here is that as number of connections exceeds n_{opt}, the sending rate of each connection has to decrease. Thus by (1), the product $rtt\sqrt{p}$ has to increase, so either rtt increases or p increases, or they both increase. In practice, as the number of connections exceeds n_{opt}, initially p remains constant and rtt increases due to the increase on queueing delay at node 2, i.e. $rtt > RTT_{min}$; if the number of connections keeps increasing and buffer on node 2 overflows, rtt will then stop increasing, and p begins to increase. Eventually we get both $rtt > RTT_{min}$ and $p > p_w$.

To summarize, if the number of TFRC connections is too small so that the aggregate throughput is smaller than $B_w(1-p_w)$, wireless channel becomes underutilized. If the number of connections is chosen optimally based on Equation (5), then wireless channel becomes fully utilized, the total throughput becomes $B_w(1-p_w)$, the $rtt = RTT_{min}$, and the overall packet loss rate is at the lower bound p_w. However, if the number of connections exceeds n_{opt}, even though the wireless channel continues to be fully utilized at $B_w(1-p_w)$, the rtt will increase beyond RTT_{min} and later on packet loss rate can exceed the lower bound p_w. In Section III, we use the above conclusions to develop a practical scheme called MULTFRC to determine the optimal number of connections.

2) Simulations and Experimental Verification: To validate the above conclusions, we carry out both NS-2 [35] simulations and actual experiments over Verizon Wireless 1xRTT CDMA data network. The topology for NS-2 simulations is the same as the one shown in Figure 1 with the following settings: $B_w = 1\ Mbps$, $RTT_{min} = 168\ ms$, $S = 1000$ bytes, and p_w varying from 0.0 to 0.16. Also, no cross traffic is introduced for illustration purposes. Within NS-2, we stream 1, 2, 4, 8, 16 and 32 TFRC connections from a fixed host to mobile hosts for 10 runs with each run lasting 1000 seconds. The wireless link is modelled as a wired link with an exponential random packet loss model.

The results of NS-2 simulations indicating throughput, packet loss rate and round trip time as a function of wireless

[1]Of course p_w may also change when packet size changes, but for the sake of simplicity, we assume p_w is stable as packet size changes. Analysis can be extended given a relation between p_w and S. The point here is to change packet size to achieve finer granularity in increase/decrease.

channel error rate, p_w, for different number of connections, are shown in Figure 2; also shown in Fig. 2(d) is the optimal number of connections computed as the ratio between the bandwidth and one connection's throughput. There are three observations to be made. First, for a given p_w, throughput increases with the number of connections up to a point, after which there is a saturation effect. For example, for $p_w = 0.04$ we need to open at least 4 connections to maximize the throughput. Second, for a fixed p_w, opening too many connections results in either higher packet loss rate, or higher round trip time than RTT_{min}, or both; for instance, seen from Figure 2, at $p_w = 0.04$, opening 8 connections results in increase in round trip time but not in packet loss rate; however, opening 16 or 32 connections results in packet loss rate to be higher than 0.04, and larger round trip time. Third, given B_w, p_w, RTT_{min}, and S, there is an "optimal" number of connections with the highest throughput and the lowest packet loss rate; for example, for $p_w = 0.04$, the optimal number of connections is a number slightly larger than 4. One might note this is higher than the corresponding optimal number of connections shown in Fig. 2(d), which is 3.85. This is because when the number of opened connections is close to the computed optimal one, every connection suffers from the variance of the wireless packet loss; when the bottleneck is fully in use, competition among connections results in lack of full utilization.

Another observation about the optimal number of connections is that it increases abruptly after $p_w > 0.1$. A careful observation into the trace file shows that TFRC suffers many timeouts when p_w is high and too many packets are lost, in which case TFRC's performance is no longer packet loss rate dominated but rather timeout dominated. TFRC resets its sending rate when timeout happens, and goes into slow start stages as if it is just started. As a result, the throughput of one TFRC drops significantly, hence requiring more TFRC connections in order to achieve full utilization, resulting in an abrupt increase in the number of connections. We should point out $p_w > 0.1$ is probably too large to be observed in practice; as such, in the later part of the paper, we focus on the $p_w \leq 0.1$ scenario.

Similar experiments are carried out on Verizon Wireless 1xRTT CDMA data network. The 1xRTT CDMA data network is advertised to operate at data speeds of up to 144 kbps for one user. As we explore the available bandwidth for one user using UDP flooding, we find the highest average available bandwidth averaged over 30 minutes to be between 80 kbps to 97 kbps. In our experiments, we stream for 30 minutes from a desktop on wired network in eecs.bekerley.edu domain to a laptop connected via 1xRTT CDMA modem using 1, 2 and 3 connections with packet size of $S = 1460$ bytes. We measure the total throughput, packet loss rate and round trip time as shown in Table I. Clearly, the optimal number of connections is 2. Specifically, the loss rate is slightly higher for 3 connections than for 2, while the throughput is more or less the same for 2 and 3 connections.

The fact that the average rtt of 3 connections is lower than that of 2 connections is somewhat counter-intuitive. However, a careful investigation of the trace files indicates that when

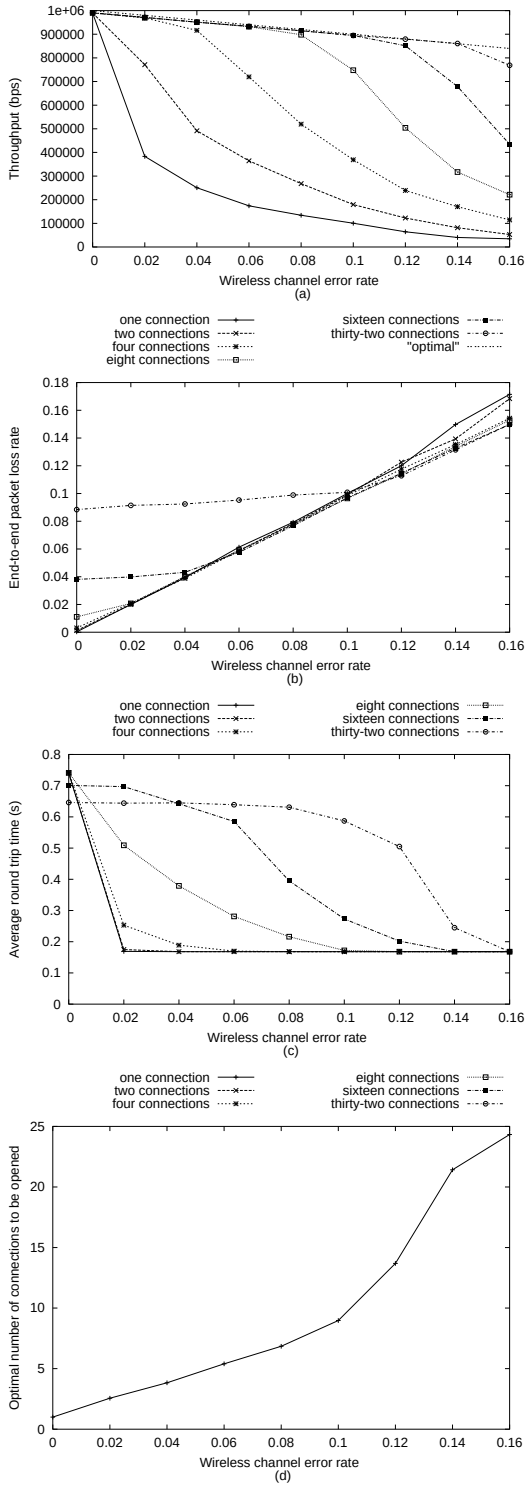


Fig. 2. NS-2 simulations showing (a) End-to-end throughput, (b) packet loss rate, (c) round trip time, and (d) optimal number of connections as a function of wireless channel error rate, p_w, for different number of connections. For clarity of illustration, we have not plotted the standard deviation associated with each point in the graphs. To provide a rough idea, the highest standard deviation for the points in the figures are (a) 9.3k bps, (b) 0.0012, and (c) 0.0024 seconds.

3 TFRC connections is opened, the wireless link is heavy congested, and TFRC sender sometimes gets into timeout states, resetting its sending rate to zero, and performing slow start again; this mimics the behavior of TCP. As a result, for a certain period after the timeout event, the total sending rate becomes low, the queue on the bottleneck is decreased, resulting in a smaller rtt. On the contrary, when we open 2 TFRC connections, the wireless link is fully utilized but not heavily congested, resulting in no observed timeout events in the trace files. Therefore, the queue is kept full. Hence, the average rtt of 3 connection case, which is affected by timeout events, is lower than the average rtt of 2 connection case, which is timeout free. The same reasoning explains the throughput of 3 connections being somewhat lower than that of 2 connections.

TABLE I
EXPERIMENTAL RESULTS FOR VERIZON WIRELESS 1XRTT CDMA DATA NETWORK.

number of conn.'s	throughput (kbps)	rtt (ms)	pkt loss rate
one	57	1357	0.018
two	48.2+45.6=94	2951	0.032
three	33.2+31.9+27.8=93	2863	0.046

Based on the above analysis and experiments, strategy leading to optimal performance can be described as follows:

Keep increasing the number of connections until an additional connection results in increase of end-to-end round trip time or packet loss rate.

As seen in Section III, in practical implementation of the above strategy, we use average round trip time measurements, rather than packet loss rate as in indicator of the optimal number of connections; this is because the increase in average round trip time typically happens before the increase in packet loss rate, and thus enables us to detect the full utilization earlier. In the next section, we propose a system called MULTFRC that uses round trip time measurements to implement the above strategy.

III. MULTIPLE TFRC (MULTFRC)

The basic idea behind MULTFRC is to measure the round trip time, and adjust the number of connections accordingly. Specifically, we increase the number of connections n by α/n or decrease it by β, depending on the rtt measurements. α and β are preset constant parameters of our control algorithm. The design goals are twofold: first, utilize the wireless bandwidth efficiently; second, ensure fairness between applications.

The framework of MULTFRC is shown in Figure 3. As seen, there are two components in the system: rtt measurement sub-system (RMS), and connections controller subsystem (CCS). The flowchart of the system is shown in Figure 4. We now describe each component in detail.

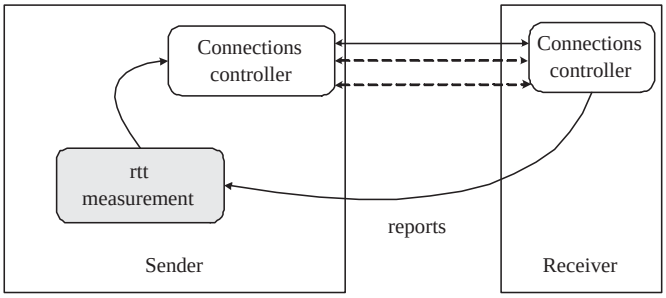
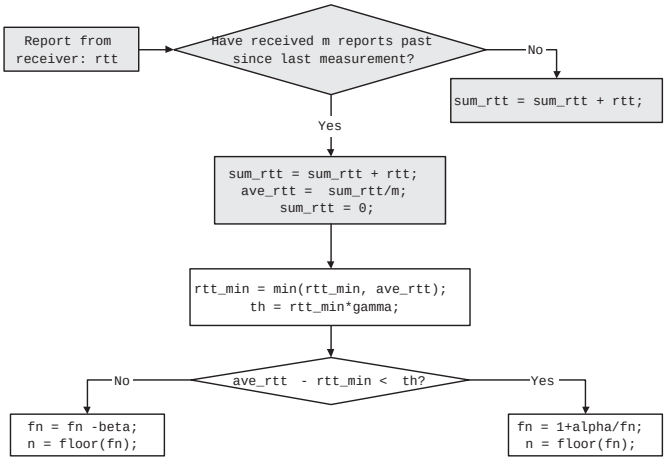


Fig. 3. MULTFRC system framework.




```
m           -- number of reports received over which we compute
               the average rtt;
n           -- number of connections opened;
fn          -- float version of n, used in inversely increase computation;
th          -- threshold used to judge whether there is an increase in
               ave_rtt
sum_rtt     -- sum of reported rtts, use internally for computing ave_rtt;
ave_rtt     -- average round trip time computed;
alpha       -- parameter alpha in IIAD
beta        -- parameter beta in IIAD
gamma       -- ratio used when updating the th
```

Fig. 4. Flow-chart for MULTFRC system. Blocks in gray represent the functionalities of rtt measurement sub-system, blocks in white represent those of connection controller sub-system.

A. rtt Measurement Sub-system (RMS)

The gray blocks in Figures 3 and 4 represent RMS that resides at the sender; it basically measures average rtt over a window, denoted by ave_rtt, and reports it to the CCS. As shown in the system flowchart in Figure 4, RMS receives reports from receiver every round trip time, containing the an average rtt_{sample} measured in the past round trip time window. RMS then further computes a smoothed version of these average rtt's every m reports, as follows:

$$ave\_rtt = \frac{\sum_{i=1}^{m} rtt\_sample_i}{m} \tag{8}$$

Setting m to large values can reduce the noise in ave_rtt, while setting it to small values makes the system more responsive to changes in round trip time.

B. Connection Controller Sub-system (CCS)

The CCS is shown as the white blocks in Figures 3 and 4. Its basic functionality is to Inversely Increase and Additively Decrease (IIAD(α, β)) the number of connections n, based on the input from RMS, as illustrated in Figure 4. Specifically, the CCS block at the sender first sets the rtt_min as the minimum ave_rtt seen so far, and then adapts the number of connection, denoted by n, as follows:

$$n = \begin{cases} n - \beta, & \text{if } ave\_rtt - rtt\_min > \gamma rtt\_min; \\ n + \alpha/n, & \text{otherwise.} \end{cases} \quad (9)$$

where γ is a preset parameter. The choice of IIAD to control n is motivated by [36]. Specifically, it is shown in [36] that sending rates controlled by IIAD can efficiently and fairly share the network bandwidth, at the same time has lower rate variations than Additive Increase and Multiplicative Decrease (AIMD). We expect the number of connections controlled by IIAD to also have this property.

In practice, the number of connections opened should be an integer. However, controlling n as shown in (9) might not result in an integer. Thus in our implementation of MULTFRC, we quantize n to its closest integer to determine the number of connections. This quantization may result in bandwidth underutlization, as shown and discussed in Section IV.

For a given route, the rtt_min is a constant representing the minimum round trip time for that route, i.e. with no queuing delay. As an example, on a wireless link with no cross traffic, the rtt_min simply corresponds to physical propagation delay. As such, $ave\_rtt - rtt\_min$ corresponds to current queuing delay, and γrtt_min is a threshold on the queuing delay that MULTFRC can tolerate before it starts to decrease the number of connections. As a result, under ideal conditions, MULTFRC keeps increasing the number of connections to make ave_rtt as close as possible to $(1+\gamma)rtt\_min$ without exceeding it. Ideally, ave_rtt becomes larger than rtt_min if and only if the link is fully utilized, and the queue on bottleneck link router is built up, introducing additional queuing delay. Thus by evaluating the relation between ave_rtt and rtt_min, MULTFRC detects full utilization the wireless link, and controls the number of connections accordingly.

When there is a route change either due to change in the wireless base station, or due to route change within the wired Internet, the value of rtt_min changes, affecting the performance of MULTFRC. Under these conditions, it is conceivable to use route change detection tools such as traceroute [37] to detect the route change, in order to reset rtt_min to a new value. Furthermore, it can be argued that the overall throughput of MULTFRC will not go to zero, resulting in starvation; this is because MULTFRC always keeps at least one connection open.

Since the video stream is transmitted using multiple connections, the receiver could potentially receive out of order video packets. In case receiver uses a buffer to reorder the arriving video packets, the out of order arrival is not a serious issue.

C. Rate of change of the number of connections

From Equation (9), it is seen that MULTFRC increases n at a rate that is inversely proportional to the number of connections. The continuous time approximation of control low in (9) for increasing stage of n is as follows:

$$\dot{n}(t) = \frac{\alpha}{m \cdot rtt \cdot n(t)} \Rightarrow n^2(t) - n^2(0) = \frac{2\alpha}{m \cdot rtt} t.$$

Therefore, to increase n from N_2 to N_1, assuming $N_1 > N_2$, it roughly takes MULTFRC $\frac{m}{2\alpha}(N_1^2 - N_2^2)$ rtts. On the other hand, MULTFRC decreases the number of connections, n, at a constant rate. The continuous time approximation of control low for the decreasing stage is:

$$\dot{n}(t) = -\frac{\beta}{m \cdot rtt} \Rightarrow n(t) - n(0) = -\frac{\beta}{m \cdot rtt} t.$$

Therefore, it takes MULTFRC $\frac{m(N_1 - N_2)}{\beta}$ rtts to decrease n from N_1 to N_2. In summary, MULTFRC is conservative in adding connections, but aggressive in closing them.

In a simple topology with one MULTFRC system over one wireless link, we can use the increasing rate result to *roughly* compute bandwidth utilization ratio. In this simple topology, MULTFRC periodically increases the number of connections n to the optimal n^* that fully utilize the wireless bandwidth, and proportionally decreases n upon reaching n^*. The approximated continuous version of this process is demonstrated in Fig. 5. In the plot, T is the time for $n(t)$ to increases from $n^* - \beta$ to n^* [2], and hence is $((n^*)^2 - (n^* - \beta)^2) m\, rtt/2\alpha$. The number of connections $n(t) = \sqrt{(t - kT)2\alpha/m\, rtt + (n^* - \beta)^2}, kT \le t \le (k+1)T, k \in Z^+$.

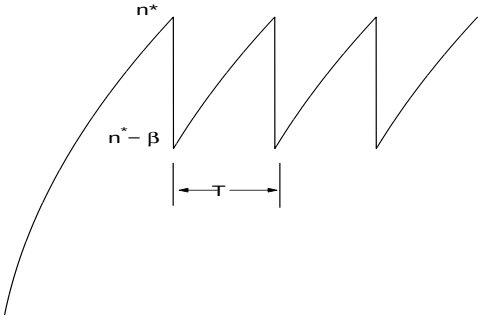


Fig. 5. Demonstration of the change on the number of connections n, controlled by single MULTFRC over single wireless link.

The utilization ratio, in the stationary stage, is then the ratio between the average number of connections and n^*, as follows:

$$\frac{1}{Tn^*} \int_0^T n(\delta) d\delta = \frac{2}{3} \frac{1 + \frac{n^* - \beta}{n^*} + (\frac{n^* - \beta}{n^*})^2}{1 + \frac{n^* - \beta}{n^*}}. \quad (10)$$

This ratio is at least 2/3, and only depends on β and is independent of wireless packet loss rate, m, and rtt. Further, this utilization ratio is a monotone function of the ratio $(n^* - \beta)/n^*$. The larger the optimal number of connections n^*, the smaller β, the better utilization. This meets the intuition that the throughput decrease due to constant decrease on n is less significant for large values than small ones.

[2]Here we assume $n^* - \beta \ge 1$; otherwise the analysis follows with the increase stage starts from 1 rather than $n^* - \beta$, since MULTFRC always has one connection opened.

IV. Simulations and Experimental Results

In this section, we carry out NS-2 simulations and actual experiments over Verizon Wireless 1xRTT CDMA data network to evaluate the performance of MULTFRC system.

A. Setup

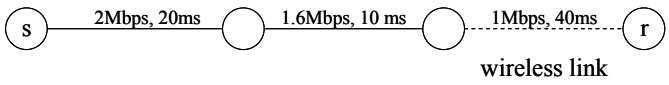


Fig. 6. Simulation topology.

The topology used in simulations is shown in Figure 6. The sender denoted by s, and the receiver denoted by r, both run MULTFRC at the application layer. For all simulations, the wireless bandwidth B_w is set be 1 Mbps and is assumed to be the bottleneck. The wireless link is modelled by an exponential error model, and p_w varies from 0.0 to 0.08 in increments of 0.02. DropTail type queue is used for each node. In order to evaluate MULTFRC's performance in the presence of wireless channel errors. We examine three issues; first, how MULTFRC performs in terms of average throughput, average round trip time, and packet loss rate, as a function of p_w. Second, whether the number of connections is stable. Third, whether or not a MULTFRC application can fairly share with an application using one TFRC or one TCP connection. In all the simulations, throughput is measured every second, packet loss rate is measured every 30 seconds, the average round trip time is measured every 100 packets, and the number of connections is sampled whenever there is a change in it.

For the actual experiments over 1xRTT, we stream from a desktop connected to Internet via 100 Mbps Ethernet in eecs.berkeley.edu domain to a notebook connected to Internet via Verizon Wireless 1xRTT CDMA data network. Thus it is quite likely that the last 1xRTT CDMA link is the bottleneck for the streaming connection. The packet size S is 1460 bytes, and the streaming takes 30 minutes. As we cannot control p_w in actual experiments, we measure the average throughput, average number of connections, and packet loss rate.

B. Performance Characterization of MULTFRC

We have empirically found the following parameters to result in reasonable performance: $\alpha = \beta = 1$, $\gamma = 0.2$ and $m = 50$. Intuitively, larger m results in more reliable estimates of round trip time, but at the expense of a lower sampling frequency, resulting in a less responsive system. Larger γ results in a system that is more robust to round trip time estimates, but at the expense of longer queues in routers and hence a longer queueing delay. Larger α and β result in a large adaptation rate, but at the expense of a large variation in the number of connections controlled, resulting in more throughput fluctuations. We have empirically found these values through simulations and experiments.

We simulate the MULTFRC system to stream for 9000 seconds, and compute the average throughput and packet loss rate for p_w =0.0, 0.02, 0.04, 0.06 and 0.08, and compare them to the optimal, i.e. $B_w(1 - p_w)$ for each p_w. The results for $B_w = 1\ Mbps$ and $RTT_{min} = 168\ ms$ are shown in Figure 7(i). As seen, the throughput is within 25% of the optimal, the round trip time is within 120% of RTT_{min}, and the packet loss rate is almost identical to the optimal, i.e. a line of slope one as a function of wireless channel error rate. As expected, the average number of connections increases with wireless channel error rate, p_w. To confirm MULTFRC's performance over a wider range of parameters, we carry out additional simulations using the same topology as in Figure 6, with $B_w = 100\ kbps$ and $RTT_{min} = 757\ ms$. The results, shown in Figure 7(ii) are as expected, and validate our earlier observations. [3] The results also show that MULTFRC throughput is reasonably close to the predicted values, which are computed using (10) with $\beta = 1$ and n^* being the average number of connections opened in the corresponding simulations. This observation demonstrates the effectiveness of the utilization ratio analysis in Section III-C.

Considering the throughput plots in Figure 7, we notice that for some values of p_w, there is a significant difference between the actual and optimal throughput. This is due to the quantization effect in situations where the number of connections is small, i.e. 2 to 4. In these situations, a small oscillation around the optimal number of connections results in large variation in observed throughput. One way to alleviate this problem is to increase γ in order to tolerate larger queuing delay and hence absorb throughput fluctuations, at the expense of being less responsive. Another alternative is to use smaller packet size in order to reduce the "quantization effect" at the expense of (a) lower transmission efficiency and (b) the slower rate of convergence to the optimal number of connections. As will be shown in the next subsection, yet anther way to alleviate this quantization effect is to combine all the connections into one.

To examine the dynamics of MULTFRC system, we show throughput, packet loss rate, and the number of connections as a function of time for $p_w = 0.04$ in Figure 8. As seen, the throughput and the number of connections are quite stable; as expected, packet loss rate is around 0.04 and round trip time is low, and is in agreement with the results corresponding $p_w = 0.04$ in Figure 7(i). Similar results are obtained for other values of p_w.

In order to examine MULTFRC's performance as a function of p_w, we use MULTFRC with p_w initially set at 0.02. Then at 3000^{th} second, p_w is switched to 0.08, and at 6000^{th} second switched back to 0.02. Here, we artificially change p_w to see how MULTFRC adapts to the change in p_w. The throughput, packet loss rate, round trip time and the number of connections opened are shown in Figure 9. As seen, the number of connections increases from around 3 to around 7 as p_w switches from 0.02 to 0.08.

We have examined two sets of parameters (α, β) in these simulations, in order to demonstrate their influence on the rate of change of the number of connections. The first set is $\alpha = 1, \beta = 1$, and the second one is $\alpha = 6, \beta = 2$. As seen from Fig. 9(a), setting $\alpha = 1$ and $\beta = 1$ leads to a

[3] Note the round trip times for $p_w = 0$ are shown neither in Figure 7(i) or 7(ii) because they represent the channel error free case in which MULTFRC reduces to one TFRC connection.

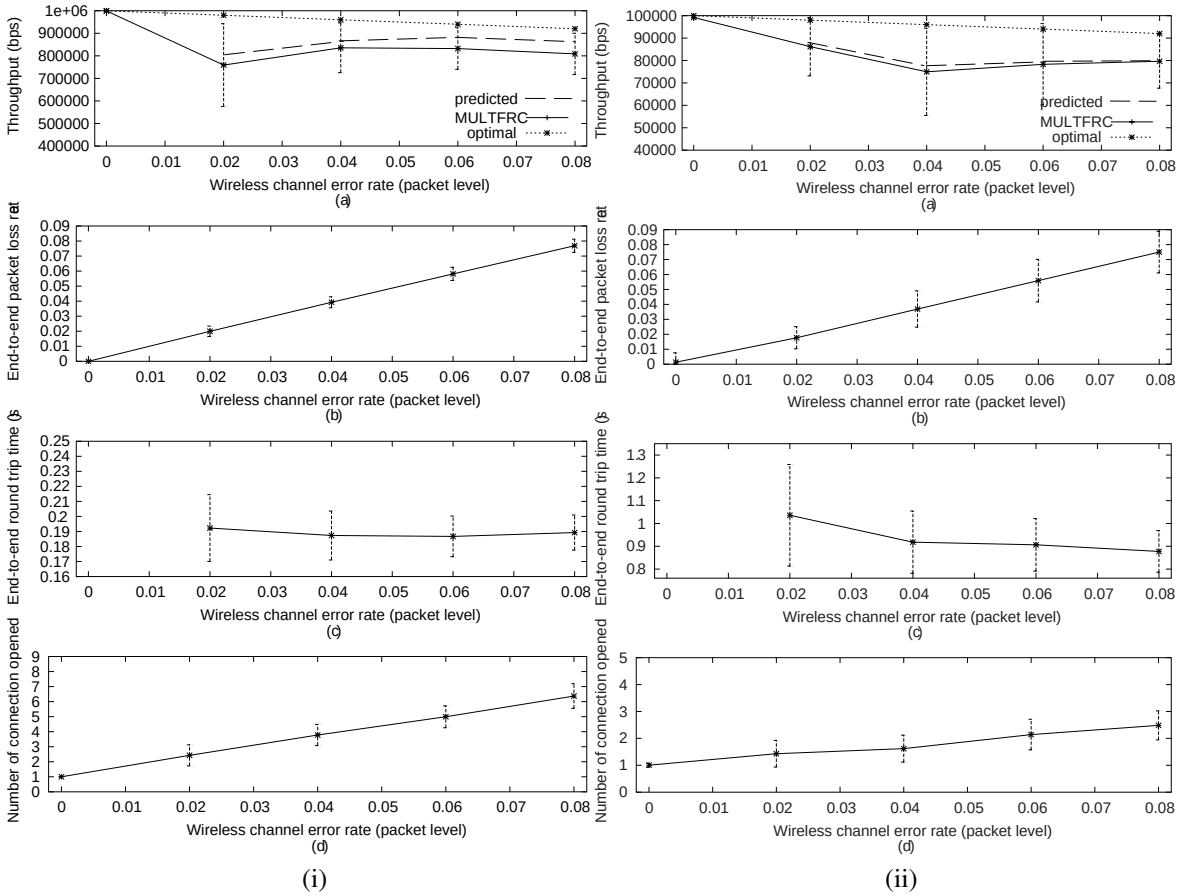


Fig. 7. NS-2 simulations for (i) $B_w = 1\ Mbps$, $RTT_{min} = 168\ ms$, and (ii) $B_w = 100\ kbps$, $RTT_{min} = 757\ ms$; (a) throughput, (b) end-to-end packet loss rate, (c) end-to-end round trip time, (d) number of connections, all as a function of packet level wireless channel error rate.

lower fluctuations on the number of connections, a low average round trip time, but a slower convergence of the number of connections. Specifically, it takes about 214 seconds for MULTFRC to increase the number of connections from 3 to 7, which are the optimal number of connections for $p_w = 0.02$ and $p_w = 0.08$, respectively. Similarly, the ramping down stage in which p_w changes from 0.08 to 0.02 takes about 87 seconds. On the other hand, as seen from Fig. 9(b), setting $\alpha = 6$ and $\beta = 2$ leads to a fast convergence of the number of connections, but a huge fluctuation on the number of opened connections, and a high average round trip time. Specifically, the increasing stage takes about 30 seconds, and the decreasing stage takes about 43 seconds. These observations confirm the intuition behind choosing values for α and β in the first paragraph of this subsection, and are in agreement with the analysis of rate of change of the number of connections shown in Section III-C.

As for actual experiments, we compare the performance of MULTFRC system and one TFRC connection in Table II with packet size of 1460 bytes. As seen, MULTFRC on average opens up 1.8 connectioins, and results in 60% higher throughput at the expense of a larger round trip time, and higher packet loss rate. Comparing the results of Tables I and II, we observe that MULTFRC achieves good performance as on average, it opens appropriate number of connections.

Table III shows packet loss details of MULTFRC for one of those 30 minutes long experiments. As expected, both the packet loss rate and burstness of the loss increase as the number of connections increases.

TABLE II

ACTUAL EXPERIMENTAL RESULTS FOR A MULTFRC SYSTEM OVER 1XRTT CDMA.

scheme	throughput (kbps)	rtt (ms)	packet loss rate	ave. # of conn.
one TFRC	54	1624	0.031	N/A
MULTFRC	86	2512	0.045	1.8

TABLE III

PACKET LOSS DETAILS OF MULTFRC

# of conn.	% of time	pkt loss rate	avg. burst err. len.	snd. dev.	max. burst length
one	24.6	0.015	2.86	3.43	7
two	60.1	0.047	2.41	3.63	10
three	15.4	0.083	3.25	9.93	11

C. All-In-One TFRC (AIO-TFRC)

There are two drawbacks associated with MULTFRC as seen from the simulations and actual experiments. First drawback has to do with bandwidth underutilization, and the second

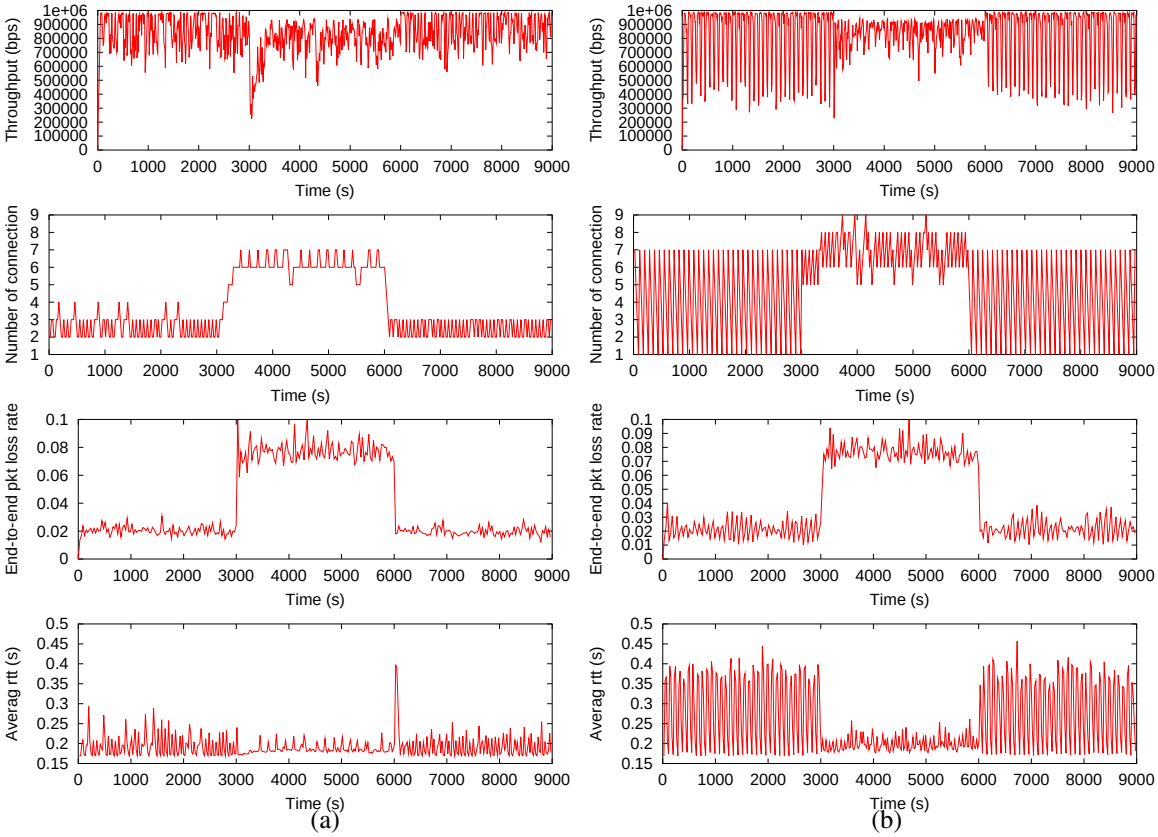


Fig. 9. NS-2 simulation results as p_w changes from 0.02 to 0.08 and back again, for case (a) $\alpha = 1, \beta = 1$, (b) $\alpha = 6, \beta = 2$. Plots from top to bottom indicate throughput, end-to-end packet loss rate, end-to-end RTT, and number of connections, all as a function of time.

one with implementation complexity. We will begin with utilization drawback. NS-2 simulations show that although MULTFRC performs reasonably well, there is still some gap between its throughput and the optimal. Specifically, MULTFRC achieves only 77% utilization for $p_w = 0.02$. This suboptimal performance has two causes. First one is the control behavior described in (9): as described, n is decreased when the full utilization of bottlenecks is detected, and is inversely increased until the next full utilization is detected. During this period, bottlenecks stay underutilized, resulting in suboptimal average throughput. It is impossible to remove this sub-optimality resulting from the control law without changing the law. The second reason for bandwidth underutilization is the "quantization effect" in MULTFRC whereby in practice the number of connections is forced to be an integer. This loss of granularity typically results in bandwidth underutilization. For example, if the optimal number of connections has been determined to be 1.5, then n is forced to take fractional values between 1 and 3, e.g. 1, 1.25, 1.45, 2.14, 1.14, ..., as dictated by (9). MULTFRC then quantizes n to the closest integer to oscillate between one and two, resulting in loss of throughput granularity. This effect can be eliminated by avoiding the quantization step.

The second drawback of MULTFRC is of a more practical nature. Operating multiple connections in one application could potentially consume too much system resources. For example, each TFRC connection uses a different port to send out data packets, carries out individual feedback process, and updates the loss event rate and RTT even though they are highly correlated for these TFRC connections. Clearly, there is unnecessary overhead associated with operating multiple connections, in terms of computation, processing power, memory, and ports, particularly for today's low power, resource-limited handheld devices.

We can propose an alternative to MULTFRC, called All-In-One TFRC (AIO-TFRC), in order to address the two drawbacks of MULTFRC, while retaining the same control law for n as in MULTFRC [38]. AIO-TFRC achieves these goals by creating one connection whose throughput is equivalent to that of the optimal number of TFRC connections even though the optimal number could be non-integer. It does so by measuring round trip times, adjusting the number of virtual connections n based on the measurements according to (9), and then controlling the sending rate of the only physical connection to be n times that of one TFRC's, using the bandwidth filtered loss detection (BFLD) technique from [39]. NS-2 simulations show that AIO-TFRC achieves similar throughput as MULTFRC in high packet loss rate situation, but better throughput than MULTFRC at the low packet loss rate scenarios, as shown in Fig. 10. For example, when $p_w = 0.02$, AIO-TFRC achieves 95% utilization of the wireless bandwidth, while MULTFRC's utilization is only 77%. Therefore, by avoiding the "quantization effect", AIO-TFRC achieves better throughput performance than MULTFRC.

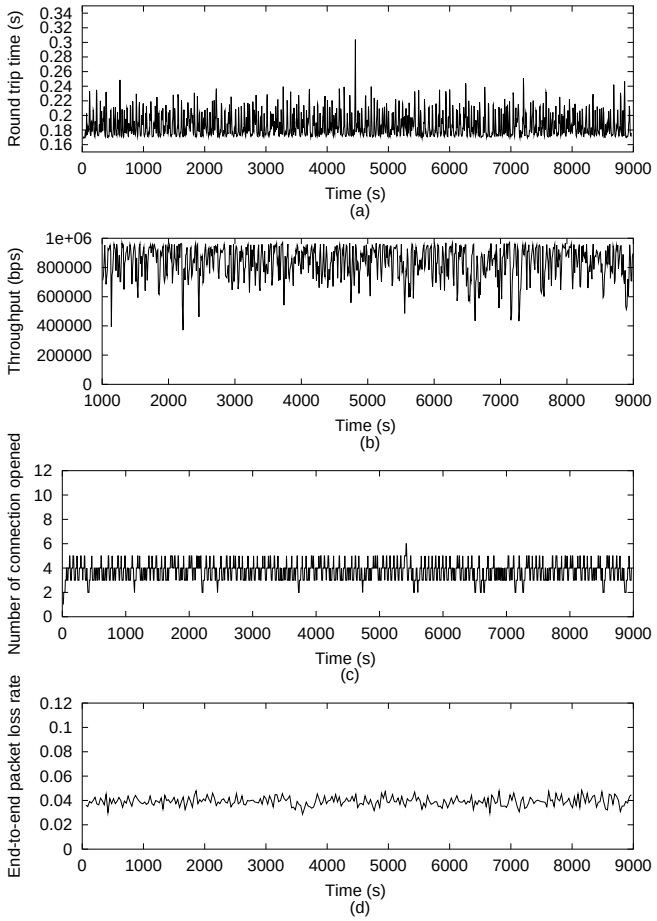


Fig. 8. NS-2 simulations for $B_w = 1\ Mbps$ and $p_w = 0.04$; (a) end-to-end round trip time, (b) throughput, (c) end-to-end packet loss rate, (d) number of connections, all as a function of time.

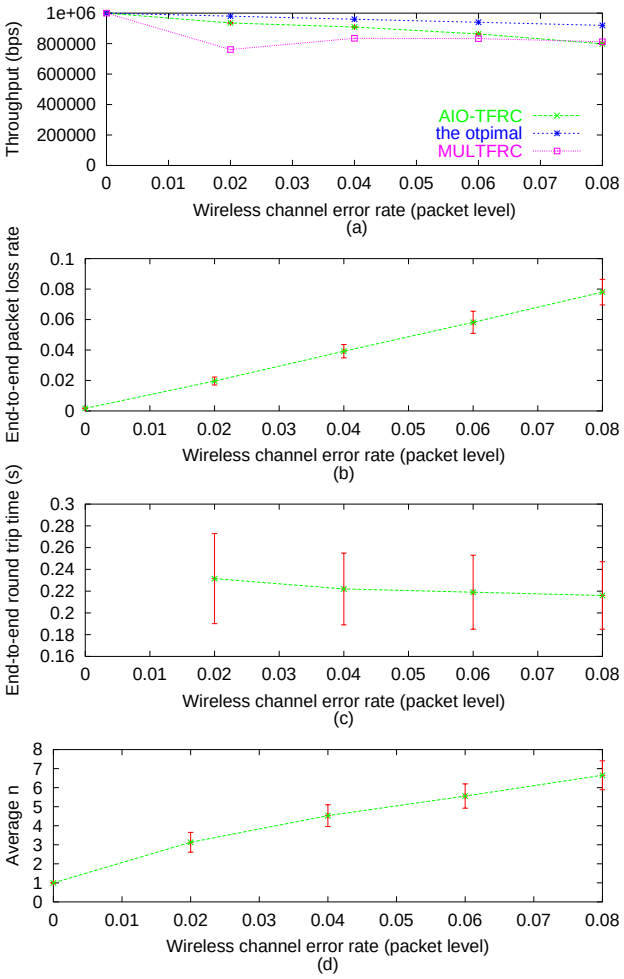


Fig. 10. NS-2 simulations for Bw = 1 Mbps and RTTmin = 168 ms; (a) throughput, (b) end-to-end packet loss rate, (c) end-to-end RTT, (d) number of connections, all as a function of packet error rate on the wireless channel.

AIO-TFRC is fundamentally similar to MULTFRC as they share the same theoretical analysis and design insights; as they only differ in implementation details, we will not go into the details of AIO-TFRC here [38].

D. Fairness between MULTFRC and TCP

To investigate the fairness of MULTFRC, we carry out NS-2 simulations based on the "dumbbell" topology shown in Fig. 11. Senders are denoted by $si, i = 1, \ldots, 16$, and receivers are denoted by $di, i = 1 \ldots, 16$. We investigate two types of fairness: the inter-protocol fairness between MULTFRC and TCP, and the intra-protocol fairness within MULTFRC.

The intra-protocol fairness is defined as the fairness between MULTFRC flows. In our simulations, we run MULTFRC on all 16 sender-receiver pairs shown in Fig. 11 for 5000 seconds, and compare their throughput. MULTFRC is said to be intra-protocol fair if all receivers get the same throughput. The fairness ratios for $p_w = 0.01$ and $p_w = 0.04$ are shown in Table IV. The fairness ratio is defined as receivers' throughput divided by the average throughput; the closer to one, the more fair the MULTFRC system is. As seen, the fairness ratio is close to one, indicating MULTFRC flows are fair to each other, at least in this simulation setting. The bandwidth utilization ratios are 98% for $p_w = 0.01$ and 99% for $p_w = 0.04$.

To show that a new MULTFRC connection consumes its fair share of bandwidth, we carry out the simulations for the same topology shown in Fig. 11 with $p_w = 0.01$. Eight MULTFRCs are spawned at $t = 0$, and 8 more are added at $t = 2000$. The average throughput for two sets of values for α and β are shown in Fig. 12, to demonstrate their effect on rate of convergence. As seen, the later connections share the bandwidth fairly with the former ones. As seen in Fig. 13, the convergence time for the average throughput is about 180 seconds for case $\alpha = \beta = 1$, and 90 seconds for case $\alpha = \beta = 2$. Although larger values of α and β lead to a faster adaptation speed, the variance on the number of connections, as well as the throughput and rtt, is also slightly higher. These observations are in agreements with those made from Fig. 9.

The inter-protocol fairness is defined as the fairness between MULTFRC and TCP[4]. In our simulations, we run MULTFRC on the first 8 sender-receiver pairs, i.e. $(si, di), i = 1, \ldots, 8$,

[4]We choose TCP SACK implementation in simulations.

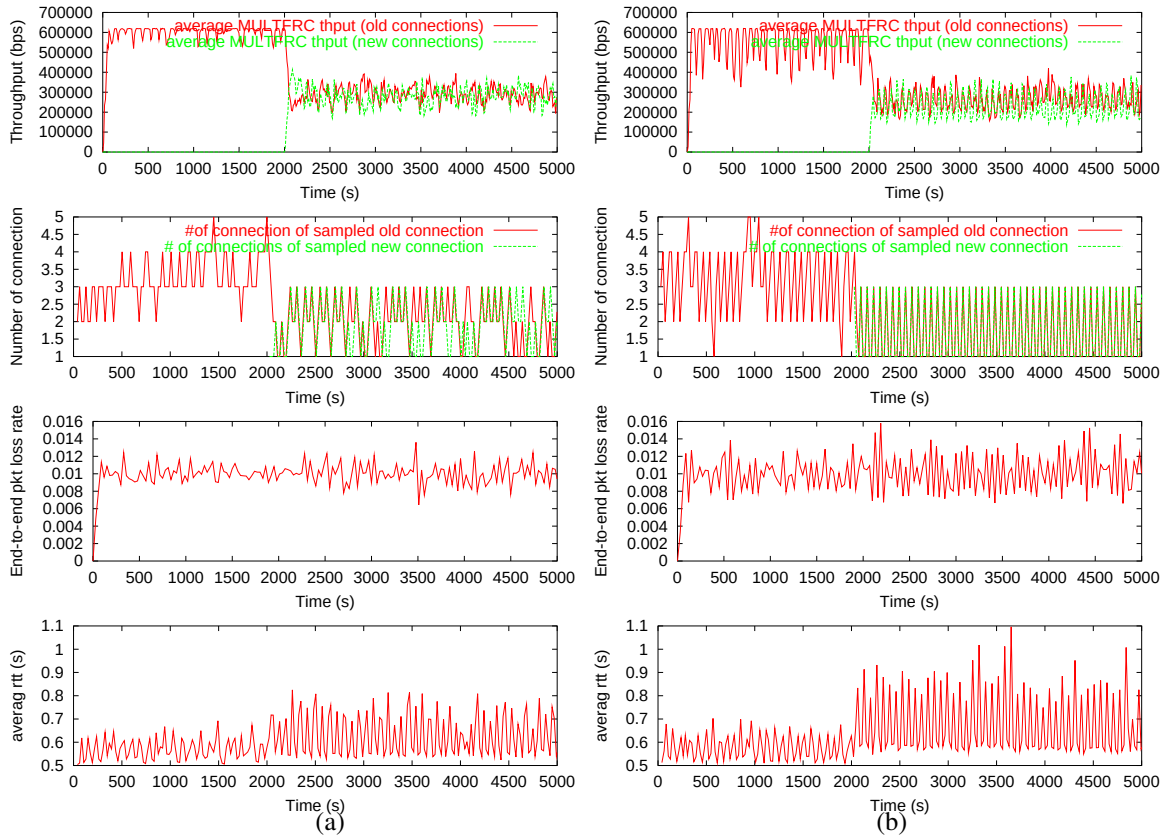


Fig. 12. MULTFRC intra-fairness for case (a) $\alpha = \beta = 1$, (b) $\alpha = \beta = 2$. As seen, new MULTFRC connections can attain fair share of bandwidth. Plots from top to bottom correspond to throughput, end-to-end packet loss rate, end-to-end RTT, and number of connections, all as a function of time.

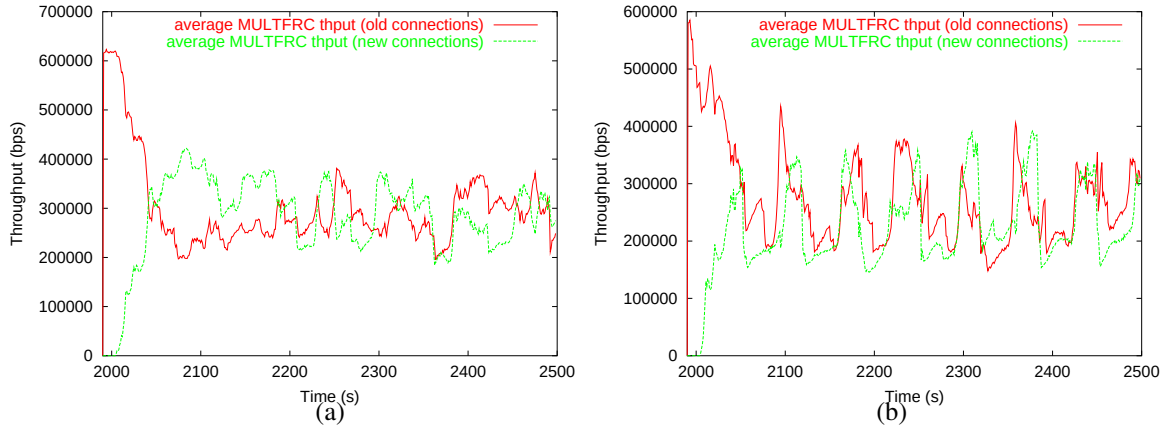


Fig. 13. MULTFRC intra-fairness for case (a) $\alpha = \beta = 1$, (b) $\alpha = \beta = 2$. As seen, new MULTFRC connections can attain fair share of bandwidth. Both plots show throughput as a function of time from $t = 1990$ to $t = 2500$ second.

and TCP on the remaining 8 sender-receiver pairs shown in Fig. 11; each session lasts 5000 seconds, and we compare their throughput for $p_w = 0.01$ and $p_w = 0.02$. Under the simulation settings, each MULTFRC consumes more bandwidth than one TCP under full utilization. This is because in this case, the wireless channel error rate is large enough to make the number of virtual connections of each MULTFRC to be larger than one. Hence, it is meaningless to define the fairness between MULTFRC and TCP as having the same throughput. As such, in our simulations, we define MULTFRC to be fair to TCP if it does not result in a decrease in TCP's throughput. Specifically for our simulations, it implies TCP retains the same throughput whether or not it coexists with MULTFRC under the same network setting. The throughput of MULTFRC and TCP, as well as the total bandwidth utilization ratios for the setup shown in Fig. 11, are shown in Table V for two scenarios: (a) 8 MULTFRC coexisting with 8 TCP connections, (b) 16 TCP connections. Fig. 14 also shows the dynamics of throughput, packet loss rate, RTT, and the number of virtual connections n. Comparing MULTFRC+TCP with TCP-alone, we see the for-

TABLE V
SIMULATION RESULTS FOR FAIRNESS BETWEEN MULTFRC AND TCP.

settings	8 MULTFRC + 8 TCP			16 TCP	
	ave. thput. (MULTFRC) (kbps)	ave. thput. (TCP) (kbps)	utilization (%)	ave. thput. (TCP) (kbps)	utilization (%)
p_w=0 γ=0.2	311.84	313.16	100	312.46	100
p_w=0.01 γ=0.2	446.73	165.67	99.0	200.168	65
p_w=0.01 γ=0.1	402.73	180.37	94.2	200.168	65
p_w=0.02 γ=0.2	488.53	122.08	99.7	139.674	46
p_w=0.02 γ=0.1	461.74	131.10	96.8	139.674	46

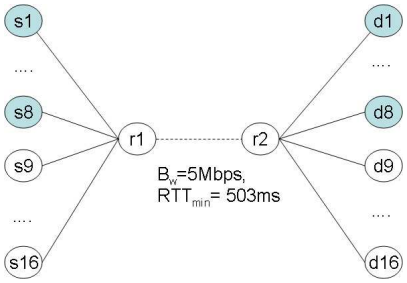


Fig. 11. The simulation topology for MULTFRC's fairness evaluation.

TABLE IV
SIMULATION RESULTS FOR INTRA-PROTOCOL FAIRNESS OF MULTFRC.

recver	fairness ratio p_w=0.01	fairness ratio p_w=0.04	recver	fairness ratio p_w=0.01	fairness ratio p_w=0.04
d1	1.07	0.93	d9	1.03	0.92
d2	1.03	0.94	d10	1.05	1.07
d3	1.04	1.14	d11	0.92	0.89
d4	0.95	1.00	d12	1.03	0.90
d5	0.89	1.08	d13	1.05	1.12
d6	0.90	0.96	d14	0.96	1.14
d7	1.10	0.87	d15	1.02	0.92
d8	1.02	1.17	d16	0.95	0.91

mer has a much higher utilization of the wireless bandwidth at the expense of lower TCP throughput. A careful examination into the traces and statistics reveals that this throughput drop is mainly caused by the higher RTT for MULTFRC+TCP as compared with TCP-alone. For example, for $p_w = 0.01$ and $\gamma = 0.2$, MULTFRC+TCP experiences around 0.58 seconds RTT, while TCP-alone only experiences 0.5 seconds RTT, i.e. the propagation delay. As TCP's throughput is known to be inversely proportional to RTT, the 16% increase in the RTT roughly explains the 17% decrease in the TCP's throughput shown in first row in Table V.

This increase in the RTT is, by design, a consequence of MULTFRC controlling n according to (9). As n is only decreased after the queuing delay exceeds the threshold γrtt_min, round trip time is increased when MULTFRC increases n to achieve full utilization. One way to address this problem is to use a smaller value for γ, in order to reduce the increase in the RTT, and hence minimize the TCP's throughput drop. However, smaller values of γ also results in lower bandwidth utilization due to increased sensitivity of MULTFRC to RTT measurements. As shown in Table V, $\gamma = 0.1$ results in a smaller drop in the TCP's throughput than $\gamma = 0.2$.

Clearly, there are situations in which MULTFRC ends up opening exactly one TFRC connection, and as such its performance is similar to one connection case. An example would be MULTFRC competing for bandwidth with TCP on wired networks. In that case, the fairness between MULTFRC and TCP is reduced to the fairness between TFRC and TCP, which has been well explored in [1]. This is because in this situation, MULTFRC only opens one TFRC connection. To verify that, we set $p_w = 0$ and carry out the above simulation with MULTFRC and TCP sharing bandwidth with each other. The results, shown in the first row in Table V, clearly validate this claim.

E. Video streaming simulations

To evaluate the performance of MULTFRC in video streaming applications, we simulate streaming of a 60 second long video clip through a channel, with throughput trace corresponding to one of the the traces obtained from actual experiments over 1xRTT CDMA as described in Section IV-B. Our goal is to compare the quality of video streaming achievable using one TFRC connection with that of MULTFRC.

We encode 300 frames of *news.cif* sequence using MPEG-4 at bit rates varying from 50kps to 100 kbps as controlled by TMN-5 [40]. The frame rate is 10 frame per second and hence the duration of the video clip is 30 seconds; the I-frame refresh rate is once every fifteen frames. The coded video bit stream is packetized with fixed packet size of 760 bytes. The packets are then protected using Reed-Solomon (RS) codes with different protection levels for one TFRC and

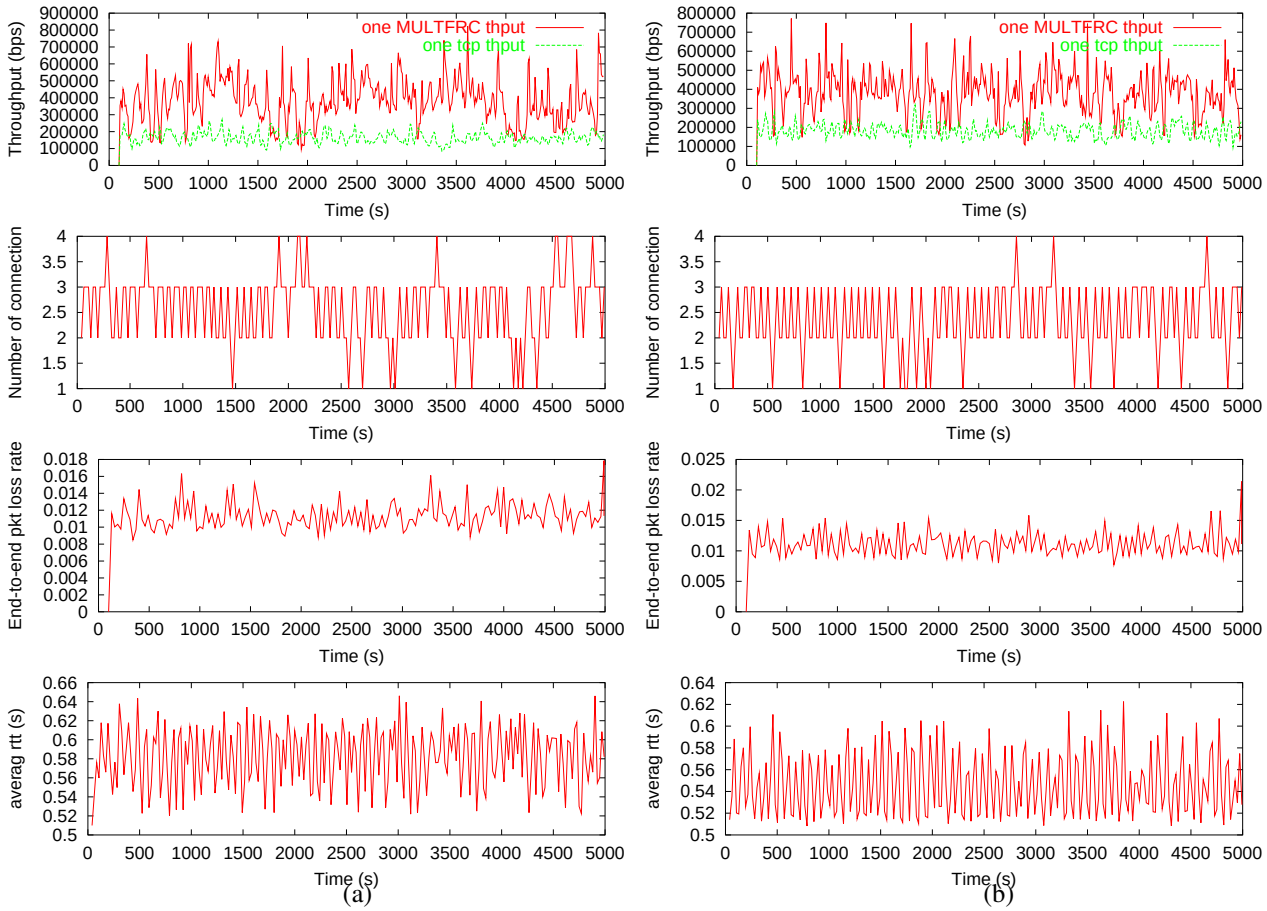


Fig. 14. NS-2 simulation results for the case (a) $p_w = 0.01, \gamma = 0.2$ and (b) $p_w = 0.01, \gamma = 0.1$. Plots from top to bottom correspond to the dynamics of throughput, number of connections, end-to-end packet loss rate, and end-to-end RTT, all as a function of time.

MULTFRC. This is because packet loss statistics are different in the two cases. Specifically, the statistics of 30 minutes long trace indicates the longest burst loss to be 6 packets long for one TFRC and 11 packets long for MULTFRC. Thus, we apply RS(56,50) to one TFRC case, and RS(61,50) to MULTFRC case in order to sufficiently protect packets in both cases.

The RS-coded packets are then passed through channels simulated using one TFRC, and MULTFRC packet level traces each lasting 70 seconds, selected from the 30 minutes long actual experiments described in Section IV-B. The throughput and packet loss details for a 70 second long segment of one TFRC and MULTFRC connections are shown in Fig. 16. As Seen, both the throughput and the packet loss rate are higher for MULTFRC than for one TFRC case.

The receiver decodes the received RS-coded packets and stores the MPEG-4 bit streams into a playback buffer. In this simulation, we fill the buffer with 10 seconds worth of data before starting the MPEG-4 decode and display process. The playback rate is fixed at 10 frames per second, and hence decoding process is stopped and the display is frozen whenever the playback buffer is empty.

To show the efficiency of MULTFRC, we compare the playback buffer occupancies of MULTFRC and one TFRC for several bit rates in Fig. 17. As seen, TFRC can only sustain a 50 kbps clip, while MULTFRC can sustain video streaming up to 90 kbps without freezes and decode buffer starvation; hence, MULTFRC can achieve higher visual quality, despite the fact that it needs stronger FEC to combat the higher packet loss rate.

V. Conclusions and Discussions

In this paper, we proposed an end-to-end rate control scheme for wireless streaming that achieves both high throughput and low packet loss rate, without having to modify network infrastructure or protocols. Our proposed strategy is based on increasing the number of connections, and selecting proper packet size when necessary. We developed a practical algorithm called MULTFRC to implement our basic approach. NS-2 simulations and actual experiments over 1xRTT CDMA data network were used to show the effectiveness of our approach. Video simulations demonstrate that it is possible to apply MULTFRC to sustain video streaming at higher bit rates, despite the fact that it needs stronger FEC to combat the higher packet loss rate. The simulations also shows that MULTFRC is relatively fair to TCP, and fair to itself.

Even though B_w and p_w are assumed to be constant in our analysis, in some networks such as wireless Local Area Networks (WLAN) and CDMA networks, B_w and p_w might be time varying or even change in a correlated fashion.

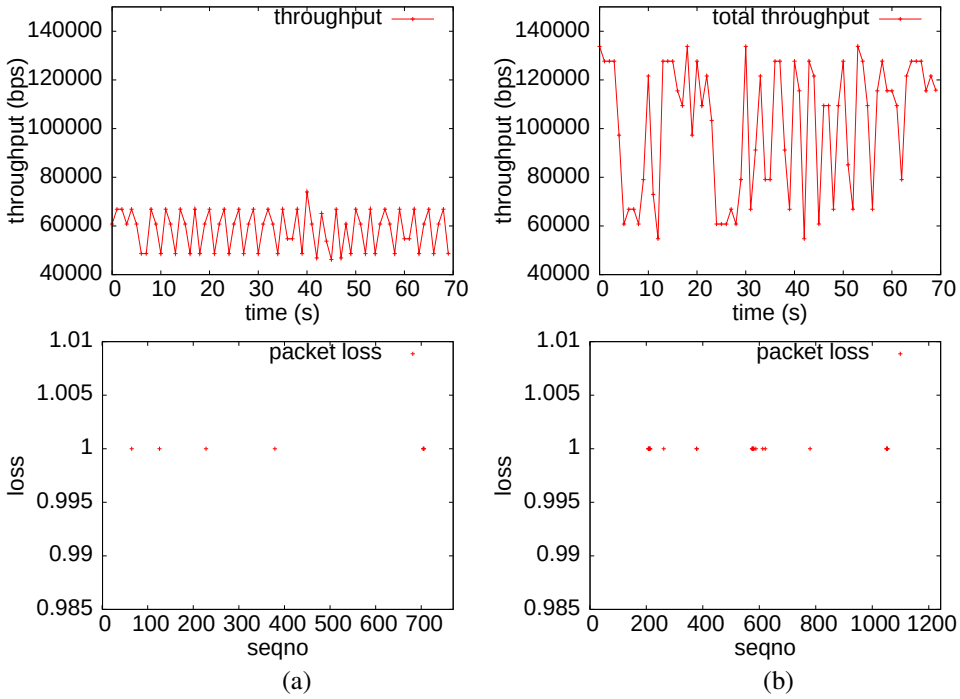


Fig. 16. Throughput and packet loss details for (a) one TFRC; (b) MULTFRC.

Nevertheless, as long as the necessary and sufficient condition in (4) is satisfied and the wireless channel is underutilized, our proposed MULTFRC approach opens an appropriate number of connections to achieve full utilization. The only issue in these time varying situations is rate of convergence to the optimal number of connections. Our experimental results in this paper have verified that in the long term, the convergence rate of our approach is not an issue in CDMA network.

Compared to other schemes such as [6], [7]that use delay or round trip time variation to infer congestions, although MULTFRC also makes use of the round trip time variation, , it differs from them in the following aspects: first MULTFRC measures the round trip time variation over a large time window, while the other schemes measure the round trip time variation instantaneously, resulting in a more noisy measurement; second, MULTFRC uses the measured variation to adapt the number of opened connections in order to adapt the rate, while the other schemes use it to differentiate between congestion loss and wireless loss.

As discussed earlier, AIO-TFRC has some advantages over MULTFRC. Specifically, it only requires opening one connection, rather than multiple connections, and it alleviates the quantization effect associated with MULTFRC. However, if we were to extend MULTFRC to MUL-TCP, and AIO-TFRC to AIO-TCP, respectively, one can easily see that the required application layer code needed to implement MUL-TCP is considerably simpler than that of AIO-TCP. This is because MUL-TCP only needs to concern itself with determining the optimal number of TCP connections to be opened. AIO-TCP, on the other hand, would have to implement a substantial portions of the TCP protocol itself in the application layer. Since TCP is the dominate protocol on the internet today, even for streaming applications, we believe that from a practical point of view, a MUL-TCP type protocol has a higher chance of being adopted than AIO-TCP.

Future work will be focused on considering the stability issues and examining the performance when both the number of connections and the sending rate of each connections are changing dynamically in a network, i.e. multiple MULTFRC connections sharing wireless or wired links with/without multiple TCP connections. Applying the idea to improve the performance of TCP over wireless would also be a future direction.

REFERENCES

[1] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000, pp. 43–56.

[2] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Trans. Networking*, no. 4, pp. 458 – 472, Aug. 1999.

[3] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving tcp performance over wireless links," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[4] H. Balakrishnan and R. Katz, "Explicit loss notification and wireless web performance," in *Proc. of IEEE Globecom Internet Mini-Conference*, Nov. 1998.

[5] D. Barman and I. Matta, "Effectiveness of loss labeling in improving tcp performance in wired/wireless networks," in *Proc. of the 10th ICNP*, Washington, DC, USA, 2002, pp. 2–11.

[6] S. Biaz and N. H. Vaidya, "Discriminating congestion loss from wireless losses using inter-arrival times at the receiver," in *Proc. of IEEE Symposium on Application-specific System and Software Engr. and Techn.*, Richardson,TX, USA, Mar. 1999, pp. 10–17.

[7] N. Samaraweera, "Non-congestion packet loss detection for tcp error recovery using wireless links," *IEE Proceedings of Communications*, vol. 146, no. 4, p. 222C230, Aug. 1999.

[8] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda, "Achieving moderate fairness for udp flows by pathstatus classification," in *Proc. of 25th Annual IEEE Conf. on Local Computer Networks*, Tampa,FL, USA, Nov. 2000, p. 252C261.

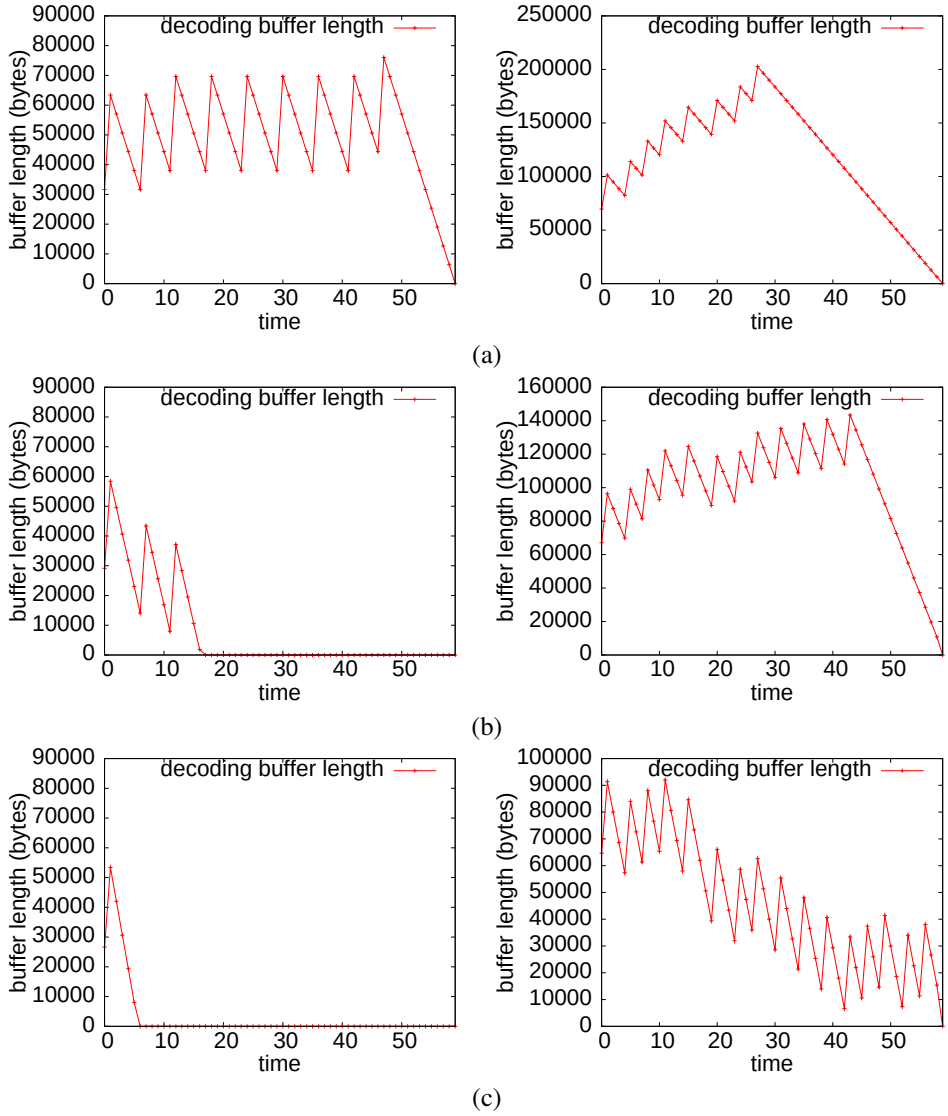


Fig. 17. Playback buffer occupancies for one TFRC (left) and MULTFRC (right): the streaming bit rate is at (a) 50kbps; (b) 70kbps; (c) 90kbps.

[9] C. JA and A. P., "Congestion or corruption? a strategy for efficient wireless tcp sessions," in *Proc. of IEEE Symposium on Computers and Communications*, Los Alamitos, CA, USA, 1995, pp. 262–268.

[10] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "A wireless transmission control protocol for cdpd," in *Proc. of IEEE Wireless Communications and Networking Conference*, Piscataway, NJ, USA, Jan. 1999, pp. 953–957.

[11] ——, "Wtcp: a reliable transport protocol for wireless wide-area networks," *Wireless Networks*, no. 2-3, pp. 301–316, 2002.

[12] W. Ding and J. A, "A new explicit loss notification with acknowledgment for wireless tcp," in *Proc. of 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Piscataway, NJ, USA, 2001, pp. B–65–9.

[13] C. CF and M. M., "Improving tcp over wireless through adaptive link layer setting," in *Proc. of IEEE Global Telecommunications Conference*, Piscataway, NJ, USA, 2001, pp. 1766–1770.

[14] J.-H. Choi, S.-H. Yoo, and C. Yoo, "A flow control scheme based on buffer state for wireless tcp," in *Proc. of the 4th International Workshop on Mobile and Wireless Communications Network*, Piscataway, NJ, USA, 2002, pp. 592–596.

[15] K. Ratnam and I. Matta, "Wtcp: an efficient mechanism for improving wireless access to tcp services," *International Journal of Communication Systems*, no. 1, pp. 47–62, Feb. 2003.

[16] J. Rendon, F. Casadevall, and J. Carrasco, "Wireless tcp proposals with proxy servers in the gprs network," in *Proc. of 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Piscataway, NJ, USA, 2002, pp. 1156–1160.

[17] Y. Yang, H. Zhang, and K. R., "Channel quality based adaptation of tcp with loss discrimination," in *Proc. of IEEE Global Telecommunications Conference*, Piscataway, NJ, USA, 2001, pp. 2026–2030.

[18] J.-J. Lee, F. Liu, and K. C-CJ, "End-to-end wireless tcp with non-congestion packet loss detection and handling," in *Proc. of the SPIE*, San Jose, USA, Jan. 2003, pp. 104–113.

[19] S. Cen, P. Cosman, and G. Voelker, "End-to-end differentiation of congestion and wireless losses," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 703–717, 2003.

[20] J. Liu, I. Matta, and M. Crovella, "End-to-end inference of loss nature in a hybrid wired/wireless environment," in *Proc. WiOpt*, 2003.

[21] T. eun Kim, S. Lu, and V. Bharghavan, "Improving congestion control performance through loss differentiation," in *Proc. ICPP Workshop*, 1999, pp. 140–145.

[22] C. Parsa and J. Garcia-Luna-Aceves, "Improving tcp congestoin control over internet with heterogeneous media," in *Proc. ICNP*, 1999, pp. 213–221.

[23] J. Tang, G. Morabito, I. F. Akyildiz, and M. Johnson, "Rcs: A rate control scheme for real-time traffic in networks with high bandwidth-delay products an high bit error rates," in *Proc. IEEE INFOCOM*, Alaska, USA, Apr. 2001, pp. 114–122.

[24] G. Yang, M. Gerla, and M. Y. Sanadidi, "Adaptive video streaming in presence of wireless errors," in *Proc. ACM MMNS*, San Diego, USA, Jan. 2004.

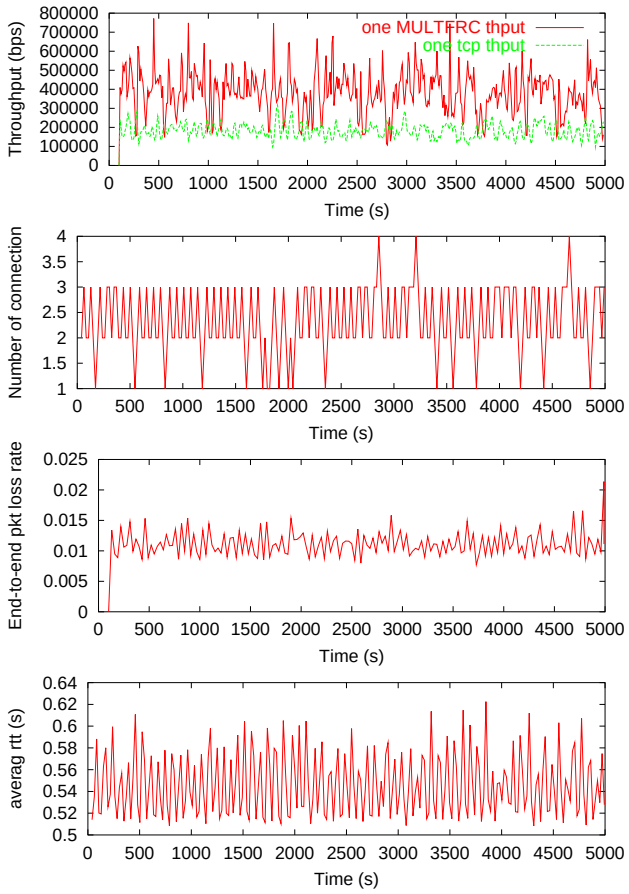


Fig. 15. NS-2 simulation results for the case $p_w = 0.01, \gamma = 0.1$: (a)throughput, (b) number of connections , (c) end-to-end packet loss rate, (d) end-to-end RTT, all as a function of time.

[25] F. Yang, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "End-to-end tcp-friendly streaming protocol and bit allocation for scalable video over mobile wireless internet," in *Proc. IEEE INFOCOM*, Hongkong, China, Mar. 2004.

[26] Ö. B. Akan and I. F. Akyildiz, "Arc: the analytical rate control scheme for real-time traffic in wireless networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 4, pp. 634–644, 2004.

[27] L. S. Brakmo and L. L. Peterson, "Tcp vegas: end-to-end congestion avoidance on a global internet," *IEEE J. Select. Areas Commun.*, no. 8, pp. 1465–1480, Oct. 1995.

[28] S. Floyd, "Tcp and explicit congestion notification," *ACM Computer Communication Review*, pp. 10–23, Oct. 1994.

[29] J. Crowcroft and P. Oechslin, "Differentiated end to end internet services using a weighted proportional fair sharing tcp," *ACM Computer Communication Review*, vol. 28, no. 3, July 1998.

[30] Netants - fast download manager. [Online]. Available: http://www.netants.com

[31] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *ACM Computer Communication Review*, no. 3, July 1997.

[32] J. Mahdavi and S. Floyd. (1997, Jan.) Tcp-friendly unicast rate-based flow control. Technical note sent to end2end-interest mailing list. [Online]. Available: http://www.psc.edu/networking/papers/tcp_friendly.html

[33] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: A simple model and its empirical validation," in *Proc. ACM SIGCOMM*, 1998, pp. 303 – 314.

[34] S. Biaz and N. H. Vaidya, "Distinguishing congestion losses from wireless transmission losses: a negative result," in *Proc. of the Seventh International Conference on Computer Communications and Networks (IC3N)*, New Orleans, USA, Oct. 1998.

[35] Network simulation version 2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[36] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic behavior of slowly-responsive congestion control algorithms," in *ACM SIGCOMM 2001*, San Diego, CA, Sept. 2001.

[37] Traceroute. [Online]. Available: http://www.traceroute.org/

[38] M. Chen and A. Zakhor, "Aio-tfrc: A light-weighted rate control scheme for streaming over wireless," in *Proc. of IEEE WirelessCom Symposium on Multimedia over Wireless 2005*, June 2005.

[39] D. E. Ott, T. Sparks, and K. Mayer-Patel, "Aggregate congestion control for distributed multimedia applications," in *Proc. IEEE INFOCOM*, Hongkong, China, Mar. 2004.

[40] *TMN (H.263) encoder/decoder, version 2.0, tmn (h.263) codec*, T. Research Std., 1996.

PLACE PHOTO HERE

Minghua Chen received the M.S. and B.Eng. degrees in Electronic Engineering from Tsinghua University in 2001 and 1999, respectively. Since 2001, he has been with Department of Electrical Engineering and Computer Science in University of California at Berkeley, where he currently is pursuing his Ph.D degree. He received a Pao Family fellowship in 2001, a Management of Technology in China Fellowship in 2004, all from U.C. Berkeley. He is co-author of the book IPv6 Principle and Practice (People's Posts and Telecommunication Publishing House, 2000). His research interests are in Flow control in wireless network, video streaming over wireless, digital signal processing, and wireless communications, with current emphasis on flow control over wireless and application layer implementation.

PLACE PHOTO HERE

Avideh Zakhor received a B. S. degree from California Institute of Technology, Pasadena, and S. M. and Ph. D. degrees from Massachusetts Institute of Technology, Cambridge, all in electrical engineering, in 1983, 1985, and 1987 respectively. In 1988, she joined the Faculty at U. C. Berkeley where she is currently Professor in the Department of Electrical Engineering and Computer Sciences. Her research interests are in the general area of image and video processing, multimedia communication, and 3D modeling. Together with her students, She has won a number of best paper awards, including the IEEE Signal Processing Society in 1997, IEEE Circuits and Systems Society in 1997 and 1999, international conference on image processing in 1999, and Packet Video Workshop in 2002. She holds 5 U.S. patents, and is the co-author of the book, "Oversampled A/D Converters" with Soren Hein.

Prof. Zakhor was a General Motors scholar from 1982 to 1983, was a Hertz fellow from 1984 to 1988, received the Presidential Young Investigators (PYI) award, and Office of Naval Research (ONR) young investigator award in 1992. From 1998 to 2001, she was an elected member of IEEE Signal Processing Borad of Governers. In 2001, she was elected as IEEE fellow. She received the Okawa Prize in 2004.

She co-founded OPC technology in 1996, which was later by Mentor Graphics (Nasdaq: MENT) in 1998, Truvideo in 2000, and Urban Scan in 2005.