# Rate Control for Streaming Video over Wireless

*Minghua Chen* and *Avideh Zakhor*
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, CA 94720
{minghua, avz}@eecs.berkeley.edu

***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 and experiments over 1xRTT CDMA wireless data network 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 for 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. For streaming over wireless where packets can be corrupted by wireless channel errors at the physical layer, rate control is still an open issue. Neither TFRC nor TCP can distinguish between packet loss due to buffer overflow and that due to bit errors. Both have been designed to deal with buffer overflow in wired networks and as such, treat any loss as a sign of congestion. Consequently, there have been a number of efforts to improve the performance of TCP over wireless [5][6][7][8][9][10]. For example, Snoop is a TCP-AWARE link layer approach which suppresses acknowledgement packets (ACK) from the TCP receiver, and does local retransmissions when a packet is corrupted by wireless channel errors [5]. Explicit Loss Notification (ELN) can also be applied to notify the TCP sender when a packet loss is caused by wireless channel errors rather than congestion [6]. End-to-end statistics can be used to help detect congestion when a packet loss happens [7][8][9][10]. For example, by examining trends in the one-way delay variation, one could interpret loss as a sign of congestion if one-way delay is increasing, and a sign of wireless channel error otherwise. All these methods 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. The disadvantages of these schemes are that they need modifications to network infrastructure or protocols.

Similar to the TCP over wireless, possible solutions for rate control for streaming over wireless include hiding end-hosts from packet loss caused by wireless channel error, or providing end-hosts the ability to distinguish between packet loss caused by congestion and that due to wireless channel error.

Cen et. al. present an end-to-end based approach to facilitate streaming over wireless [11]. 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, examining 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 [11]; it also requires modifications to congestion control mechanism in protocol.

Other schemes such as [7][8][9][10] that use end-to-end statistics to detect congestion, can be also combined with TFRC to achieve 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. The disadvantage of this approach is that congestion detection schemes based on statistics are not accurate enough, and require some modifications to the congestion control part of the protocol stack.

Another way to achieve rate control for streaming over wireless is to insert a TFRC-aware Snoop-like module, similar to [5], into the network to do local retransmissions when packets are corrupted by wireless channel errors, and to apply TFRC on end-hosts. In this way, streaming rate is not affected by wireless channel errors. The advantage of this approach is its simplicity, and robustness to unpredictable wireless channel conditions. The disadvantages are as follows. First, it requires modifications to the network infrastructure. Second, Snoop-like module does not work when the forward route is different from the reverse route. This is because Snoop can not block ACK packets sent from the receiver to the sender when doing

local retransmissions; hence the sender interprets the packet loss caused by wireless channel error as a sign of congestion, and reduces the sending rate unnecessarily.

ELN [6] can also be applied to streaming over wireless. By setting ELN bits on consecutive packet headers when packets are lost due to wireless channel errors, the end-host can differentiate between congestion and channel errors. In this case, TFRC can take into account only the packet loss caused by congestion when adjusting the streaming rate. Fundamentally, this achieves the same objective as Snoop-like module does, i.e. it enables TFRC not to respond to packet loss caused by wireless channel errors. The disadvantage of ELN approach is that it also needs modifications to the network infrastructure.

Other similar works, but not related to our approach include MULTCP [15] and NetAnts [16]. 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.

In this paper, we show that using one TFRC connection in wireless streaming applications results in under-utilization 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.

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 and actual experimental results 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 have 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 nodes 1 and 2, denoted by $p_c^1$ and $p_c^2$, respectively. We use $p_c^{i1}$ and $p_c^{i2}$ to represent the packet loss rate at node $i$ caused by streaming traffic itself, i.e. self congestion, and by cross traffic, i.e. cross congestion, respectively. Thus we have $p_c^i = p_c^{i1} + p_c^{i2}$. 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 want to analyze. Third, it makes our analysis easy to understand and evaluate. We believe the analysis and solution are also applicable to other general scenarios.

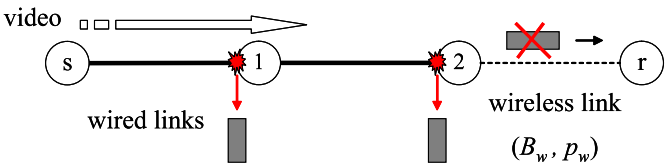



Fig. 1. Typical scenario for streaming over wireless.

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 self congestion at node 1, i.e. $p_c^{11} = 0$.
2) There is no self congestion at node 2 , i.e. $p_c^{21} = 0$, if an only if the wireless bandwidth is underutilized, i.e. $T(1-p) \leq B_w(1-p_w)$. Also, $p_c^{21} = 0$ implies no queuing delay due to self congestion, and hence results in the round trip time for a given route to be at a minimum, i.e. $RTT_{min}$. Thus, this assumption can be restated as follows: for a given route, $rtt = RTT_{min} \Leftrightarrow T(1-p) \leq B_w(1-p_w)$. This in turn implies that if $T(1-p) > B_w(1-p_w)$ then $rtt \geq RTT_{min}$.
3) The packet loss caused by cross traffic is independent of the streaming rate at the sender , i.e. $p_c^{12}$ and $p_c^{22}$ are constant and independent of $T$.
4) $B_w$ and $p_w$ are assumed to be constant.
5) The packet loss caused by wireless channel error is assumed to be random and stationary.

6) Our objective is to optimize long-term streaming throughput and packet loss rate performance rather than short term behavior; this is because the short term behavior is taken care of by TFRC.
7) Packet size $S$ for all connections of one application are the same, unless otherwise stated.
8) We assume one TFRC connection does not fully utilize $B_w$, otherwise it already achieves optimal performance, and no improvement is to be expected.
9) For simplicity, the backward route is assumed to be error-free and congestion-free.

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 Under-utilization*

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 [13] and 1.3 [12], depending on the particular derivation of Eqn. (1). Although this model has been refined to improve accuracy [1][3], 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].

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 $p_c^i$ $(i = 1, 2)$ as follows:

$$p = p_c^1 + (1-p_c^1)p_c^2 + (1-p_c^1)(1-p_c^2)p_w$$

Using the fact that $p_c^i = p_c^{i1} + p_c^{i2}$, for $i = 1, 2$, and invoking the no self congestion assumption 1, i.e. $p_c^{11} = 0$, $p$ can be re-written as:

$$\begin{aligned} p &= p_c^{12} + (1-p_c^{12})p_w + (1-p_c^{12})(1-p_w)p_c^{22} \\ &\quad +(1-p_c^{12})(1-p_w)p_c^{21} \\ &= \hat{p}_w + \hat{p}_c \geq \hat{p}_w \end{aligned} \tag{2}$$

where

$$\hat{p}_w = p_c^{12} + (1-p_c^{12})p_w + (1-p_c^{12})(1-p_w)p_c^{22} \tag{3}$$

and

$$\hat{p}_c = (1-p_c^{12})(1-p_w)p_c^{21} \tag{4}$$

$\hat{p}_w$ is independent of packet loss due to self congestion, i.e. $p_c^{11}$ or $p_c^{21}$, and hence also independent of streaming rate $T$. In a sense, $\hat{p}_w$ is similar to $p_w$ in that it lumps cross congestion and wireless channel error in one quantity. Therefore it can be interpreted as equivalent wireless channel packet loss rate with no cross congestion on nodes 1 and 2. On the other hand, $\hat{p}_c$ depends on packet loss due to self congestion, i.e. $p_c^{21}$, and thus may vary with the streaming rate. Eqn. (2) shows that $\hat{p}_w$ is a lower bound for $p$, and that the bound is reached if and only if there is no self congestion, i.e. $p_c^{21} = 0$ and hence, $\hat{p}_c = 0$. Combining Eqn. (1) and (2), an upper bound, $T_b$, on the streaming rate of one TFRC connection can be derived as follows:

$$T \leq \frac{kS}{RTT_{min}\sqrt{\hat{p}_w}} \equiv T_b \tag{5}$$

If there is no self congestion, i.e. $p_c^{21} = 0$, and hence no queuing delay caused by self congestion, we get $rtt = RTT_{min}$, $\hat{p}_c = 0$, $p = \hat{p}_w$, and therefore $T = T_b$ in Eqn. (5). In this case, the throughput is $T_b(1-\hat{p}_w)$, 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 under-utilized 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 under-utilize wireless link is*

$$T_b(1-\hat{p}_w) < B_w(1-p_w) \tag{6}$$

*When there is no cross congestion, i.e. $p_w = \hat{p}_w$, the condition is simplified to $T_b < B_w$.*

*Proof:* Since $T_b(1-\hat{p}_w)$ is the upper bound of one TFRC's throughput, clearly Eqn. (6) implies under-utilization of the wireless channel, and hence the "sufficient" part of the Theorem is obvious. To see the necessary part, note that if under-utilization happens, i.e. $T(1-p) < B_w(1-p_w)$, then invoking assumption 2 in Section II.A, no self congestion happens, thus $rtt = RTT_{min}$, $p = \hat{p}_w$ and $T = T_b$, resulting in $T_b(1-\hat{p}_w) < B_w(1-p_w)$. ■

If the condition in (6) is satisfied, then direct application of TFRC or TCP to wireless scenario results in under-utilization. In essence, the approaches taken in [5][6][7][8] [9][10][11] ensure the condition in (6) 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 $T_b$ become independent of the wireless channel packet loss rate $p_w$, and hence ensures that condition in (6) is independent of the wireless channel errors. 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 (6) 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 $\hat{p}_w$. To see this, let us consider a simple example in which

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

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

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 $\hat{p}_w$ for each of the two TFRC connections remain unchanged from the case with one TFRC connection. This is because according to Eqn. (3), packet loss rate due to cross congestion, $p_c^{12}$ and $p_c^{22}$, are independent of the streaming rate, $T$. Thus the throughput upper bound for each of the two TFRC connections is $\frac{kS}{RTT_{min}\sqrt{\hat{p}_w}}(1-\hat{p}_w) = T_b(1-\hat{p}_w)$, and the aggregate throughput upper bound for both of them is $2\frac{kS}{RTT_{min}\sqrt{\hat{p}_w}}(1-\hat{p}_w) = 2T_b(1-\hat{p}_w)$, which is smaller than $B_w(1-p_w)$, implying channel under-utilization. Invoking assumption 2, we conclude that there is no self congestion and hence $rtt = RTT_{min}$, $p_c^{21} = 0$ and $\hat{p}_c = 0$, and thus $p = \hat{p}_w$. The throughput for each connections is then $\frac{kS}{RTT_{min}\sqrt{\hat{p}_w}}(1-\hat{p}_w)$. Consequently, the total throughput for both connections is $2\frac{kS}{RTT_{min}\sqrt{\hat{p}_w}}(1-\hat{p}_w)$ with packet loss rate at $\hat{p}_w$.

A similar argument can be repeated with three TFRC connections, except that the wireless channel is no longer under-utilized and $rtt > RTT_{min}$. Furthermore, if the buffer on node 2 overflows then $\hat{p}_c$ will no longer be zero and hence using Eqn. (2) we get $p > \hat{p}_w$. 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 $RTT_{min}$; and overall packet loss rate $p$ could exceed $\hat{p}_w$, i.e. the overall packet loss rate in the two connections case.

In general, given $B_w$, $\hat{p}_w$, 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:

$$\begin{aligned} B_w(1-p_w) &= n_{opt}\,\frac{kS}{RTT_{min}\sqrt{\hat{p}_w}}(1-\hat{p}_w) \\ \Rightarrow\ n_{opt}S &= B_w\,\frac{1-p_w}{1-\hat{p}_w}\,\frac{RTT_{min}\sqrt{\hat{p}_w}}{k} \end{aligned} \tag{7}$$

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

$$\left\lfloor B_w\,\frac{1-p_w}{1-\hat{p}_w}\,\frac{RTT_{min}\sqrt{\hat{p}_w}}{kS}\right\rfloor \equiv \hat{n}_{opt} \tag{8}$$

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

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 $\delta n$ 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

$$(n_{opt}+\delta n)S = B_w\,\frac{1-p_w}{1-p'}\,\frac{rtt\sqrt{p'}}{k} \tag{9}$$

Comparing the above equation with Eqn. (7), and taking into account that the right hand sides of Eqn. (7) and (9) are monotonically increasing functions with respect to overall packet loss rate and round trip time, we conclude that either $rtt > RTT_{min}$ and/or $p' > \hat{p}_w$.

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 $p > \hat{p}_w$.

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 under-utilized. If the number of connections is chosen optimally based on Eqn. (7), then wireless channel becomes fully utilized, the total throughput becomes $B_w(1-p_w)$, the $rtt = RTT_{min}$, and

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

the overall packet loss rate is at the lower bound $\hat{p}_w$, given in Eqn. (3). 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 $\hat{p}_w$. 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 [14] 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 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 channel error rate, $p_w$, for different number of connections, are shown in Figure 2. 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 around 4 or 5.

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 department at U.C. Berkeley 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.

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*

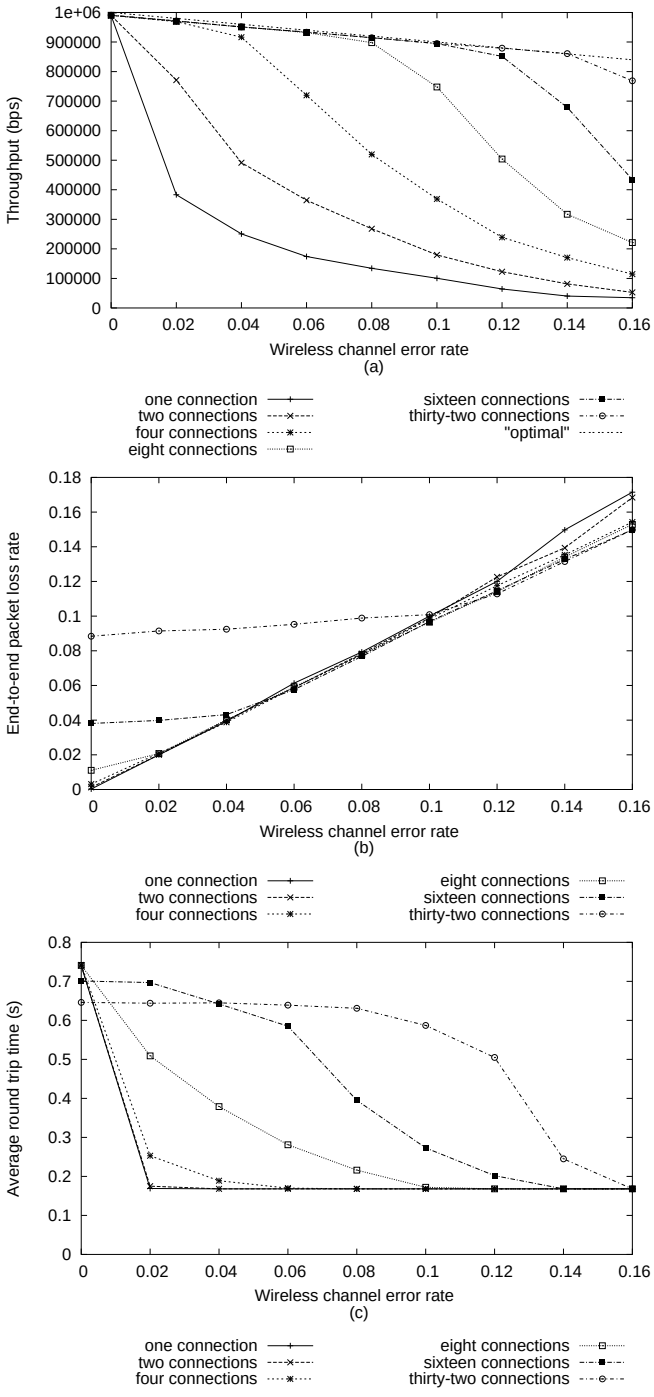



Fig. 2. NS-2 simulations showing (a) End-to-end throughput, (b) packet loss rate, and (c) round trip time as a function of wireless channel error rate, $p_w$, for different number of connections.

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 |

*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 $\alpha/n$ or decrease it by $\beta$, depending on the $rtt$ measurements. $\alpha$ and $\beta$ 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 sub-system (CCS). The flowchart of the system is shown in Figure 4. We now describe each component in detail.

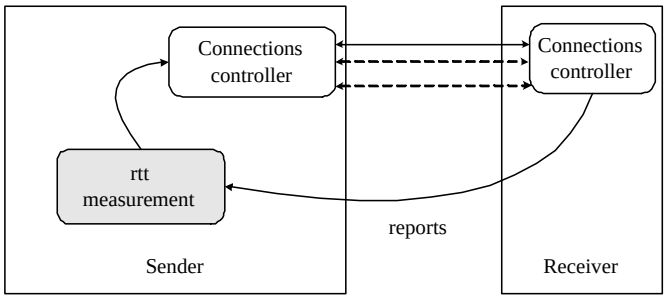
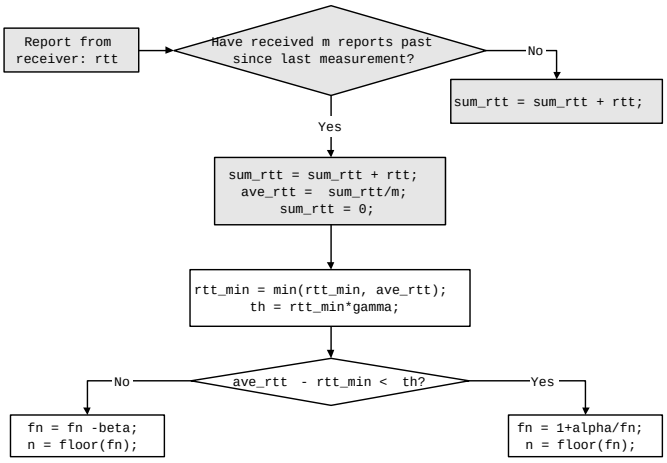



Fig. 3. MULTFRC system framework.

### 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.




```
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.

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:

$$ave\_rtt = \frac{\sum_{i=1}^{m} rtt\_sample_i}{m} \tag{10}$$

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($\alpha, \beta$)) the number of connections $n$, based on the input from RMS, as illustrated in Figure 4. Specifically, it first sets the $rtt\_min$ as the minimum $ave\_rtt$ seen so far, and then adapts the number of connection $n$ as follows:

$$n = \begin{cases} n - \beta, & \text{if } ave\_rtt - rtt\_min > \gamma rtt\_min; \\ n + \alpha/n, & \text{otherwise.} \end{cases} \tag{11}$$

where $\gamma$ is a preset parameter. The reason for this is fair and efficient sharing among multiple MULTFRC applications, and between MULTFRC and TCP or TFRC connections.

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 $\gamma rtt\_min$ 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 [17] 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.

## 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

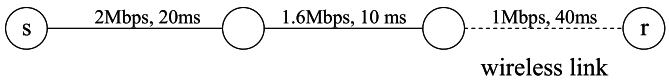



Fig. 5. Simulation topology.

The topology used in simulations is shown in Figure 5. 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 10 seconds, 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, 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 $m = 50$.

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 6. 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 5, with $B_w = 100\ kbps$ and $RTT_{min} = 757\ ms$. The results, shown in Figure 7 are as expected, and validate our earlier observations. [2]

Considering the throughput plots in Figures 6 and 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 $\gamma$ 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.

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 6. 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

[2]Note the round trip times for $p_w = 0$ are shown neither in Figure 6 or 7 because they represent the channel error free case in which MULTFRC reduces to one TFRC connection.

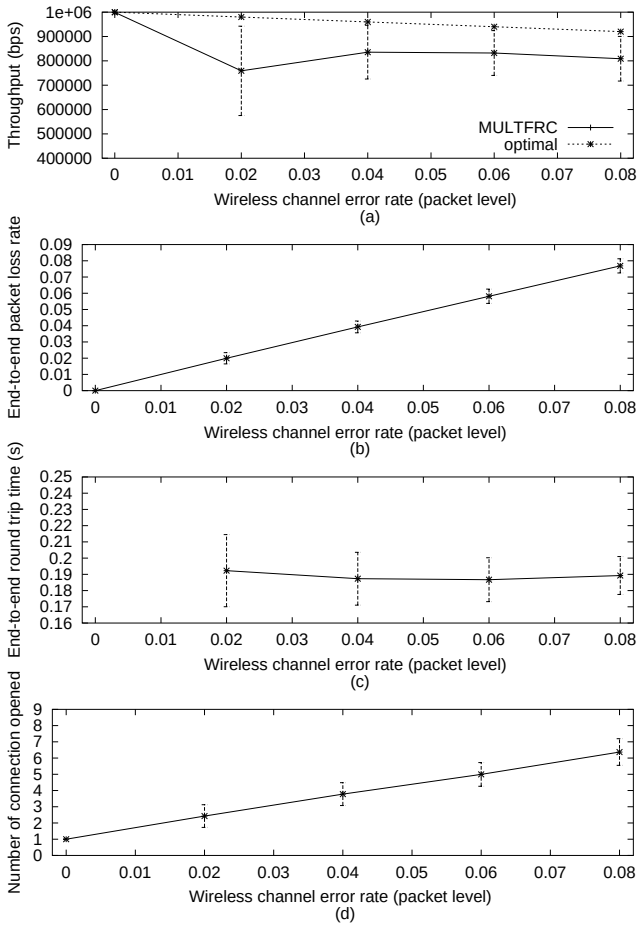



Fig. 6. NS-2 simulations for $B_w = 1\ Mbps$ and $RTT_{min} = 168\ ms$; (a) throughput, (b) end-to-end packet loss rate, (c) end-to-end round trip time, (d) number of connections, all as a function of packet level wireless channel error rate.

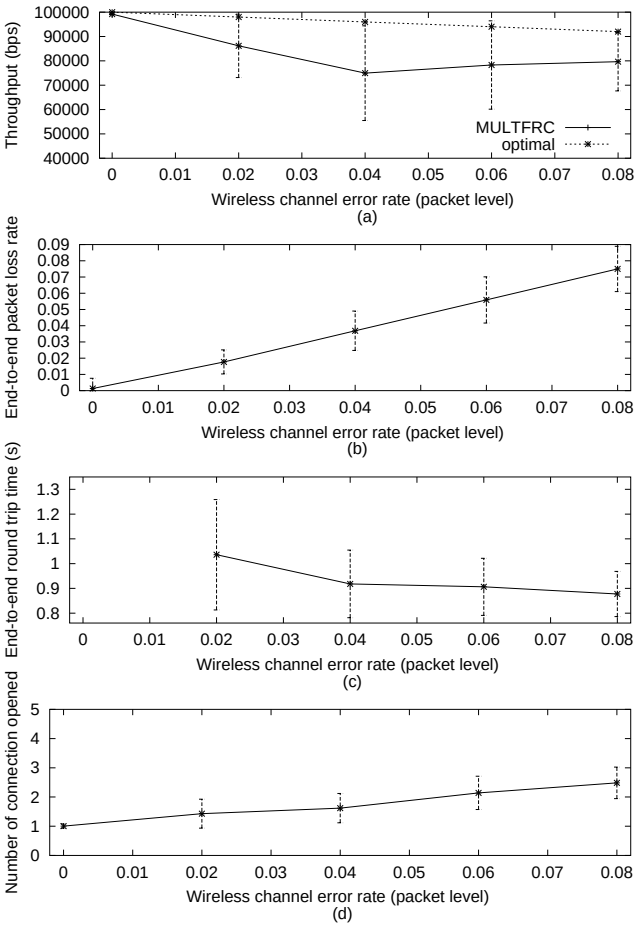



Fig. 7. NS-2 simulations for $B_w = 100\ kbps$ and $RTT_{min} = 757\ ms$; (a) throughput, (b) end-to-end packet loss rate, (c) end-to-end round trip time, (d) number of connections, all as a function of packet level wireless channel error rate.

throughput, packet loss rate, round trip time and the number of connections opened are shown in Figure 9. As seen, the number of connections varies from around 3 to around 7 as $p_w$ switches from 0.02 to 0.08.

As for actual experiments, we compare the performance of MULTFRC system and one TFRC connection in Table II. 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.

### C. Fairness between MULTFRC and TCP or TFRC

We now use NS-2 simulation for the topology shown in Figure 1 to show that MULTFRC does not starve applications using one TCP or one TFRC connection; we start with one TCP or TFRC connection and add a MULTFRC at $3000^{th}$ second. At $6000^{th}$ second, the MULTFRC is terminated. The results are shown in Figures 10 and 11 for comparison with

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 |

TCP and TFRC respectively. As shown, MULTFRC system starves neither TCP nor TFRC.

## 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

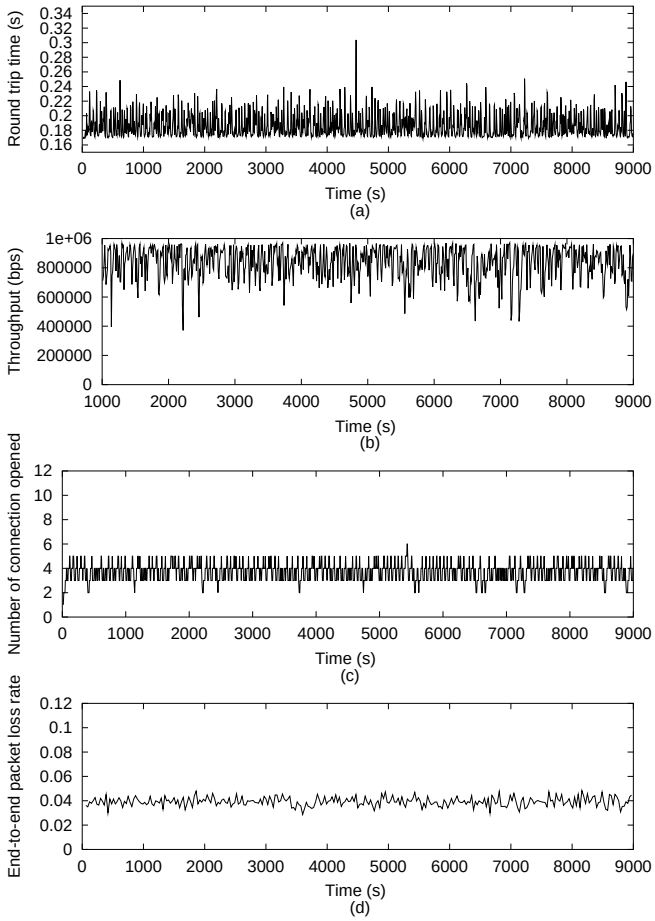



Fig. 8. NS-2 simulations for $B_w = 1Mbps$ and $p_w = 0.04$; (a) end-to-end round trip time, (b) throughput, (c) end-to-end packet loss rate, (d) number of connections, all as a function of time.

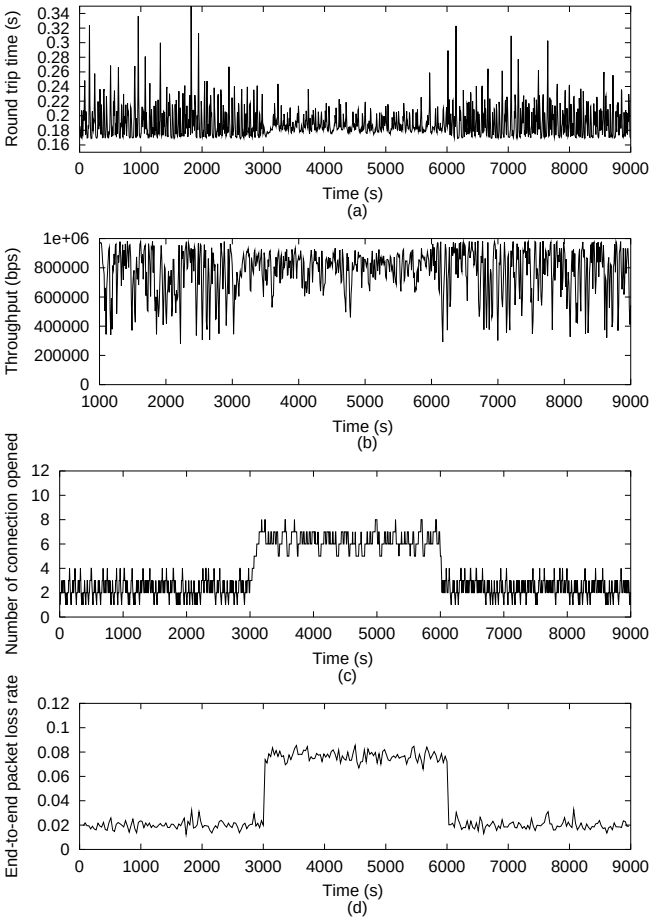



Fig. 9. NS-2 simulation results as $p_w$ changes from 0.02 to 0.08 and back again; (a) end-to-end round trip time, (b) throughput, (c) numbers of connections, (d) end-to-end packet loss rate, all as a function of time.

data network were used to show the effectiveness of our approach.

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. Nevertheless, as long as the necessary and sufficient condition in (6) 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.

Future work will be focused on (a) examining the performance of multiple MULTFRC connections sharing a wireless channel, and (b) considering the stability issue when both the number of connections and the sending rate of each connections are changing dynamically in a network.

## ACKNOWLEDGEMENT

This work was supported by NSF grant ANI-9905799 and AFOSR contract F49620-00-1-0327.

## REFERENCES

[1] Sally Floyd, Mark Handley, Jitendra Padhye and Joerg Widmer, "Equation-Based Congestion Control for Unicast Applications", *Proc. ACM SIGCOMM 2000*, 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 Transactions on Networking*, Aug. 1999

[3] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", *ACM SIGCOMM 98*, pp. 303 - 314

[4] W. Tan and A. Zakhor, "Real-Time Internet Video Using Error Resilient Scalable Compression and TCP-Friendly Transport Protocol", *IEEE Transactions on Multimedia*, June 1999, vol. 1, no. 2, pp. 172-186

[5] H. Balakrishnan, V. Padmanabhan, S. Seshan, R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links", *ACM SIGCOMM'96*, pp.256 - 269

[6] H. Balakrishnan and R. Katz, "Explicit Loss Notification and Wireless Web Performance", *IEEE Globecom Internet Mini-Conference*, Nov. 1998

[7] S. Biaz, N. H. Vaidya, "Distinguishing congestion losses from wireless transmission losses: a negative result", *Computer Communications and Networks*, 1998

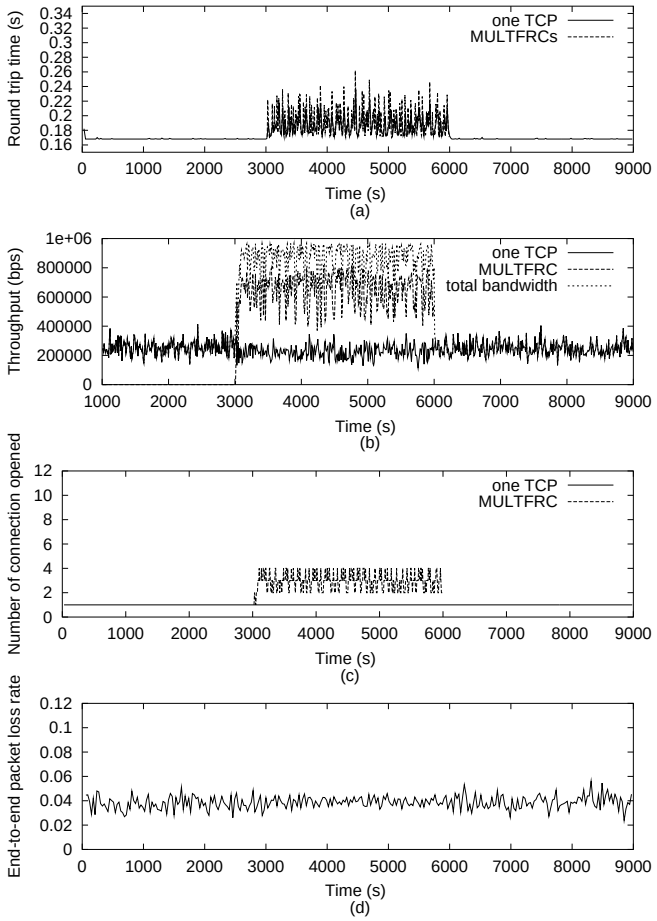



Fig. 10. MULTFRC systems does not starve one TCP connection; (a) end-to-end round trip time, (b) throughput, (c) number of connections, (d) end-to-end packet loss rate, all as a function of time.


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

[9] N. Samaraweera, "Non-congestion packet loss detection for TCP error recovery using wireless links", in *IEE Proceedings of Communications*, 146(4), pp. 222C230, Aug 1999.

[10] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda, "Achieving moderate fairness for UDP flows by pathstatus classification," in *Proc. 25th Annual IEEE Conf. on Local Computer Networks (LCN 2000)*, pp. 252C61, Tampa, FL, Nov 2000.

[11] S. Cen, P.C. Cosman, and G.M. Voelker, " End-to-end differentiation of congestion and wireless losses", *Proc. Multimedia Computing and Networking (MMCN) conf. 2002*, pp. 1-15, San Jose, CA, Jan 23-25, 2002

[12] J. Mahdavi and S. Floyd, "TCP-Friendly unicast rate-based flow control", *Technical note sent to end2end-interest mailing list, http://www.psc.edu/networking/papers/tcp_friendly.html*, Jan. 1997

[13] M. Mathis, J. Semke, J. Mahdavi, T. Ott, "The macroscopic behavior of the TCP congestion avoidance Algorithm", *CCR*, July 1997, vol. 27, No. 3

[14] Network Simulation version 2, *http://www.isi.edu/nsnam/ns/*

[15] Jon Crowcroft and Philippe Oechslin, "Differentiated End to End Internet Services using a Weighted Proportional Fair Sharing TCP", *ACM Computer Communication Review*, vol. 28, no. 3, July 1998

[16] NetAnts - fast download manager, *http://www.netants.com/*

[17] tranceroute, *http://www.traceroute.org/*

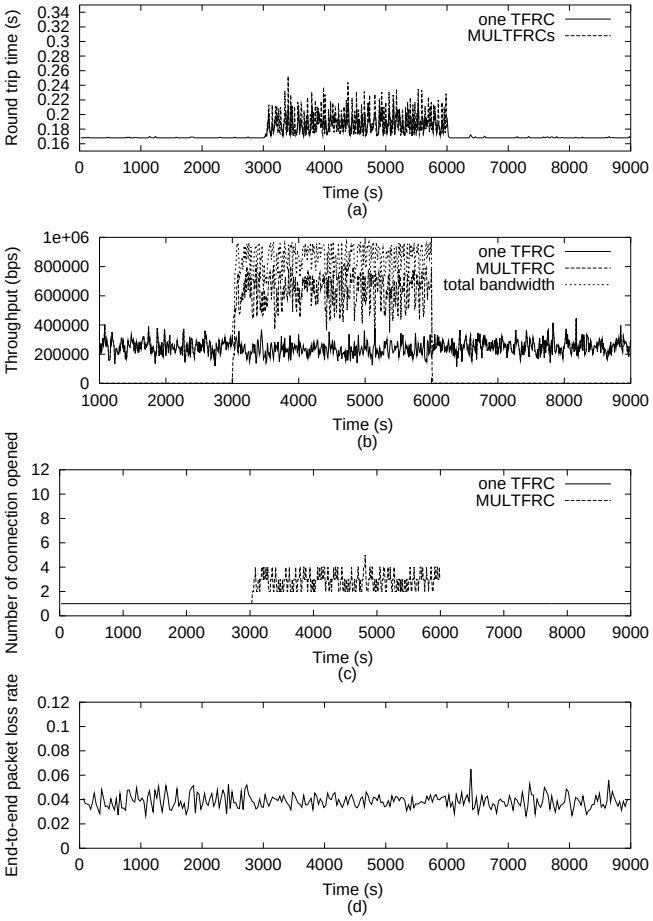




Fig. 11. MULTFRC systems does not starve one TFRC connection; (a) end-to-end round trip time, (b) throughput, (c) number of connections, (d) end-to-end packet loss rate, all as a function of time.