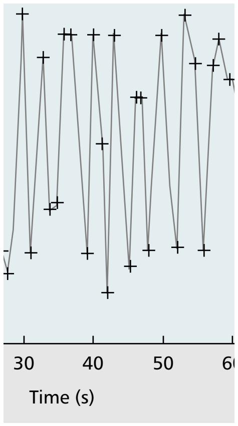
RATE CONTROL FOR STREAMING VIDEO OVER WIRELESS

MINGHUA CHEN AND AVIDEH ZAKHOR, UNIVERSITY OF CALIFORNIA AT BERKELEY



The authTFRC assumes that packet loss in wired networks is primarily due to congestion, and as such is not applicable to wireless networks in which the main cause of packet loss is at the physical layer. The authors review existing approaches to solve this problem.

This work was supported by AFOSR contract P49620-00-1-0327.

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 TCP-friendly rate control (TFRC) [1]. It is equation-based rate control in which the TCP-friendly rate is determined as a function of packet loss rate, round-trip time, and packet size. 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 main cause of packet loss is at the physical layer. In this article we review existing approaches to solve this problem. Then 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 a 1xRTT CDMA wireless data network, and video streaming simulations using traces from the actual experiments are carried out to characterize the performance and show the efficiency of our proposed approach.

INTRODUCTION

Rate control is important to multimedia streaming applications in both wired and wireless networks. First, it results in full utilization of bottleneck links by ensuring that sending rates are not too low. Second, it prevents congestion collapse by ensuring that sending rates are not too aggressive. For example, there was an actual network collapse of the Internet in October 1986 at the University of California at Berkeley, resulting in serious performance degradation [2, Sec. 1]. Finally, proper rate control ensures fairness between users sharing common links in a given network.

A widely popular rate control scheme for streaming in wired networks is equation-based rate control [1, 3, 4], also known as TCP-friendly rate control (TFRC). In TFRC the TCP-friendly rate is determined as a function of packet loss rate, round-trip time (RTT), and packet size, to mimic the long-term steady performance of TCP. There are basically three advantages to rate control using TFRC: first, it can fully utilize bottleneck capacities while preventing congestion collapse. Second, it is fair to TCP flows, which are the dominant source of traffic on the Internet. Third, the TFRC results in small rate fluctuation, making it attractive for streaming applications that 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 is dominated by physical channel errors, violating this key assumption. Neither TFRC nor TCP can distinguish between packet loss due to buffer overflow and that due to physical layer errors. As we show later, this results in underutilization of the wireless channel. For example, our experiments later show that TFRC can only utilize 56 percent of the wireless bandwidth on a Verizon 1xRTT wireless data network [5]. Hence, rate control for streaming applications over wireless is still an open problem.

There have been a number of efforts to improve the performance of TCP or TFRC over wireless [6–27]. 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 [6]. A Snoop module resides on a router or base station on the last hop (i.e., the wireless link) and records a copy of every forwarded packet. Assuming a Snoop module can access TCP acknowledgment (ACK) packets 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 [7–9, 12–14]. 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 a TCP/TFRC sender when packet loss is caused by wireless channel errors rather than congestion [10, 11]. In these cases, 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 [15–26]. For example, by examining trends in the one-way delay variation, Parsa and Garcia-LunaAceves [25] 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 queuing delay, and remains constant otherwise. Similarly, Barman and Matta proposed a loss differentiation scheme based on the assumption that the variance of RTT is high when congestion occurs, and low otherwise [21].

Cen *et al.* present an end-to-end-based approach to facilitate streaming over wireless [18]. They combine packet interarrival 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, interarrival 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 underutilizing the wireless bandwidth, as shown in [18]. Yang *et al.* [23] also propose a similar approach to improve video streaming performance in the presence of wireless error, under the assumption that the wireless link is the bottleneck.

Other schemes, such as [15–17, 19, 20, 22] 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 the streaming rate.

Tang *et al.* proposed an idea of using small dummy packets to actively probe whether the network is congested in case of packet loss, to differentiate between packet loss due to congestion and that due to channel error [27]. Yang *et al.* [28] 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 queries the link layer about recent signal strength. The packet loss is recognized as due to wireless channel error if recent signal strength is low, and due to congestion otherwise.

The disadvantage of end-to-end statistics-based approaches is that congestion detection schemes based on statistics are not sufficiently accurate, and require either 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 [29], in its congestion avoidance stage, uses queuing 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 routers with ECN marking capability to do rate control using ECN as the measure of congestion [30]. As packet loss no longer corresponds to congestion, ECN-based rate control does not adjust sending rate upon observing a packet loss.

In this article we explore the necessary and sufficient condition under which 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, it has the potential to fully utilize the wireless bandwidth, provided the number of connections and packet size are selected appropriately. A more detailed exposition of our proposed approach can be found in [5].

The rest of this article is structured as follows. We analyze the performance of one TFRC connection over wireless and show conditions under which it underutilizes the wireless channel. We then propose an optimal strategy based on multiple TFRC connections to fully utilize the wireless channel. We propose a practical system called MULTFRC to implement the approach discussed previously. NS-2 simulations, actual experimental results, and video streaming simulations using traces from the actual experiments are included. Conclusions and future work are then given.

PROBLEM FORMULATION

In this section we analyze the performance of one TFRC over wireless and show conditions under which it underutilizes the wireless channel. We then propose a rate control strategy based on opening multiple TFRC connections that has the potential to achieve optimal performance (i.e., maximum throughput and minimum end-to-end packet loss rate).

SETUP AND ASSUMPTIONS

The typical scenario for streaming over wireless is shown in Fig. 1, where a video server s in the wired network is streaming video to a receiver r 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. 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 the receiver is denoted by p, and the streaming rate is denoted by T. We refer to the wireless channel as underutilized if the streaming throughput is less than the maximum possible throughput over the wireless link, $T(1 - p) < B_w (1 - p_w)$.

There have been a number of efforts to improve the performance of TCP or TFRC over wireless. 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.

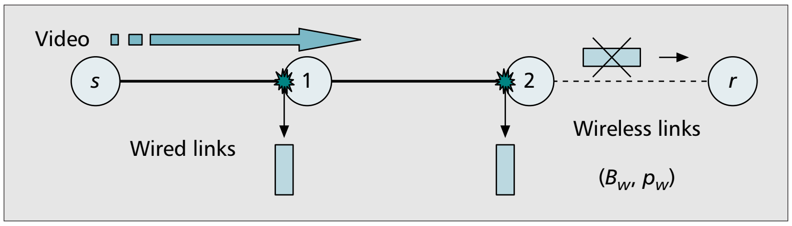


Figure 1. *Typical scenario for streaming over wireless.*

Given this scenario, we assume the following. First, there is no cross traffic at either node 1 or node 2; for a case with cross traffic, see [5]. Second, in the long term the wireless link is assumed to be the bottleneck. By this we mean there is no congestion at node 1. Third, we assume there is no congestion and queuing delay at node 2 if and only if wireless bandwidth is underutilized (i.e., we achieve $p_c = 0$ and minimum RTT, defined as RTT_{min}, if and only if $T < B_w$).When $T > B_w$, we have $p_c \geq 0$ and $rtt \geq RTT_{min}$. Fourth, B_w and p_w are assumed to be constant, at least on the timescale in which the analysis is carried; packet loss caused by wireless channel error is assumed to be random and stationary. Fourth, for simplicity, the backward route is assumed to be error-free and congestion-free.

Based on this scenario, two goals of our rate control scheme 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., highest possible throughput[1] and lowest possible packet loss rate).

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

A Sufficient and Necessary Condition for Underutilization

We use the following model for TFRC in the analysis [3]:

$$T = \frac{kS}{rtt\sqrt{p}}, \qquad (1)$$

where T represents the sending rate, S is the packet size, rtt is the end-to-end RTT, p is the end-to-end packet loss rate, and k is a constant factor. Although this model has been refined to improve accuracy [1, 31], it is simple, easy to analyze, and, more important, captures all the fundamental factors that affect the sending rate. Furthermore, the results we derive based on this simple model can be extended to more sophisticated models, such as the one used in [1].

The overall packet loss rate is p, a combination of p_w and p_c, and can be written as

$$p = p_w + (1 - p_w)p_c. \qquad (2)$$

This 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 this observation and Eq. 1, an upper bound, T_b, on the streaming rate of one TFRC connection can be derived as follows:

$$T \leq \frac{kS}{RTT_{\min}\sqrt{p_w}} \equiv T_b. \qquad (3)$$

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 T achieves the upper bound $T = T_b$ in Eq. 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 Fig. 1. Based on these, we can state the following:

Theorem 1: Given the scenario and assumptions earlier, a sufficient and necessary condition for one TFRC connection to underutilize a wireless link is

$$T_b < B_w. \qquad (4)$$

Proof: See the proof of Theorem 1 in [5].

This implies that if the available bandwidth is larger than the highest sending rate one TFRC can achieve, underutilization happens. In essence, the approaches taken in [6–20, 22–26] ensure that the condition in Eq. 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 after local retransmissions; thus, Eq. 4 can never be satisfied. By effectively setting $p_w = 0$, a Snoop-like module translates the new problem (rate control for streaming over wireless) into an old one (rate control for streaming over a wired network) for which a known solution exists. Similar observations can be made for other solutions such as the end-to-end statistics-based approaches [15–20, 22–26].

A Strategy to Reach Optimal Performance

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

Given the scenario shown in Fig. 1, and the assumptions stated earlier, we now argue that multiple connections can be used to achieve optimal performance: 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

[1] *Clearly, in situations where the network bandwidth is not a bottleneck, achieving highest possible throughput might not necessarily be the appropriate metric to optimize. An example of this would be a single video session in an 802.11b wireless LAN.*

$$T = \frac{kS}{RTT_{\min}\sqrt{p_w}} = T_b.$$

Let us now consider the case with two TFRC connections from sender s to receiver r in Fig. 1. Following the assumptions and analysis in earlier sections, since p_w for each of the two TFRC connections remain unchanged from the case with one TFRC connection, the throughput upper bound for each of the two TFRC connections is

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

and the aggregate throughput upper bound for both of them is

$$2\frac{kS}{RTT_{\min}\sqrt{p_w}}(1-p_w) = 2T_b(1-p_w),$$

which is smaller than B_w $(1 - p_w)$, implying channel underutilization and no congestion. Consequently, end-to-end packet loss rate p is at p_w, and the total throughput for both connections is

$$2\frac{kS}{RTT_{\min}\sqrt{p_w}}(1-p_w).$$

A similar argument can be repeated 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, p_c will no longer be zero; hence, using Eq. 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 RTT is no longer at the minimum value RTT_{min}, and 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 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

$$B_w(1-p_w) = \frac{n_{opt}kS(1-p_w)}{RTT_{\min}\sqrt{p_w}} \Rightarrow \qquad (5)$$
$$n_{opt}S = B_w\frac{RTT_{\min}\sqrt{p_w}}{k}.$$

Thus, what really matters is the product of n_{opt} and S; as such, it is always possible to achieve full wireless channel utilization by choosing n_{opt} to be an integer, and selecting S accordingly.[2] It is also possible to analyze the case with different packet sizes for different connections, but it is not fundamentally different from the case with the same packet size for all connections. For the rest of the article, we assume the packet size S is fixed. Then, the optimal number of connections is given by

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

resulting in throughput of

$$\hat{n}_{opt}\frac{RTT_{\min}\sqrt{p_w}}{kS}(1-p_w)$$

and packet loss rate of p_w. Opening more than n_{opt} connections results in larger rtt, or possibly higher end-to-end packet loss rate.

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 Eq. 5, then wireless channel becomes fully utilized, the total throughput becomes $B_w(1 - p_w)$, with $rtt = RTT_{min}$, and the overall packet loss rate achieves 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. The intuition here is that as number of connections exceeds n_{opt}, the sending rate of each connection has to decrease. Thus by Eq. 1, the product $rtt\sqrt{p}$ has to increase, so either rtt increases or p increases, or they both increase. For NS-2 simulations and actual experiments to validate this, see [5].

Based on the above, a strategy leading to optimal performance can be described as follows: *keep increasing the number of connections until an additional connection results in an increase of end-to-end round trip time or packet loss rate*. Next, we use this observation to develop a practical scheme called MULTFRC to determine the optimal number of connections.

A strategy leading to optimal performance can be described as follows: keep increasing the number of connections until an additional connection results in an increase of end-to-end round trip time or packet loss rate.

The Proposed Solution: Multiple TFRC

The basic idea behind MULTFRC is to measure the round trip time, and adjust the number of connections accordingly so as to utilize the wireless bandwidth efficiently, and ensure fairness between applications. There are two components in our proposed system: an rtt measurement subsystem (RMS), and a connections controller subsystem (CCS), both of them residing at the sender.

RMS measures average rtt over a window, denoted ave_rtt, and reports it to the CCS. Specifically, RMS receives average rtt_{sample}, measured in the past RTT window, from the receiver every RTT. RMS then further computes a smoothed version of these average rtts every m reports (i.e., $ave\_rtt = 1/m\ \Sigma_{i=1}^{m},\ rtt\_sample_i$). Here one can set m to be large values to reduce the noise in ave_rtt, or small values to make the system more responsive to changes in RTT.

Inspired by TCP, CCS's basic functionality is to inversely increase and additively decrease (IIAD(α, β)) the number of connections n, based on the input from RMS with α and β being preset constant parameters. 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:

$$n = \begin{cases} n-\beta & \text{if } ave\_rtt - rtt\_min > \gamma rtt\_min; \\ n+\alpha/n & \text{otherwise}, \end{cases} \qquad (7)$$

where γ 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, $ave\_rtt - rtt\_min$ corre-

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

Scheme	Throughput (kb/s)	RTT (ms)	Packet loss rate	Average # of connections
One TFRC	54	1624	0.031	N/A
MULTFRC	86	2512	0.045	1.8

Table 1. *Actual experimental results over a 1XRTT CDMA.*

sponds 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. 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 due to either change in the wireless base station or 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 [32] 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.

In [5] we evaluated the performance of MULTFRC system through NS-2 simulations and actual experiments over a Verizon Wireless 1xRTT CDMA data network. We showed via simulations that MULTFRC can achieve reasonable utilization of the wireless bandwidth, and does not starve applications that use one TCP connection.

For actual experiments over lxRTT, we stream from a desktop connected to the Internet via 100 Mb/s Ethernet in the electrical engineering and computer science (EECS) domain at the University of California (UC) at Berkeley, to a notebook connected to the Internet via the Verizon Wireless 1xRTT CDMA data network. In this case it is quite likely that the 1xRTT CDMA link is the bottleneck for the streaming connection. The lxRTT CDMA data network is advertised to operate at data speeds of up to 144 kb/s for one user. As we explore the available bandwidth for one user using UDP flooding, we find the average available bandwidth averaged over eight 30-min-long streaming sessions to be between 80 and 97 kb/s. The packet size S is 1460 bytes. As we cannot control p_w in actual experiments, we measure the average throughput, average number of connections, and packet loss rate. We compare the performance of the MULTFRC system and one TFRC connection in Table 1. As seen, MULTFRC on average opens 1.8 connections, and results in 60 percent higher throughput at the expense of a larger RTT and higher packet loss rate.

Table 2 shows packet loss details of MULTFRC for a 30-min-long experiment with packet size of 760 bytes. As expected, both the packet loss rate and burstiness of the loss increase as the number of connections increases.

Video Streaming Simulations

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

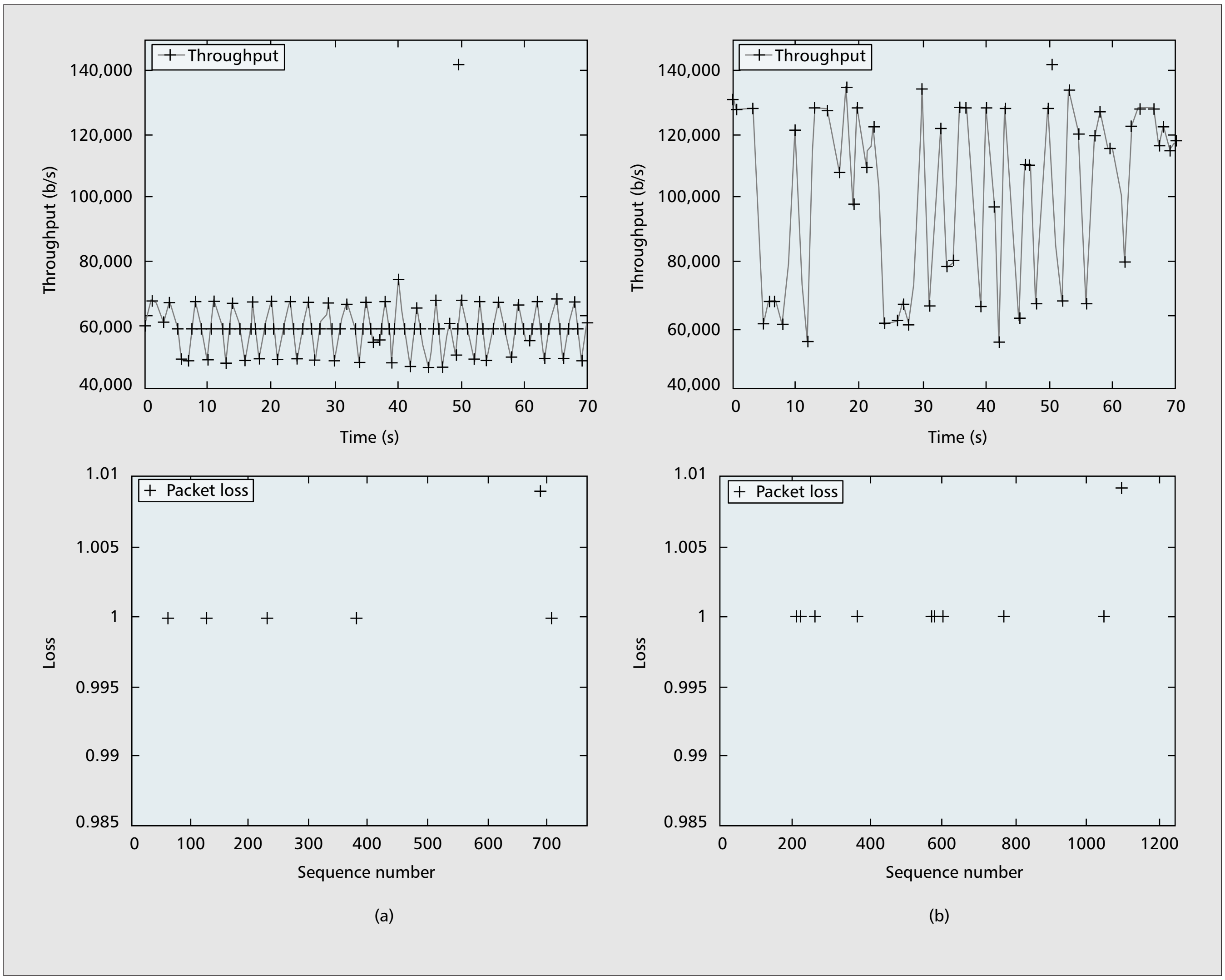
We encode 300 frames of the *news.cif* sequence using MPEG4 at bit rates varying from 50 to 100 kb/s[3] as controlled by TMN-5 [33]. The frame rate is 10 frames/s; the intra-frame refresh rate is once every 15 frames. The coded video bitstream is packetized with fixed packet size of 760 bytes. The packets are then protected using Reed-Solomon (RS) codes with different protection levels for one TFRC and MULTFRC. This is because packet loss statistics are different in the two cases. Specifically, the statistics of the 30 min trace indicates the longest burst loss to be 6 packets for one TFRC and 11 packets for MULTFRC. Thus, we apply RS(56,50) to one TFRC case and RS(61,50) to the MULTFRC case in order to sufficiently protect packets in both cases. Under ideal conditions where all packets in both schemes get through, the decoded video quality is identical between the two schemes. This is because before adding RS code, the source video bit rate is chosen to be the same for both schemes.

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 min actual experiments described earlier. The throughput and packet loss details for a 70 second segment of one TFRC and MULTFRC connections are shown in Fig. 2. As seen, both throughput and

Connections (#)	Time (%)	Packet loss rate	Average burst length	Snd. dev.	Maximum 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

Table 2. *Packet loss details of MULTFRC.*

[3] *Our choices of video bit rates are related to the available bandwidth in today's cellular telephony networks.*



■ **Figure 2**. *Throughput and packet loss details for a) one TFRC; b) MULTFRC.*

packet loss rate are higher for MULTFRC than for one TFRC case. The large throughput fluctuations in MULTFRC due to changing number of connections can potentially be argued not to be suitable for video applications in general; however, proper buffering can absorb these fluctuations in non-delay-sensitive streaming applications.

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. 3. As seen, compared to one TFRC case, MULTFRC can sustain video streaming at higher average bit rates and hence higher visual quality, despite the fact that it needs stronger forward error correction (FEC) to combat the higher packet loss rate.

DISCUSSION AND FUTURE WORK

Other work similar but unrelated to our approach includes MULTCP [34] and NetAnts [35], which open multiple connections to increase throughput. MULTCP was originally proposed to provide differential service, and was later proposed to improve the performance in high bandwidth-RTT product networks [34]. NetAnts achieves higher throughput by opening multiple connections to compete for bandwidth against others' applications [35]. Since fairness of TCP is more important at the connection level than at the application level, opening more connections can result in higher individual throughput. The differences between NetAnts and our approach are as follows. First, opening more connections than needed in wired networks unnecessarily increases the end-to-end packet loss rate experienced by an end host. Second, unlike our approach, there is no mechanism to control the number of connections in NetAnts.

Future work includes the stability, scalability, and fairness analysis of our proposed approach. In particular, we are currently investigating fairness between MULTFRC and TCP [36]. We also plan to investigate the possibility of applying our approach to improve the performance of

TCP over wireless. Finally, it would be interesting to quantify the achieved improvement in video quality resulting from MULTFRC.

References

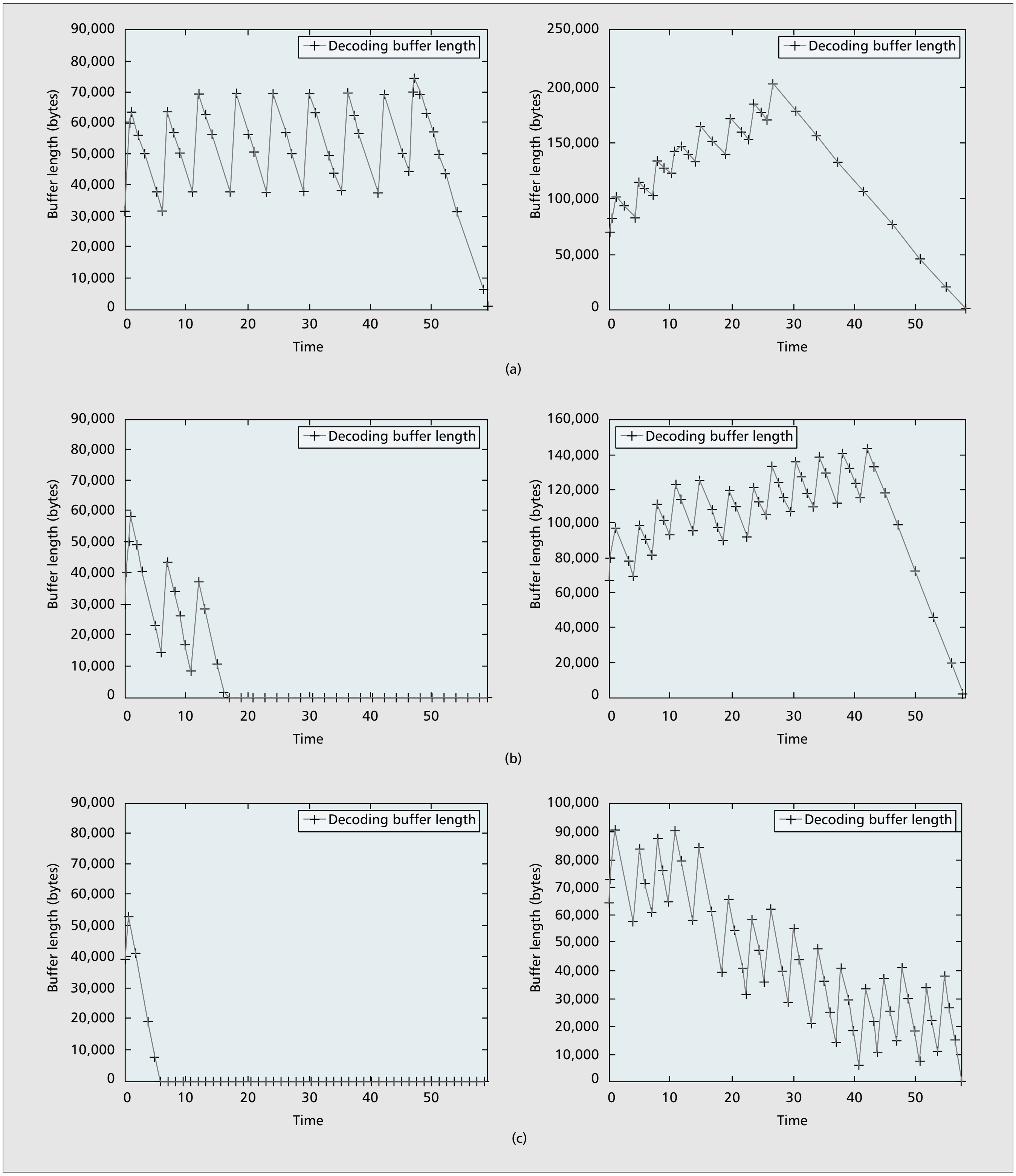
[1] S. Floyd *et al.*, "Equation-Based Congestion Control for Unicast Applications," *Proc. ACM SIGCOMM 2000*, Aug. 2000, pp. 43–56.

[2] V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM*, 1998, pp. 314–29.

[3] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Trans. Net.*, Aug. 1999.

[4] W. Tan and A. Zakhor, "Real-Time Internet Video Using Error Resilient Scalable Compression and TCP-Friendly Transport Protocol," *IEEE Trans. Multimedia*, June 1999, vol. 1, no. 2, pp. 172–86.

[5] M. Chen and A. Zakhor, "Rate Control for Streaming



■ **Figure 3**. *Throughput and packet loss details for one TFRC (left) and MULTFRC (right): the streaming bit rate is at a) 50 kb/s; b) 70 kb/s; c) 90 kb/s.*

Video over Wireless," *Proc. INFOCOM 2004*, Hongkong, China, Mar. 2004.
[6] H. Balakrishnan *et al.*, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *ACM SIGCOMM '96*, pp. 256–69.
[7] W. Ding and A. Jamalipour, "A New Explicit Loss Notification with Acknowledgment for Wireless TCP," *12th IEEE Int'l. Symp. Pers., Indoor and Mobile Radio Commun.*, vol. 1, 2001, pp. B-65–69.
[8] K. Ratnam and I. Matta, " WTCP: an Efficient Mechanism for Improving Wireless Access to TCP Services," *Int'l. J. Commun. Sys.*, vol.16, no. 1, Feb. 2003, pp. 47–62.
[9] J. Rendon, F. Casadevall, and J. Carrasco, "Wireless TCP Proposals with Proxy Servers in the GPRS Network," *13th IEEE Int'l. Symp. Pers. Indoor and Mobile Radio Commun.*, vol. 3, 2002, pp. 1156–60.
[10] H. Balakrishnan and R. Katz, "Explicit Loss Notification and Wireless Web Performance," *IEEE GLOBECOM Internet Mini-Conf.*, Nov. 1998.
[11] Y. Yang, H. Zhang, and R. Kravets, "Channel Quality Based Adaptation of TCP with Loss Discrimination," *IEEE Global Telecommun. Conf.*, vol. 2, 2002, pp. 2026–30.
[12] C. F. Chiasserini and M. Men, "Improving TCP over Wireless through Adaptive Link Layer Setting," *IEEE Global Telecommun. Conf.*, vol. 3, 2001, pp. 1766–70.
[13] J. A. Cobb and P. Agrawal, "Congestion or Corruption? A Strategy for Efficient Wireless TCP Sessions," *Proc. IEEE Symp. Comp. and Commun.*, 1995, Los Alamitos, CA, pp. 262–68.
[14] J.-H. Choi, S.-H. Yon, and C. Yon, "A Flow Control Scheme based on Buffer State for Wireless TCP," *4th Int'l. Wksp. Mobile and Wireless Commun. Net.*, 2002, pp. 592–96.
[15] S. Biaz and N. H. Vaidya, "Discriminating Congestion Loss from Wireless Losses using Interarrival Times at the Receiver," *Proc. IEEE Symp. Application-specific System and Software Eng. and Tech.*, Richardson, TX, Mar. 1999, pp. 10–17.
[16] N. Samaraweera, "Non-congestion Packet Loss Detection for TCP Error Recovery using Wireless Links," *IEE Proc. Commun.*, vol. 146, no. 4, Aug 1999, pp. 222–30.
[17] Y. Tobe *et al.*, "Achieving Moderate Fairness for UDP flows by Pathstatus Classification," *Proc. LCN 2000*, Tampa, FL, Nov 2000, pp. 252–61.
[18] S. Cen, P. Cosman, and G. Voelker, "End-to-End Differentiation of Congestion and Wireless Losses," *IEEE/ACM Trans. Net.*, vol. 11, no. 5, 2003.
[19] P. Sinha *et al.*, "A Wireless Transmission Control Protocol for CDPD," *IEEE Wireless Commun. and Net. Conf.*, vol. 2, 1999, pp. 953–57.
[20] P. Sinha *et al.*, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," *Wireless Networks*, vol. 8, no. 2–3, 2002, pp. 301–16.
[21] D. Barman and I. Matta, "Effectiveness of Loss Labeling in Improving TCP Performance in Wired/Wireless Networks," *Proc. 10th ICNP*, Washington, DC, USA, 2002, pp. 2–11.
[22] J.-J. Lee, F. Liu, and C. C. J. Kuo, "End-to-end Wireless TCP with Non Congestion Packet Loss Detection and Handling," *Proc. SPIE*, vol. 5100, 2003, pp. 104–13.
[23] G. Yang, M. Gerla, and M. Y. Sanadidi, "Adaptive Video Streaming in Presence of Wireless Errors," *MMNS 2004*, San Diego, CA, 2004.
[24] J. Liu, I. Matta, and M. Crovella, "End-to-End Inference of Loss Nature in a Hybrid Wired/Wireless Environment," *Proc. WiOpt '03*, 2003.
[25] T. Kim, S. Lu, and, V. Bharghavan, "Improving Congestion Control Performance through Loss Differentiation," *ICPP Wksp.*, 1999, pp.140–45.
[26] C. Parsa and J. J. Garcia-Luna-Aceves, "Improving TCP Congestion Control over Internet with Heterogeneous Media," *Proc. ICNP '99*, 1999, pp. 213–21.
[27] J. Tang *et al.*, "RCS: A Rate Control Scheme for Real-Time Traffic in Networks with High BandwidthDelay Products an High Bit Error Rates," *INFOCOM 2001*, Anchorate, AK, Apr. 2001, pp. 114–22.
[28] F. Yang *et al.*, "End-to-End TCP-Friendly Streaming Protocol and Bit Allocation for Scalable Video over Mobile Wireless Internet" *IEEE INFOCOM 2004*, Hongkong, China, Mar. 2004.
[29] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end-to-end Congestion Avoidance on a Global Internet," *IEEE JSAC*, vol. 13, no. 8, Oct. 1995, pp. 1465–80.
[30] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Comp. Commun. Rev.*, vol. 24, Oct. 1994, pp. 10–23.
[31] J. Padhye *et al.*, "Modeling TCP Throughput: A Simple Model and Its Empirical Validation," *ACM SIGCOMM '98*, pp. 303–14.
[32] tranceroute, http://www.traceroute.org/
[33] T. Research, "TMN (H.263) Encoder/Decoder, v. 2.0, tmn (h.263) Codec," June 1996.
[34] J. Crowcroft and P. Oechslin, "Differentiated End to End Internet Services Using a Weighted Proportional Fair Sharing TCP," *ACM Comp. Commun. Rev.*, vol. 28, no. 3, July 1998.
[35] NetAnts: fast download manager, http://www.netants.com/
[36] M. Chen and A. Zakhor, "Rate Control for Streaming Video over Wireless," to be submitted to *IEEE Trans. Multimedia*.

Additional Reading

[1] J. Mahdavi and S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control," Tech. note sent to end2end-interest mailing list, http://www.psc.edu/networking/papers/tcp_friendly.html, Jan. 1997.
[2] M. Mathis et al., "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *CCR*, vol. 27, no. 3, July 1997.
[3] Network Simulation v. 2, http://www.isi.edu/nsnani/ns/

Biographies

AVIDEH ZAKHOR [F'01] (avz@eecs.berkeley.edu) received a B.S. degree from the 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 UC Berkeley where she is currently a 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 five U.S. patents, and is co-author of the book *Oversampled A/D Converters* with Soren Hein. She was a General Motors scholar from 1982 to 1983, a Hertz fellow from 1984 to 1988, and 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 the IEEE Signal Processing Board of Governors. She received the Okawa Prize in 2004.

MINGHUA CHEN [StM] (minghua@eecs.berkeley.edu) received M.S. and B.Eng. degrees in electronic engineering from Tsinghua University in 2001 and 1999, respectively. Since 2001 he has been with the Department of Electrical Engineering and Computer Science at UC Berkeley, where he is currently pursuing his Ph.D. degree. He is co-author of the book *IPv6 Principle and Practice* (Beijing: People's Posts and Telecom Press, 2000). He received a Pao Family fellowship and a Management of Technology in China fellowship from UC Berkeley in 2001 and 2004, respectively. His research interests are in signal processing, networking, control, and wireless communication. His current focus is flow control over wireless, with practical implementations for data and multimedia transmissions over commercial wireless networks.

We are currently investigating fairness issue between MULTFRC and TCP. We also plan to investigate the possibility of applying our approach to improve the performance of TCP over wireless.