AN ENHANCED ALL-IN-ONE TFRC PROTOCOL FOR STREAMING VIDEO IN WIRELESS NETWORKS

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. The widely accepted rate control method in wired networks, i.e. TCP friendly equation based rate control (TFRC) [1], is known to perform suboptimally in wireless networks. In previous work, we proposed MULTFRC as an end-to-end approach to fully utilize wireless bandwidth, modifying only application layer by opening appropriate number of TFRC connections. We have recently proposed two extensions to MULTFRC, namely, a light-weight approach AOI-TFRC, and a scalable approach Enhanced MULTFRC (E-MULTFRC). AIO-TFRC not only enjoys all the advantages of MULTFRC, but also reduces implementation complexity, and the undesirable "quantization effect" associated with MULTFRC by combining multiple TFRC connections into one. E-MULTFRC not only shares all the properties of MULTFRC, but also embraces provable optimality and scalability properties. In this paper, we propose an Enhanced AIO-TFRC (EAIO-TFRC) protocol which combines benefits of both AIO-TFRC and E-MULTFRC. NS-2 simulations are carried out to characterize EAIO-TFRC's performance.

1. INTRODUCTION

Rate control is an important issue in both wired and wireless streaming applications. A popular rate control scheme over wired networks is TCP Friendly Rate Control (TFRC) [1], in which the TCP Friendly rate is determined as a function of packet loss rate, round trip time (RTT) and packet size. However, TFRC assumes that packet loss in wired networks is primarily due to congestion, as a result of buffer overflow. As such, TFRC is not applicable to wireless networks in which the bulk of packet loss is due to errors at the physical layer. Therefore, rate control in wireless networks is still an open problem.

Consequently, a number of techniques have been combined with TFRC to improve its performance over wireless [2]. 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. This in turn requires modifying either network protocols or infrastructure, potentially making them hard to deploy in practice.

We have recently proposed a new approach to solve this problem [3]. MULTFRC [3] improves the performance of TFRC over wireless networks by measuring the RTT, and decreasing the number of connections by a constant if RTT shows an increasing trend, and inversely increasing it otherwise. This Inversely Increasing, Additively Decreasing approach is referred to as IIAD. We have shown in [3] that MULTFRC can control the number of connections around the optimal value to achieve the highest throughput and lowest packet loss rate, with no modifications to the network infrastructure or the protocol stack. This makes MULTFRC different from all the existing approaches in that it is fairly straightforward to deploy in practice.

Nevertheless, there are three issues that need to be addressed in deploying MULTFRC. First, if the optimal number of connections is non-integer, MULTFRC oscillates around the fractional optimal, resulting in both large throughput variability and underutilization; we refer to this as "quantization effect". Second, operating multiple connections in one application requires more resources , such as memory and computation power, than operating one connection. These make MULTFRC inefficient, especially for implementation on low power, resource-limited handheld devices. Third, it is unclear whether MULTFRC is scalable, or whether it can converge to an optimal solution in a network as large as the Internet. Optimality and scalability properties are crucial before any large scale deployment of any rate control protocol on the Internet.

To address the first two issues, we have recently proposed an improved scheme, called ALL-IN-ONE TFRC (AIO-TFRC) [4]. We achieve this by integrating multiple connections in MULTFRC into one TFRC connection, and have shown that it results in a better utilization performance than MULTFRC.

To address the third concern, we have recently proposed an Enhanced MULTFRC (E-MULTFRC) [5]. E-MULTFRC measures RTT and decreases the number of connections multiplicatively if RTT shows an increasing trend, and inversely

increases it otherwise. This Inversely Increasing, Multiplicatively Decreasing approach is referred to as IIMD. E-MULTFRC is similar to MULTFRC in its design, except for the control law for the number of connections: E-MULTFRC applies IIMD, while MULTFRC applies IIAD. An inherent advantage of E-MULTFRC over MULTFRC is its provable optimal convergence and scalability properties as shown in [5].

In this paper, we combine AIO-TFRC with E-MULTFRC to obtain a new protocol called Enhanced AIO-TFRC (EAIO-TFRC). It enjoys both the advantages of AIO-TFRC and E-MULTFRC, and as such addresses all three issues associated with MULTFRC. This is achieved by integrating the IIMD control law for the number of connections in E-MULTFRC into one TFRC connection, while achieving a better utilization performance than E-MULTFRC. NS-2 simulations are carried out to evaluate its performance.

The rest of the paper is structured as follows. In Section 2, we briefly review our previous work, including MULTFRC, E-MULTFRC, and AIO-TFRC. We propose EAIO-TFRC in Section 3, followed by NS-2 simulation results in Section 4.

2. OVERVIEW OF PREVIOUS WORK

2.1. MULTFRC

We have shown that for a given network setting, there is an optimal number of connections[1] for an application to achieve the highest possible throughput and minimum packet loss rate [3]. Opening more connections than the optimal results in an increase in RTT, and subsequently an increase in end-to-end packet loss rate [3]. MULTFRC measures the RTT, and adjusts the number of connections so as to (a) utilize the wireless bandwidth efficiently, and (b) ensure fairness between applications. Specifically, it measures the average RTT, denoted by ave_rtt, and inversely increases and additively decreases the number of virtual connections, denoted by n, based on the following law:

$$n = \begin{cases} n - \beta, & \text{if } ave\_rtt - rtt\_min > \gamma\ rtt\_min; \\ n + \alpha/n, & \text{otherwise.} \end{cases} \quad (1)$$

where rtt_min is the minimum ave_rtt seen so far, and α, β, and γ are preset parameters empirically chosen to be $\alpha = \beta = 1, \gamma = 0.25$ [3]. MULTFRC quantizes n to its closest integer number, denoted by $\bar{n}$, and opens multiple TFRC connections accordingly.

For a given route, $ave\_rtt - rtt\_min$ corresponds to current queuing delay, and γrtt_min is a threshold on the queuing delay that MULTFRC can tolerate before it starts to decrease n. Thus by evaluating the relation between ave_rtt and rtt_min, MULTFRC detects full utilization of network bottleneck, and controls n accordingly.

In [3], we have evaluated the performance of MULTFRC system through NS-2 simulations and actual experiments over Verizon Wireless 1xRTT CDMA data network. Simulations and experiments have shown that MULTFRC can achieve better utilization of the wireless channel than traditional TFRC or TCP, is fair to them, and significantly improves video streaming performance.

However, there are three concerns associated with MULTFRC. First one has to do with bandwidth underutilization. NS-2 simulations in [3] show that although MULTFRC performs reasonably well, there is still some gap between its throughput and the optimal. There are two causes for this. First one is the control behavior described in (1): n is decreased when the full utilization of bottlenecks is detected, and is inversely increased until the next full utilization is detected. As such, bottlenecks stay underutilized during this period, resulting in suboptimal average throughput. It is impossible to remove this sub-optimality due to the control law without changing it. The second reason 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. 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.

Last but not the least, although MULTFRC provides new practical insight on how to improve the performance of TFRC over wireless, it is unclear whether its performance can easily scale to a network as large as the Internet, and whether it is optimal. Hence a general framework for flow control over wireless is needed to address the issues of optimality and scalability, and to provide guidelines and performance prediction prior to any implementation.

2.2. ALL-IN-ONE TFRC (AIO-TFRC)

We recently proposed an alternative to MULTFRC, called AIO-TFRC [4], in order to address the two drawbacks of MULTFRC, while retaining the same control law for n as in MULTFRC. To achieve this goal, we integrate the Bandwidth Filtered Loss Detection (BFLD) technique from [6], together with the control law in (1) to construct the AIO-TFRC system. Basically, the receiver feeds back the RTT and loss event rate to the sender. The sender then adjusts n based on (1), and sends out the data packets at a rate of n times that of one TFRC's sending rate.

NS-2 simulations show that AIO-TFRC achieves similar

[1] Not necessarily an integer.

throughput as MULTFRC in high packet loss rate situations, but achieves higher throughput than MULTFRC at the low packet loss rate scenarios. The simulations also show that AIO-TFRC is relatively fair to TCP, and highly fair to itself.

2.3. Enhaced MULTFRC (E-MULTFRC)

In [7], we have formulated rate control in wireless networks as a concave optimization problem, of which rate control in wired networks can be shown to be a special case. This formulation results in a new class of end-to-end based solutions, in which an appropriate number of connections are opened at the application layer by the sender. The solutions require only one bit of information on whether or not the route is congested, making it easy to estimate accurately at the application layer. Hence no modifications to either existing protocols, e.g. TCP, or infrastructure, e.g. routers, are needed. We have shown that our proposed solution has a unique, stable equilibrium that solves the concave optimization problem, implying the scalability and optimality of the solution.

To demonstrate the generality of our proposed solution, we have developed a practical scheme called E-MULTFRC for streaming over wireless networks. In design, this is exactly the same as MULTFRC except that E-MULTFRC adapts the number of connections, n, using IIMD, rather than the IIAD control law. IIMD takes this form:

$$n = \begin{cases} \beta n + \alpha/n, & \text{if } ave\_rtt - rtt\_min > \gamma rtt\_min; \\ n + \alpha/n, & \text{otherwise.} \end{cases} \quad (2)$$

where $\alpha = 1 - \beta < 1$ and γ is a preset parameter. E-MULTFRC's efficient performance, and fairness to both itself and TCP are characterized and evaluated using both NS-2 simulations and 1xRTT wireless experiments in [5]. Analysis and simulation results also indicate E-MULTFRC in fact works in both wired and wireless scenarios. We have also extended TCP to E-MULTCP as a way to improve its throughput in wireless networks for data transmission applications [7].

3. EAIO-TFRC

In this section, we propose EAIO-TFRC in order to address all three issues of MULTFRC, by combining AIO-TFRC and E-MULTFRC. To achieve this goal, similar to AIO-TFRC, we integrate BFLD technique from [6] to be described shortly, with the control law in (2) to construct the EAIO-TFRC system. The system framework is shown in Fig. 1. Basically, the Sink at the receiver feeds back the RTT and loss event rate to the sender. The sender then adjusts n based on (2), and sends out the data packets at a rate of n times that of one TFRC's sending rate.

Sender: There are two functional components at the sender. One component is represented by the "compute n" block. It receives the RTTs from the receiver, computes an ave_rtt, by averaging these RTT samples over a 20 second window, then

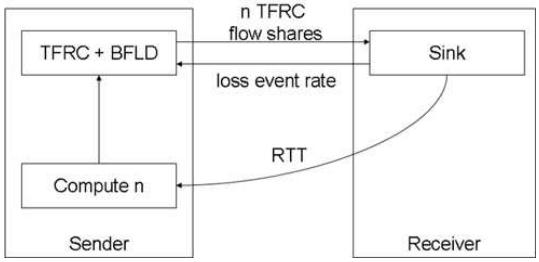

Fig. 1. The system framework of EAIO-TFRC.

updates n according to (2) every 20 seconds, i.e. using the same law as E-MULTFRC. For EAIO-TFRC, we empirically choose $\alpha = \beta = 1$, and $\gamma = 0.5$.

The other component is represented by the "TFRC+BFLD" block, and has two functionalities: first, it obtains the updated n from the "compute n" component, as well as the loss event rate from the receiver. It then computes the TCP friendly rate of one TFRC connection as the standard TFRC does [1], and adjusts the sending rate to be n times that of one TFRC.

The second functionality of the "TFRC+BFLD" block in the sender is to mark the headers of selected data packets before they are sent out. The data packets to mark are selected in such a way that they form a virtual single TFRC flow, and hence correspond to $1/n$ of all the outgoing packets. For example, if $n = 1.5$, then "TFRC+BFLD" evenly marks $2/3$ of all outgoing packets. The reason for the marking is to facilitate the loss event rate measurement at the receiver. Otherwise, if the deflated loss event rate is reported to the sender, the aggregate sending rate will in fact be higher than n TFRCs' flow shares. This is because the aggregate sending rate is inversely proportional to the square root of its measured loss event rate, which is smaller than those measured by individual TFRC flow. Consequently, this aggressive sending rate could potentially cause congestion collapse or unfairness to TCP.

Receiver: The EAIO-TFRC Sink component reports the RTT and the loss event rate of the virtual TFRC connection with marked packets to the sender every RTT.

The operation flow of EAIO-TFRC can be summarized as follows. Every RTT, the receiver sends back the measured RTT and loss event rate to the sender. Based on the RTT, the sender adjusts n according to (2) every 20 seconds. At any moment, the sender sends at a rate equivalent to n TFRC flow shares, and marks the selected outgoing packets to form a virtual stream, which is used by the receiver to carry out loss event rate measurements.

4. SIMULATION RESULTS

In this section, we carry out NS-2 simulations to evaluate the performance of EAIO-TFRC. Specifically, we examine how EAIO-TFRC performs in terms of average throughput

and packet loss rate, as a function of wireless packet loss rate, denoted by p_w. The scalability simulations are not included here due to space limitation. Since EAIO-TFRC and E-MULTFRC share the same control law on n, the stability results for EAIO-TFRC are similar to those of E-MULTFRC in [5]. In all the simulations, throughput is measured every 10 seconds, packet loss rate is measured every 30 seconds, the average RTT is measured every 100 packets, and the number of connections is sampled whenever there is a change.

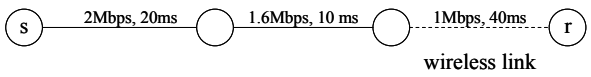

Fig. 2. The simulation topology for EAIO-TFRC's utilization evaluation.

The topology used in simulations for utilization evaluation is shown in Fig. 2. The sender is denoted by s, and the receiver is denoted by r. They both run EAIO-TFRC at the application layer. For the simulations, the wireless bandwidth, denoted by B_w, is set be 1 Mbps and is assumed to be the bottleneck. The wireless link is modeled 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.

We simulate the EAIO-TFRC system to stream for 9000 seconds. The average throughput, end-to-end packet loss rate, average RTT, and average n for p_w =0.0, 0.02, 0.04, 0.06 and 0.08 are shown in Fig. 3, where $RTT_{min} = 168\ ms$. As seen, the throughput and end-to-end packet loss rate are close to the optimal. As expected, the average n increases with wireless channel error rate, p_w.

The throughput of E-MULTFRC is also shown in Fig. 3(a) for comparison. As seen, EAIO-TFRC has almost the same throughput as E-MULTFRC when p_w is high, while it significantly outperforms E-MULTFRC when p_w is low. For example, when $p_w = 0.02$, EAIO-TFRC achieves 92% utilization of the wireless bandwidth, while MULTFRC's utilization is only 81%. Therefore, by avoiding the "quantization effect", EAIO-TFRC achieves better throughput performance than E-MULTFRC.

Similar experiments are carried out on Verizon Wireless 1xRTT CDMA data network. Using UDP flooding and averaging over 30 minutes, we find the highest average available bandwidth of the 1xRTT CDMA data network to be between 80 kbps to 97 kbps. In our experiments, we stream for 30 minutes from a desktop on wired network in eecs. berkeley.edu domain to a laptop connected via 1xRTT CDMA modem using EAIO-TFRC and TFRC. The results are shown in Table 1 for packet size of 1460 bytes. As seen, on average EAIO-TFRC mimics 1.6 connections, and results in 65% higher throughput, at the expense of a larger round trip time, and higher packet loss rate.

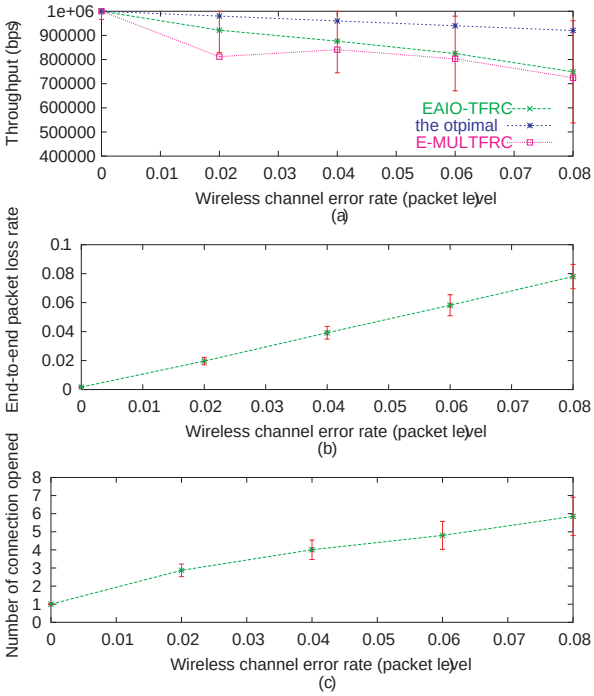

Fig. 3. NS-2 simulations for Bw = 1 Mbps and RTTmin = 168 ms; (a) throughput, (b) end-to-end packet loss rate, (c) the number of connections, all as a function of packet error rate on the wireless channel.

Table 1. Actual experimental results for a EAIO-TFRC system over 1xRTT CDMA.

scheme	throughput (kbps)	rtt (ms)	packet loss rate	ave. # of conn.
one TFRC	54	1624	0.031	N/A
EAIO-TFRC	90	2354	0.039	1.6

5. 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] M. Chen and A. Zakhor, "Rate control for streaming video over wireless," *IEEE Wireless Commun. Mag.*, vol. 12, no. 4, Aug. 2005.

[3] M. Chen and A. Zakhor, "Multiple TFRC connections based rate control for wireless networks," *IEEE Trans. Multimedia*, accepted.

[4] 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.

[5] M. Chen and A. Zakhor, "Enhanced MULTFRC (E-MULTFRC)," Technical report of EECS Department, University of California at Berkeley, June 2005.

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

[7] M. Chen and A. Zakhor, "Flow control over wireless network and application layer implementation," in *Proc. IEEE INFOCOM*, Barcelona, Apr. 2006, to appear.