Adaptive Video Streaming in Presence of Wireless Errors

Guang Yang, Mario Gerla and M. Y. Sanadidi

Computer Science Department, UCLA
Los Angeles, CA 90095, USA
{yangg, gerla, medy}@cs.ucla.edu

Abstract. Real-time video streaming with rate adaptation to network load/congestion represents an efficient solution to its coexistence with conventional TCP data services. Naturally, the streaming rate control must be efficient, smooth and TCP friendly. As multimedia clients become mobile, these properties must be preserved also over wireless links. In particular, they must be robust to random wireless losses. Existing schemes such as TCP Friendly Rate Control (TFRC) perform well in the wired Internet, but show serious performance degradation in the presence of random wireless losses. In this paper we introduce the Video Transport Protocol (VTP) with a new rate control mechanism based on the Achieved Rate (AR) estimation and Loss Discrimination Algorithm. We show that VTP can preserve efficiency without causing additional performance degradation to TCP, in both error-free and error-prone situations.

1 Introduction

Real-time video streaming is becoming increasingly important on the Internet. Unlike conventional applications, real-time streaming generally requires a minimum, continuous bandwidth guarantee as well as stringent bounds on delays and jitters. Earlier work largely relied on the unresponsive UDP traffic and imposed potential menace to network stability. Thus the more recent research is focused on adaptive schemes that respond to the network dynamics and avoid possible congestion collapses.

TCP, the dominant transport protocol on the Internet, has also been considered for streaming [11]. However, the instantaneous sending rate of TCP changes drastically such that buffering is needed at the receiver to accommodate rate fluctuations [14]. Buffering smoothes the playback rate but also brings up two concerns. First, it causes a startup delay. For Video-on-Demand (VoD) applications, startup delays of a few seconds or slightly longer are tolerable, but for real-time, interactive applications, e.g. video conferencing and online gaming, startup delays have to be tightly bounded [16]. The second concern is that more and more mobile/wireless devices are connected to the Internet. These devices are often small and inexpensive with limited computation and buffer capacities; storing a large amount of data is simply impractical.

To address the concerns, real-time streaming needs more intelligent *rate adaptation* or *rate control* mechanisms. Solutions are usually based on two types of feedback: *a)* cross-layer feedback from lower layers [9], or *b)* end-to-end feedback. On the Internet, cross-layer approaches require modifications on both end hosts and intermediate nodes, which is not practical, thus end-to-end rate control has been the preferred choice [3][7].

TCP Friendly Rate Control (TFRC) [7] is one of the most popular end-to-end streaming protocols and often used as the reference and benchmark. TFRC attempts to match the long-term throughput of legacy TCP (e.g. Reno) and is smooth, fair and TCP friendly in wired networks. However, with the increasing popularity of wireless Internet terminals and the demand for delivering multimedia to mobile users, it is necessary for streaming protocols to work efficiently also on wireless links, withstanding the high random wireless errors. Legacy TCP does not work well in this case; it tends to over-cut its window, leading to a severely degraded performance. Since TFRC attempts to faithfully match the throughput of TCP, it suffers the same low efficiency in the presence of moderate to high random errors [18].

Our goal is to develop a real-time streaming protocol that behaves well in the wired Internet, and moreover is robust to random errors and can be deployed with wireless links. We have proposed the Video Transport Protocol (VTP) [2], which measures the Achieved Rate (AR) and adapts its sending rate according to the network dynamics. However, we have recently found that the original VTP tends to be unfriendly to TCP in some scenarios. The main contribution of this paper is to refine the VTP rate control. The new mechanism should provide efficient and smooth rate control in both error-prone and error-free situations, while maintaining fairness and friendliness to coexisting flows.

The rest of the paper is organized as follows: Section 2 lists our design goals of the VTP rate control. The Achieved Rate (AR) estimation and Loss Discrimination Algorithm (LDA) are introduced in Section 3, followed by the VTP rate control mechanism in Section 4. We evaluate the performance of VTP in the Ns-2 simulator in Section 5. Related work is summarized in Section 6 and finally Section 7 concludes the paper.

2 Design Goals

In this section we discuss the main design goals of the VTP rate control mechanism, namely robustness to random errors and TCP friendliness.

2.1 Robustness to Random Errors

As the Internet evolves into a mixed wired-cum-wireless environment, more and more devices are interconnected via wireless technologies. Wireless links are usually error-prone due to interference, noise, fading, mobility, etc. [13]. However, popular error recovery techniques, such as Automatic Repeat reQuest (ARQ) and Forward Error Correction (FEC), may not completely solve this problem.

First of all, ARQ increases both the end-to-end delay and its variance, which is undesirable for real-time streaming. Applying ARQ in a single FIFO queue, as performed in the majority of commercial MAC layer implementations, also introduces the problem of head-of-of-line blocking, where retransmission of a packet forces subsequent packets in the same queue to wait. On the other side, FEC is more effective when errors are sporadic. In practice, errors are usually bursty due to the interference by external sources. In conclusion, after applying limited ARQ/FEC where appropriate, packet error rates of a few percent or higher are still expected in wireless networks [17]. This is the key working assumption that motivates the rest of the paper. The first design goal of VTP is to provide efficient streaming rate control in presence of random wireless errors.

2.2 TCP Friendliness

TCP is deployed virtually on every computer. Years of operation have proved that the well-designed congestion control in TCP contributes significantly to the stability of the Internet. New protocols must be *TCP friendly* to avoid potential congestion collapses.

Different definitions of "TCP friendliness" exist in the literature. A widely used one is based on *Jain's fairness index* [8], which belongs to the class of *max-min fairness*. Applying Jain's fairness index to TCP friendliness results in a statement like "a flow of the new protocol under evaluation must achieve a rate similar to the rate achieved by a TCP (usually Reno/NewReno) flow that observes the same round-trip time (RTT) and packet loss rate". VTP must comply with this definition in the region where TCP performs efficiently (i.e., with zero random errors) and can potentially use the entire bandwidth. In the case of frequent random errors, however, legacy TCP cannot achieve full bandwidth utilization. Thus the conventional definition of friendliness must be modified to allow a new, more efficient protocol to opportunistically exploit the unused bandwidth, even beyond the "fair share".

In this paper we introduce the notion of *opportunistic friendliness* to refer to the ability of a new flow to use the bandwidth that would be left unused by legacy flows. More precisely, a new protocol *NP* is said to be opportunistically friendly to legacy TCP if TCP flows obtain no less throughput when coexisting with *NP*, compared to the throughput that they would achieve if all flows were TCP (i.e., *NP* flows replaced by TCP). The second design goal of VTP is to have opportunistic friendliness to legacy TCP.

3 Achieved Rate and Loss Discrimination Algorithm

3.1 Achieved Rate

The Achieved Rate (AR), together with the Loss Discrimination Algorithm (LDA) that will be introduced shortly, are two important components in VTP. AR is the rate that the sender succeeds in pushing through the bottleneck. This

is the rate that the receiver can measure, plus the fraction corresponding to packet losses at the exit of the bottleneck due to random errors.

For the time being, let us assume zero errors. The receiver samples and filters the receiving rate, using an Exponentially Weighted Moving Average (EWMA). AR has an intuitive interpretation. Assuming we start with an empty bottleneck, each sender can safely transmit for an unlimited time at AR and expect its packets to be delivered to the receiver with no buffer overflow. If the sender transmits at a rate higher than AR, there is a chance that the extra packets will get buffered at the bottleneck queue. The sender will typically transmit, over limited periods of time, at rates higher than AR to probe the bandwidth. However, following a packet loss, it will step back and restart at or below AR.

An AR sample S_k is obtained, by the receiver, as the number of received bytes during a time period of T, divided by of T. AR samples are reported back to the sender, which updates its smoothed AR value AR_k as

$$AR_k = \sigma \cdot AR_{k-1} + (1 - \sigma) \cdot \frac{1}{2}(S_k + S_{k-1}) \tag{1}$$

where σ is a fraction close to 1.

The above scheme works well when no random errors are present. If packets can get lost at the exit of the bottleneck due to errors, they will not be received and counted by the receiver, although they do have succeeded in squeezing through the bottleneck. These packets should be included in the sender's AR value. This is done jointly with the LDA. Via the LDA, the VTP sender is able to estimate the fraction of packet losses that are error-induced, i.e., the error rate e. The AR sample reported by the receiver is then prorated by $1 + e$.

3.2 Loss Discrimination Algorithm

The Loss Discrimination Algorithm (LDA) allows VTP to distinguish error losses from congestion losses. Intuitively, it suffices to measure the RTT. If RTT is close to RTT_{min} measured on this connection, we know the bottleneck is not congested; the loss must be an error loss. On the contrary, if RTT is quite larger than RTT_{min}, the loss is likely to be due to congestion. We propose to use the Spike [4] scheme as the LDA in VTP. Spike, as illustrated in Figure 1, is an end-to-end algorithm based on RTT measurement. A flow enters the *spike* state if 1) it was not in the *spike* state, and 2) RTT exceeds a threshold B_{start}. Similarly, the flow exits the *spike* state if 1) it was in the *spike* state, and 2) RTT falls below another threshold B_{end}. B_{start} and B_{end} are defined as:

$$B_{start} = RTT_{min} + \alpha \cdot (RTT_{max} - RTT_{min}) \tag{2}$$

$$B_{end} = RTT_{min} + \beta \cdot (RTT_{max} - RTT_{min}) \tag{3}$$

where α and β are adjustable parameters. If a loss occurs when the flow is in the *spike* state, it is believed to be congestion-induced; otherwise it is error-induced.

We must point out that the above LDA works only if the error-prone link is also the bottleneck. If not, flows that share the bottleneck but do not traverse

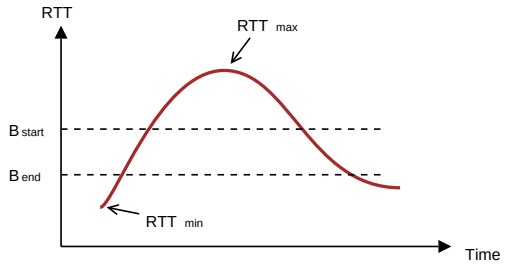

Fig. 1. Spike as a loss discrimination algorithm.

the "error" link will keep the bottleneck loaded and the value of RTT high. Other flows that also traverse the error link will suffer extra losses. However, the corresponding senders will not be able to classify these losses as error-induced due to the consistently high value of RTT. Those latter flows will reduce their rates and be "suppressed" by the flows that do not experience random errors. Fortunately, in virtually all wireless scenarios the wireless error-prone link is also the bottleneck, e.g. a satellite link or last-hop wireless segment. Thus all bottlenecked flows are subject to random errors. Our LDA scheme thus applies to most wireless situations.

4 VTP Rate Control

In this section we present the rate control mechanism in VTP. Similar to the Additive Increase in TCP congestion control, VTP linearly probes the bandwidth until congestion is detected. VTP does not perform Multiplicative Decrease though; instead it reduces the rate to AR, with extra adjustments required to mimic the TCP behavior.

4.1 TCP Behavior in Terms of Rate

While most streaming protocols operate on the concept of *rate*, TCP is window-based: a congestion window *cwnd* is used to control the number of outstanding packets. Due to this difference, streaming protocols must first understand the TCP behavior, in terms of its instantaneous sending rate rather than the window size, in order to achieve TCP friendliness.

We now consider TCP NewReno operating in congestion avoidance. We ignore slow start since it has less impact on the steady state performance. We also focus on the losses caused by congestion and assume no random errors. Consider the topology in Figure 2. C, B and P are the link capacity, queue buffer size and round-trip bandwidth-delay product (namely the pipe size), respectively. Assuming the buffer size is equal to the pipe size, we have $B = P$. $Cwnd$ oscillates between P and $B + P = 2P$ as the left diagram in Figure 3 shows.

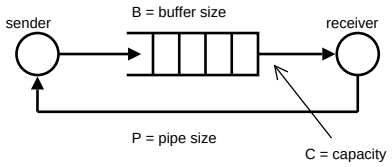

Fig. 2. A simple topology with buffer size equal to pipe size.

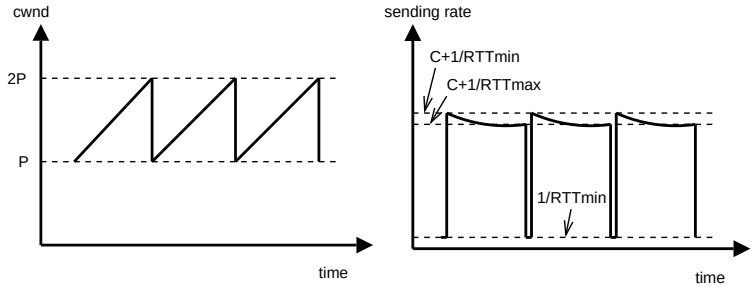

Fig. 3. Congestion window and instantaneous sending rate of TCP NewReno.

Although TCP increases *cwnd* at the speed of 1 packet/RTT, it does not necessarily increase the sending rate, since the extra packets may be buffered in the queue. With the assumption of $B = P$ in Figure 2, the sender detects a packet loss when the queue is full, i.e., $cwnd = 2P$. $Cwnd$ is then halved to P. Since there are $2P$ outstanding packets, the sender must temporarily stop sending and wait until P ACKs have been received, before it can resume the transmission. Having P outstanding packets means that the queue is drained while the pipe is full. Next, *cwnd* will increase by 1 packet/RTT, allowing the sender to transmit an extra packet every RTT. Other than this, TCP is regularly transmitting at the rate of the bottleneck capacity C, limited by the arriving rate of ACKs (i.e., self-clocked).

The right diagram in Figure 3 illustrates the instantaneous *sending rate* of TCP as we discussed above. The sending rate of TCP in congestion avoidance, in the topology of Figure 2, is $C + 1/RTT$. Note that as RTT grows with more packets get buffered, the sending rate actually decreases slightly.

4.2 VTP Rate Control

As we explained earlier, TCP instantaneous sending rate drops drastically when *cwnd* is cut by half, due to the fact that the sender must wait until half of the outstanding packets are drained from the queue/pipe. This rate reduction, as shown in Figure 3, can not be implemented as such in VTP. Yet VTP must respond to congestion by reducing, on average, its rate in the same way as TCP

in order to be TCP friendly. The tradeoff is between the amount of rate reduction and the length of time this rate is maintained. Simply speaking, VTP may reduce the rate by less but keep it longer.

Figure 4 illustrates the VTP rate control mechanism and compares it to TCP. Note that Figure 4 reflects just one of the three cycles in Figure 3. Also, curves are approximated by line segments. In Figure 4, TCP chooses the "deep but short" strategy where its rate is cut to near zero for RTT_{min} and then restored to $C+1/RTT$ immediately. In contrast, VTP reduces its rate by a smaller portion but keeps it longer. The two shaded areas $A1$ and $A2$ in Figure 4 represent the amount of extra data that TCP and VTP would be able to transmit if the loss did not happen. To make VTP friendly to TCP, these areas should be equal.

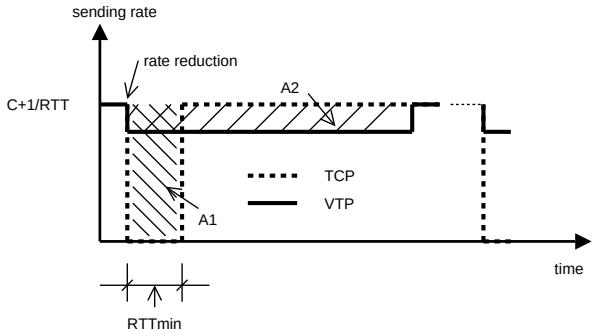

Fig. 4. Comparsion of rate control between TCP and VTP.

A parameter γ between 0 and 1 is selected as the tolerable rate reduction ratio. When congestion is detected, VTP reduces its rate to $\gamma \cdot AR$, where AR is equal to C in this case. Since $A1 = A2$, the interval over which the reduced rate is maintained is $\tau = A1/((1-\gamma) \cdot AR) = RTT/(2(1-\gamma)) = RTT_{max}/(2(1-\gamma))$. When τ has elapsed, VTP has given up the same amount of data transmission as TCP and should then enter congestion avoidance.

During congestion avoidance, VTP must match the TCP behavior. For convenience we introduce the concept of *equivalent window* in VTP and denote it as *ewnd*. *Ewnd* is defined as the number of packets transmitted by the sender during one RTT. The VTP sender computes its sending rate as follows:

1. The sender measures its current *ewnd* by counting the packets transmitted within the current RTT. Letting R be the current transmit rate, we have

$$ewnd = R \cdot RTT \tag{4}$$

2. Following Additive Increase, the new window $ewnd'$ is given as

$$ewnd' = R \cdot RTT + 1 \tag{5}$$

3. Converting from window to rate, we have the new rate R' as

$$R' = (R \cdot RTT + 1)/(RTT + \Delta RTT) \tag{6}$$

where ΔRTT is the RTT increase after a round.

4. Assuming RTT increases linearly at each round, then

$$R' = (R + 1/RTT)/(2 - RTT^{[-1]}/RTT) \tag{7}$$

where $RTT^{[-1]}$ is the round-trip time during the previous round. Note that R' is lower than what would be derived from the conventional linear rate increase by 1 packet/RTT. All necessary quantities can be readily measured by the sender.

5 Performance Evaluation

In this section, we evaluate the performance of VTP in terms of efficiency, intra-protocol fairness and opportunistic friendliness, for both error-prone and error-free cases. We also examine the ability of VTP to adapt to bandwidth changes. The experiments are carried out with the Ns-2 simulator.

5.1 Simulation Setup

The topology in Figure 5, representing a mixed wired-cum-wireless scenario, is used throughout this paper. The Internet segment is abstracted as a set of error-free links, while the wireless segment, e.g. a wireless LAN, is abstracted as a shared error-prone link. The wireless link is the only bottleneck in the system which all traffic goes through. The round-trip propagation delay is 72 msec, a typical value for a cross-continent path. All queues are drop-tail; the bottleneck buffer size is 99 packets, equal to the pipe size. Simulation time is 300 seconds for all runs.

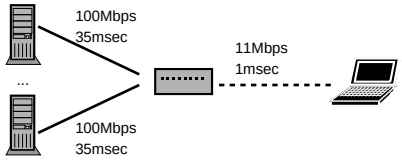

Fig. 5. Simulation setup.

5.2 Rate Adaptation and Robustness to Random Errors

First we test the rate adaptation of VTP to bandwidth changes under different error rates, where a solo VTP flow runs against varying rate CBR traffic. This test is indicative of the "agility" of the protocol and also of its efficiency in different loss rate conditions. For comparison we also replace VTP with TFRC and repeat all experiments.

Figure 6 shows the sending rates of VTP and TFRC as they adapt to the bandwidth changes caused by CBR traffic. Bandwidth available to VTP/TFRC, computed as the bottleneck capacity minus the aggregate CBR rate, is included in Figure 6 for reference. In the absence of random errors, both VTP and TFRC manage to utilize the bandwidth efficiently, maintain a smooth rate, and react to bandwidth changes quickly. When errors are present, e.g. 1% or 5% as shown, VTP is able to maintain its efficiency in bandwidth utilization, while TFRC suffers the inefficiency problem as we have discussed.

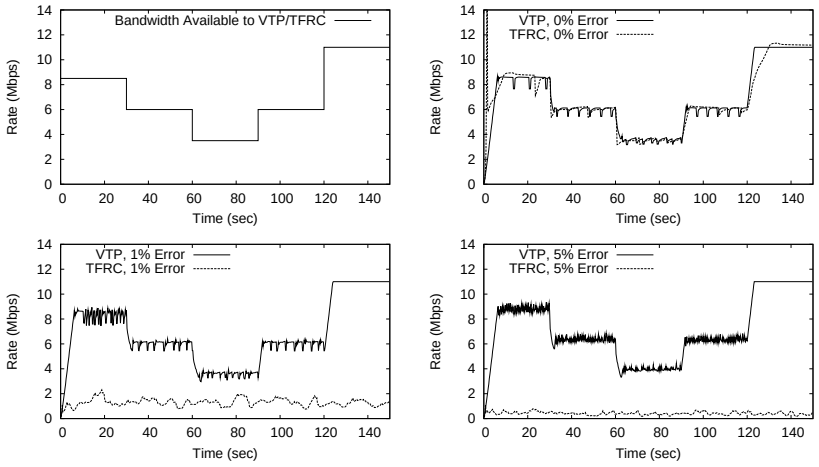

Fig. 6. Solo VTP/TFRC under different random error rates.

5.3 Inter-protocol Opportunistic Friendliness

We now evaluate the opportunistic friendliness between VTP and TCP under different error rates. For each error rate we run a couple of experiments. In the first experiment, one VTP flow and one TCP flow share the bottleneck. VTP is then replaced by TCP and the experiment is repeated. According to the definition, if the TCP throughput in the first experiment is comparable to that in the second, VTP is said to be opportunistically friendly to TCP.

We show the experiment results in Figure 7. Graphs are grouped in pairs by different error rates. In each pair, the left graph presents the instantaneous sending rates of VTP and TCP when they coexist; the right graph presents the rates when the VTP flow is replaced by TCP. More precisely, Table 1 lists the long-term[1] throughput of a TCP flow when it coexists with either VTP or another TCP flow. In all cases, impact of VTP on TCP throughput is minimal. Moreover, VTP is able to utilize the residual bandwidth when errors are present, a perfect reflection of opportunistic friendliness.

We need to point out that in the zero error case, TCP slow start tends to overshoot the TCP window and result in multiple losses, where NewReno takes multiple RTTs to recover. During this period the TCP sending rate drops to near zero. Since VTP rate probing is less aggressive than slow start, TCP overshoots its window even higher when coexisting with VTP than with another TCP. Thus in Figure 7 the TCP recovery time is longer when coexisting with VTP. Once TCP is recovered, VTP is able to share the bandwidth in a friendly manner. We are still investigating this issue to make TCP recover faster.

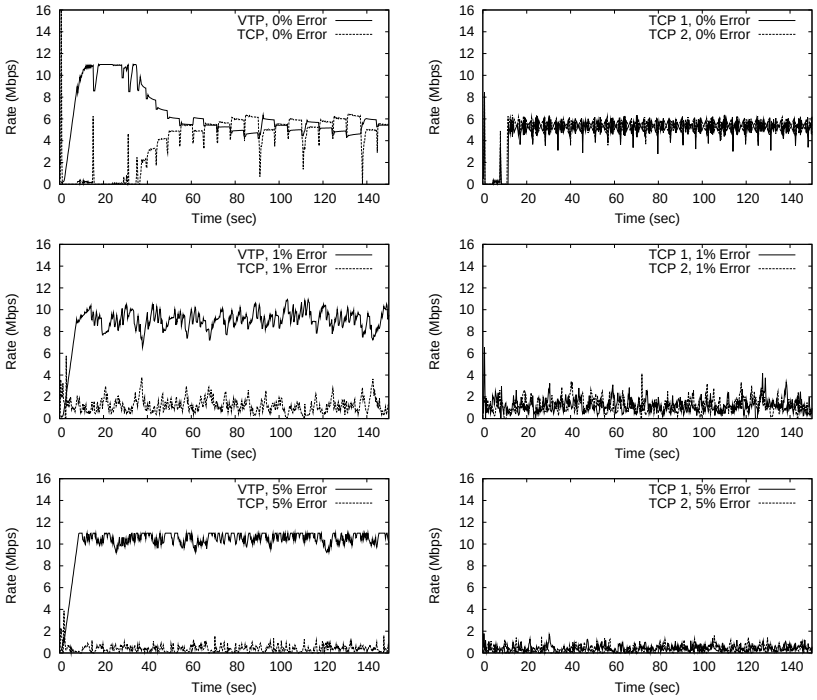

Fig. 7. Opportunistic Friendliness of VTP to TCP under different error rates.

[1] 500 seconds.

Table 1. TCP throughput: coexisting with VTP or itself under different error rates

	TCP Throughput (Mbps): Coexisting with VTP	TCP Throughput (Mbps): Coexisting with Another TCP	TCP Performance Degradation
0% Error	5.09	5.14	1%
1% Error	1.22	1.21	0%
5% Error	0.38	0.38	0%

5.4 Intra-protocol Fairness

By *fairness* we refer to the relationship between flows of the same protocol. To evaluate the fairness property of VTP, we run a set of experiments with two VTP flows sharing the bottleneck. Since both flows have identical network parameters, they should equally share the bottleneck bandwidth. Figure 8 presents the VTP sending rates in different error situations. In all tested cases, VTP flows are able to fairly share the bottleneck while achieving high link utilization in presence of random errors. This confirms that VTP has an excellent property of fairness to itself.

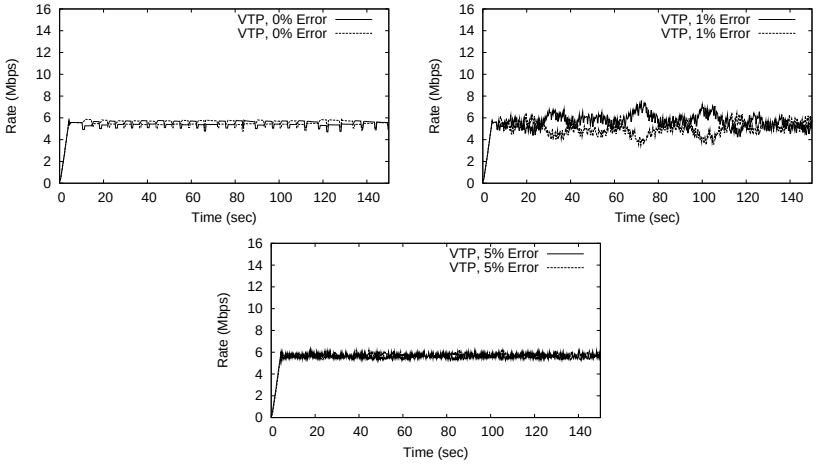

Fig. 8. Fairness between VTP flows.

6 Related Work

In this section we briefly summarize some of the related work. TCP Friendly Rate Control (TFRC) [7] is an equation-based protocol aiming to achieve the

equivalent long-term throughput of TCP with less short-term fluctuation. TFRC per se is not robust to wireless losses, but extensions that can operate in mixed wired-cum-wireless environments have been recently reported. For example, [4] proposes to apply a LDA to the TFRC receiver and exclude error losses from the calculation of packet loss rate p used in the equation. [5] proposes another approach that creates multiple parallel TFRC connections when a single connection is inefficient. We plan to compare VTP to these TFRC extensions in the future work.

Rate Control Scheme (RCS) [17] uses low-priority dummy packets to probe the available bandwidth on the path. RCS is able to estimate an admissible rate similar to the AR and effectively distinguish between error and congestion losses. However, RCS requires all intermediate gateways to implement a multiple-priority mechanism. This feature is currently not available on the Internet.

Explicit Congestion Notification (ECN) [12] is a general method to handle error losses. It is often implemented in conjunction with Random Early Detection (RED) [6]. It stamps a packet with a designated bit to indicate buffer congestion. If the sender detects a loss but no ECN bit is reported, it assumes the loss to be random and does not trigger congestion recovery. The ECN scheme is generally used for TCP but can also be used for streaming. The main limitation again is the cost of the network layer implementation.

Binomial algorithms [3] aim to provide smoother rate control than AIMD for real-time streaming. Wireless losses are not specifically addressed. Streaming Control Transmission Protocol (SCTP) [15] and Datagram Congestion Control Protocol [10] are transport protocols that can be used for real-time streaming. Again they were designed mostly for wired networks and lack a rate control mechanism that handles wireless losses efficiently.

7 Conclusion and Future Work

In this paper we have proposed a new rate control mechanism for the adaptive real-time video streaming protocol VTP. This new protocol measures the end-to-end Achieved Rate (AR) and adjusts the sending rate accordingly when congestion is detected by the Loss Discrimination Algorithm (LDA). Rate decreases and increases are carefully designed so as to mimic the TCP behavior and maintain intra-protocol fairness and opportunistic friendliness to legacy TCP. We have shown via Ns-2 experiments that under all tested error rates (up to 5%), VTP is able to utilize the bandwidth efficiently, while at the same time keeping excellent properties of fairness and friendliness.

To our knowledge, VTP is one of the few truly end-to-end schemes that perform well in the wireless environment without requiring the support from lower layer feedback and AQM mechanisms. In the future we plan to compare VTP to the recently proposed TFRC extensions for the wireless scenario. We are currently in the process of developing a Linux-based implementation of VTP and will carry out testbed/Internet measurements in the near future.

References

1. Allman, M., Paxson, V., Stevens, W.: TCP Congestion Control. RFC 2581 (1999)
2. Balk, A., Maggiorini, D., Gerla, M., Sanadidi, M. Y.: Adaptive MPEG-4 Video Streaming with Bandwidth Estimation. QOS-IP 2003, Milano, Italy (2003)
3. Bansal, D., Balakrishnan, H.: Binomial Congestion Control Algorithms. IEEE Infocom 2001, Anchorage, AK (2001)
4. Cen, S., Cosman, P., Voelker, G.: End-to-end Differentiation of Congestion and Wireless Losses. IEEE/ACM Transactions on Networking, Vol. 11, Iss. 5 (2003)
5. Chen, M., Zakhor, A.: Rate Control for Streaming Video over Wireless. IEEE Infocom 2004, Hong Kong, China (2004)
6. Floyd, S., Jacobson, V.: Random Early Detection gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking, V.1, N.4 (1993)
7. Floyd, S., Handley, M., Padhye, J., Widmer, J.: Equation-Based Congestion Control for Unicast Applications. SIGCOMM 2000, Stockholm, Sweden (2000)
8. Jain, R.: The Art of Computer Systems Performance Analysis. John Wiley and Sons (1991)
9. Kazantzidis, M.: Adaptive Wireless Multimedia. Ph.D. Thesis, Computer Science Dept, UCLA (2002)
10. Kohler, E., Handley, M., Floyd, S.: Datagram Congestion Control Protocol (DCCP). Internert Draft (2004)
11. Mehra, P., Zakhor, A.: TCP-based Video Streaming Using Receiver-driven Bandwidth Sharing. Int'l Packet Video Workshop 2003, Nantes, France (2003)
12. Ramakrishnan, K., Floyd, S., Black, D.: The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (2001)
13. Rappaport, T.: Wireless Communications: Principles and Practice. Prentice Hall PTR, Upper Saddle River, NJ (1996)
14. Rejaie, R., Handley, M., Estrin, D.: Quality Adaptation for Congestion Controlled Video Playback over the Internet. ACM SIGCOMM 1999, Cambridge, MA (1999)
15. Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., Paxson, V.: Stream Control Transmission Protocol. RFC 2960 (2000)
16. Sun, M., Reibman, A. (ed.): Compressed Video over Networks. Marcel Dekker, Inc. (2001)
17. Tang, J., Morabito, G., Akyildiz, I., Johnson, M.: RCS: A Rate Control Scheme for Real-Time Traffic in Networks with High Bandwidth-Delay Products and High Bit Error Rates. IEEE Infocom 2001, Anchorage, AK (2001)
18. Yang, Y., Kim, M., Lam, S.: Transient Behaviors of TCP-friendly Congestion Control Protocols. IEEE Infocom 2001, Anchorage, AK (2001)