Streaming Media Congestion Control Using Bandwidth Estimation[1]

Nadeem Aboobaker, David Chanady, Mario Gerla, and M.Y. Sanadidi

Computer Science Department, University of California, Los Angeles
{nadeem, chanady, gerla, medy}@cs.ucla.edu

Abstract. The fundamental challenge in streaming media over the Internet is to transfer the highest possible quality, adhere to the media play out time constraint, and efficiently and fairly share the available bandwidth with TCP, UDP, and other traffic types. This work introduces the Streaming Media Congestion Control protocol (SMCC), a new adaptive media streaming congestion management protocol in which the connectionís packet transmission rate is adjusted according to the dynamic bandwidth share of the connection. In SMCC, the bandwidth share of a connection is estimated using algorithms similar to those introduced in TCP Westwood. SMCC avoids the Slow Start phase in TCP. As a result, SMCC does not exhibit the pronounced rate oscillations characteristic of traditional TCP, thereby providing congestion control that is more suitable for streaming media applications. Furthermore, SMCC is fair, sharing the bandwidth equitably among a set of SMCC connections. An important advantage is robustness when packet losses are due to random errors, which is typical of wireless links and is becoming an increasing concern due to the emergence of wireless Internet access. In the presence of random errors, SMCC is also friendly to TCP New Reno. We provide simulation results using the ns2 simulator for our protocol running together with TCP New Reno.

1 Introduction

TCP has successfully supported data applications with end-to-end reliable in-order packet communication. With the increase in link bandwidth over the past decade many multimedia applications that stream video and audio over the Internet have emerged. The popularity of such applications has caused multimedia data to be increasingly present in the Internet. Protocols must be developed that fairly share network bandwidth between multimedia and other connection types such as TCP, UDP, etc.

Multimedia is generally not transferred using the pure TCP paradigm because the services provided by TCP are more beneficial to applications requiring reliable data transfer, such as the Web, E-mail, and FTP. The latter applications place greater priority on reliable delivery, rather than the data transfer time. In contrast, the video

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-540-45812-8_28

[1] This research was supported by NSF Grant ANI 9983138

K.C. Almeroth and M. Hasan (Eds.): MMNS 2002, LNCS 2496, pp. 89-100, 2002.

streaming application has stricter transfer latency requirements. The key distinguishing characteristic of such applications is that audio and video data needs to be continuously played out at the client end. This associates a play out time with each transferred data packet. If the packet arrives late, it is useless to the application, and might as well have been lost. A sending rate that does not fluctuate dramatically is therefore more suited for multimedia communications.

SMCC estimates the bottleneck bandwidth share of a connection at the client (receiver) side, using algorithms similar to those introduced in TCP Westwood [4,5,10]. The client does not send acknowledgements to the sender for each received video packet. Instead, a negative acknowledgment (NACK) is returned to the sender when the client perceives a packet loss. This NACK serves two purposes. First, it is a request for retransmission of the lost packet. Depending on whether it can be delivered on time the sender may or may not retransmit the packet. Secondly, in SMCC, a NACK carries to the sender the current Bandwidth Share Estimate (BSE). Thus a NACK message conveys a measure of congestion, and the sender adjusts its sending rate accordingly.

At all times, the sender mimics TCPís congestion avoidance phase by increasing its sending rate by one packet per round trip time (RTT) until a NACK message is received, upon which the sending rate is set to the BSE. After this readjustment in the sending rate, the server resumes a linear sending rate increase of one packet per RTT.

As will be shown in the remainder of this paper, each SMCC sender gets an accurate estimate of the connectionís fair share of the bottleneck bandwidth, and effectively adjusts the sending rate to the changing estimate. This enables the receiver to continuously play out the delivered media, albeit at varying quality, even under highly dynamic network conditions. SMCC is also a fair protocol in that it shares the bottleneck bandwidth fairly among a set of SMCC connections. Finally, SMCC behaves well in random loss environments, since the bandwidth estimate is robust to sporadic packet loss, unlike TCP Reno-like adaptive schemes, which tend to overreact to random errors with significant throughput degradation. Moreover, SMCC is TCP friendly when packet losses are due to random errors.

The remainder of this paper is organized as follows: Section 2 overviews related work. Section 3 presents SMCC, the streaming protocol proposed in this paper. This is followed by an evaluation of SMCC bandwidth estimation accuracy, throughput performance, fairness, and its friendliness to TCP, all in Section 4. Section 5 concludes by summarizing results and discussing current and possible future research directions.

2 Related Work

Recent research has focused on designing TCP friendly protocols that meet the demands of multimedia applications. In many cases these efforts have produced multimedia protocols that behave very similar to TCP.

Time-lined TCP (TLTCP) [11] is one such protocol that adheres strictly to the TCP congestion mechanism, while allowing data to be associated with deadlines. Once the deadline of a section of data has passed the rest of the data associated with that

deadline is dropped. Although this protocol eliminates some of the timing issues introduced by TCP, it still adheres to the reliability and in-order delivery constraints of TCP, which hinder the transfer of streaming media.

SCTP [7] adapts the basic mechanisms of TCP congestion management to multimedia traffic requirements. SCTP allows the disabling of in-order packet delivery and reliable packet delivery, in times of network congestion. However, SCTP still employs the slow-start mechanism of TCP, leading to undesirable sending rate fluctuations.

RAP [13] also mimics the additive increase multiplicative decrease congestion scheme of TCP. While in congestion avoidance phase, the sending rate is increased by one per round-trip-time. Upon detection of congestion, the sending rate is multiplicatively decreased. RAP filters out the throughput variation, caused by the multiplicatively sending rate reductions, by relying on receiver buffering strategies.

Feamster et al. [3] extend the RAP protocol by using non-AIMD algorithms along with receiver buffer management to further reduce sending rate oscillations. Their SQRT algorithm in particular improves throughput variations by reducing the sending rate by sqrt(window size), as opposed to AIMD algorithms where the reduction in transmission rate is proportional to the window size.

TCP Friendly Rate Control (TFRC) [6] adjusts the serverís sending rate upon congestion according to a TCP throughput equation. This equation includes packet loss rate (due to congestion) and RTT, which are monitored and maintained at the receiver. TFRC, however cannot handle random losses in the sense that such losses will cause unwarranted loss of throughput.

One key design distinction between SMCC and the protocols mentioned above is that SMCC does not require the acknowledgement of every data packet. This is a significant saving since multimedia data packets are typically small in size, ~200 bytes. Requiring a 40byte acknowledgment for each data packet automatically adds a 20% overhead to the protocol.

An issue that has recently been addressed in the Rate Control Scheme (RCS) proposed by Tang et al. [14] is robustness of the rate adaptation algorithm to high link loss rates on wireless channels with high Bandwidth X Delay products. Such cases correspond to satellite links as well as wireless access links to remote Internet servers. This scenario is likely to attract considerable interest as wireless Internet access to multimedia services is gaining popularity. The RCS scheme is a clever mechanism based on dummy packets to distinguish between congestion loss and random loss. The proper operation of RCS, however, requires the implementation of a priority level ìbelow TCP best effortî in the router. Such support is not provided by most Internet routers. SMCC on the other hand provides robustness to random errors by virtue of the intrinsic insensitivity of the Bandwidth Share Estimate to isolated losses. No extra network level support is required.

3 Streaming Media Congestion Control Protocol (SMCC)

The protocol we propose in this paper, SMCC, 1) is able to adapt the sending rate according to the connection estimated bandwidth share, 2) is fair to existing

connections, and 3) does not suffer from pronounced sending rate oscillations typical of most window protocols.

There is no congestion window in SMCC, although SMCC adjusts its sending rate so as to mimic TCPís congestion avoidance phase with its linear bandwidth probing, i.e. one extra packet is sent per round trip time so long as no congestion is detected. When congestion is encountered in SMCC, the sending rate is reduced to the current Bandwidth Share Estimate (BSE), and linear probing is continued. Never is the sending rate dropped to 1 packet per round trip time following a timeout, as in TCP, unless the current BSE dictates this value. In other words, the ì slow-startî phase with exponential growth as in TCP is not present in SMCC.

3.1 SMCC Sender Operation

After connection setup using a three-way-handshake between SMCC server and client is complete, the sender starts to send media data. In our implementation, we assume the network can handle the lowest quality sending rate, and use that rate as the initial sending rate.

The initial round trip time estimate is gathered from the handshaking process. After each successive RTT, the sender increases the sending rate by one packet. This has the effect of mimicking TCP congestion avoidance phase. The sender readjusts its RTT estimate once every round trip time, by requesting an acknowledgement for the first data packet in every new RTT window.

Whenever the sender receives a NACK, it resets the sending rate to the BSE contained in the NACK message (the BSE computation is discussed below), and resumes linear probing. The sender can determine if there is sufficient time to retransmit the lost packet, based on information it gets from the client regarding the receiverís current buffer and current play out time.

3.2 SMCC Receiver Operation

With each data packet received, the client may recalculate the BSE. Upon receiving a data packet that the sender requests to be acknowledged, the receiver sends an ACK for that packet. As mentioned above, this message exchange is used to set the RTT at the sender side. Once a lost packet is detected, a NACK message is sent containing the current bandwidth estimate. As noted above, it is the senderís responsibility to determine if the packet should be retransmitted.

3.3 Bandwidth Calculation in SMCC

Just as in TCP Westwood, the Bandwidth Share Estimate, i.e. the estimate of the rate to be used by the sender in order to share the bottleneck bandwidth fairly, is determined by exponentially averaging rate samples. However, where TCP Westwood measures ACK arrivals at the sender, SMCC uses the inter-arrival time between two subsequent packets at the client (receiver) to calculate a rate sample on

the forward path of the connection. The advantage of the SMCC approach is that it measures the rate on the forward path of the connection, filtering out the effects of congestion on the reverse path.

Basically, the client side calculates the connection bandwidth share using the Receiver Based Packet Pair (RBPP) method described in [12]. RBPP requires the use of two consecutively sent packets to determine a bandwidth share sample. The adherence of SMCC to the RBPP sequence number requirement is fundamental for not overestimating bandwidth share and thus enhancing fairness and friendliness of SMCC.

A second constraint is to make sure that ëtime compressioní has not occurred on the consecutive packets. Here we define time compression as occurring when the difference between arrival times of the packets is less than the difference between sending times of the packets [12]. If the packets are time compressed, they are not used in the BSE calculation, as the resulting estimate would be too aggressive. The intent is to estimate bandwidth only up to the serverís (sender) instantaneous sending rate, even if more bandwidth is available.

If two consecutively received packets have passed the two tests above, they are used to calculate a bandwidth sample as follows:

$$\text{Bandwidth} = s_2 / (a_2 - a_1) \quad (1)$$

Where s_2 is the size of the second packet, a_1 and a_2 are the arrival times of the first and second packet respectively. This bandwidth sample is then used just as the rate estimation in TCP Westwood; it is plugged into the exponential filter to produce the current BSE.

4 Experimental Results

We performed several simulation experiments to assess the performance of SMCC with respect to throughput, fairness, and TCP-friendliness. Unless otherwise stated, all simulations use the topology depicted in Figure 1, with all connections going through the same bottleneck link using RED queue management, and sending data in the same direction. Simulations were run for 200sec. To filter out transient system behavior, all simulations were run for 100 seconds before measurements were taken. Table 1 shows the constant configuration parameters.

Table 1. Simulation properties

Delay of all side links	3ms
Side links capacity	5Mbps
Bottleneck buffer	10*RTT*bottleneck B/W
FTP/SMCC data packets	200 bytes
FTP/SMCC ack packet size	40 bytes
Min. Threshold	0.05*Bottleneck buffer
Max. Threshold	0.5*Bottleneck buffer
q_weight	0.002

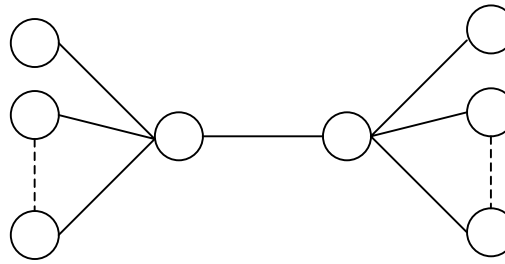

Figure 1. Simulation Topology

4.1 Accuracy of Bandwidth Estimation

First, we want to verify the accuracy of SMCCís bandwidth estimation. This is achieved by changing the link capacity during the lifetime of a SMCC connection. We alter the link capacity by introducing CBR traffic at different rates over the bottleneck link. Figure 2 shows how SMCC adapts to changing bottleneck bandwidth with no competing SMCC or TCP connections. The figure shows both SMCCís BSE value, and its throughput as a function of time. The figure shows that SMCC calculates the BSE accurately to within 10% of the available bottleneck bandwidth.

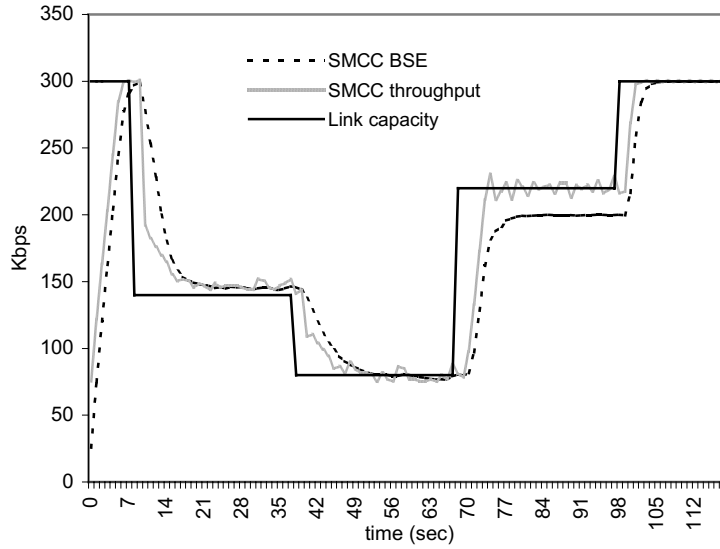

Figure 2. Adaptation of BSE to changing bottleneck bandwidth

4.2 Throughput Behavior

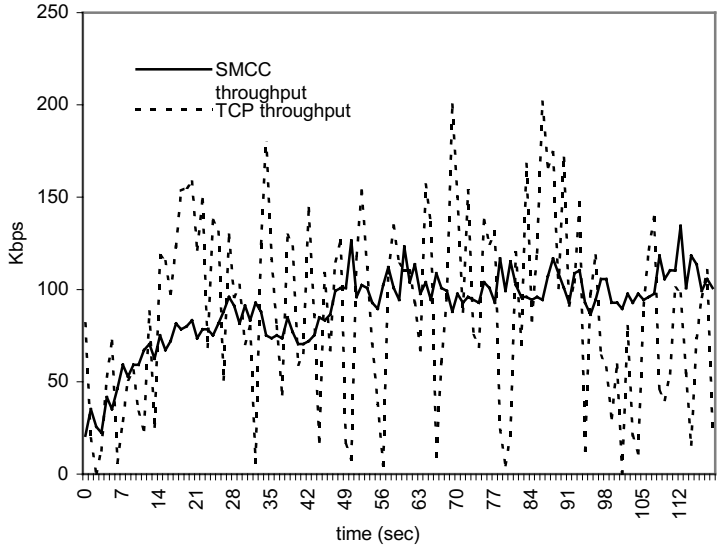

Figure 3. Throughput curve of 1 SMCC and 1 FTP/TCP New Reno connections

Figure 3 plots the throughput curve of one SMCC and one TCP New Reno connection, in 1-second intervals for a 120 second simulation. A total of 5 TCP and 5 SMCC connections were present, but only one throughput curve of each type of connection is shown if Figure 3. The bottleneck capacity was 1Mbps, with a 10msec delay, thus the fair share per connection is 100Kbps. Figure 3 shows that while still receiving its fair share value of 100Kbps, SMCCís throughput curve is much smoother than TCP New Renoís oscillatory sending rate. This is because SMCC does not reduce its sending rate by one half or to a minimum of one packet per round trip time as TCP does. Rather, SMCC adjusts its sending rate to the BSE, and never performs an exponential rate increase. Thus, SMCC takes longer to reach the fair share value, but it does not suffer from pronounced sending rate oscillations, and is thus suitable for streaming media such as audio and video.

TCP was originally designed to work in a totally wired environment, where a packet loss meant that a buffer overflow occurred somewhere in the network, and therefore, served as a sign of congestion. However, in the case of lossy links, packets may not only be lost due to congestion, but also due to random errors. TCP Reno incorrectly interprets this event as congestion, and unnecessarily reduces its window size with consequent reduction in throughput. This prohibits TCP from fully utilizing the error-prone link.

SMCC is more robust to random loss because of the relative insensitivity of bandwidth estimation to sporadic errors. In fact, the loss of an isolated packet has only marginal impact on the bandwidth estimation. No time out and slow start are incurred in SMCC.

To illustrate the robustness of SMCC in the wireless domain, Figure 4 presents results from a simulation with one SMCC and one TCP New Reno connection running over a 0.3Mbps bottleneck link with 35ms delay, and an error rate varying from 1 to 20%. We see that SMCC is able to maintain a large percentage of the 150Kbps fair share value independent of the error rate, while TCP throughput continues to degrade as the error rate increases.

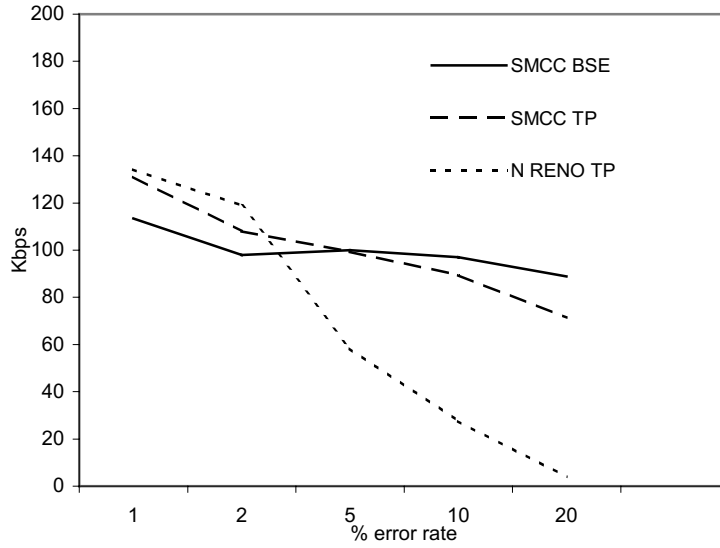

Figure 4: 1 SMCC versus 1 TCP New Reno connection running over a 300Kbps bottleneck lossy link with a 35ms delay

4.3 Fairness

The fairness of a protocol is a measure of how a number of connections running the same protocol share the bottleneck bandwidth. The fairness of SMCC is studied below by simultaneously running 10 SMCC connections. The time averaged throughput and BSE of each connection is shown in Figure 5. The results show that the connections vary little in their throughput and BSE averages. We also ran experiments changing the number of connections from 2 to 50. The bottleneck capacity is scaled to maintain a fair share value of 100Kbps, while the delay was held constant at 10ms. All runs showed good fairness behavior: Jainís Fairness Index was higher than .99 for all runs.

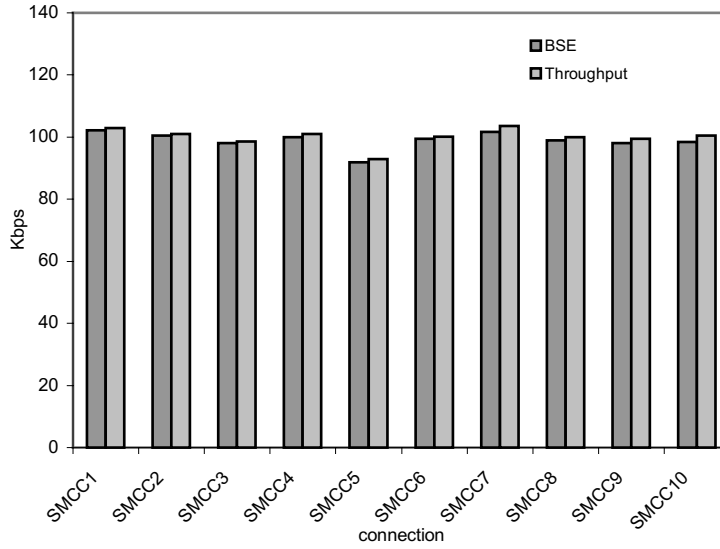

Figure 5: Time Averaged BSE and Throughput on 10 SMCC connections

4.4 TCP Friendliness

We also tested friendliness for a variety of network conditions, including number of flows, bottleneck link delay, bottleneck bandwidth, and random errors. Varying the number of flows did not show significant impact on friendliness. Therefore we will limit our discussion in this section to how bottleneck delay, bandwidth and random errors impact friendliness. All simulations in this section were performed using a balanced set of 10 SMCC and 10 TCP New Reno connections.

We use the Efficiency/Friendliness Tradeoff Graph introduced in [4] to better visualize the efficiency/friendliness tradeoff behavior when introducing SMCC connections to share a common bottleneck link with TCP New Reno connections. The following two experiments produce a single point on the graph:

1. A simulation with 20 TCP New Reno flows is run for calibration, to establish a reference value. The throughput of each flow, and the total link utilization, are measured.
2. Another simulation is then run with half of the TCP flows replaced with SMCC flows. The new throughput and utilization are measured.

Let t_{R1} be the average throughput of the TCP New Reno flows in the first simulation, and U1 be the total link utilization. Similarly, let t_{R2} be the average throughput of the TCP New Reno flows in the second simulation, and U2 be the total link utilization by all connections, that is SMCC as well as TCP connections. We then define:

$$\text{Efficiency Improvement } E = U2/U1 \tag{2}$$

$$\text{Friendliness } F = (t_{R2}/t_{R1}) \tag{3}$$

For each network scenario of interest, E and F are determined and an (F, E) point is placed on the Efficiency / Friendliness Tradeoff graph. We expect that, in most cases, an increase in Efficiency (E) is compensated for by a decrease in Friendliness (F).

We performed a set of experiments, varying the bottleneck capacity from 5 to 20Mbps. The bottleneck delay was also varied between 10, 50, 100 and 150ms. We also introduced random dropping of packets over the bottleneck link to simulate random loss associated with data transfer over a wireless channel.

Figure 6 shows the efficiency/friendliness tradeoff behavior when SMCC connections share a bottleneck link with TCP New Reno connections. Each curve in Figure 6 corresponds to fixed bottleneck bandwidth and error rate. Moreover, the points on each curve represent the impact of sequentially changing the bottleneck delay. For convenience, the arrows mark the direction of increasing bottleneck delay.

We see that for all three 0% error rate cases, as the delay increases, SMCC reduces TCPís throughput without any resulting gain in efficiency. The reason for this is that in these scenarios there is not much room for improvement in efficiency, since TCP by itself can achieve a high utilization of the bottleneck link. There is a friendliness problem in the current version of SMCC, represented by the flat portions of the 0% error curves. This is one area of further needed enhancement, and can be addressed by applying adaptive share estimation methods, as in [4,5], as opposed to the current packet pair model. The packet pair model and the associated available bandwidth estimation have been shown to be too aggressive at times. Results presented in [4,5] have shown major improvements in friendliness using the rate estimation model.

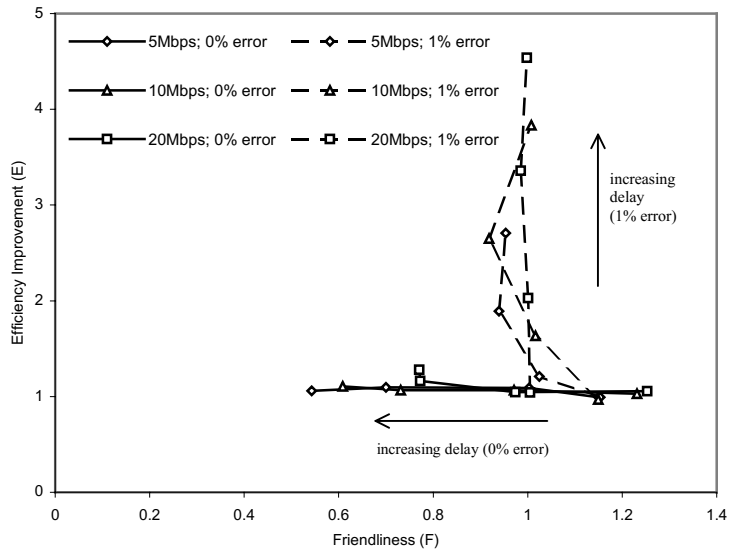

Figure 6: Efficiency / Friendliness tradeoff graph

Figure 6 also shows that SMCC is friendly to TCP New Reno in the random error case (wireless scenario).

As we mentioned in Section 4.2, when a packet loss is due to random error, TCP unnecessarily reduces its window size, with consequent reduction in throughput. This prohibits TCP from fully utilizing the capacity of an error-prone link. This inefficiency is more pronounced as the bandwidth X delay product increases.

In SMCC, the loss of an isolated packet has only marginal impact on the bandwidth estimation, thus it achieves a higher utilization of an error-prone link. When SMCC connections are introduced, they readily use the residual bandwidth that TCP is incapable of utilizing. This is apparent in Figure 6 where the 1% error rate curves show that SMCC remains friendly to TCP irrespective of bottleneck capacity and delay. Furthermore, the gain in total utilization resulting from the introduction of SMCC increases as the bandwidth X delay product increases.

4.5 Comparison of SMCC to Other Streaming Protocols

As mentioned in Section 2, the common challenge for any streaming protocol is to adjust its sending rate to an appropriate value once congestion occurs. Most protocols so far proposed adhere to the multiplicative decrease paradigm of TCP. Others, such as TFRC attempt to calculate the throughput that TCP would achieve on the same connection, and use that calculation to set the sending rate. The drawback of these methods is that in striving to achieve the same throughput as TCP with lower sending rate oscillations, such protocols can inherit the fundamental limitations of TCP. [13] shows that the RAP protocol using fine-grain rate adaptation can achieve the same throughput as TCP regardless of the bottleneck delay. Similarly, TFRC [6] is based on an underlying control equation that ensures that the protocol uses no more bandwidth, in steady-state, than a TCP conforming protocol running under comparable conditions. From figures 5 and 6, we see that in the random error cases, achieving the same throughput as TCP, leaves the bottleneck link severely underutilized. SMCC allows the streaming application to access this unused bandwidth in these cases, without hindering existing TCP connections. As wireless Internet access becomes increasingly popular and technology increases the capacity of links, this issue will become increasingly important. Naturally, in the attempt to improve utilization, SMCC at times also grabs bandwidth from TCP (and vice-versa). We plan to address these issues in depth in future work.

5 Conclusion

This paper has presented Streaming Media Congestion Control (SMCC), a protocol based on bandwidth estimation concepts, which provides smooth sending rate changes and good utilization when available network bandwidth fluctuates. SMCC is fair with respect to other SMCC flows. It is reasonably friendly to TCP, particularly with lossy links. SMCC mimics the linear probing for additional bandwidth in TCP congestion avoidance phase. Upon detection of a lost packet, which the receiver explicitly

NACKs along with the current BSE, the sender adjusts its sending rate to the current Bandwidth Share Estimate. SMCC mimics the Congestion Avoidance phase of TCP congestion control, but never the Slow Start phase.

SMCC dictates the sending rate of a streaming media application, but not the quality of the stream. The quality adaptation is a separate issue. It not only depends upon the current sending rate, but also upon the receiverís buffer, and the user preferences and profile as well.

The resulting behavior shows that SMCC throughput is quite smooth, as compared to the fluctuating behavior of conventional TCP. This is a desirable property for video streaming applications. Robustness to link errors and random loss was also demonstrated via simulations.

Areas of further research include: friendliness of bandwidth estimation method particularly when congestion, not random error, is the predominant cause of packet losses; and the extension to multicasting applications (both single and multi-source). This protocol will also be tried on interactive media, although the tight time delay constraints will introduce more stringent buffer space requirements.

References

1. I.F Akyildiz, G. Morabito, S. Palazzo. TCP Peach: A New Congestion Control Scheme for Satellite IP Networks, IEEE/ACM Transactions on Networking, Vol. 9, No. 3, pp. 307-321, June 2001.
2. D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker. Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms. In Proceedings of Sigcomm 2001.
3. N. Feamster, D. Bansal, and H. Balakrishnan. On The Interactions Between Layered Quality Adaptation and Congestion Control for Streaming Video. 11th International Packet Video Workshop, Kyongju, Korea. May 2001.
4. M. Gerla, M. Sanadidi, K. Ng, M. Valla, R.Wang. Efficiency/Friendliness Tradeoff in TCP Westwood. ISCC 2002, Taormina, Italy.
5. M. Gerla, M. Sanadidi, M. Valla, R. Wang. Adaptive Bandwidth Share Estimation in TCP Westwood. To appear in Globecom 2002, Taipei, Taiwan.
6. M. Handley, J. Padhya, S. Floyd, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. Internet Engineering Task Force, July 2001.
7. Internet RFC 2960
8. K. Lai, M. Baker. Measuring Link Bandwidths Using a Deterministic Model of Packet Delay. Proceedings of SIGCOMM 2000.
9. A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel. Scalable On-Demand Media Streaming with Packet Loss Recovery. ACM SIGCOMM Aug. 2001.
10. S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In Proceedings of Mobicom 2001.
11. B. Mukherjee and T. Brecht, Time-lined TCP for the TCP-friendly Delivery of Stremining Media, Proceedings of ICNP 2000.
12. V. Paxon. Measurements and Analysis of End-to-End Internet Dynamics. Ph.D. thesis, University of California, Berkeley, April 1977.
13. R. Rejaie, M. Handley, D. Estrin. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. University of Southern California, Information Sciences Institute. July 1998.
14. J. Tang, G. Morabito, I. Akyildiz, and M. Johnson. RCS: A Rate Control Scheme for Real-Time traffic with High Bandwidth-Delay Products and High Bit Error Rates. Infocom 2001.