Portland State University

PDXScholar

Computer Science Faculty Publications and Presentations

Computer Science

12-1997

Flow and Congestion Control for Internet Streaming Applications

Shanwei Cen

Calton Pu

Jonathan Walpole
Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/compsci_fac

 Part of the Computer Sciences Commons, and the Digital Communications and Networking Commons

Let us know how access to this document benefits you.

Citation Details

Shanwei Cen ; Jonathan Walpole and Calton Pu, "Flow and congestion control for Internet media streaming applications", Proc. SPIE 3310, Multimedia Computing and Networking 1998, 250 (December 29, 1997); doi:10.1117/12.298426; http://dx.doi.org/10.1117/12.298426.

This Article is brought to you for free and open access. It has been accepted for inclusion in Computer Science Faculty Publications and Presentations by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

DOCKET ALARM

Find authenticated court documents without watermarks at docketalarm.com.

Flow and Congestion Control for Internet Media Streaming Applications

Shanwei Cen[a], Jonathan Walpole[b] and Calton Pu[b]

[a]Tektronix, Inc., Video and Networking Division
P.O.Box 500, M/S 50-490 Beaverton, Oregon 97077 USA

[b]Department of Computer Science, Oregon Graduate Institute
P.O.Box 91000, Portland, Oregon 97291 USA

ABSTRACT

The emergence of *streaming* multimedia players provides users with low latency audio and video content over the Internet. Providing high-quality, best-effort, real-time multimedia content requires adaptive delivery schemes that fairly share the available network bandwidth with reliable data protocols such as TCP. This paper proposes a new flow and congestion control scheme, SCP (Streaming Control Protocol), for real-time streaming of continuous multimedia data across the Internet. The design of SCP arose from several years of experience in building and using adaptive real-time streaming video players. SCP addresses two issues associated with real-time streaming. First, it uses a congestion control policy that allows it to share network bandwidth fairly with both TCP and other SCP streams. Second, it improves smoothness in streaming and ensures low, predictable latency. This distinguishes it from TCP's jittery congestion avoidance policy that is based on linear growth and one-half reduction of its congestion window. In this paper, we present a description of SCP, and an evaluation of it using Internet-based experiments.

Keywords: Internet, flow control, congestion control, adaptive systems, streaming multimedia systems

1. INTRODUCTION

The real-time distribution of continuous audio and video data via streaming multimedia applications accounts for a significant, and expanding, portion of the Internet traffic. Many research prototype media players have been produced, including Rowe's MPEG player,[1] the authors' distributed video player,[2] and McCanne's vic,[3] etc. Over the past two years, many commercial streaming media players have also been released, such as RealAudio and RealVideo players, Microsoft Netshow, and Vxtreme.

Common characteristics of media streaming applications include their need for high bandwidth, smooth data flow, and low and predictable end-to-end latency and latency variance. In contrast, the Internet is a best-effort network that offers no quality of service guarantees, and is characterized by a great diversity in network bandwidth and host processing speed, wide-spread resource sharing, and a highly dynamic workload. Hence, in practice, Internet-based applications experience large variations in available bandwidth, latency, and latency variance.

To address these problems the new generation of Internet-based multimedia applications use techniques such as buffering and feedback-based adaptation of presentation quality. Buffering is used at the sender, receiver, or both, to mask short-term variations in the available bandwidth or latency. Through adaptation, applications scale media quality in one or more quality dimensions to better utilize the currently available bandwidth. For example, real-time play-out can be preserved in the presence of degraded bandwidth by adaptively sacrificing other presentation quality dimensions such as video frame rate or spatial resolution.

Dynamic adaptation is a powerful approach, but it requires new flow and congestion control mechanisms for accurate discovery and appropriate utilization of the available network bandwidth. Congestion control mechanisms must determine, dynamically, the share of the network bandwidth that can be fairly used by the adaptive application in the presence of competing traffic. If the mechanisms are not sensitive enough

Please send correspondence to Shanwei Cen at shanwei.cen@tek.com

SPIE Vol. 3310 • 0277-786X/97/$10.00

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

to competing traffic the potentially high multimedia data rates could cause serious network congestion. On the other hand, if they are too sensitive they will under-utilize the bandwidth and presentation quality will be forced to degrade unnecessarily. In practice, such congestion control mechanisms must share bandwidth fairly among competing multimedia streams and must be compatible with TCP congestion control,[4] since TCP is the base protocol for currently-dominant HTTP and FTP traffic.

Flow control mechanisms should attempt to minimize end-to-end latency due to buffering, and maximize the smoothness of the data stream. Reliability through indefinite data retransmission is usually not desirable, since streaming applications can often tolerate some degree of quality degradation due to data loss, but are usually less tolerant of the delay introduced by the retransmission of lost data.

This paper presents a flow and congestion control scheme, called SCP (Streaming Control Protocol), for unicast media streaming applications. Like TCP, SCP employs sender-initiated congestion-detection through positive acknowledgment, and uses a similar window-based policy for congestion avoidance. The similarities in congestion avoidance between SCP and TCP make SCP robust and a good network citizen, and enable SCP and TCP to share the Internet fairly.

Unlike TCP, when the network is not congested, SCP invokes a hybrid rate- and window-based flow control policy that maintains smooth streaming with maximum throughput and low latency. Conversely, TCP repeatedly increases its congestion window size until packets are lost, then halves its window size (a behavior of TCP-Reno). One consequence of this behavior is that TCP sessions exhibit burstiness and develop long end-to-end latency due to the build up of packets in network router buffers. This behavior is particularly problematic over PPP links since some PPP servers have buffers that hold 15 seconds of data, or more. Also unlike TCP, SCP does not retransmit data lost in the network. Thus it avoids the associated unpredictability in latency and wasted bandwidth.

Further salient features of SCP include its support for rapid adaptation in the presence of drastic bandwidth fluctuations, such as those that occur when mobile computers migrate among different network types,[5] and its support for streaming-specific operations such as pausing.

This paper is organized as follows. Section 2 presents the design of SCP. Section 3 follows with an analysis of SCP's steady-state bandwidth sharing. Section 4 describes the implementation of SCP. Section 5 explains the experimental results. Section 6 then briefly discusses other congestion control schemes, and finally, Section 7 concludes the paper and discusses future work.

2. THE DESIGN OF SCP

A unicast streaming scenario with SCP involves a sender, which streams media packets over a network connection in real-time to a receiver. SCP policies are implemented at the sender side. Each packet contains a sequence number, and for each packet, SCP records the time at which it is sent, and initiates a separate timer*. The receiver acknowledges each packet it receives by returning an ACK containing the sequence number of the packet to the sender. Based on the reception of ACKs and expiration of timers, SCP monitors the available bandwidth, detects packet loss, and adjusts the size of its congestion window to control the flow and avoid network congestion. To achieve this task SCP maintains the following internal state variables and parameter estimators.

- *state* — The current state. SCP has several states, each corresponding to a specific network and session condition and associated flow and congestion control policy.
- W_l — The size of the congestion window (in number of packets).
- W — The number of outstanding packets sent but not acknowledged. When $W < W_l$, the congestion window is open, and more packets can be sent, otherwise it is closed and no packets can be sent.
- W_{ss} — The threshold of W_l for switching from the slow-start policy to the steady-state policy.
- $\hat{T}_{brtt}$ — An estimator of the base RTT (round trip time) – the RTT of a packet sent when the network is otherwise quiet.
- $\hat{T}_{rtt}$ — An estimator of the recent average RTT.
- $\hat{D}_{rtt}$ — An estimator of the standard deviation of the recent RTT.

*For clarity, we explain the design of SCP based on a timer per packet, but this aspect is optimized in the implementation.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

State	Network and session condition	Congestion window adjustment policy
slowStart	Available bandwidth not discovered yet	SCP opens the congestion window exponentially by increasing the window size by one upon the receipt of each ACK.
steady	Available bandwidth being fully utilized	SCP maintains appropriate amount of buffering inside the network to gain sufficient throughput, avoid excessive buffering or buffer overflow, and trace the changes in available bandwidth.
congested	The network is congested.	SCP backs off multiplicatively by halving the window size. Persistent congestion results in exponential back-off.
paused	No outstanding packet in the network	When a new packet is sent, SCP shrinks the window size and invokes the slow-start policy.

Table 1. SCP states, network and session conditions, and flow and congestion control policies

- $\widehat{rto}$ — An estimator of the timer duration.
- $\hat{r}$ — An estimator of the receiving packet rate.

As long as the congestion window is open, the sender streams packets at a rate no more than $\frac{W_l}{\hat{T}_{brtt}}$, instead of sending them out in bursts, so as to improve the smoothness of the stream. When an ACK is received, or a timer expires, the congestion window size W_l is adjusted using a policy associated with the current state.

SCP adopts window-based policies similar to those of TCP for slow-start, and exponential back-off upon network congestion. It also estimates RTT in a way similar to TCP. Therefore, many parameters and state variables have counterparts in TCP. However, SCP differs from TCP in that it has a base RTT estimator $\hat{T}_{brtt}$ and a packet rate estimator $\hat{r}$. The use of these estimators to ensure smooth streaming and low latency is discussed in the following sections.

2.1. Overall architecture

SCP is based on the observation that excessive packets in the round-trip network connection accumulate in buffers at network routers and switches and lead to an increased RTT. As the accumulation of packets increases, these buffers overflow and packets are dropped. The goal of SCP is to ensure smooth streaming at a suitably high throughput, but without causing excessive buffering or congestion in the network. To achieve this goal, SPC monitors fluctuations in the available bandwidth and pushes an appropriate number of additional packets into the network connection. If network congestion is detected, SCP reacts immediately with exponential back-off. Depending on the condition of the network and the streaming session under control, SCP is in one of the following four states: *slowStart*, *steady*, *congested* or *paused*. Each state is associated with a specific condition and congestion window size adjustment policy as listed in Table 1.

SCP handles events that indicate changes in network and session conditions. Such events include indications that: the SCP session becomes paused or active; the available network bandwidth is fully utilized; the network is congested; the network interface has been switched. Upon these events, SCP updates its internal state and possibly switches to a new policy. Figure 1 shows the events that SCP handles and its associated state transitions.

2.2. Congestion window adjustment policies and state transitions

Initialization: Upon initialization, SCP sets $W = 0$, $W_l = 1$ and $W_{ss} = L_w$, where L_w is a limit on the congestion window size. $\hat{T}_{brtt}$, $\hat{T}_{rtt}$ and $\hat{D}_{rtt}$ are set to infinity. $\widehat{rto}$ is set to an initial default value. After initialization, SCP enters the *paused* state. Transmission of the first packet brings SCP to the *slowStart* state.

Slow-start: The slow-start policy is invoked after initialization or when SCP resumes from a pause. Its goal is to quickly grow the congestion window and discover the available network bandwidth. W_l is incremented by 1 when an ACK is received in-order, thus doubling the congestion window after each RTT amount of time. SCP leaves the *slowStart* state upon detecting events indicating network congestion, full utilization of available bandwidth, pause in streaming, or network interface switch.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

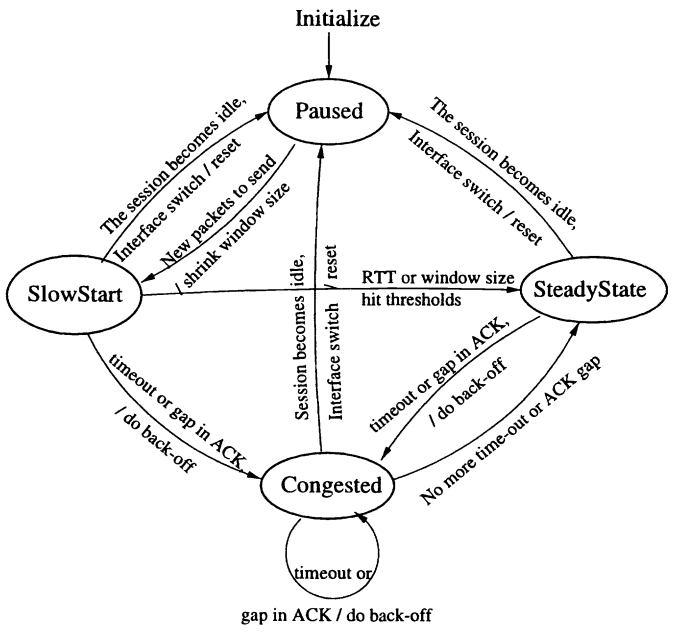

Figure 1. SCP state transition diagram

Steady-state smooth streaming: By pushing an appropriate number of extra packets, the steady-state policy enables sufficient utilization of the available bandwidth while avoiding over-buffering. It also traces the fluctuations in available bandwidth. In the *steady* state, estimation of packet rate $\hat{r}$ is enabled. Whenever a new $\hat{r}$ value is available, congestion window size W_l and threshold W_{ss} are adjusted according to Equation 1 below, where W_Δ is a constant referred to as the window-size incremental coefficient.

$$W_l = W_{ss} = \hat{r}\hat{T}_{brtt} + W_\Delta \tag{1}$$

The idea behind the flow control defined by Equation 1 is that SCP assumes that $\hat{r}$ is an approximation of the network bandwidth (in terms of packet rate) available to the session, calculates the bandwidth-delay product of the network connection with minimum buffering $\hat{r} \times \hat{T}_{brtt}$, and adjusts its congestion window size W_l accordingly. The product $\hat{r} \times \hat{T}_{brtt}$ is the amount of data SCP should keep outstanding inside the network in order to maintain sufficient throughput with minimum buffering. Since the network is shared and highly dynamic, the bandwidth available to the session under control fluctuates. If SCP just sets its congestion window size to $\hat{r} \times \hat{T}_{brtt}$, when the available bandwidth decreases, $\hat{r}$ would decrease, and SCP would reduce its congestion window size and thus the amount of outstanding data. However, if the available bandwidth increases, there is no way for SCP to detect that. This is because the receiving packet rate $\hat{r}$ could not be increased unless SCP increases the sending packet rate first, but SCP would not increase its sending packet rate by increasing the congestion window size unless it observes a increased $\hat{r}$. To solve this "chicken-and-egg" type of problem, SCP pushes W_Δ amount of extra outstanding data, which are held in router and switch buffers. When the network available bandwidth increases, releasing the extra data buffered in the network results in an increase in packet rate $\hat{r}$, which in turn results in larger amount of outstanding data as defined by the increased W_l. It can be seen that the increase in W_l is no more than additive, which is what TCP does in its steady state. Due to the smoothing effect of lowpass filtering in estimation of $\hat{r}$, W_l will be changed gradually, causing smooth changes in throughput. Also, with the flow control policy stated in Equation 1, though at the beginning $\hat{r}$ may not be a reliable estimation of the actual available bandwidth, if the network is stable enough, SCP will eventually bring $\hat{r}$ to the actual bandwidth through iterations. Overall, the steady-state policy is able to converge to a stable throughput, and to trace the fluctuation in available bandwidth smoothly.

Upon detecting network congestion, SCP backs off and enters the *congested* state. SCP enters the *paused* state when the session is paused, or when a switch in network interface is detected.

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.