

Coordinating Congestion Management and Bandwidth Sharing for Heterogeneous Data Streams

Venkata N. Padmanabhan
Microsoft Research
padmanab@microsoft.com

Abstract

Many of the busiest servers in the Internet consist of clusters of nodes that serve out data of heterogeneous types. At a given point in time, a user could be receiving multiple concurrent data streams, both real-time and non-real-time, that originate at a single server node, multiple nodes in the same server cluster, or nodes in separate clusters. In this paper, we argue that significant performance benefits can accrue from a coordinated approach to congestion management and bandwidth sharing across the concurrent data streams. We discuss several challenges that arise in this context and outline a coordination architecture that addresses these challenges.

1 Introduction

Many of the busiest servers¹ in the Internet consist of large clusters of nodes that serve out data of heterogeneous types. The kinds of information served include text, images, and streaming audio/video. The information is typically transferred from servers to clients via unicast. This tends to be the case even for audio/video streams because often these correspond to playback rather than live transmission, so unicast rather than multicast is used.

A user may, in general, be receiving multiple data streams, both real-time and non-real-time, simultaneously. For example, as the HTML and the inline images for a news story are being downloaded, the user may also play a short video clip of the news event together with background audio commentary for a richer multimedia experience. In some sense, the audio and video clips are "inline" just as images have always been in Web pages. The relative importance of the data streams could vary dynamically. At certain times, the quality of the video stream may be of the utmost importance to the user whereas at other times quick download of static images may take priority.

From the viewpoint of data transport, the challenge is to coordinate these data streams, both to ensure overall

network well-being and to optimize user-perceived performance. Coordinated congestion management ensures that the data streams learn about and respond appropriately to congestion events at shared bottleneck link(s). Coordinated bandwidth sharing enables bandwidth to be apportioned to the streams according to their dynamically-varying, user-specified priorities.

The task of coordinating the disparate data streams is particularly challenging because of the many possible configurations of the server node(s) transmitting the data:

1. All of the data streams could originate from a single physical server node.
2. The data streams could originate from multiple nodes in the same server cluster. For instance, in large Internet server sites such as CNN and MSNBC, distinct nodes typically serve the real-time and the non-real-time streams. This is so because of the administrative convenience of managing the two kinds of nodes separately as well as because of their specialized hardware requirements (e.g., large and fast disks are needed for video servers).
3. The streams could originate from distinct server clusters. For instance, the user may be receiving a traffic cam video stream from the highway department site and listening to background audio commentary from a sports site while also browsing other unrelated Web sites. Even if the user is receiving content that is originally from a single site (e.g., CNN), the individual data streams may originate from widely separated server nodes if distinct dissemination infrastructures are used for different kinds of content (e.g., Squid [SQD] for Web content and the Real Broadcast Network [RBN] for streaming media).

Thus we believe that when dealing with heterogeneous data streams, it is imperative to consider the possibility of the streams originating from distinct locations.

The location of the servers transmitting the data streams impacts not only the feasibility and granularity of coordination across the server nodes, but also the appropriateness of such coordination. Coordination is appropriate only if the streams share a bottleneck link. Moreover, differential treatment of the streams within the network (e.g., *diffserv* [RFC2475]) would complicate the coordination issue.

¹ In this paper, we use the terms "sender" and "server" interchangeably, and likewise the terms "receiver", "client", and "user". This is solely for ease of exposition.

In this paper, we discuss the potential benefits of coordination, the challenges that arise, and an architecture that addresses these challenges. We focus specifically on the problem of coordination of a heterogeneous collection of real-time and non-real-time streams. This problem is more general than the coordination of a homogeneous set of streams (e.g., the set of concurrent TCP connections corresponding to the inline images on a Web page). Also, users may find it quite natural to tune in to a set of concurrent, heterogeneous data streams. For instance, people often like to read while listening to audio in the background.

In the remainder of this paper, we present a brief overview of our prior work on TCP Session (Section 2), discuss potential benefits of coordination between concurrent data streams (Section 3), discuss specific challenges that arise and potential solutions (Section 4), outline a coordination architecture (Section 5), survey related work (Section 6), and finally present a summary (Section 7).

2 Overview of TCP Session

The goal of TCP Session [Pad98] is to coordinate multiple concurrent TCP connections between a pair of hosts. A typical application of TCP Session is the coordination of concurrent TCP connections between a Web server and a client, where the connections correspond to the individual constituents of a Web page. TCP Session has three components:

1. **Integrated congestion control:** maintains a session-wide congestion window to make the ensemble of TCP connections as well-behaved and responsive to congestion as an individual TCP connection.
2. **Connection scheduling:** implements a scheduler and provides an API for applications to explicitly control the (relative) bandwidth share of concurrent connections.
3. **Integrated loss recovery:** exploits packet ordering information across concurrent connections to improve the effectiveness of data-driven loss recovery.

While quite effective in the problem space it is targeted at, TCP Session falls short in the general setting outlined in Section 1 because it only considers TCP connections between a host-pair. Nevertheless, its key ideas carry over to the new setting, as we elaborate in the following sections.

3 Opportunities for Optimization

We begin by discussing several opportunities for performance optimization via coordination between concurrent real-time and non-real-time data streams.

3.1 Initialization of Congestion State

It is desirable for short connections, such as Web transfers, to utilize the available network bandwidth effectively. However, the short length of these connections does not permit extensive probing, so the sender has to resort to guesswork based on past

information (such as the old congestion window size). This has the potential of degrading overall network performance when the guess is incorrect.

A long-running real-time stream could be an excellent source of information for a new TCP connection that shares a bottleneck link with it and is trying to determine the state of the network in order to initialize its congestion control state. RTCP receiver reports [RFC1889] would keep the real-time sender informed of the packet loss rate and the delay jitter along the end-to-end network path. The new TCP connection could start up aggressively if the real-time stream has experienced relatively little congestion in the recent past and conservatively if the opposite is true.

3.2 Improved Loss Recovery

Short TCP connections tend to suffer because of the reduced effectiveness of data-driven loss recovery due to the insufficient number of packets in the pipe to trigger the threshold number of duplicate acks needed for TCP fast retransmission [Ste94]. The duplicate acks are needed to disambiguate packet loss from packet reordering. In a manner akin to integrated loss recovery in TCP Session, packets belonging to a (long-lived) real-time stream could be used to augment the effectiveness of such disambiguation by providing a regular “heartbeat” in the form of a steady stream of packets. Synchronized sender timestamps or a monotonically-increasing sequence number (such as the IP ID) could be used to infer the relative ordering of packets belonging to the TCP connection and the real-time stream. The underlying assumption is that all packets traverse a common path through the network.

3.3 Bandwidth Sharing Across Streams

As noted in [Pad98,GB99], user-perceived performance in the context of the Web is not just a function of the overall data transfer rate but is also dependent on the (dynamically varying) perceptual importance of the individual streams. When the total bandwidth available to all the data streams being received by a client is limited, it is desirable to be able to explicitly control the (relative) allocation to each stream (based on a user/application-level notion of their relative importance) rather than have the streams arrive at a bandwidth share based purely on network dynamics.

The options available for adjusting the data rate of a real-time stream includes changing the frame rate, frame size, quantization granularity, sampling rate, coding algorithm, etc. For non-real-time streams, there is the option of speeding up or slowing down the download of objects (since there is not a real-time constraint) in addition to transcoding it or choosing one of the appropriate size among multiple discrete object formats.

4 Challenges and Potential Solutions

We now turn to the challenges that arise in exploiting the opportunities mentioned above and discuss ways of addressing these challenges.

4.1 Shared Bottleneck Link

Coordinated congestion management and bandwidth sharing across data streams are both predicated on the streams sharing a common bottleneck link. After all, if there is not a common bottleneck link, the data streams might as well operate independently of one another.

We propose three techniques, to be used in some combination, to determine when the data streams share a bottleneck link:

1. **Topology discovery:** If the streams share (much of) the same end-to-end path, it is likely that they also share a bottleneck link. This would tend to be the case when data streams originate at the same or nearby servers and are destined for the same or nearby clients (cases 1 and 2 in Section 1). However, this technique could fail if the bottleneck is close to network periphery, e.g., a dialup access link.

2. **Correlating delay and/or loss observed at the receiver:** If the (one-way) delay and/or loss rate observed at the receiver on two streams is highly correlated, it is likely that the streams share a bottleneck link. Sender timestamps carried by the individual packets may be used to compute the one-way delay. Since we are only interested in the *correlation* between the one-way delay observed by multiple streams, synchronization between the clocks at the sender and the receiver is not essential. Figure 1 plots the one-way delay as a function of time for two connections that share a bottleneck link. In general, we expect a delay-based scheme to be more robust than a loss-based one, especially in the presence of drop-tail routers. And both schemes are likely to be more effective with long-lived streams than with short-lived streams (e.g., Web connections).

We believe the correlation technique would be useful when there is a single (dominant) bottleneck link, such as a dialup access link. However, the presence of multiple congested links, only a subset of which are shared among the streams, would render the technique ineffective. We believe the findings of research such as MINC [CDH+99] would be indicative of the potential of the correlation technique.

3. **Enhanced ECN:** As we proposed in [Pad96], routers could augment their explicit congestion notifications (ECN) with unique, router-specific IDs. A receiver could then deduce the common points of congestion, if any, shared by multiple streams. Such a mechanism would be quite useful even if it were deployed only on the subset of routers that are likely to be attached to bottleneck links (e.g., access routers).

The router ID could take the form of an IP option. Multiple instances of the option may be permitted to

accommodate multiple congested links along the end-to-end path. The router ID could be the IP address of the outgoing interface or a shorter hash to save space. In the

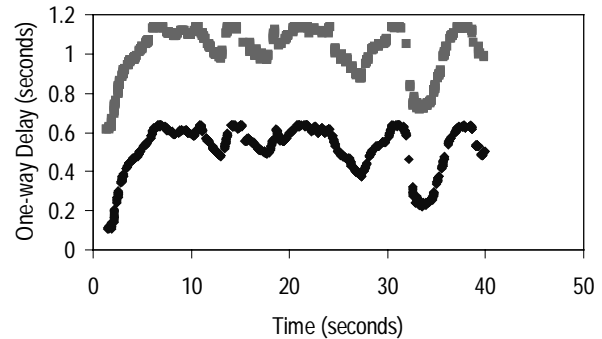


Figure 1 The one-way delay experienced by packets of two TCP connections that share a 1.5 Mbps bottleneck link. We have shifted the upper curve up by 0.6 seconds to improve clarity.

latter case, it may be a good idea for a router to periodically re-seed the hash computation to avoid a persistent collision with another (congested) router.

4.2 Differentiated Services Mechanisms

The use of differentiated services (*diffserv*) mechanisms, such as priority dropping or priority scheduling, could adversely impact the schemes discussed in Section 3. For instance, if packets of a real-time audio stream are assigned a higher drop priority than those of non-real-time TCP streams, then the packet loss rate observed by the audio stream can no longer be used to determine how aggressive TCP should be upon start up (Section 3.1). However, if packet scheduling is still FIFO, the utility of the real-time stream in aiding loss recovery in the TCP connections (Section 3.2) would remain unaffected.

However, if priority scheduling were employed, neither state initialization nor loss recovery using real-time streams would be appropriate, but bandwidth sharing (Section 3.3) may still be appropriate.

In general, *diffserv* poses an interesting challenge to determining if there is a shared bottleneck link. While a low-priority and a high-priority stream may experience very different loss rates, the bandwidth used by the high-priority stream may well have a bearing on the loss rate observed by the low-priority one, so coordinated bandwidth sharing may still be appropriate. While the loss rates experienced by the two streams may not be correlated, the delays would still be if the scheduling is FIFO. The enhanced ECN scheme from Section 4.1 would also be effective in this context.

4.3 Differing Notions of Fairness and Priority

Even with a shared bottleneck link, it is unclear exactly what would constitute fair sharing of the bottleneck link bandwidth. For example, consider two clients, A and B. Suppose that A is receiving one data

stream each from the CNN and the MSNBC servers whereas B is only receiving a data stream from CNN. If all three streams traverse a common bottleneck link, how should they share its bandwidth? From the viewpoint of the clients, an equal allocation of bandwidth to the two streams destined for A put together and the single stream destined for B may be considered fair. However, this may not necessarily be fair from the viewpoint of the servers, since CNN would be receiving a larger share of the bottleneck bandwidth than MSNBC. As another example, consider that, in general, only a subset of the streams destined for a client contain data sought by the user; the rest contain data "thrust" upon the user (e.g., advertisements) by servers. So the servers and the clients may be at odds on how to apportion bandwidth to the streams. We believe that the resolution of these issues is essentially a matter of policy.

5 Outline of Coordination Architecture

In this section, we present a brief outline of an architecture to coordinate congestion management and bandwidth sharing across streams that do not necessarily share end points (Figure 2).

5.1 Multiple Levels of Coordination

Our architecture includes coordination at three different levels:

1. **Intra-host:** Each server node coordinates the set of data streams that it transmits to the same client host, using fine-grained, packet-level mechanisms, much as in TCP Session.

2. **Intra-cluster:** Multiple nodes in a server cluster (and likewise clients on a LAN) share congestion information amongst themselves over a local multicast group (à la the coordination bus mechanism [McC96]). The granularity of coordination would tend to be coarser than in the intra-host case — congestion epoch-level rather than packet-level. The nodes adapt the frequency of updates sent on the coordination bus based both on the overall volume of control traffic and the magnitude of change that the updates correspond to. For instance, an update is likely to be sent sooner if it corresponds to a more significant change in the network conditions.

3. **Inter-cluster:** Servers on different clusters coordinate their actions using feedback from the common client (or neighboring clients) they are transmitting data to. The granularity of coordination is even more coarse-grained than an intra-cluster setting — typically at the level of user actions indicating a shift in the relative importance of the individual data streams.

5.2 Receiver-based Control

The receiver (client) plays a key role in our architecture:

1. It conveys the user's (dynamically-varying) preferences (translated into relative bandwidth shares for individual streams) to the servers.

2. It provides congestion feedback to the servers. Ideally, the feedback message would identify the congested router (if the enhanced ECN scheme from Section 4.1 is in use), so that server nodes can effectively

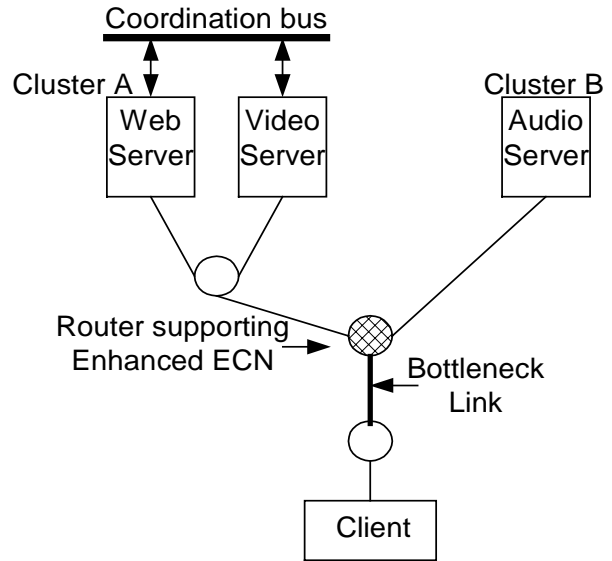


Figure 2 A depiction of the proposed architecture.

aggregate congestion information from multiple sources.

3. It facilitates inter-cluster coordination where otherwise server-to-server communication would be expensive, and moreover, the servers may not be aware of the need to coordinate.

5.3 Distributed Coordination

Bandwidth sharing among streams originating from the same server node can be accomplished using a server-local mechanism akin to connection scheduling in TCP Session. However, the task is much harder when the streams originate from widely separated servers. We consider two alternative approaches:

1. **Congestion feedback filtering:** The receiver adjusts the relative frequency of congestion notifications sent to each server, while ensuring that the aggregate congestion response is at the appropriate level.

2. **Explicit flow control:** The receiver clamps the bandwidth used by a stream to be no more than a fixed (absolute) level using a flow control mechanism such as the TCP receiver-advertised window, or the RTSP "speed" header field or "pause" method [RFC2326].

While explicit flow control is conceptually simpler, it suffers from the drawback that it does not adapt well to dynamic variation in the available bandwidth. We believe that both alternatives have a place in our architecture.

6 Related Work

There has been a fair amount of research in the general area of coordinating multiple concurrent data streams. Schemes aimed at coordinating congestion

management include TCP Control Block Interdependence [RFC2140], our previous work on TCP Session [Pad98], extensions thereof to include UDP streams [BRS99], and router-based coordination [SAA+99]. Efforts aimed at improving user-perceived performance via the explicit coordination of concurrent streams include connection scheduling in TCP Session [Pad98], progressive Web data delivery [GB99], and receiver-driven data transport in WebTP [GCM+99]. While this collective body of research has addressed several relevant problems, the main limitation, in our opinion, is that the individual proposals focus only on coordination across data streams between a pair of hosts and/or they only consider TCP streams. Neither of these simplifications is appropriate in the context of this paper.

7 Summary

In this paper, we have argued that clients on the Internet are increasingly more likely to be communicating simultaneously with one or more servers via real-time and non-real-time data streams. As such, the coordination of congestion management and bandwidth sharing across the data streams is highly desirable. We have outlined the potential benefits of such coordination, some challenges that arise, and a coordination architecture that addresses these challenges. We are investigating these issues in greater detail in ongoing research.

Acknowledgements

We thank Rajeev Koodli, Lili Qiu, K. K. Ramakrishnan, and the anonymous NOSSDAV reviewers for their thoughtful comments.

References

- [BRS99] H. Balakrishnan, H. Rahul, S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts", Proc. ACM SIGCOMM, September 1999.
- [CDH+99] R. Caceres et al., "Multicast-based Inference of Network-Internal Characteristics: Accuracy of Packet Loss Estimation", Proc. IEEE Infocom, March 1999
- [GCM+99] R. Gupta, M. Chen, S. McCanne, J. Walrand, "WebTP: A Receiver-Driven Web Transport Protocol", University of California at Berkeley, USA, 1998.
- [GB99] J. Gilbert, R. Brodersen, "Globally Progressive Interactive Web Delivery", Proc. IEEE Infocom, March 1999.
- [McC96] S. McCanne, "Scalable Compression and Transmission of Internet Multicast Video", Ph.D. Thesis, University of California at Berkeley, USA, December 1996.
- [Pad96] V. N. Padmanabhan, "Receiver-oriented Data Transport with Persistent Sessions", Qualifying Exam Proposal, University of California at Berkeley, USA, October 1996. <http://www.cs.berkeley.edu/~padmanab/research/quals-proposal.ps>
- [Pad98] V. N. Padmanabhan, "Addressing the Challenges of Web Data Transport", Ph.D. Thesis, University of California at Berkeley, USA, September 1998. <http://www.cs.berkeley.edu/~padmanab/phd-thesis.html>.
- [RBN] "Real Broadcast Network", <http://www.real.com/solutions/rbn>
- [RFC1889] H. Schulzrinne et al., "RTP: A Transport Protocol for Real-Time Applications", RFC-1889, IETF, January 1996.
- [RFC2140] J. Touch, "TCP Control Block Interdependence", RFC-2140, IETF, April 1997.
- [RFC2326] H. Schulzrinne et al., "Real Time Streaming Protocol", RFC-2326, IETF, April 1998.

- [RFC2475] S. Blake et al., "An Architecture for Differentiated Services", RFC 2475, IETF, December 1998.
- [SAA+99] S. Savage et al., "Detour: A Case for Informed Internet Routing and Transport", IEEE Micro, Vol. 19, No. 1, January/February 1999.
- [SQD] "Squid Internet Object Cache", <http://squid.nlanr.net>
- [Ste94] W. R. Stevens, "TCP/IP Illustrated, Volume 1", Addison-Wesley Publishers, 1994.