ISSN No. 0976-5697

Volume 2, No. 3, May-June 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

# Multimedia Streaming in Multicast Environment

Mayank Sharma
Associate Professor (IT),
Aurora's Engineering College
Bhongir, Andhra Pradesh, India
Mayank_sharma04@yahoo.com

Pragati.G
Aurora's Engineering College
Andhra Pradesh, India
Pragati_g@yahoo.com

O.Nagamani*
Aurora's Engineering College
Bhongir, Andhra Pradesh, India
nagmani53@gmail.com

***Abstract:*** **Streaming media** is multimedia that is constantly received by and presented to an end-user while being delivered by a streaming provider. The name refers to the delivery method of the medium rather than to the medium itself. The distinction is usually applied to media that are distributed over telecommunications networks, as most other delivery systems are either inherently streaming or inherently non-streaming.[3] The verb 'to stream' is also derived from this term, meaning to deliver media in this manner. Internet television is a commonly streamed medium. TCP is widely used in commercial multimedia streaming systems, with recent measurement studies indicating that a significant fraction of Internet streaming media is currently delivered over HTTP/TCP. These observations motivate us to develop analytic performance models to systematically investigate the performance of TCP for both live and stored-media streaming.

***Keywords :***Media Streaming, Multimedia, TCP, Multicast.

## I. INTRODUCTION

In recent years, there have been an explosive growth of multimedia applications over the Internet. All major news networks such as ABC and NBC now provide news with accompanying video clips. Several companies, such as MovieFlix [1], also offer video on demand to broadband subscribers. However the quality of videos being streamed over the Internet is often of low quality due to insufficient bandwidth, packet loss, and delay. To view a DVD quality video from an on demand video service, a customer must down- load either the entire video or a large portion of the video before playback time in order to avoid pauses caused by insufficient bandwidth during a streaming session. Thus, many techniques have been proposed to enable efficient multimedia streaming over the Internet. The source coding community has proposed scalable video [2][3], error-resilient coding, and multiple description [4] for efficient video streaming over the best-effort networks such as the Internet. A scalable video bit stream is coded in such a way to enable the server to easily and efficiently adapt the video bit rate to the current available bandwidth. Error-resilient coding and multiple description are aimed at improving the quality of the video in the presence of packet loss and long delay caused by retransmission. Channel coding techniques are also used to mitigate long delay for real-time applications such as video conferencing or IP-telephony [5]. The main disadvantages of these approaches are first, specialized codecs are required and second, their performances are highly affected by the network traffic conditions.

From a network infrastructure perspective, Differentiated Services [6][7] and Integrated Services [8][7] have been proposed to improve the quality of multimedia applications by providing preferential treatments to various applications based on their bandwidth, loss, and delay requirements. More recently, path diversity architectures that combine multiple paths and either source or channel coding have been proposed to provide larger bandwidth, and to combat efficiently against packet loss [9][10][11]. Nonetheless, these approaches cannot be easily deployed as they require significant changes in the network infrastructure.

The most straightforward approach is to transmit standard-based multimedia via existing IP protocols. The two most popular choices are TCP and UDP. A single TCP connection is not suitable for multimedia transmission because its congestion control may cause a large fluctuation in the sending rate. Unlike TCP, an UDP-based application is able to set the desired sending rate. If the network is not too much congested, the UDP throughput at the receiver would approximately equal to the sending rate. Since the ability to control the sending rate is essential to interactive and live streaming applications, majority of multimedia streaming systems use UDP as the basic building block for sending packets over the Internet.

However, UDP is not a congestion aware protocol since it does not reduce its sending rate in presence of network congestion, and therefore potentially results in a congestion collapse. Congestion collapse occurs when a router drops a large number of packets due to its inability to handle a large amount of traffic from many senders at the same time. TCP-Friendly Rate Control Protocol (TFRC) has been proposed for multimedia streaming with UDP in order to incorporate TCP-like congestion control mechanism [12]. Another drawback of using UDP is its lack of reliable transmission and hence the application must deal with the packet loss.

Based on these drawbacks of UDP, we propose a new receiver-driven, TCP-based system for multimedia streaming over the Internet. In particular, our proposed system, called MultiTCP, is aimed at providing resilience against short-term insufficient bandwidth by using Multicast for the same application. Furthermore, our system enables the application to achieve and control the sending rate during congested period, which in many cases, cannot be achieved using a single TCP connection. Finally, our proposed system is implemented at the application layer, and hence, no kernel modification to TCP is necessary.

The rest of the paper is organized as follows. In Section 2, we describe the two major drawbacks of using TCP for multimedia streaming: short-term in- sufficient bandwidth and lack of precise rate con- trol. These drawbacks motivate the use of multiple TCP connections in our proposed system, which is described in Section 3. In Section 4, we demonstrate the performance of our system based on simulations results using NS[13]. We then describe other related works that utilize multiple network connections in Section 5. Finally, we summarize our contributions in Section 6.

## II. DRAWBACKS OF TCP FOR MULTIMEDIA STREAMING

As discussed briefly in Section 1, TCP is unsuitable for multimedia streaming due partly to its fluctuating throughput and its lack of precise rate control. TCP is designed for end-to-end reliability and fast congestion avoidance. To provide end-to-end reliability, a TCP sender retransmits the lost packets based on the packet acknowledgment from a TCP receiver. In order to have fast response to network congestion, TCP controls the sending rate based on a window-based congestion control which works as follows. The sender keeps track of a window of maximum number of unacknowledged packets, i.e., packets that have not been acknowledged by the receiver. In the steady state, the sender increases the window size W by 1/W upon successfully receiving an acknowledged packet, or equivalently, it increases the sending rate by one packet per round trip time. Upon encountering a loss, the window size is reduced by half, or equivalently, the sending rate is cut in half. In TCP, the receiver has the ability to set a maximum window size for the unacknowledged packets, hence imposing a maximum sending rate. Thus, in a non-congestion scenario, the application at the receiver can control the sending rate by setting the window size appropriately. On the other hand, during congestion, the actual throughput can be substantially low as the maximum window size may never be reached.

Based on the above discussion, we observe that a single packet loss can drop the TCP throughput abruptly and the low throughput lingers due to the slow increase of the window size. If there is a way to reduce this throughput reduction effect without modifying TCP, we can effectively provide higher throughput with proper congestion control and reliable transmission. In addition, if there is a way to control the TCP sending rate during congestion, then TCP can be made suitable for multimedia streaming. Unlike non real-time applications such as file transfer and email, precise control of sending rate is essential for interactive and live streaming applications due to several reasons. First, sending at too high a rate can cause buffer overflow in certain receivers with limited buffer such as mobile phones and PDAs. Second, sending at a rate lower than the coded bit rate results in pauses during a streaming session, unless a large buffer is accumulated before playback. In the following section, we propose a system that can dynamically distribute streaming data over Multicast using Multiple TCPs per application to achieve higher throughput and precise rate control. The control is performed entirely at the receiver side and thus, suitable for streaming applications where a single server may serve up to thousands of receivers simultaneously.

## III. MULTICASTING USING TCP

As mentioned in Section 2, the throughput reduc- tion of TCP is attributed to the combination of (a) reduction of the sending rate by half upon detection of a loss event and (b) the slow increase of sending rate afterward or congestion avoidance. To alleviate this throughput reduction, one can modify TCP to (a) reduce the sending rate by a small factor other than half upon detection of a loss, or (b) speed up the congestion avoidance process, or (c) combine both (a) and (b). There are certain disadvantages associated with these approaches. First, these changes affect all TCP connections and must be performed by recompiling the OS kernel of the sender machine. Second, changing the decreasing multiplicative factor and the additive term in isolated machines may potentially lead to instability of TCP in a larger scale of the network.

Third, it is not clear how these factors can be changed to dynamically control the sending rate. As such, we propose a different approach: instead of using a traditional, single TCP connection, we use multiple TCP connections for a multimedia streaming application. Our approach does not require any modification to the existing TCP stack or kernel. Figure 1 shows a diagram of our proposed Multicast using TCP system.

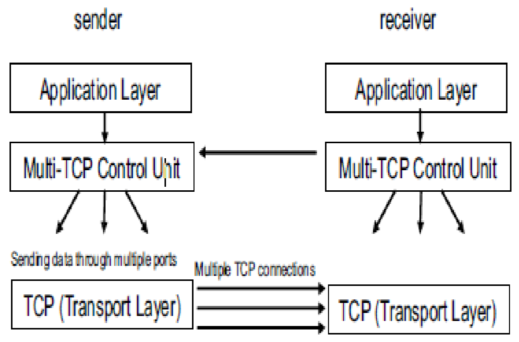

Figure 1: Multicast using TCP system

The MultiTCP control unit is implemented immediately below the application layer and above the transport layer at both the sender and the receiver. The MultiTCP control unit at the receiver receives the input specifications from streaming application which include the streaming rate and the throughput resilience. The throughput resilience can be thought of as the amount of throughput reduction an application can tolerate in presence of sudden burst traffic. A higher throughout resilience leads to a lower short-term throughput reduction.

TheMultiTCP control unit at the receiver measures the actual throughput and uses this information to control the rate and the throughput reduction by using multiple TCP connections and dynamically changing receiver's window size for each connection. In the next two sections, we show how multiple TCP connections can mitigate the throughput reduction problem in a lightly loaded network and describe our mechanism to maintain the desired throughput in a congested network.

### *A. Alleviating Throughput Reduction In Lightly Loaded Network*

In this section, we analyze the throughput reduction problem in a lightly loaded network and show how it can be alleviated by using multiple TCP connections. When there is no congestion, the receiver can control the streaming rate in a single TCP connection quite accurately by setting the maximum the receiver's window size Wmax. The effective throughput during this period is approximately equal to

$$T = \frac{W_{max}MTU}{RTT} \qquad (1)$$

where RTT denotes the round trip time, including both propagation and queuing delay, between the sender and the receiver. MTU denotes the TCP maximum transfer unit, typically set at 1000 bytes. If a loss event occurs, the TCP sender instantly reduces its rate by half as shown in Figure 2(a). As a result, the area of the inverted triangular region in Figure 2(a) indicates the amount of data that would have been transmitted if there were no loss event. Thus, the amount of data reduction D equals to

$$D = (\frac{1}{2})(\frac{W_{max}MTU RTT}{2})(\frac{W_{max}}{2RTT}) = \frac{W^2_{max}MTU}{8} \qquad (2)$$

Note that the time it takes for the TCP window to increase from Wmax/2 to Wmax equals to WmaxRTT/2 since the TCP window increases by one every round trip time. Clearly, if there are a burst of loss events during a streaming session, the total throughput reduction can potentially be large enough to deplete the start up buffer, causing pauses in the playback.

Now let us consider the case where two TCP connections are used for the same application. Since we want to keep the same total streaming rate Wmax/RTT as in the case of one TCP connection, we set Wmax = Wmax/2 for each of the two connections as illustrated in Figure 2(b). Assuming that only a single loss event happens in one of the connection, the total throughput reduction would be equal to

$$D' = \frac{(W'_{max}MTU)}{8} = \frac{(W^2_{max}MTU)}{32} = \frac{D}{4} \qquad (3)$$

Equation (3) shows that, for a single loss event, the throughput reduction of using two TCP connections is four times less than that of using a single TCP connection. Even in the case when there are simultaneously losses on both connections as indicated in Figure 2(c), the throughput reduction is half of that of the single TCP. In general, let N denote the number of TCP connections for the same application and n be the number of TCP connections that suffer simultaneous losses during short congestion period, the amount of throughput reduction equals to

$$D_N = \frac{nW^2_{max}MTU}{N^2} \qquad (4)$$

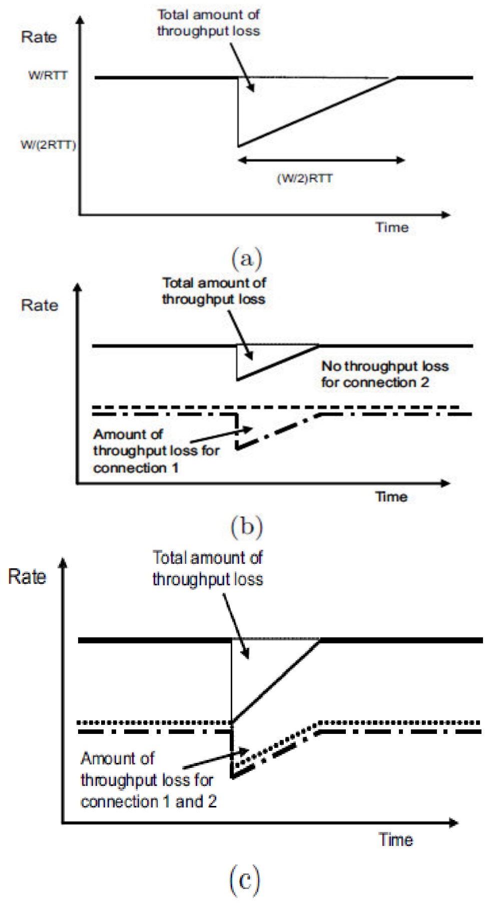

Figure 2: Throughput reduction for a) one TCP connection with singe loss b) Two TCP connections with single loss c) Two TCP connections with Double Losses.

### *B. Control Streaming Rate in a Congested Network*

In the previous section, we discuss the throughput reduction problem in a lightly loaded network and show that using multiple TCP connections can alleviate the problem. In a lightly loaded network condition, one can set the desired throughput Td by simply setting the receiver window Wmax = TdRTT/MTU.

However, in a moderately or heavily congested network, the throughput of a TCP does not depend on Wmax, instead, it is determined by the degree of congestion. This is due to the fact that in a non-congested network, i.e. without packet loss, TCP rate would increase additively until maxMTU/RTT is reached, after that the rate would remain approximately constant at WmaxMTU/RTT. However, in a congested network, a loss event would most likely occur before the sending rate reaches its limit and cut the rate by half, resulting in a throughput lower than WmaxMTU/RTT.

A straightforward method for achieving a higher throughput than the available TCP throughput would be to use multiple TCP connections for the same application. Using multiple TCP connections results in a larger share of the fair bandwidth. Hence, one may argue that this is unfair to other TCP connections.

On the other hand, one can view this approach as a way of providing higher priority for streaming applications over other non time-sensitive applications under resource constraints. We also note that one can use UDP to achieve the desired throughput. However unlike UDP, using multiple TCP connections can provide (a) congestion control mechanism to avoid congestion collapse, and (b) automatic retransmission of lost packets. Assuming multiple TCP connections are used, there are still issues associated with providing the desired throughput in a congested network.

In order to maintain a constant throughput during a congested period, one possible approach is to increase the number of TCP connections until the measured throughput exceeds the desired one. This approach suffers from a few drawbacks. First, the total resulting throughput may still exceed the desired throughput by a large amount since the sending rate of each additional TCP connection may be too high. Second, if only a small number of TCP connections are required to exceed the desired throughput, this technique may not be resilient to the sudden increase in traffic as analyzed in Section 3.1. A better approach is to use a larger number of TCP connections but adjust the receiver window size of each connection to precisely control the sending rate. It is undesirable to use too many TCP connections as they use up system resources and may further aggravate an already congested network. In practice, our algorithm maintains a relatively stable number of TCP connections while varies the size of the receiver windows to achieve the desired throughput.

## IV. RESULTS

In this section, we show simulation results using NS to demonstrate the effectiveness of our MultiTCP system in achieving the required throughput as compared to the traditional single TCP approach. Our simulation setup consists of a sender, a receiver, and a traffic generator connected together through a router to form a dumb bell topology as shown in Figure 3. The bandwidth and propagation delay of each link in the topology are identical, and are set to 6 Mbps and 20 milliseconds, respectively. The sender streams 800 kbps video to the receiver continuously for a duration of 1000s, while the traffic generator generates cross traffic at different times by sending packets to the receiver using either long term TCPs or short bursts of UDPs. In particular, from time t = 0 to t = 200s, there is no cross traffic. From t = 200s to t = 220s and t = 300s to t = 340s, bursts of UDPs with rate of 5.5 Mbps are generated from the traffic generator node to the receiver. At t = 500s the traffic generator opens 15 TCPs connections to the receiver, and 5 additional TCP connections at t = 750s. We now consider this setup under three different scenarios: (a) the sender uses only one TCP connection to stream the video, while the receiver sets the receiver window size to 8, targeting at 800 kbps throughput, (b) the sender and the receiver use our MultiTCP system to stream the video with the number TCP connections limited to two, and (c) the sender and the receiver also use our proposed MultiTCP system, except the number of TCP connections are now set to five.

## V. RELATED WORK

There has been previous work on using multiple network connections to transfer data. For example, path diversity multimedia streaming framework [10][11][9] provide multiple connections on different path for the same application. These work focus on either efficient source or channel coding techniques in conjunction with sending packets over multiple approximately independent paths. On the other hand, our work aims to increase and maintain the available throughput using multiple TCP connections on a single path. There is also a related work using multiple connections on a single path to improve throughput of a wired-to-wireless streaming video session [15]. This work focuses on obtaining maximum possible throughput and is based TFRC rather than TCP. On the other hand, our work focuses on eliminating short term throughput reduction of TCP due to burst traffic and providing precise rate control for the application. As such, the analysis and rate control mechanism in our paper are different from those of [15]. Another related work is Streaming Control Transmission Protocol (SCTP)[16], designed to transport PSTN signaling messages over IP networks. SCTP allows user's messages to be delivered within multiple streams, but it is not clear how it can achieve the desired throughput in a congestion scenario. In addition, SCTP is a completely new protocol, as such the kernel of the end systems need to be modified. There is also other work related to controlling TCP bandwidth. For example, the work in [17] focuses on allocating bandwidth among flows with different priorities. This work assumes that the bottleneck is at the last-mile and that the required throughput for the desired application is achievable using a single TCP connection. On the other hand, our work does not assume the last-mile bottleneck, and the proposed MultiTCP system can achieve the desired throughput in variety of scenarios.

Also, the authors in [18], use weighted proportional fair sharing web flows to provide end-to-end differentiated services. The work in [19] uses the receiver advertised window to limit the TCP video bandwidth in VPN link between video and proxy servers. Finally, the authors in [20] propose a technique for automatic tuning of receiver window size in order to increase the throughput of TCP.

## VI. CONCLUSION

We conclude our paper with a summary of contributions. First, we propose and implement a receiver-driven, TCP-based system MultiTCP for multimedia streaming over the Internet using multiple TCP connections for the same applications. Second, our proposed system is able to provide resilience against short-term insufficient bandwidth due to traffic bursts.

Third, our proposed system enables the application to control the sending rate in a congested scenario, which cannot be achieved using traditional TCP. Finally, our proposed system is implemented at the application layer, and hence, no kernel modification to TCP is necessary. The simulation results demonstrate that using our proposed system, the application can achieve the desired throughput in many scenarios, which cannot be achieved by traditional single TCP approach.

## VII. REFERENCES

[1] MovieFlix, http://www.movieflix.com

[2] W. Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and tcpfriendly transport protocol," IEEE Transactions on Multimedia, vol. 1, pp. 172–186, june 1999.

[3] G. De Los Reyes, A. Reibman, S. Chang, and J. Chuang, "Error-resilient transcoding for video over wireless channels," IEEE Transactions on Multimedia, vol. 18, pp. 1063–1074, june 2000.

[4] A. Reibman, "Optimizing multiple description video coders in a packet loss environment," in Packet Video Workshop, April 2002.

[5] H. Ma and M. El Zarki, "Broadcast/multicast mpeg-2 video over wireless channels using header redundancy fec strategies," in Proceedings of The International Society for Optical Engineering (SPIE), November 1998, vol. 3528, pp. 69–80.

[6] S. Blake, D. Black, M. Carson, E. Davis, Z.Wang, and W. Weiss, "An architecture for differentiated services," in RFC2475, December 1998.

[7] Z. Wang, Internet QoS, Architecture and Mech- anism for Quality of Service, Morgan Kaufmann Publishers, 2001.

[8] P.White, "Rsvp and integrated services in the internet: A tutorial," IEEE Communication Mag- azine, pp. 100–106, May 1997.

[9] T. Nguyen and A. Zakhor, "Multiple sender dis- tributed video streaming," IEEE Transactions on Multimedia and Networking, vol. 6, no. 2, pp. 315–326, April 2004.

[10] J. Apostolopoulos, "Reliable video communication over lossy packet networks using multiple state encoding and path diversity," in Proceeding of The International Society for Optical Engineering (SPIE), January 2001, vol. 4310, pp. 392–409.

[11] J. Apostolopoulos, "On multiple description streaming with content delivery networks," in InfoComm, June 2002, vol. 4310.

[12] S. Floyd, M. Handley, J. Padhye, and J. Wid- mer, "Equation-based congestion control for uni- cast application," in Architectures and Protocols for Computer Communication, October 2000, pp. 43–56.

[13] Information Sciences Institute, http://www.isi.edu/nsnam/ns, Network simula- tor.

[14] J. Leigh, O. Yu andD. Schonfeld, and R. Ansari, "Adaptive networking for tele-immersion," in Immersive Projection echonology/Eurographics Virtual Environments Workshop(IPT/EGVE), May 2001.

[15] M. Chen and A. Zakhor, "Rate control for streaming over wireless," in INFOCOM, July 2004.

[16] Internet Engineering Task Force, RFC 1771, Stream Control Transmission Protocol, October 2000.

[17] P. Mehra and A. Zakhor, "Receiver-driven bandwidth sharing for tcp," in INFOCOM, San Francisco, April 2003.

[18] J. Crowcroft and P.Oeschlin, "Differentiated end-to-end internet services using weighted propor- tional fair sharing tcp," 1998.

[19] Y. Dong, R. Rohit, and Z. Zhang, "A practical technique for supporting controlled quality assur- ance in video streaming across the internet," in Packet Video, 2002.

[20] J. Semke, J. Mahdavi, and M. Mathis, "Auto- matic tcp buffer tuning," in SIGCOMM, 1998.