Multimedia Streaming; Can TCP Handle It ?

Saugath Joshi, Dushan Aththidiyavidanalage Don

Abstract

TCP plays a major role in the protocol stack of the Internet, providing a wrap of reliability over the underlying best effort IP network. However, with the introduction of the multimedia streaming over the Internet, there has been an argument that the very same features make TCP an indispensible part of the Internet stability produce some undesirable conditions for streaming. Various solutions varying from workarounds to alternative service models have been suggested to mitigate these adverse effects of TCP. Recently there have been proposals challenging the long –held argument of the unfriendly behavior of TCP over streaming applications and they argue that using techniques like client side buffering, receiver-driven bandwidth sharing, streaming can be implemented over TCP without significant issues. In this brief survey of ours, we try to summaries the issues, their validity and a few proposed solutions.

1. Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) is the dominant transport protocol in today's Internet. This protocol adjusts how much data may be sent for a particular connection between end applications over each interval of time. TCP is designed to "back off " the rate of transmission when network congestion occurs and to increase the rate over time when there is no congestion. The increase is TCPs mechanism to probe the network to determine if excess capacity is available. TCP congestion control can be viewed as a nonlinear feedback control system that dynamically adjusts its transmission rate according to the network's congestion state [1].

2. Multimedia Streaming

The Internet's ubiquity has long made it an attractive platform for distributed multimedia applications. A particularly elusive goal has been effective streaming solutions. In streaming, both transferring and viewing or listening can be done simultaneously where as in downloading these two are temporally sequential; the transfer of the content must be completed before the playing started [2].

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. It is expected that real-time media-streaming traffic will increase rapidly, and will soon make up a significant portion of the total Internet bandwidth. The key characteristics of such real-time

streaming applications are the potential for high data rates, and the need for low and predictable latency and latency variance [3].

3. The issues with the current set up of the Internet

The current set up of the Internet relies heavily on TCP stability and expects other protocols to be TCP friendly for perfect harmony. However, at least two major features of TCP which compensate for best-effort, no guarantee IP network on which the Internet is formed, have been asserted as undesirable for multimedia streaming.

3.1 Reliability Through Retransmission

TCP uses packet retransmissions to recover from packet losses and packet re-ordering. It's argues that retransmissions can introduce unacceptable end-to-end latency. In multimedia streaming, unlike in other cases, this is not appropriate given the real-time nature of the content. This may result in recent data arriving at the receiver end too late to be played.

3.2 Congestion Control

TCP congestion algorithm is designed to monitor available bandwidth through manipulation of the transmission rate which results in some deliberate waste. In steady state, TCP congestion control converges on an average transmission rate close to a fair-share of available bandwidth. TCP instantaneous rate takes on a familiar saw-tooth shape, over a shorter time scales. It cycles between periods of additive increase separated by multiplicative decrease (AIMD). This rate fluctuation is asserted undesirable for smooth functioning of multimedia streaming applications.

4. Overcoming the Hurdle

Many solutions, in the form of work-abounds, extensions and alternatives have been suggested to this clash of interests between multimedia streaming on the internet and functional requirements of the internet operations. In the proceeding sections, we will attempt to summarize a few of them.

The anti-TCP argument has gone to the extent of introducing new service models even. But given the indispensable role TCP is playing in the current set up of the Internet and the large number of other applications, protocols and infrastructures developed TCP in mind and the widespread use of them encourages the work-around approach and building extensions rather than completely new service models.

Furthermore in certain situations multimedia streaming over TCP is an unavoidable option such as situations where the end users sitting behind a firewall permitting only HTTP traffic is allowed.

4.1 Streaming Control Protocol (SCP) [3]

A new flow and congestion control scheme called Streaming Control Protocol (SCP) addresses two issues associated with real-time streaming

1. It uses a congestion control policy that allows it to fairly share network bandwidth with both TCP and other SCP streams.
2. It improves smoothness in streaming and ensures low, predictable latency.

The protocol is different from TCP's jittery steady-state congestion avoidance policy that is based on linear growth and one-half rate reduction.

SCP (Streaming Control Protocol), a unicast streaming and congestion control scheme. It consists of a media sender and receiver linked by a network connection with SCP residing at the sender side. Each packet has a sequence number. For each packet, SCP has a separate timer and records the time when it is sent. The receiver acknowledges each packet received with an ACK carrying the sequence number of the packet. Based on the reception of ACKs and timer expiration events, the sender adjusts the size of its congestion window to control the flow and avoids network congestion. The SCP operation is self evident from the state transition diagram given below (Figure 1) and each state is briefly described in Table 1.

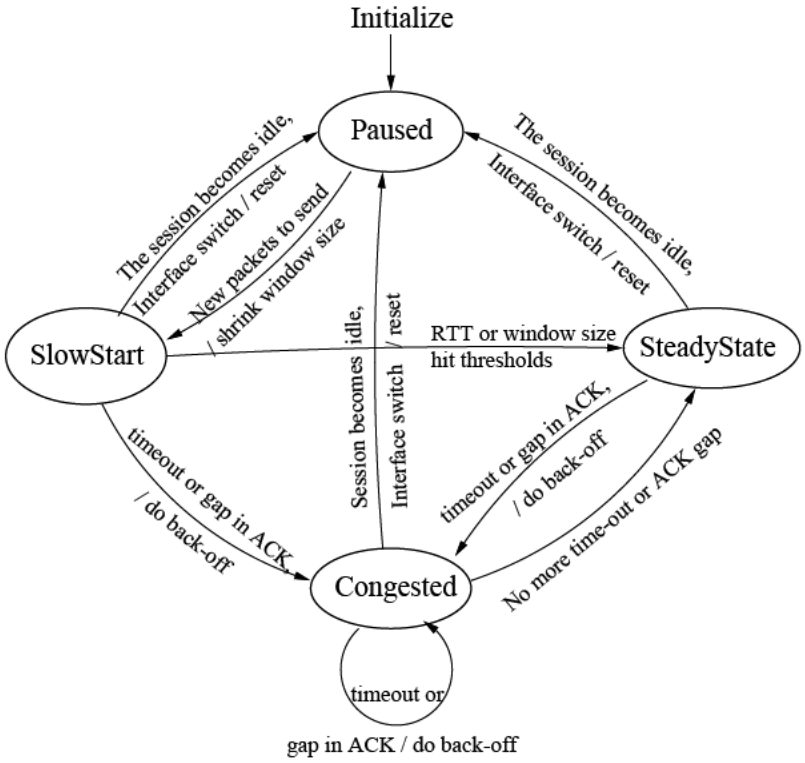

Figure 1 – SCP State Transition Diagram [3]

State	Network and session condition	Congestion window adjustment policy
Slow start	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 maximum throughput, avoid excessive buffering or buffer over flow 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 slow start policy

Table 1 - SCP States, network and session conditions and congestion policies

4.1.1 Similarities between TCP and SCP

1. Both employ sender-initiated congestion detection through positive acknowledgement, and uses a congestion-window based policy for congestion avoidance

2. During the start-up phase, SCP uses TCP-style slow-start policy to quickly discover the available network bandwidth. The similarities are important because it makes SCP as robust as TCP, and enables the two of them to share the Internet fairly.

4.1.2 Differences between TCP and SCP

1. Unlike TCP, when the network is not congested, SCP invokes a combined rate and window-based flow control policy that maintains smooth streaming with maximum throughput and low latency.
2. SCP does not retransmit data lost in the network. Thus it avoids the unpredictability in latency and wasted bandwidth.
3. While TCP linearly increases its congestion window size, detects packet loss, and halves its rate, SCP tries to maintain an appropriate amount of buffering in the network for sufficient utilization of available bandwidth.

4. SCP also has an interface to support drastic bandwidth fluctuations by embedding mobile computing [4].

SCP based on various experimental results, out stands existing TCP in the grounds of an effective multimedia streaming protocol. SCP reduces the latency caused by retransmission (which is permissible in multimedia except for that the delay should be minimum, which is of utmost importance). SCP also enables stable sharing of network bandwidth between multiple streams. SCP backs off exponentially. SCP has features to handle the limited data rate and possible pause in streaming. With the ability to reset its internal state and parameter estimators, SCP also has mobile awareness.

4. 2 DiffServe [5]

The Internet Diffserve architecture (RFC 2475) suggests service differentiation , allowing internet to handle different classes of traffic in different ways in a scalable and flexible manner . The Diffserve architecture is flexible in the sense that it does not define specific services and or service classes considering that the existing services or classes may become obsolete in future and new ones may arise. Instead, Diffserve provides the functional components; a network architecture with which such services can be built [6]. The DiffServe architecture has two sets of functional elements;

Edge functions: packet classification and traffic conditioning - At the incoming edge of the network, arriving packets are marked by setting the differentiated service (DS) field of the packet is set to some value. This mark will represent the class of traffic to wish a certain packet belongs.

Core function: forwarding – When a packet with its DS field marks arrived at a Diffserve capable router, the packet is forwarded onto its next hop according to the so-called per-hop behavior associated with that packet's class. The per-hop behavior influences how a routers buffers and link bandwidth are shared among competing classes of traffic.

The IETF Diffserv working group has specified the Assured Forwarding (AF) per hop behavior (PHB). The AF PHB is intended to provide different levels of forwarding assurances for IP packets at a node, and therefore, can be used to implement multiple priority service classes [5].

Hou et el, presents a Diffserv implementation architecture, in the context of IETF Diffserv AF PHB, with the aim of providing different levels of reliability in terms of packet delivery over the Internet [7]. Their Differv architecture is targeted at integrated support for both real-time streaming applications and traditional data applications.

Under their Diffserv architecture, two types of services are defined,

High Reliable (HR) service - intended for certain high priority traffic in real-time streaming applications while LA service is. Packets under HR service are considered critical to overall perceptual quality for a multimedia streaming application and should be delivered as reliably as possible.

Less Assured (LA) service - for low priority traffic in real-time streaming applications and traditional TCP applications. Packets under LA service either have less impact on the perceptual quality (if they belong to real-time streaming applications) or can be retransmitted (if they are traditional TCP-type applications).

They propose a node mechanism, called Selective Push-out with Random Early Detection (SPRED), to perform packet discarding during network congestion and achieve our Diffserv AF PHB. By employing a single shared queue and storing and serving packets in the queue in the order of their arrival, SPRED does not introduce any packet re-ordering at the node. SPRED performs selective packet discarding from an embedded queue at a shared buffer and does not maintain any state information for each flow. For HR service, when network is congested and buffer is full, SPRED selectively pushes out LA packets in the buffer to make room for the incoming HR packets. Thus, SPRED offers more reliable delivery for HR service than RED/RIO. For LA service, SPRED employs RED to resolve the global TCP synch synchronization problems.

According to the paper, their simulation results conclusively demonstrated that under the same link speed and network topology, network nodes employing our Diffserv/SPRED architecture has substantial performance improvement over the current best effort architecture for real-time multimedia streaming applications.

4.3 Receiver –Driven Bandwidth Sharing

A common case in multimedia streaming is users whose last-mile connection to the Internet is bandwidth-limited and act as a bottleneck. End users generally run more than one networked applications simultaneously and they compete for bandwidth resources. This may lead TCP, which shares bottleneck network capacity based on connection round trip time (RTT), to deprive the multimedia streaming applications of the required bit-rates.

Mehra and Zakhos work focuses of this common case [8]. In their previous work, they presented a receiver-based bandwidth sharing system (BWSS) for allocating the capacity of last mile bottlenecks among TCP flows according to a user's preferences.

This system does not require modifications to the TCP protocol, network infrastructure or sending hosts, making it easy to deploy. Their approach was to limit the throughput fluctuations of high-priority applications by breaking TCP fairness between flows on the access link.

The bandwidth allocation model used by the BWSS categories applications into two groups: those that require a minimum throughput guarantee, and those which do not have any such requirements.

Each TCP connection destined to the receiver is assigned a priority, a minimal rate, and a weight. The bandwidth is allocated to applications according to the following algorithm. First, the minimal rate
is provided to every connection, in decreasing order of application priority. Then, the remaining bandwidth is shared proportionally to the weight of the connection.

The BWSS effectively deals with the case when the overall access link bandwidth is reduced during the streaming session, and ensures that the streaming application receives its minimum required bit-rate. The BWSS is also effective even if the flows causing congestion on the access link are not under the direct control of the BWSS.

However, the BWSS is not useful if there is congestion on the path to the streaming source which limits the throughput of the streaming. Furthermore, the BWSS does not provide any benefits if there is congestion on the access link which reduces the available capacity below the required streaming rate.

5. Countering anti-TCP Argument

With the increasing popularity of multimedia steaming applications fueled by the recent proliferation of broadband Internet access, there have been many suggestions to improve the current TCP model and even alternative service models claiming that some of the key features, which makes TCP an indispensible component contributes to the stability of the Internet, are undesirable for multimedia streaming.

However, there have been recent several proposals which challenge this long-held belief that TCP is not suitable streaming applications. Krasic et el provide a strong qualitative argument for the possibility of multimedia streaming using TCP [2]. They argue that client side buffering can handle throughout variations due to both retransmission delays and congestion control.

Retransmissions can be caused by both packet re-ordering rather and packet loss, however the latency penalty for re-ordered packets will be small, since TCP will still accept an out of order packet when it arrives. The latency penalty for retransmission of lost packets will be on the order of one RTT.

RTTs vary for numerous reasons on the wide-area internet. The following is a rough taxonomy of RTT scales, and consequently the latency penalties resulting from TCP retransmission: 20ms between sites in the same region, 100ms for sites on the same continent, and about 200ms between sites requiring oceanic crossings. We now consider how these latencies would relate to video applications.

For purely-interactive applications such as tele-conferencing or distributed gaming, users are highly sensitive to end-to-end delays of sub-second timescales, typically in the range of 150 to 200 milliseconds.

Unlike purely-interactive applications, on demand multimedia streaming has interactive requirements only for control events such as start, pause, fast-forward, etc, which are relatively infrequent compared to the normal streaming state. While streaming, the quality perceived by the user is not directly affected by end-to-end latency, as the interaction is strictly uni-directional. Such application may gradually increase buffering, hence end-to-end delay, by dividing its use of

available bandwidth between servicing content play-out and buffer accumulation. After a time, the end-to-end delay will actually be quite large, but the user perceives it only indirectly, in the sense that quality during the buffer accumulation period might have been slightly decreased.

Therefore on demand streaming may not have end –to- end the latency issue as real time interactive streaming hence operational with TCP.

The rate variations in an applications TCP flow due to two distinct reasons; the competing traffic in the network and flow's own congestion control behavior. These rate variations can be persistent or transient. The distinction between transient and persistent rate changes is whether the buffer capacity is large enough to smooth them out. The purpose of buffering is precisely to smooth out transient changes.

For any amount of buffering, competing traffic can have persistent effects on a stream's rate. Streaming video applications must deal with persistent rate changes, before the client-side buffers are overwhelmed. The usual way is to employ quality-adaptation, adjusting the basic quality-rate trade-off of the video.

The TCP saw-tooth or the congestion controls increase and decrease phases suggest it should be treated strictly as a source of transient rate changes. The quality-adaptation should address this not only the persistence component.

In conclusion, Krasic et el argue that both TCP congestion control and retransmission induced issues can be handled using client side buffering and hence on demand multimedia streaming is viable with TCP and suggest that further investigation is required to examine to what extent interactivity can be accommodated.

6. Minimal Buffering Requirements

As Krasic et el suggest, buffering and rate adoption can be used to avoid problems related to TCP congestion control, effectively. Li et el discuss series of analysis and experiments conducted to appreciate the minimum buffering requirements of congestion controlled multimedia streaming applications [9].

They have studied the relationship between buffering requirements and adaptation policies. They focused particularly on a widely pursued policy that adapts an application's sending rate exactly to the average available bandwidth to maximize throughput. Under this adaptation policy, at least a minimal amount of buffering is required to smooth the rate oscillation inherent in congestion control, and they assert this minimal buffering requirement as a cost of maximizing throughput.

Their results indicate that choosing an AIMD based TCP-friendly congestion control with a small increment parameter can reduce the buffer requirement, because the buffer requirement is proportional to the increment parameter. However, the buffer requirement is also proportional to

the square of the application's sending rate and round-trip-time. Thus, they conclude that adapting application sending rate closely to the average available bandwidth is not a preferable Adaptation policy for interactive applications with high rate and long RTT.

The Table 2 summarizes their results

Adaptation Policy	Minimal Buffer Requirement	Bandwidth Efficiency	Number of quality Adjustments
Aggressive adaption	0	92%	105
Conservative adaption	0	58%	5
Lazy adaption	> 300KB	92%	0
Ideal adaption	7.8 KB	92%	5

Table 2 – Comparison of Various Adaption Policies And Minimum Buffer Requirement

7.Conclusions

There have been a wide speculation that TCP causes undesirable conditions for multimedia streaming over the Internet. While many attempts were made and are being made to address these issues, there have been counter arguments about this anti-TCP speculation. The pro-TCP camps argue that TCP is an indispensible part of the Internet and helps reliable communication using the underline best-effort IP network. Due to already developed architecture which relies on TCP and the vast amount of the TCP friendly protocols and applications and their widespread use favors any solution which does not demand changes to the exiting TCP based systems.

References

[1] Kang Li", Molly H. Shory, Jonathan Walpole", Calton Pu, David C. Steere, "Modeling the Effect of Short-term Rate Variations on TCP-Friendly Congestion Control Behavior", Proceedings of the American Control Conference Arlington, VA June 25-27,2 001

[2] Charles Krasic, Kang Li, and Jonathan Walpole, "The Case for Streaming Multimedia with TCP- [3] Flow and Congestion Control for Internet Media Streaming Applications", Proceedings of SPIE , December 1997-- Volume 3310, pp. 250-264

[4] Inouye, J., Cen, S., Pu, C., and Walpole, J. "System support for mobile multimedia applications", In NOSSDAV'97 (May 19-21 1997), pp. 143{154.

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An architecture for differentiated services", RFC 2475, Internet Engineering Task Force, December 1998.

[6] James F. Kurose,Keith W Ross , Computer Networking –A Top to Down Approach, 4th Edition - , Addision Wesley (2007) ; pp. 652-657.

[7] Yiwei Thomas Hou , Dapeng Wu , Bo Li , Takeo Hamada , Ishfaq Ahmad ,H. Jonathan Chao "A differentiated services architecture for multimedia streaming in next generation Internet" , Computer Networks, Volume 32, issue 2 Elsevier Science (February, 2000), pp. 185-209

[8] Puneet Mehra , Avideh Zakhor, "TCP based video streaming using receiver-driven bandwidth sharing" , IEEE Transactions on Multimedia, Volume 7, Issue 4, pp 740-752.

[9] Kang Li1, Charles Krasic, Jonathan Walpole1, Molly H. Shor, and Calton Pu, "The Minimal Buffering Requirements of Congestion Controlled Interactive Multimedia Applications", Lecture Notes in Computer Science- Interactive Distributed Multimedia Systems, Springer Berlin/Heidelberg (2001), Volume 2158/2001, pp 181-192.