STREAMING APPLICATION QOS: CONTROLLING CONGESTION VIDEO STREAMING MEDIA DATA QUALITY THROUGH SENDING THE FINEST DATA PACKAGE SUBSEQUENT

Ashish Parejiya[1], Dr. Vinod Desai[2], Dr.V. K. Chaubey[3]

Ashish Parejiya[1],PhD Scholar(CSS), Mewar University, Chittorgarh, INDIA

Dr. Vinod Desai[2],Computer Science Departments, Govt. Science College, Navsari, INDIA

Dr. V. K. Chaubey[3], Department of CSS, Mewar University, Chittorgarh, INDIA

ABSTRACT

In a traditional network stack, data is transmitted in the order that it is received. An algorithm is proposed where priority of packets and expiry times is used by the transport layer to reorder or discard packets to optimize the use of available bandwidth. This scheme is implemented and compared to unmodified Datagram Congestion Control Protocol (DCCP) using traffic modeled on video streaming conferencing media data software. The results show improvement can be made during periods of congestion – more audio packets arrive on time and in many cases video packet arrival rates also increase.

Keywords: *Streaming Congestion Control; DCCP; datagram congestion control protocol; API; multimedia; quality of service.*

1 INTRODUCTION

Typically an application will prepare data and transfer it to the transport layer where it is segmented into packets. A packet is queued and then sent when it reaches the head of the queue, after all other data before it has been transmitted. This however takes no account of time requirements on the delivery of data, or the possibility of different priority among different parts of the data.

This paper discusses a scheme where the transport layer, in cooperation with the application, determines which packets should be sent and in what order they should be sent in. The motivation for this is two-fold. It seems reasonable for the transport layer to discard packets which will no longer be useful for the receiver. It would also seem reasonable for audio packets to be sent before video packets, as Sasse et al. (1994) shows that audio is more important than video to a user's experience in a video conference situation. This is particularly applicable when the number of layers of data being transmitted is being reduced, or is already at the lowest possible.

Our unique contribution is that we combine priority queues with tracking of expiry times which take into account the Round Trip Time (RTT). This is tested and implemented using the DCCP transport protocol (Kohler et al., 2006) which is congestion controlled but unreliable.

This paper begins with a survey of previous work in Section 2. Section 3 introduces the Sending the Finest Data Package Subsequent (SFDPS) algorithm and discusses why DCCP was used. In Section 4 the methodology and the test environment are discussed. In Section 5 the findings from testing SFDPS are discussed, and in the last section possibilities for further work are outlined.

2 Previous Work

In this section previous work aimed at improving the transmission of video is examined. By nature video generates a considerable amount of network traffic that has strict requirements of the network. As such it is often constrained by the network conditions.

It is commonly proposed that video codecs should add and remove video layers as network conditions permit (Feamster et al., 2001; Rejaie et al., 1999). While these codecs match the traffic generated to the available capacity, they do not ensure that the most important data is transmitted first. In a study of multimedia streaming (Guo et al., 2006) it was shown that the median time to change to a lower bit-rate stream was around 4 s. This indicates that there is scope to improve the user experience during the transition time as it is presumed that a lower bit-rate is being used because of loss. There are also cases where the bit-rate cannot be lowered further and loss occurs. Feng et al. (1999) builds upon changing which video layers are used and proposes a priority queue for delivery of prerecorded video streams where lower priority video layers are only sent after higher priority video layers have been sent. Krasic et al. (2003) investigates storing streaming video in multiple discrete layers which are tailored to the type of client that is requesting the data. The streaming server, which they call priority-progress, then decides the most appropriate layers for the client and maps these to a priority order within each time segment. At the end of each time segment, if the data has not been transmitted, then the data is discarded by a 'progress regulator'.

In Tsaoussidis and Wei (1999) Multimedia Transmission Protocol (MTP) is introduced, which is based on TCP Reno but without guaranteed reliability. Packet priority information is sent as a 2 bit field which determines whether packets are retransmitted or not. This enables MTP to not retransmit data which is of lower priority, as compared to TCP which retransmits all missing data. Time-lined TCP (TLTCP) (Mukherjee and Brecht, 2000) introduces the concept of tracking expiry time for data. TLTCP marks data with an expiry time as we also propose, and discards data if the expiry time is reached before the packet is sent. Unlike the approach in this paper, TLTCP does not take the RTT into consideration but proposes that it is worthwhile to do so. TLTCP is a partially reliable protocol – if the data has not expired and retransmission is requested by the receiver then the data is retransmitted. If the data has expired, a more recent packet is sent instead as the expired packet no longer has value.

In Papadimitriou Scalable Streaming Video Protocol (SSVP) (Papadimitriou and Tsaoussidis, 2007) and Video Transport Protocol (VTP) (Balk et al., 2003) are introduced. These are layered on top of UDP. SSVP adjusts the send rate through altering the inter-packet gap, VTP smoothes the transmit rate and sends at a fixed rate. Both protocols adaptively alter the layers of video being transmitted. Amer et al. (1994) introduce a partial order transport service that allows reliable or unreliable partial order service. This allows marking of importance and time values by the application. This is used by the receiver to determine whether lost data is ignored or requested to be retransmitted depending on their temporal value.

Research by Lai and Kohler (2004) investigated using late binding ring buffers with DCCP for transmission that were accessible by the application. The application could replace a packet in the buffer with a more recent packet if it had not been transmitted. The aim is for the most timely data to be sent and old data to be discarded.

The previous work shows that there is scope to improve the video transmission by considering which packets get sent when, and whether packets get sent at all. Our research draws together the range of different algorithms from the previous work to improve video streaming conferencing media data and is described in the next section.

3 Sending the Finest Data Package Subsequent

In this section a concept is introduced called Sending the Finest Data Package Subsequent (SFDPS), different variants of SFDPS are also described, and the use of DCCP to implement SFDPS is explained.

SFDPS is a unique contribution of combining a priority queue, with expiry times for packets that take RTT into account. Discarding expired packets allows newer packets to be sent that are more likely to arrive on time. SFDPS is completely sender based; it could be implemented on a server without requiring changes to the receiver.

3.1 DCCP

Datagram Congestion Control Protocol (DCCP) is a session-based unreliable transport protocol for datagrams. DCCP is a Standards Track RFC. Further discussion on the usage of DCCP is contained in the Problem Statement for DCCP (Floyd et al., 2006a).

DCCP was used for SFDPS as it is a recent protocol designed to accommodate applications such as multimedia streaming and VoIP (Phelan, 2005) that do not need reliability. DCCP is session based, tracks RTT and has congestion control in the base protocol.

UDP was not used because it does not track the RTT and does not have congestion control in the base protocol, and it was not desired to implement this at the application level. UDP also has difficulty traversing some firewalls due to its lack of session control and often applications fall back to TCP. It was also decided not to use TCP as it is a reliable protocol and TCP often combines packets, even when the (Nagle, 1984) algorithm is turned off. With TCP being a reliable protocol if data is not received it is retransmitted when, for many applications, it would be better to discard the data.

DCCP has pluggable Congestion Control Identifiers (CCIDs) which allow different congestion control methods to be used within the protocol. CCID3 (Floyd et al., 2006b) implements TCP-Friendly Rate Control (TFRC) (Handley et al., 2003) for DCCP. This was used because it has a smoother response to loss than CCID2 (Floyd and Kohler, 2006) and is thus considered to be more suitable for multimedia (Phelan, 2005). CCID2 is TCP-like in using a congestion window and following many of the TCP semantics although it does not have retransmission.

3.2 SFDPS1

The SFDPS1 algorithm sorts the transmit queue as a priority queue. All higher priority packets get sent prior to any lower priority packets being sent. Audio packets are marked as higher priority than video packets. The packets in the queue are also marked with an expiry time. When a packet is due to be sent the expiry time is checked against the current time added to half the RTT. If the expiry time is less than this calculation the packet is discarded rather than being sent.

3.3 SFDPS2

One aspect of CCID3 is that it reduces the allowed transmit rate after short idle periods. The formula for allowed sending rates is $X = \max(\min(X\_calc, 2 * X\_recv), s/t\_mbi)$ where X_calc is the TCP throughput equation (Padhye et al., 1998), X_recv is the receive rate since the previous feedback (at least once per RTT), s is the average packet size and t_mbi is 64.

This means that in practice the allowed transmit rate falls very quickly when no packets are being transmitted and thus not being received also. This has the impact of reducing the quality of the video conference, particularly during quiet periods. There is an internet draft for TFRC faster restart (Kohler et al., 2007) that partially addresses this issue.

To address the issue of reduced transmit rate the SFDPS1 algorithm was modified to send an expired packet if it is the only packet in the queue to reduce the likelihood of the transmit rate being reduced.

3.4 Ring buffers

As a comparison to SFDPS2 Lai and Kohler's (2004) ring buffer scheme was implemented as we interpreted it. Their application generates one packet every 10, 20, 25 or 30ms and is marked as expired after three packets are queued. Their network was constrained to 50 Kbits/s. They then also implemented preferential packet dropping; that

is, if the buffer of three is full then their algorithm drops B and P frames before I frames so that as many I frames as possible are sent – they experimented this with 10, 20, and 25ms but keeping at three packets buffer. I frames are full frames in MPEG and B/P frames are incremental frames. In our implementation we used a variable length buffer (as opposed to 3) and if the buffer is full and there are video packets the oldest packet is dropped, otherwise we drop the oldest audio packet.

SFDPS2 determines whether to discard a packet at time of transmission. Ring buffers determine whether to discard a packet at time of queueing for transmission. It was decided to combine the two approaches in one algorithm for comparison to see if the combination further improves the number of packets received on-time.

Test methodology

4.1 Video conference model

Packet traces were captured from Ekiga which is an open-source application which includes video transmission. The traces were captured on a 100Mbps link and consisted of 10 s of talking/motion, followed by 5 s of silence/still, followed by 5 s of talking/motion. The traces were then analysed to see their protocol, timing, packet size, and whether each packet contained audio or video.

UDP was used by Ekiga to transmit audio and video. A minimal amount of TCP was used in the connection setup and for control purposes due to a requirement for reliability. The effects of TCP traffic were excluded from our experiments as they constituted less than 0.1% of the traffic. Ekiga had a fixed time period between audio packets of 20ms and employs silence suppression. All audio packets were exactly 214 bytes.

Ekiga had a simple transmission scheme with variable frame sizes that was dependent on the amount of motion in the frame. For high motion the average is 4450 bytes per video frame with a standard deviation of 1000 bytes, while for low motion the average is 1075 bytes with a standard deviation of 475 bytes. Frames are sent at the rate of ten frames per second and are composed of packets with a maximum size of 1026 bytes which are sent as a bunch.

A model of the traffic was then built. Fixed rate audio was used with a payload size of 214 bytes every 20ms and silence suppression was not used. Video frames in the model were created with an average and a standard deviation for their type that matched the traffic Ekiga generated. With this model audio equates to 86Kbps and video equates to 289Kbps which gives a total of 374 Kbps. These rates include all packet overheads, including the MAC headers.

4.2 Test setup

All of the packets in an audio frame or video frame were queued immediately after each other by the application and the time to send was determined by the DCCP CCID3 congestion control. This differs from the applications which used UDP and determined the send time themselves as UDP does not have congestion control at the transport level. If audio and video frames were both due to be transmitted at the same time the application randomly chose which to send first to avoid bias.

An application was written to then replay this stream between two PCs with a Linux PC in the middle using the netem (Hemminger, 2005) software to add delay, and to rate limit the traffic or create loss. There was a separate application written to track the difference in time between the two PCs so that arrival time could be tracked at the receiver relative to the sender. The receiver recorded how long each packet took to arrive and whether it arrived at all

Tests were run with symmetric one way delays of 15, 40, 75ms producing minimum RTTs of 30, 80, 150 ms.For the purpose of simplicity symmetric links were used. It is recognised that not all links are symmetric (Paxson, 1999) and the algorithm could easily be adapted if the one way delay could be measured for a link. Two priorities were used in the tests, one for audio and one for video. The number of priorities could be extended for multiple video layers.

Initial testing with the 20 s runs showed a high rate of variability due to the initial connection setup and the random nature of loss/congestion. The test was then done with 60 s runs with the data being repeated three times in each run.

The first 3 s of data was discarded and the remaining data had significantly reduced variation due to the connection speed stabilising over a longer run. In all of the results in this paper data was generated from 30 runs and an average calculated with 95% confidence intervals from a T-test as recommended in Law (2004).

The ITU recommend that delivery time for audio be less than 150ms and anything between 150ms and 400ms has lower quality (ITU, 1993). With SFDPS packets get marked with an expiry time which is 200ms from the time that the packets are created. The expiry time includes both time in the queue and time in transit. The choice of 200ms is arbitrary and it is envisaged that an application would allow the user to adjust this setting to meet their quality requirements.

4.3 Linux kernel implementation

The implementation was done on Linux 2.6.20 with a series of patches to ensure performance of CCID3 matched the TFRC (Handley et al., 2003) throughput equation.

The sendmsg system call in Linux allows control data to be sent as well as the data. The DCCP implementation of sendmsg was modified to transmit expiry time, priority and method in a structure passed into the msg_control field in the msghdr structure. This data is received by the dccp_sendmsg function, which uses the priority information to store the packet in the correct location in the transmit queue. The transmit queue remained as one queue and was changed to a queue sorted first by priority and secondly by expiry time. This was implemented as one queue for reasons of code simplicity, but for wider use it could be implemented as a set of queues. The dccp_write_xmit function starts transmission from the beginning of this priority queue. The function checks the expiry time just before a packet is transmitted and decides whether to discard the packet if the packet has expired.

The method field is used to select which algorithm will be used in testing. The expiry field contains the time that the data will be of no use to the receiver. In the tests this was set to be 200ms from the creation of the data by the application. These fields are used by all of the SFDPS algorithms with the exception of unmodified DCCP.

5 Results

The desired outcome is that the number of on time audio packets increase and that the number of on time video packets decrease less than the audio increase. It should be noted that with the unmodified transmission scheme more audio packets arrived on time than video packets. The reason for this is that all packets for a single video frame get queued for transmission at the same time which means there is a higher probability the buffer will be full and a video packet discarded.

5.1 Testing with loss

SFDPS1 was implemented and compared to unmodified DCCP using netem to discard packets between two nodes. A queue length of five packets was used which is similar to average UDP buffering. When run under these conditions the SFDPS1 algorithm performed worse than unmodified DCCP as shown in Figure 1. Audio and video packet on time arrivals both decreased.

Testing was carried out with the SFDPS2 algorithm which only discards expired packets if there are other packets in the transmit queue. The results for the SFDPS1 and SFDPS2 algorithms with queue length of five packets can be seen in Figure 2. This did improve results compared to the SFDPS1 algorithm but did not achieve results that were better than unmodified DCCP. This required some further investigation.

Figure 3 shows the difference between the data received and the data received on time. In the graph SFDPS1 has almost no difference between the packets received and the packets received on time as it only transmits when the packet will reach the other end without expiring.

Figure 1: Testing with loss – 80ms RTT, queue length 5

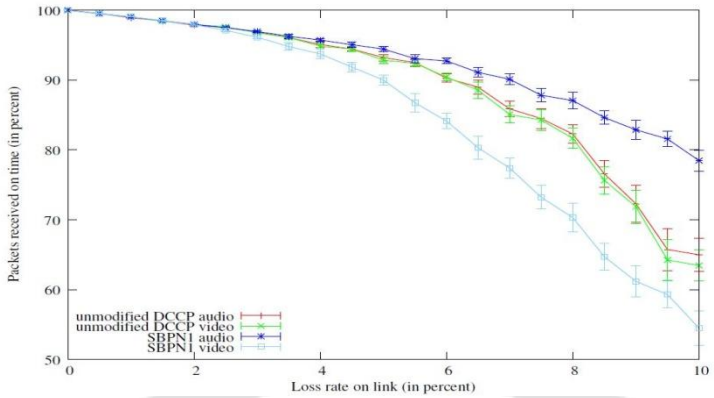
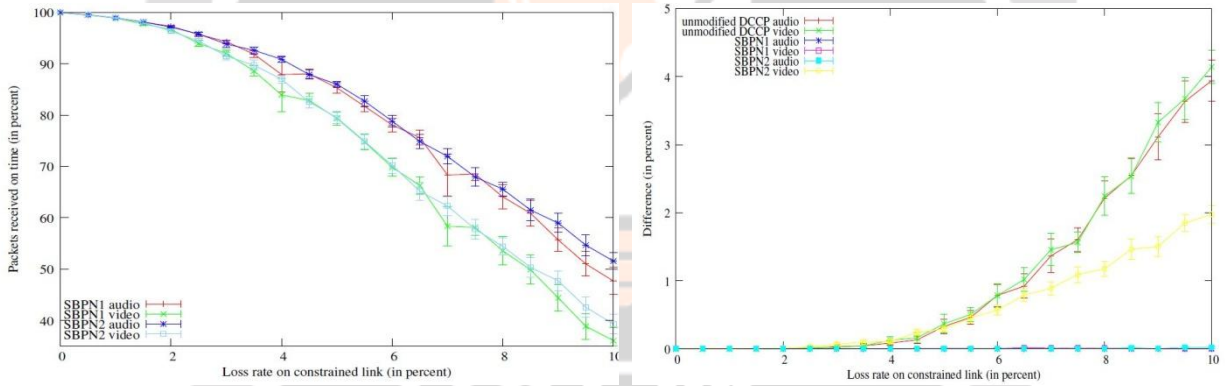


Figure 2: Testing with loss – SFDPS1 vs. SFDPS2 – 80ms RTT

Figure 3: Difference in on time arrival – 80ms RTT, queue



SFDPS2 has slightly more packets received than packets received on time as it sends data to keep the connection from going idle, and unmodified DCCP has even more packets received as it always tries to send. This indicates the improvement is not occurring because of running at a fixed rate of loss and if less packets are being sent then a lesser number of relevant packets will be received due to the smaller number of packets being transmitted. It is interesting to note on this graph that all the packets used to keep alive the connection for the SFDPS2 algorithm are video packets – this shows that the audio packets are being transmitted and being received on time successfully. This can be seen from the graph as the difference between received and on time packets are zero (which is also the case for SFDPS1).

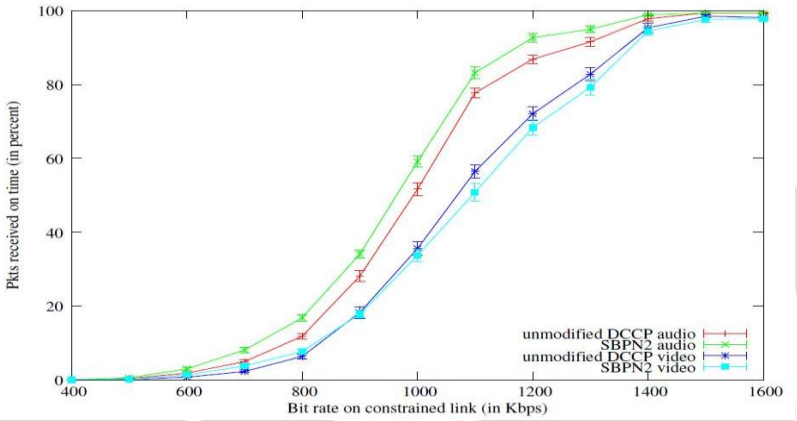
This effect can also be seen in congestion control on mobile networks where loss is an indication of packet corruption, not congestion. Congestion control algorithms such as Vegas (Brakmo and Peterson, 1995) and Veno (Fu and Liew, 2003) attempt to find the level of congestion in a network, as well as responding to packet loss – which is not always an indicator of loss. As such if these, or similar, algorithms were implemented for DCCP the results in this section may have been quite different. Implementing these algorithms for DCCP is outside the scope of this paper however.

5.2 Testing with congestion

As it was concluded that fixed loss does not give a realistic test tests were carried out with congestion. This was done by running tests with two competing TCP Reno flows, produced using (Iperf, 2007) that was modified to allow continuous running. Within the video streaming conferencing media data flow there are a range of larger packets for video and smaller packets for audio as described in Section 4, resulting in an average packet size of 519 bytes. A fixed packet size was used with iperf of this size as DCCP effectively works on a rate of packets per second (Widmer et al., 2004). BIC or Cubic are the default TCP congestion control in recent Linux kernels but TCP Reno was chosen as DCCP CCID3 uses the TCP throughput equation (Padhye et al., 1998) modelled on Reno.

When the tests were run with competing TCP flows we obtained the results shown in Figure 4. This shows a clear difference for audio between unmodified DCCP and SFDPS2. This result shows that more audio packets can be transmitted when there is congestion than under unmodified DCCP.

Figure 4: Unmodified DCCP vs. SFDPS2 – 80ms RTT, queue



The algorithm was also run with a buffer size matching TCP. Often media applications will use TCP due to UDP having difficulties traversing NAT and firewall devices. Figure 5 shows the results from using a queue length of 32 which matches the default TCP buffer size of 16 Kbytes in Linux 2.6.20. These results show a larger improvement in audio and smaller reduction in the number of video packets dropped.

To test if the algorithm is RTT dependent, the tests were also run with 30ms and 150ms RTT, shown respectively in Figures 6 and 7. These represent scenarios such as a relatively local network connection, or across greater distances.

TFRC faster restart (Kohler et al., 2007) was also implemented to determine if this would counteract the effect of reduced transmit rates and ran a number of tests. A representative example of the results for this are shown in Figure 8 and this shows that TFRC does not produce significantly better results. (It should be noted that TFRC faster restart was implemented on Linux 2.6.23 so this graph cannot be compared directly to the other graphs.)

Figure 5: Unmodified DCCP vs. SFDPS2 – 80ms RTT, queue

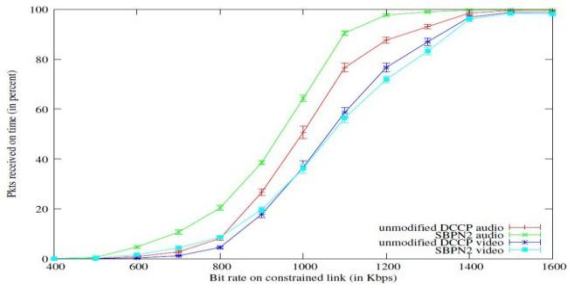
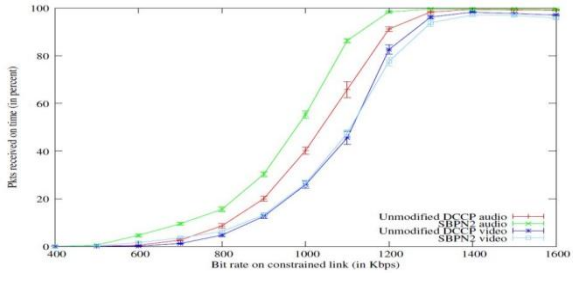


Figure 6: Unmodified DCCP vs. SFDPS2 – 30ms RTT, queue



Figures 9 and 10 compare the results for SFDPS2 with the results for ring buffers. These tests, and other tests that were conducted, showed that in some cases ring buffers performed better than SFDPS2.

Figure 7: Unmodified DCCP vs. SFDPS2 – 150ms RTT, queue

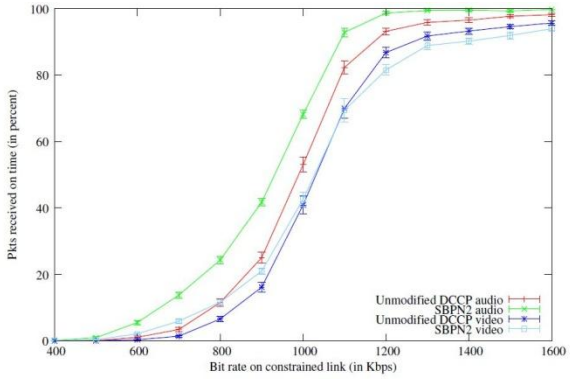


Figure 8: Unmodified DCCP vs. Faster Restart – 80ms RTT,

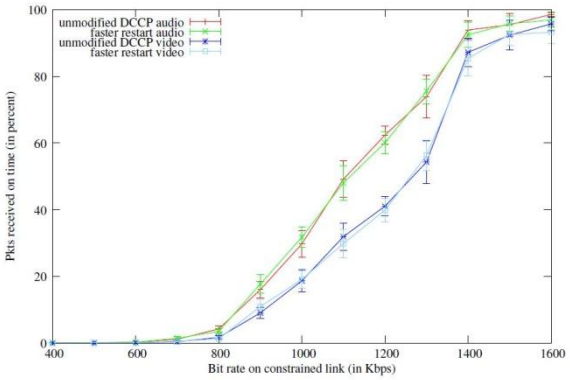


Figure 9: SFDPS2 vs. ring buffers – 80ms RTT, queue length 5

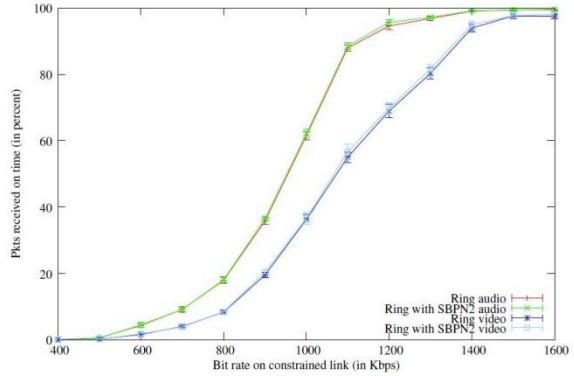
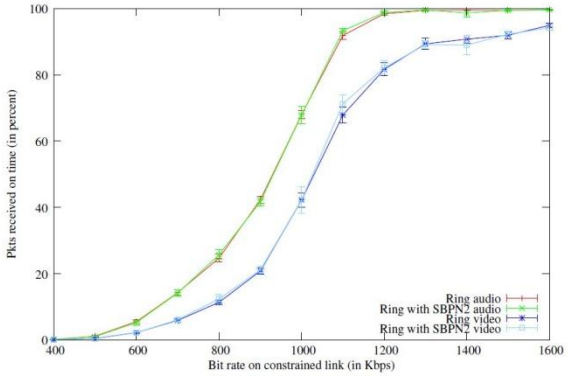


Figure 10: SFDPS2 vs. ring buffers – 150ms RTT, queue length 5



As ring buffers modify the transmission scheme in a different way to SFDPS2 we combined the two algorithms and the results are shown in Figures 11 and 12. These tests, and others that were conducted, show that combining the two algorithms together produces the best results.

Figure 11: Ring buffers vs. ring with SFDPS2 – 80ms RTT,

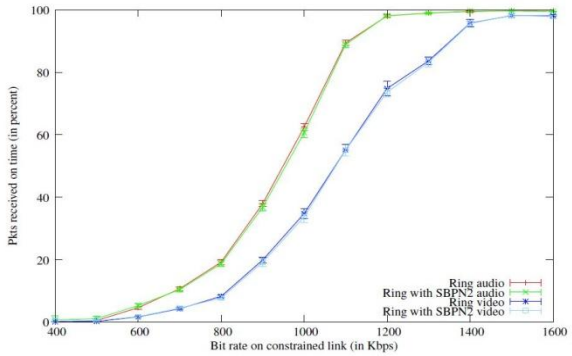
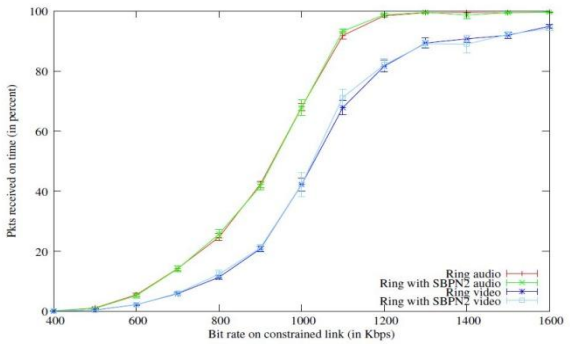


Figure 12: Ring buffers vs. ring with SFDPS2 – 150ms RTT,

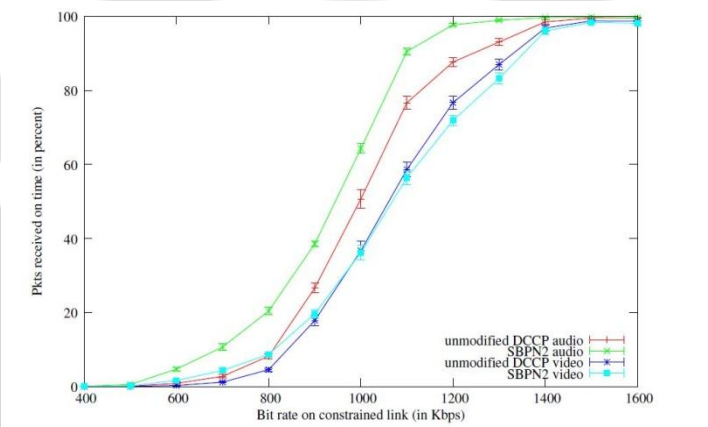


6 Conclusion and further work

This paper shows that it is possible to gain a real improvement to video streaming conferencing media data applications when congestion is experienced. A new class of algorithms have been introduced that automatically bases packet delivery on "Send the Finest Data Package Subsequent" principle. This has resulted in a clear improvement in terms of audio packets received on time when tested with competing TCP flows which will give a higher quality video conference experience. These results have been shown across a range of RTTs and with differing queue lengths. Using the best algorithm (SFDPS2) shows audio on time improves by over 10% across most of the range while video on time does not decrease or decreases by a much smaller amount as shown in Figure 13.

TFRC faster restart gave no statistically significant improvement. The best results occurred when SFDPS2 was combined with ring buffers.

Figure 13: Unmodified DCCP vs. ring with SFDPS2 – 80ms



There are many areas for potential further research. A form of weighting could be considered for the queues which calculates a score based on packet type, expiry time and any other variables which would then be used to sort the transmit queue. To counter the effects of reduced transmit rate as shown in Section 5.1 it may be worthwhile to implement a version that only discards packets if the buffer is full and sends expired packets if there are not any unexpired packets.

Another aspect being considered is to modify an open source application, such as Ekiga, to use the transmission scheme discussed in this paper. Improvements to video streaming conferencing media data could then be measured subjectively also.

As the algorithms proposed in this paper are made at the sender only, servers can implement these algorithms and give a significant benefit to clients without a change being required to the clients. The algorithms presented could also be used in a range of services such as real time IPTV.

References

[1]. Network Simulator. http://www.isi.edu/nsnam/ns.

[2]. TCP Probe. http://www.linuxfoundation.org/.

[3]. Wireless Home Digital Interface. http://www.whdi.org.

[4]. IETF 68 Proceedings. http://www.ietf.org/proceedings/68/, 2007.

[5]. MSN Messenger. http://msn.messenger.com, Accessed 2009.

[6]. Internet World Stats. http://www.internetworldstats.com/stats.htm, Accessed 2011.

[7]. Ashish Parejiya, Dr. Vinod Desai, "Congestion Control for Streaming Data Broadcasting over Internet", IJMTER Vol.1 Issue. 5, Novmber 2014, http://www.ijmter.com/

[8]. Ashish Parejiya, Dr. Vinod Desai, "A Comparative Study and Survey on Broadcasting Multimedia Streaming Data Congestion Control Mechanisms", IJCSMC, Vol. 3, Issue. 12, December 2014, http://www.ijcsmc.com

[9]. Ashish Parejiya, Dr. Vinod Desai, "Multicasting Of Adaptively-Encoded MPEG4 Over Qos-Cognizant IP Networks",IJMTER Vol.2 Issue. 1, January 2015, http://www.ijmter.com/

[10]. Ashish Parejiya, Dr. Vinod Desai, Bhavana Hotchandani, Purvi Trivedi "Transforming Network Coding with TCP for Broadcasting Streaming Data in Multi-Hop Wireless LAN", IJESRT, Vol. 4, Issue. 2, February 2015, http://www.ijesrt.com

[11]. Ashish Parejiya, Dr. Vinod Desai, "MULTI-PATH MECHANISM FOR BROADCASTING AUDIO / VIDEO STREAMING BASED ON BANDWIDTH ESTIMATION", IJESRT, Vol. 4, Issue. 5, May 2015, http://www.ijesrt.com