Effect of TCP and UDP Parameters on the quality of Video streaming delivery over The Internet

MAZHAR B. TAYEL[1], ASHRAF A. TAHA[2]
[1]Electrical Engineering Department, Faculty of Engineering
[1]Alexandria University, EGYPT
[1]dr mbasyouni@egsysexperts.com
[2]Informatics Institute, Mubarak City for Scientific Researches and Technology Applications
[2]New Borg El-Arab, Alexandria, EGYPT
[2]ashraf_taha_1968@yahoo.com

Abstract :- Delivering real-time video over the Internet is an important issue for many Internet multimedia applications. Transmission of real-time video has bandwidth, delay, and loss requirements. The application-level quality for video streaming relies on continuous playback, which means that neither buffer underflow nor buffer overflow should occur. Since the Best Effort network such as the Internet does not provide any Quality of Service (QoS) guarantees to video transmission over the Internet. Thus, mapping the application-level QoS requirements into network-level requirements, namely, limited delay jitters. End-to-end application level QoS has to be achieved through adaptation. Since the QoS of video streams over IP networks depends on several factors such as video transmission rate, packet loss rate, and end-to-end transmission delay. The objectives of this paper are to simulate an adaptation scheme to include the effect of User Datagram Protocol (UDP) parameters on delay jitter and datagram loss values to increase the efficiency of UDP protocol to prevent the network congestion and increase the adaptivity and also, simulate an adaptation scheme to include the effect of Transmission Control Protocol (TCP) parameters on the transmission rates to increase the adaptivity of the transmission.

Key-Words: - Quality of Service, Video streaming, Internet, TCP Window, UDP, Iperf.

1. Introduction

Many multimedia applications rely on video streaming techniques. Streaming is the only technology that is capable of transmitting video and audio events across the Internet in real time, i.e. while they are happening [11]. The term "Real Time" means that the user receives a continuous stream (with a minimum delay), and that the duration of the transmitted and received streams is exactly the same. In the streaming mode, however, the content file need not be downloaded in full, but plays out while parts of the content are being received and decoded. It can deliver live content such as a football match, a concert or a political speech as it happens.

There are three main obstacles in video streaming over the Internet. The first is the variable network performance due to load changes. The second, the bandwidth availability is highly unpredictable which makes quality adaptation difficult. Finally, network congestion, manifests itself by low arrival rate and packet losses. Two fundamental challenges for network-aware applications are how to translate the application level Quality of Service requirements into network-level Quality of Service requirements, and how to determine and deal with the nature of network resource availability [7].

The quality of video traffic transmission over Internet depends on the available bandwidth. The traffic load along a path, changes throughout transmission in an unpredictable way. Hence, bandwidth requirements have to be modified accordingly. When available bandwidth is not sufficient, video transmission rate can be adjusted lowering the quality of the video in a controlled way. The quality adjustment is done by excluding some frames from the transmission or by reducing frame sizes. Later, the frame sizes reduction can be achieved in a number of ways: adjusting the compression ratio of on-line encoder [3], switching to a lower quality pre-encoded version, transcoding a pre-encoded version [4] or dropping a layer of a hierarchical encoding scheme [5]. In all cases two issues have to be considered; the first is how to determine the available

bandwidth, and the second is how to adjust the quality as needed.

The major sources of QoS degradation in a packet-based video streaming service are packet loss and transmission delay. Packet loss disrupts the video coding, while transmission delays can decrease the throughput or cause underflow events of the receiving buffer for decoding [6].

To investigate the cause of video quality degradation and to observe video QoS level for end users, a method proposed for monitoring the QoS-related parameters detected in both the transport and application layers for TCP/IP based video streams. Based on sequential measurement using an MPEG-4 video system in a real environment it found that the buffer usage rate in the application layer tended to decrease before detection of sudden decreases in bit-rate of video streams [6].

TCP congestion control algorithm increases the transmission rate to probe for the available bandwidth and decreases it in response to packet loss [5]. The video quality is affected by these changes unless a large buffer is available to absorb the variability. A need for a smoother rate adjustment for multimedia application has been introduced in [8, 9].

Later, a congestion control mechanism is designed to react to a drop event and not to a single packet drop. Rate is adjusted less frequently than once per Round Trip Time (RTT). The rate oscillations are reduced but the behavior is also less responsive. Rate adaptation is performed independently of quality adaptation and based only on the network status. Quality adaptation on the other hand, depends on the rate and its adaptation. The quality of video can be increased depending on the rate adaptation mechanism. The two levels of adaptation, rate and quality, operate at different time scales and a buffer is used to cushion the difference between them.

The method presented in [1], frame-rate adaptation, combines rate and quality adaptation into one. The available bandwidth is estimated by measuring the incoming data rate at the client and evaluating the RTT. Each frame is requested from the server separately. Hence the rate at which the requests are sent is one of the factors determining the video arrival rate at the client. The rate is adjusted by adjusting the quality, i.e., including or excluding frames from the transmission. The requested rate and the quality at the same time are not only adjusted in response to the changes in the network status, but also to low buffer occupancy. The observed arrival rate and buffer occupancy at the client form basis for defining thresholds triggering the request rate adjustment. The 10% difference between the request and arrival rates is the first of them. The second threshold is determined based on the length of time interval after which client buffer underflow may occur. In both cases when the threshold is reached, the request rate is decreased. The rate is set to a value smaller than the observed arrival rate to allow the network backlog to clear out. Next the rate is increased to the value of the previously measured arrival rate. We argue that a decrease of the transmission rate is not always necessary. Without a more detailed information about network status, we must consider the worst case scenario, i.e., network congestion.

In [10], The paper present two new congestion control protocols for streaming media like applications as examples of protocol design in this framework: the first protocol, LOG, has the objective of reconciling the smoothness requirement of an application with the need for a fast dynamic response to congestion. The second protocol, SIGMOID, guarantees a minimum bandwidth for an application but behaves exactly like TCP for large window.

However, if the network is not congested, keeping the rate unchanged when the rate threshold is reached, and increasing transmission rate when the buffer occupancy threshold is reached can improve the video quality. The key element in making a decision whether, the rate reduction is needed or not, is the network status information. Therefore, an idea of an application obtaining this information and acting is based on both end-to-end performances observed and network status indication. In [7], the paper present an approach allows to obtain better user-perceived video quality by providing additional information to properly interpret the arrival rate observed at the end-point. In this scheme, transmission rate can be changed without affecting the quality. In contrast to [5, 8], however, there is a two-way dependence between them. If the rate has to be lowered because of congestion, the video quality is affected. The quality adaptation dictates the rate value if there is no congestion. A higher rate may be requested either to sustain the current quality level or to increase it. In this way the frequency of quality changes can be controlled. Maximizing the user-perceived quality does not necessarily mean

using all available bandwidth at any time as assumed in [3].

Most adaptive delivery mechanisms for streaming multimedia content do not explicitly consider user-perceived quality when making adaptation decisions. An optimal adaptation trajectory (OAT) through the set of possible encodings exists, and that it indicates how to adapt encoding quality in response to changes in network conditions in order to maximize user-perceived quality. The OAT is related to the characteristics of the content, in terms of spatial and temporal complexity. A method to automatically determine the OAT in response to the time-varying characteristics of the content. In this way, as the characteristics of the content change over time, the system can dynamically and intelligently adjust the adaptation process in order to maximize the user-perceived quality. The OAT can be used with any sender-based transmission adaptation policy. The paper [12] demonstrate content-based adaptation using the OAT in a practical system using two different adaptation algorithms. Furthermore, it show how this form of adaptation can result in differing adaptation behavior not only as a result of the dynamics of the content but also as a result of the adaptation algorithm being used. Finally, it show how increased feedback frequency does not necessarily improve the behavior of the adaptation algorithm being used.

The objectives of this paper are to simulate an adaptation scheme to include the effect of UDP parameters on delay jitter and datagram loss values to increase the efficiency of UDP protocol to prevent the network congestion and increase the adaptivity of the network and also, simulate an adaptation scheme to include the effect of TCP parameters on the transmission rates to increase the adaptivity of the transmission.

2. Adaptive QoS management for video streaming delivery over the Internet – TCP/UDP

Since most of the adaptation process, including statistic gathering and decision making, is done on the client side, the server is left with the job of reading and sending video data only. This video delivery system can be more easily scaled up to support many clients [1].

A TCP window is the amount of outstanding data (unacknowledged by the recipient), a sender can send on a particular connection before it gets an acknowledgment back from the receiver that it has gotten some of it. By adjusting the TCP window size and buffer lengths to different values through the interval times, measuring the Time To live (TTL) using the “ping” command for different hops, it is possible to measure the BW for every interval and also the total bandwidth for number of streams.

2.1 TCP/UDP Bandwidth Measurement Tool

Iperf was developed as a modern tool for measuring TCP and UDP bandwidth performance. It measures the maximum TCP bandwidth, allowing the tuning of various parameters, and UDP characteristics and reports the bandwidth, delay jitter, and datagram loss. The server detects UDP datagram loss by ID numbers in the datagram. Usually a UDP datagram becomes several IP packets. Losing a single IP packet will lose the entire datagram. Jitter calculations are continuously computed by the server. The server computes the relative transit time as (server's receive time - client's send time). The client's and server's clocks do not need to be synchronized; since any difference is subtracted out in the jitter calculation. Jitter is the smoothed mean of differences between consecutive transit times [2].

The primary reason for the window is a congestion control. The whole network connection, which consists of the hosts at both ends, the routers in between and the actual connections themselves (be they fiber, copper, satellite or whatever) will have a bottleneck somewhere that can only handle data so fast. Unless the bottleneck is the sending speed of the transmitting host, then if the transmitting occurs too fast the bottleneck will be surpassed resulting in lost data. The TCP window throttles the transmission speed down to a level where congestion and data loss do not occur.

2.2 Simulation Results

2.2.1 UDP Protocol

Comparison between the delay jitter and lost datagram values, obtained at server and client for number of streams (1, 2, and 3) with different combinations of UDP buffer lengths and UDP packet sizes (8k, 16k, 32k and 64k) in 16 categories at UDP protocol, are studied.

During simulations, it was noticed that higher delay jitter using reassembled 32k datagram (each split into 23 packets of 1500 bytes) causes a lost in datagram due to burstiness of the traffic lead to poor performance.

Figures (1) and (2) show delay jitter values obtained at server and client respectively for different packet sizes and UDP buffer lengths for one stream

Server – one stream

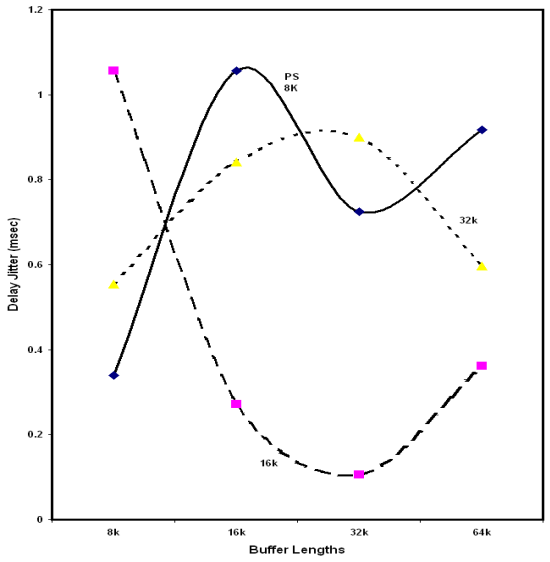

Fig. 1 Delay jitter values obtained at server for different packet sizes and UDP buffer lengths for one stream

Client – one stream

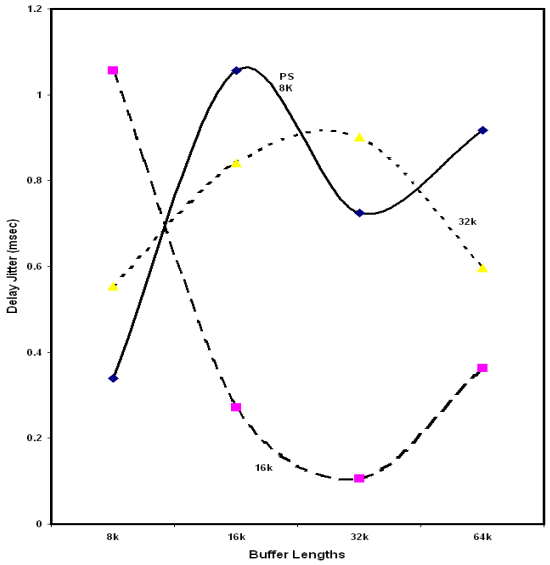

Fig. 2 Delay jitter values obtained at client for different packet sizes and UDP buffer lengths for one stream

From figures (1) and (2), it is seen that the delay jitter attains its minimum delay value of 0.106 msec at server and client respectively at UDP buffer length = 32k and Packet size = 16k.

Figures (3) and (4) show delay jitter values obtained in logarithmic scales at server and client respectively for different packet sizes and UDP buffer lengths for two streams.

Server – two streams

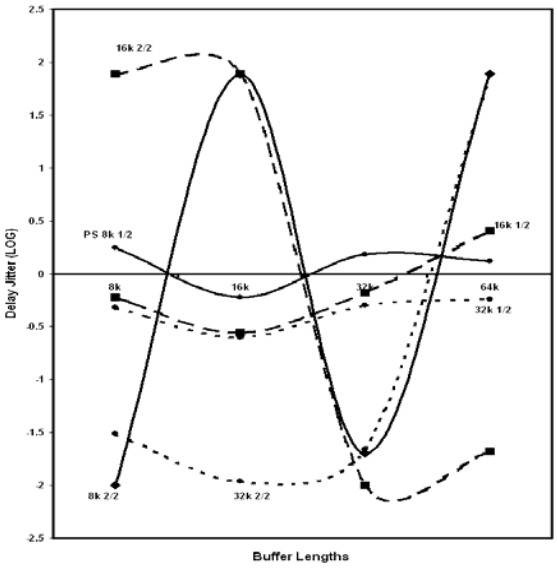

Fig. 3 Delay jitter values obtained in LOG scale at server for different packet sizes and UDP buffer lengths for two streams

Client – two streams

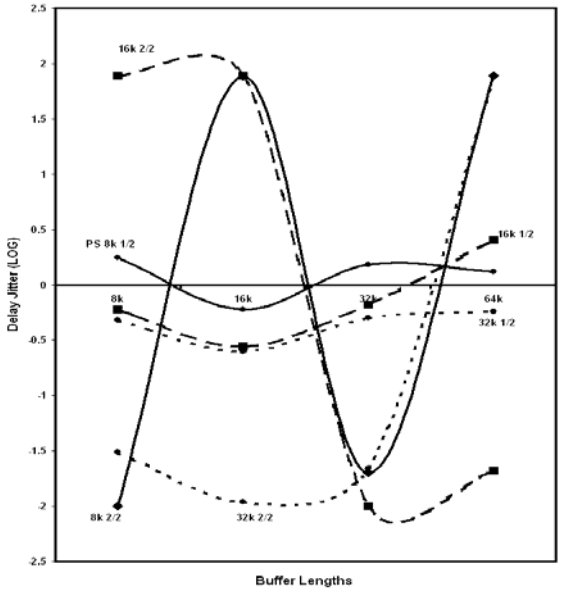

Fig. 4 Delay jitter values obtained in LOG scale at client for different packet sizes and UDP buffer lengths for two streams

From figures (3) and (4), it is seen that delay jitter attains its minimum optimal values in logarithmic scale at server and client respectively at UDP buffer length = 32k for the two streams. The first stream, with UDP buffer length = 32k according to packet size values, has delay jitter values = (1.517, 0.633, 0.502) at server and delay jitter = (1.518, 0.633, 0.502) at client.

The second stream with UDP buffer length = 32k according to packet size values has delay jitter values = (0.02, 0.01, 0.022) at server and delay jitter values = (0.021, 0.009, 0.023) at client.
The problems with the two streams in the lost datagram are shown in figures (5) and (6).

Server – two streams

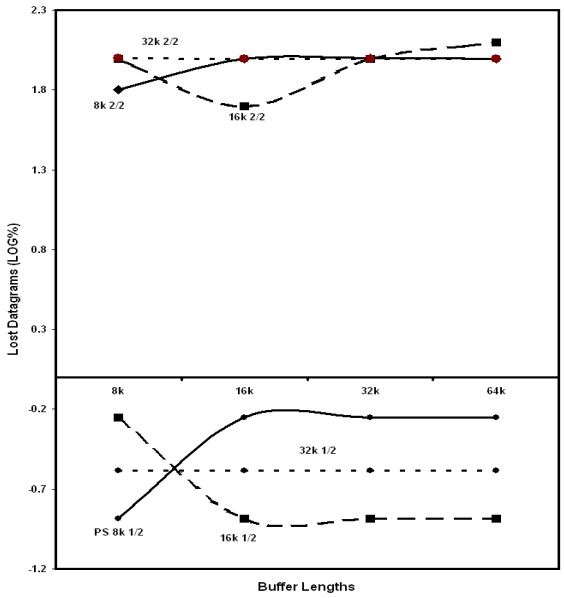

Fig. 5 lost datagram in LOG% scale obtained at server for different packet sizes and UDP buffer lengths for two streams

Client – two streams

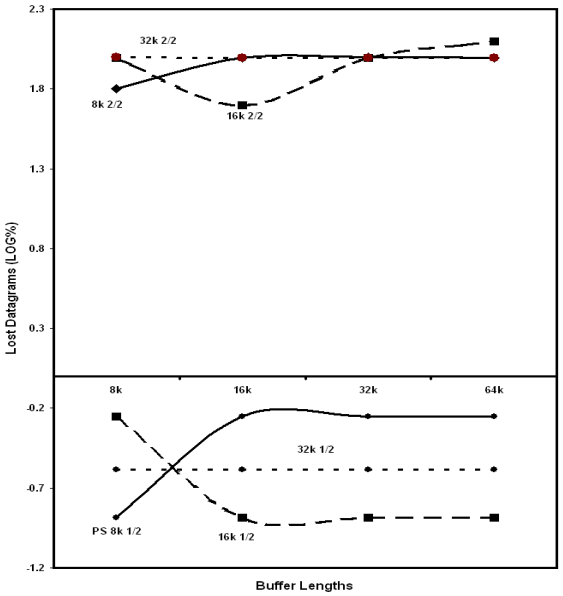

Fig. 6 lost datagram in LOG% scale obtained at client for different packet sizes and UDP buffer lengths for two streams

From figures (5) and (6), it is seen that lost datagram obtained in percentage logarithmic scales and attains its minimum optimal values at the first stream for all packet sizes at server and client respectively, whereas the second stream attains higher values, thus, the problem occurred in the second stream.
Figures (7) and (8) show delay jitter values obtained in logarithmic scales at server and client respectively for different packet sizes and UDP buffer lengths for three streams.

Server – three streams

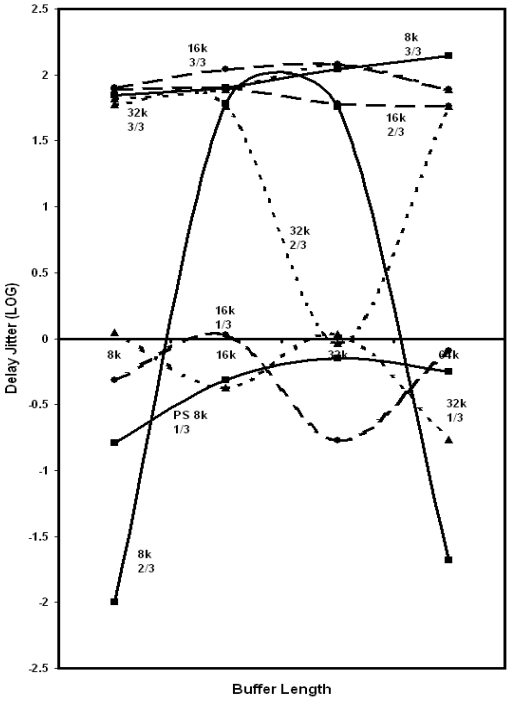

Fig. 7 Delay jitter values obtained in LOG scale at server for different packet sizes and UDP buffer lengths for three streams

Client – three streams

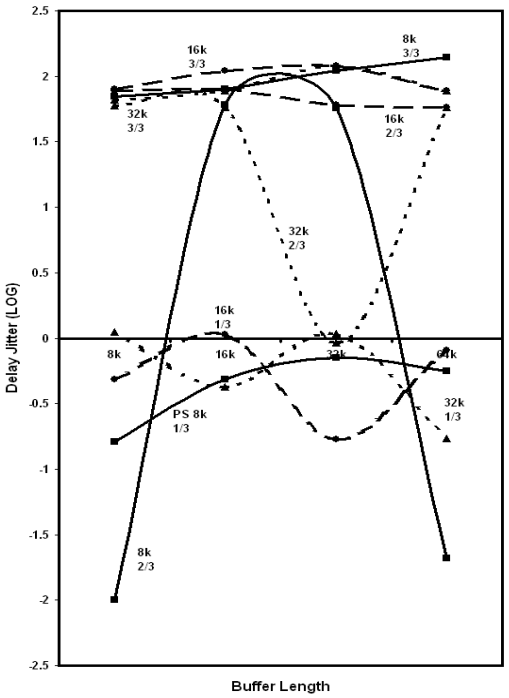

Fig. 8 Delay jitter values obtained in LOG scale at client for different packet sizes and UDP buffer lengths for three streams

From figures (7) and (8), it is seen that the delay jitter attains its minimum values in logarithmic scales at server and client respectively, UDP buffer length = 32k using three streams. The first stream, with UDP buffer length= 32k according to packet size values, has delay jitter values = (0.713, 0.17, 1.087) at server and delay jitter values = (0.713, 0.169, 1.087) at client.
The 2nd and 3rd streams attain higher values of delay jitter, thus, the problems are occurred in them as shown in figures (9) and(10).

Server – three streams

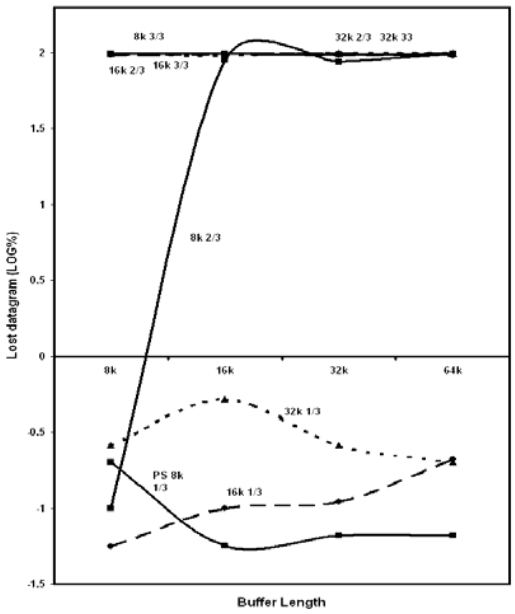

Fig. 9 lost datagram in LOG % scale obtained at server for different packet sizes and UDP buffer lengths for three streams

Client – three stream

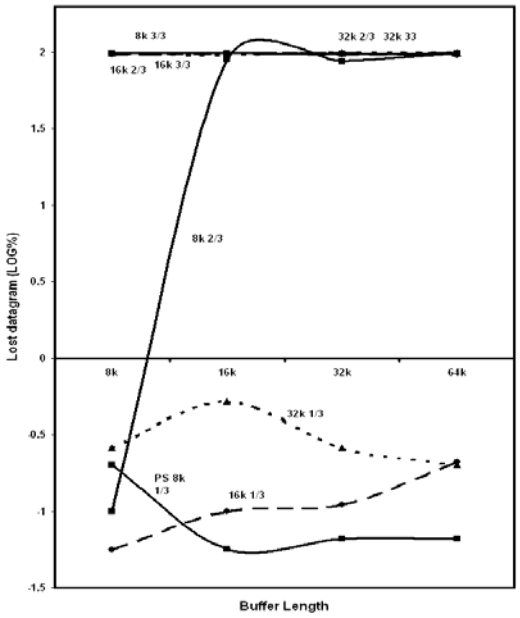

Fig. 10 lost datagram in LOG % scale obtained at client for different packet sizes and UDP buffer lengths for three streams

From figures (9) and (10), it is seen that the lost datagram obtained in percentage logarithmic scale, attains its minimum optimal values at the first stream for all packet sizes at server and client respectively, whereas the 2nd and 3rd streams attain higher values.
From the above discussions it is seen that, the one stream with UDP buffer length = 32k is the best choice at server and client due to the lowest delay jitter value at packet size = 16k.

2.2.2 TCP Protocol

A comparison between the Maximum Bandwidth (MBW) values, obtained at server and client for (1, 2, and 3) streams with different combinations of TCP window size and buffer lengths (BLs) = (8k, 16k, 32k and 64k) in 16 categories at TCP protocol.
Figures (11) and (12) show the MBW values obtained at server and client for different buffer lengths (BLs) and TCP window sizes for one stream.

Server – one stream

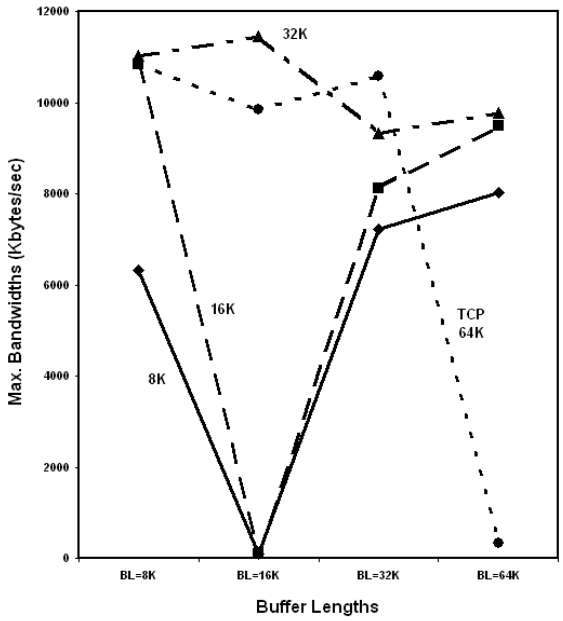

Fig. 11 MBW values obtained at server for different buffer lengths and TCP window sizes for one stream

Client – one stream

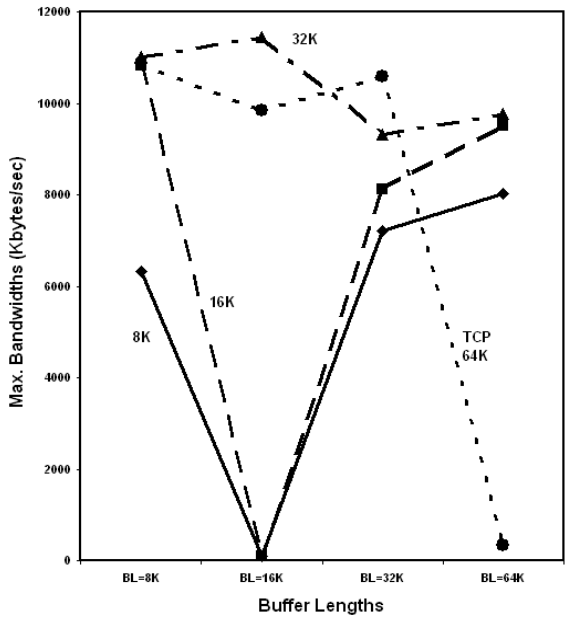

Fig.12 MBW values obtained at client for different buffer lengths and TCP window sizes for one stream

From figures (11) and (12), it is seen that the MBW attains its optimum value of 11453 Kbytes/sec and 11448 Kbytes/sec at server and client respectively at TCP window size = 32k and BL = 16k.
Figures (13) and (14) show the MBW values obtained at server and client for different buffer lengths and TCP window sizes for two streams.

Server – two streams

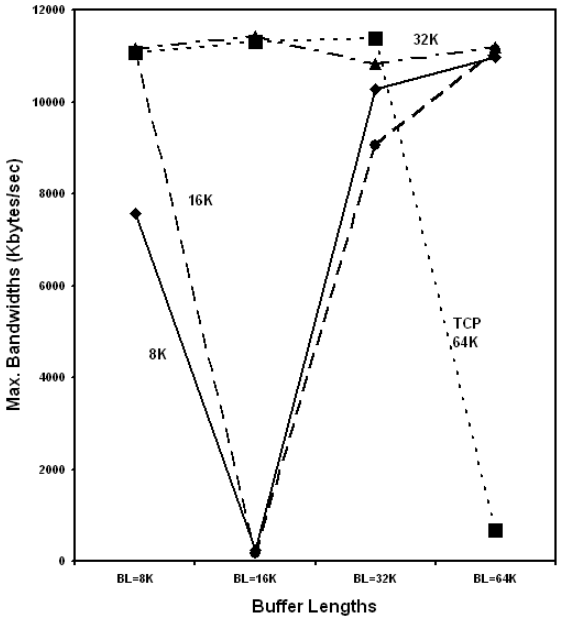

Fig. 13 MBW values obtained at server for different buffer lengths and TCP windows sizes for two streams

Client – two streams

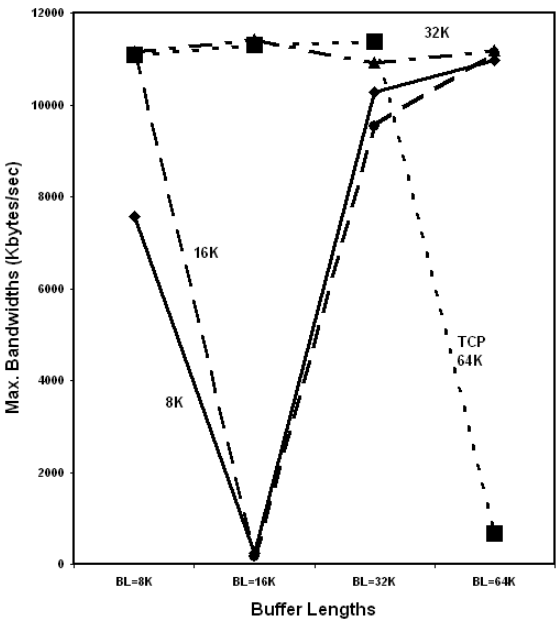

Fig. 14 MBW values obtained at client for different buffer lengths and TCP windows sizes for two streams

From figures (13) and (14), it is seen that the MBW attains its optimum value of 11416 Kbytes/sec and 11415 Kbytes/sec at server and client respectively at TCP window size=32k and BL=16k. However, the values for TCP window size = 32k and TCP window size = 64k almost are closed values for BL=8k, 16k, and 32k. Thus, there is a trade off for selecting TCP window size=32k or 64k.
Figures (15) and (16) show the MBW values obtained at server and client for different buffer lengths and TCP window sizes for three streams.

Server – three streams

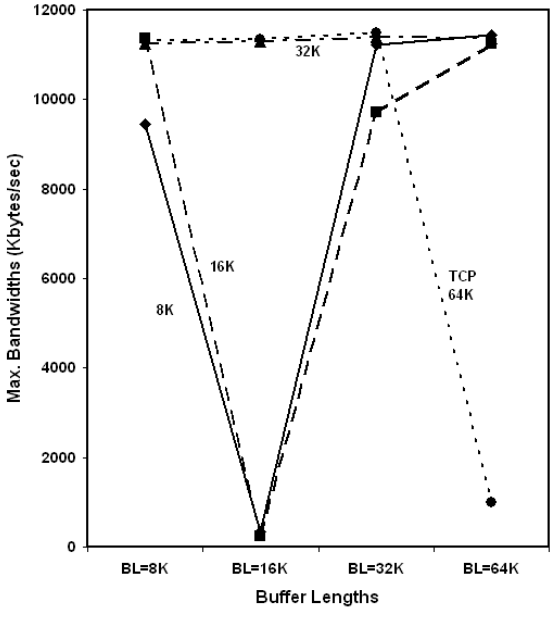

Fig. 15 MBW values obtained at server for different buffer lengths and TCP windows sizes for three streams

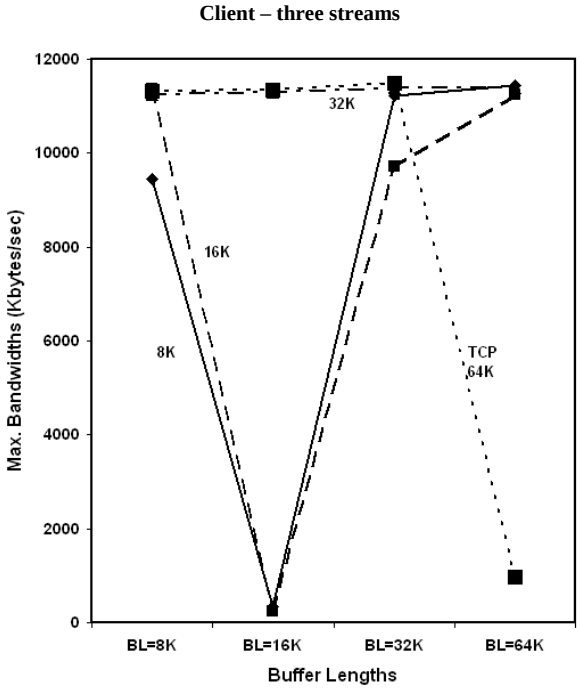

Fig. 16 MBW values obtained at client for different buffer lengths and TCP windows sizes for three streams

From figures (15) and (16), it is seen that the MBW has almost flat coincidental optimum response for BL= 8k, 16k, and 32k for both TCP window size = 32k and 64k. Thus, there is a trade-off for selecting the TCP window size = 32k or 64k.

The percentage of difference between the optimum values obtained at TCP window = 32k and TCP window =64k for BLs values (8k, 16k, and 32k) as follow:

Deviation% between (TCP window=32k,64k) with (BL=8k)=((11317-11235)/11317)x100%=0.7246%

Deviation% between (TCP window=32k,64k) with (BL=16k)=((11350-11290)/11350)x100%=0.529%

Deviation% between (TCP window=32k,64k) with (BL=8k)=((11491-11389)/11491)x100%=0.888%

Therefore, for one stream, TCP window size = 32k is the best choice at server and client due to the highest BW values according to BLs values (8k, 16k, 32k, and 64k).

MBW = (11019, 11453, 9325, and 9773) at server and MBW = (11020, 11448, 9325, and 9773) at client.

For two streams, TCP window size = 32k is the best choice at server and client due to the highest BW values according to BLs values (8k, 16k, 32k, and 64k).

MBW= (11150, 11416, 10838, and 11181) at server and MBW=(11152, 11415, 10926, and 11181) at client.

For three streams, the MBW has almost flat coincidental optimum response for BL= 8k, 16k, and 32k for both TCP window size = 32k and 64k. The percentage of difference between the optimum values obtained at TCP window = 32k and TCP window =64k for BLs values (8k, 16k, 32k) is small and for BL=64k, TCP window =32k has a higher value of MBW than for TCP window = 64k.

Therefore, for three streams, TCP window size = 32k is the best choice at server and client according to BLs values (8k, 16k, 32k, and 64k). The optimum value at TCP window = 32k and BL = 16k.

MBW= (11235, 11290, 11389, 11379) at server and MBW=(11236, 11296, 11388, 11382) at client.

These best results can be accumulated in one table as shown in table (1) and (2).

Table (1) The best choices for MBW at server for TCP window size 32k and different BLs (8k, 16k, 32k, and 64k) using (1, 2, and 3) streams

Buffer Lengths / Streams	8k	16k	32k	64k
1	11019	11453	9325	9773
2	11150	11416	10838	11181
3	11235	11290	11389	11379

Table (2) The best choices for MBW at client for TCP window size 32k and different BLs (8k, 16k, 32k, and 64k) using (1, 2, and 3) streams

Buffer Lengths / Streams	8k	16k	32k	64k
1	11020	11448	9325	9773
2	11152	11415	10926	11181
3	11236	11296	11388	11382

From table (1) and (2) it is seen that the best results obtained at TCP Window size = 32k for different values of Buffer Lengths = (8k, 16k, 32k, and 64k) using the three types of the streams (1, 2, and 3) streams.

For one stream, it is seen that the MBW attains its optimum value of 11453Kbytes/sec and 11448Kbytes/sec at server and client respectively at TCP window size = 32k and BL = 16k.

For two streams, it is seen that the MBW attains its optimum value of 11416Kbytes/sec and 11415Kbytes/sec at server and client respectively at TCP window size = 32k and BL = 16k.

For three streams, it is seen that TCP window size = 32k is the best choice at server and client according to BLs values (8k, 16k, 32k, and 64k).

The optimum value at TCP window = 32k and Buffer Length = 16k.
Figures (17) and (18) show the best values of MBW with applying (1, 2, and 3) streams at server and client according to BLs values (8k, 16k, 32k, and 64k).

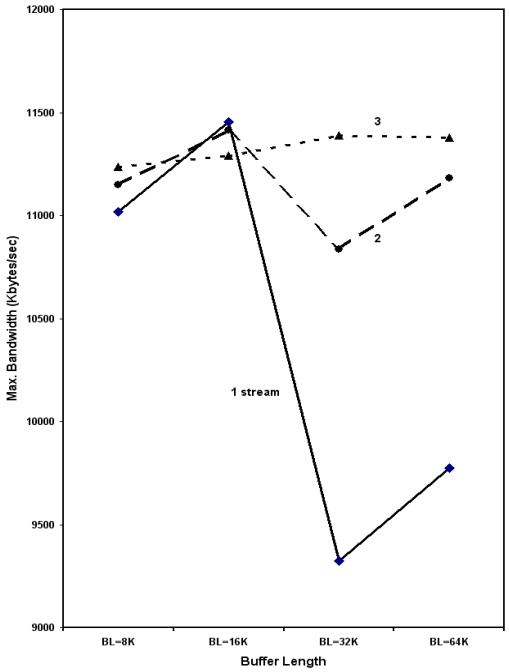

Fig. 17 MBW values obtained at server for different BLs for (1, 2, and 3) streams

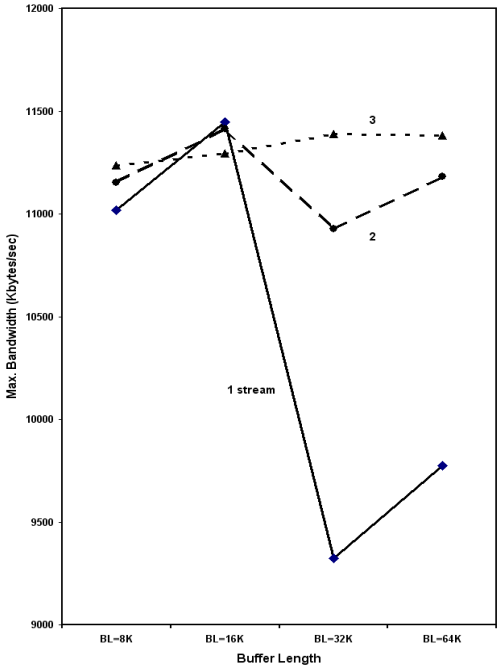

Fig. 18 MBW values obtained at client for different BLs for (1, 2, and 3) streams

From figures (17) and (18), it is seen that the TCP window size = 32k is the best choice at server and client using (1) and (2) streams due to the highest values of maximum bandwidth values according to Buffer Length values (8k, 16k, 32k, and 64k). Also, the maximum bandwidth has almost flat coincidental optimum response for Buffer Length = 8k, 16k, and 32k for both TCP window size = 32k and 64k. For Buffer Length = 64k, TCP window = 32k has a higher value of maximum bandwidth than for TCP window = 64k. Therefore, TCP window size = 32k is the best choice at server and client using three streams according to BLs values (8k, 16k, 32k, 64k),

3 Discussion and Conclusion

Comparisons between the delay jitter and lost datagram values obtained at server and client for (1, 2, and 3) streams with different combinations of UDP Buffer lengths and UDP Packet Sizes = (8k, 16k, 32k and 64k) in 16 categories at UDP protocol. The paper introduce UDP buffer length = 32k with the smallest delay jitter value (0.106 msec) at Packet size =16k and zero lost datagram using one stream.
A comparison between maximum bandwidth values, obtained at server and client for (1, 2, and 3) streams with different combinations of TCP window size and Buffer Lengths = (8k, 16k, 32k and 64k) in 16 categories are obtained at TCP protocol.
The paper concluded that the TCP window size = 32k is the best choice at server and client using (1) and (2) streams due to the highest values of maximum bandwidth values according to Buffer Length values (8k, 16k, 32k, and 64k).
Also, the maximum bandwidth has almost flat coincidental optimum response for Buffer Length = 8k, 16k, and 32k for both TCP window size = 32k and 64k. For Buffer Length = 64k, TCP window = 32k has a higher value of maximum bandwidth than for TCP window = 64k. Therefore, TCP window size = 32k is the best choice at server and client using three streams according to BLs values (8k, 16k, 32k, 64k), Because the paper measure the Quality of Service - related parameters in both the transport and application layers, the proposed simulation can be applied to TCP/IP - based video systems available on the market.
A future study will include Quality of Service evaluations of other Quality of Service - related parameters, further evaluations of the precision

and applicability in real environments, and application to several kinds of multimedia communications.

References :-

[1] F. Kozamernik, " Media Streaming Over The Internet - An overview of delivery technologies", EBU Technical Review, October 2002.

[2] E. Kusmierek, H. David, "Streaming video delivery over internet with adaptive end-to-end QoS", Journal of Systems and Software, Elsevier Science Inc., Vol.75, No.3, March 2005, pp. 237-252.

[3] N. Duffield, K. Ramakrishnan, A. Reibman, "Save: An algorithm for smoothed adaptive video over explicit rate network", IEEE/ACM Transactions on Networking, 1999.

[4] Z. Lei, N. Georganas, "Rate adaptation transcoding for video streaming over wireless channels". In: IEEE International Conference on Multimedia and Expo, 2003.

[5] R. Rejaie, M. Handley, D. Estrin, "Quality adaptation for congestion controlled video playback over the Internet", In: Proceedings of ACM SIGCOMM, 1999a pp. 189–200.

[6] M. Masugi, T. Takuma and M. Matsuda, "Quality of Service assessment of video streams over IP networks based on monitoring transport and application layer processes at userclients", IEE Proceeding of Communications, Vol. 152, No. 3, June 2005.

[7] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate based congestion control mechanism for real time streams in the Internet". In: IEEE INFOCOM, 1999b, pp. 1337–1345.

[8] S. Floyd, M. Handley, and J. Widmer, "Equation based congestion control for unicast applications", In: ACM SIGCOMM, 2000, pp. 43–56.

[9] C. Fung, S. Liew, "End-to-End Frame-Rate Adaptation Streaming of Video Data", EGC Earmarked Research Grant of the Hong Kong University and Polytechnic Grant Council: CUHK 336/96E, IEEE, 1999.

[10] N. Sastry, S. Lam, "CYRF: a theory of window - based unicast congestion control", IEEE/ACM Transactions on Networking (TON), Vol.13, No.2, April 2005, pp. 330-342,.

[11] N. Cranley, P. Perry, L. Murphy, "Dynamic content-based adaptation of streamed multimedia, Journal of Network and Computer Applications, Vol. 30, No.3, August, 2007, pp. 983-1006.

[12] T. Brethour, K. Gibbs, "Jperf version 1.0, The Iperf Front-End", The Board of Trustees of the University of Illinois, All Rights Reserved. Copyright (c) 2002, 2003