BEHAVIOUR INVESTIGATION USING SIMULATION FOR REDUNDANT MULTICAST TRANSMISSION SUPPORTING ADAPTIVE QOS

Ch. Bouras[1,2] *A. Gkamas*[1,2] *An. Karaliotas*[1,2] *K. Stamos*[1,2]

[1]Computer Engineering and Informatics Dep., Univ. of Patras, GR-26500 Patras, Greece

[2]Research Academic Computer Technology Institute, Riga Feraiou 61, GR-26221 Patras, Greece

ABSTRACT

In this paper we describe a mechanism for redundant multicast transmission of multimedia data supporting adaptive QoS over the Internet and we investigate its behaviour using simulation. There are two major issues that have to be considered when designing and implementing such a mechanism, the fairness, which is the ability to cope with receiver heterogeneity, and the adaptation schemes. The proposed mechanism uses a friendly to the network users congestion control policy to control the transmission of the data. We evaluate the adaptive multicast transmission mechanism through a number of simulations in order to examine its behaviour to a heterogeneous group of receivers and its behaviour against TCP and UDP data streams. Main conclusion of the simulation was that the proposed mechanism has friendly behaviour against the dominant traffic types of today’s Internet and treats a heterogeneous group of receivers with fairness.

1. INTRODUCTION

The heterogeneous network environment that Internet provides to the real time applications as well as the lack of sufficient Quality of Service (QoS) guarantees, many times forces applications to embody adaptation elements in order to work efficiently. The main goal of such an approach is to adapt the data rate that is sent to the network every time that network conditions change. Many researchers believe that this end-to-end control scheme must be implemented in the application layer because today’s Internet architecture does not provide such a mechanism in the network layer ([7]).

In addition any application that sends data (mostly multimedia) over the Internet should have a friendly behaviour towards the other flows that coexist in today’s Internet and especially towards the TCP flows that comprise the majority of flows. Applications must meet some special characteristics in order to be characterized as TCP friendly ([12]).

This paper presents the evaluation through simulation of a mechanism for redundant multicast transmission, which supports adaptive QoS. The proposed mechanism is based on multicast video transmission with the use of RTP/RTCP (Real-time Transport Protocol / Real-time Transport Control Protocol)([13]). The main perspectives, which the proposed mechanism tries to fulfil, are 1) each receiver should receive the best video quality that it is capable of and 2) the generated multicast data flow should not be a constraint for the other flows. Paper [3] presents the architecture and implementation of a prototype, which is based on the proposed mechanism. Paper [4] gives an evaluation of the implemented prototype through experiments in a real network environment.

The methods proposed for the multicast transmission of time sensitive data in the Internet can be generally divided in three main categories:

- The source uses a single multicast stream for all receivers ([1], [14], [15]). This results to the most effective use of the network resources, but on the other hand the fairness problem (cope with receiver heterogeneity) arises.
- Simulcast: The source transmits versions of the same video encoded in varying degrees of quality. This results to the creation of a small number of multicast streams with different rates, responsible for a range of receivers with similar capabilities ([8], [6]). The different streams carry the same video information but in each one the video is encoded with different bit rates, and even different video formats. The main disadvantage in this case is that the same video information is replicated over the network but recent research has shown that under some conditions simulcast has better behavior that multicast transmission of layered encoded video ([9]).
- The source uses layered encoded video, which is video that can be reconstructed from a number of discrete data streams and transmit each layer into different multicast stream ([17], [11], [5]). The receivers subscribe to one or more multicast streams depending on the available bandwidth into the network path to the source.

This work is based on the simulcast approach and it is an extension of the work, which has been presented in [2] and [6].

2. ARCHITECTURE OF THE PROPOSED MECHANISM

The proposed mechanism is based on the simulcast transmission of video and follows the client –server architecture and uses the RTP/RTCP protocol for the transmission of the data. The transmission rate within each stream is adapting within its limits according to the capabilities and the state of the Clients participating in it.

The Server is unique and responsible of: 1) creating the *n* different multicast streams (in our simulations we use three), 2) setting each one’s bandwidth limits, 3) tracking if there are any Clients that are not handled with fairness and 4) providing the mechanisms to the Clients to change stream whenever they consider that they should be in another stream closer to their capabilities.

Figure 1 shows the organisation and the architecture of the Server entity. The Server generates *n* different Stream Managers. In each Stream Manager an arbitrary number of Client Managers is assigned. Each Client Manager corresponds to a unique receiver that has joined the stream controlled by this Stream

Manager. The Synchronisation Server is responsible for the management, synchronization and intercommunication between Stream Managers.

The Stream Manager entity is responsible for the maintenance and the monitoring of one of the *n* different multicast streams that are generated in the beginning of the application. Also the Stream Manager entity has all the intra-stream adaptation mechanisms for the adjustment of the transmission rate. The Stream Manager periodically gathers the states reported by all Client Managers belonging to it at the end of a specific, fixed time period (from now on called an epoch). It then uses an algorithm described in a following paragraph that tries to improve fairness between Clients by determining whether a lower or a higher bit rate is more appropriate. Whenever a Client cannot be satisfied by a stream due to the fact that most of the other Clients have much higher or much lower reception capabilities, the Stream Manager informs it that it has to move to a lower or higher quality stream.

Each Client Manager Corresponds to a unique Client. It processes the RTCP reports generated by the Client and can be considered as a representative of the Client at the side of the Server. It can interact only with one Stream Manager at a given time, the Stream Manager controlling the stream from which the Client is receiving the video. Client Manager receives the RTCP reports from the Client and processes them based on packet loss rate and delay jitter information. It then makes an estimation of the state of the Client, based on the current and a few previous reports that it stores in a buffer. The exact operation of the algorithm is described in the following paragraph.

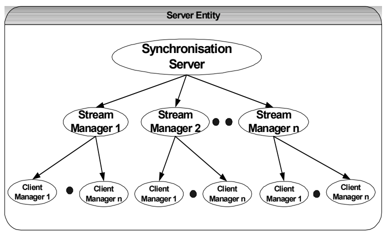

Figure 1. The architecture and the data flow of the Server.

3. DESCRIPTION OF MECHANISM OPERATION

The Server initially constructs a number of streams. When a Client joins a multicast session, a dedicated Client Manager is created to represent the Client at the side of the Server and manipulates the RTCP reports of that Client. Information in RTCP reports contains two values that describe the quality of the transmission: packet loss rate and delay jitter. These values are passed through the following filters used to avoid wrong estimations and determine the aggressiveness of the feedback analysis protocol: For the packet loss rate:

$$LR_{new} = a * LR_{old} + (1-a) * LR_{net}$$

Where: LR_{new}: The new filtered value of packet loss rate. LR_{old}: The previous filtered value of packet loss rate (for the first report after the start of transmission, this value is 0). LR_{net}: The packet loss value that was contained in the RTCP report received from the Client. a: a parameter that determines the aggressiveness of the adaptation concerning the packet loss value (its value ranges from 0 to 1). For delay jitter:

$$J_{new} = b * J_{old} + (1-b) * J_{net}$$

Where: J_{new}: The new filtered value of delay jitter. J_{old}: The previous filtered value of delay jitter (for the first report after the start of transmission, this value is 0). J_{net}: The delay jitter that was contained in the RTCP report received from the Client. b: a parameter that determines the aggressiveness of the adaptation concerning the delay jitter value (its value ranges from 0 to 1).

For the sake of clarity, a distinction has to be made between two kinds of states, that both can take the values of UNLOADED, LOADED or CONGESTED: we call the first one the “unprocessed state” and the second the “processed state”. The unprocessed state is derived directly from the filtered values of packet loss rate and delay jitter, according to the following rules:

$$\textit{if } (LR_{new} >= LR_c) \textit{ unprocessed state} = CONGESTED$$
$$\textit{if } (LR_u < LR_{new} < LR_c) \textit{ unprocessed state} = LOADED$$
$$\textit{if } (LR_{new} <= LR_u) \textit{ unprocessed state} = UNLOADED$$
$$\textit{if } (J_{new} > \gamma * J_{old}) \textit{ unprocessed state} = CONGESTED$$

We have defined LR_u as the maximum value of the unloaded packet loss rate and LRc as the minimum value of the congested packet loss rate. Where γ is a parameter which specifies how aggressive the network condition estimation component will be to the increase of delay jitter.

The state that will be reported to the Stream Manager is called the processed state. It is computed by taking into account the last *n* unprocessed states, which are held in an n-sized buffer in the Client Manager. This buffering mechanism contributes to the conservative behaviour of the Optimal Rate Estimation module. A CONGESTED unprocessed state does not necessarily impose that the processed state will also be congested, especially if the majority of the previous “unprocessed states” were UNLOADED. The way the processed state is computed is presented below: We first introduce a new variable, USV (Unprocessed State Variable), which takes a new value for each unprocessed state as shown:

$$\textit{if (unprocessed state}_i == CONGESTED) \textit{ then } USV_i = -1$$
$$\textit{if (unprocessed state}_i == LOADED) \textit{ then } USV_i = 0$$
$$\textit{if (unprocessed state}_i == UNLOADED) \textit{ then } USV_i = 1$$

The processed state is then determined by the value of

$$f(i) = USV_i * w_i + USV_{i-1} * w_{i-1} + \ldots + USV_{i-n+2} * w_{i-n+2} + USV_{i-n+1} * w_{i-n+1}$$

where w1 < w2 < … wn are weights used to quantify the decreasing importance of old unprocessed states. We have chosen 1/wi = (1/wi-1) – 1, with w1 = 1/n, although any monotonous increasing sequence could have been used. We consider that all states i-k where k>5 have no real significance in estimating the current state because they are too old. So we chose n equal to 5.

$$\textit{if } (f(i) < 0) \textit{ then processed state}_i = CONGESTED$$
$$\textit{if } (f(i) == 0) \textit{ then processed state}_i = LOADED$$
$$\textit{if } (f(i) > 0) \textit{ then processed state}_i = UNLOADED$$

Information update in Client Managers is made asynchronously, every time an RTCP report arrives. We have chosen to completely ignore the first RTCP report since the moment a

Client joins a new stream, because we observed that this report usually contains a very high packet loss rate value. That value is due to temporary transition load and does not reflect an actual congestion reason. Had we taken it into account, it would force the next few processed states to be found CONGESTED, and would therefore tend to invoke a new unwanted transition towards a lower stream. Stream Managers update their rates synchronously and therefore time in system operation is divided in epochs of certain length. At the end of an epoch, each Stream Manager polls the states of all the Client Managers that correspond to a Client receiving this stream and determines the improvement or degradation in this stream's video quality. Whether there will be an improvement or degradation is determined as follows: If all receivers (the number n of the receivers can easily computed by the RTCP protocol) are in the UNLOADED state, video quality is improved. If more than a certain threshold of receivers is CONGESTED, video quality is degraded. In other cased we keep the current video quality. The threshold used for our simulations was one-second of all receivers listening to the stream.

The new bit rate is estimated using an Additive Increase, Multiplicative Decrease (AIMD) algorithm, just like TCP. Increase is achieved by adding a standard small value to the previous bit rate, and is therefore quite conservative in bandwidth consumption, while decrease is achieved by multiplying the previous bit rate with a number in the range of 0…1 (typically around 0.75) and so the algorithm is more aggressive when trying to react to congestion.

There are three cases in this phase that will lead to a Client's transition towards another stream:

- If the stream from which the Client is currently receiving video has already reached its lowest transmitting rate and the Client is still in CONGESTED state then the Client stops listening to this stream and joins the stream of a lower quality stream (if such a stream exists).
- If the stream from which the Client is currently receiving video has already reached its highest transmitting rate and the Client is still in UNLOADED state then the Client stops listening to this stream and joins the stream of a higher quality stream (if such a stream exists).
- The third case applies to a Client that co-exists in a stream with low capacity receivers but is capable of handling better quality video, so it has been unable to improve the video quality of the current stream. The mechanism used aims in making the protocol more conservative and operates by counting the number of consecutive times the receiver was UNLOADED but failed to improve the video quality. When this number exceeds a certain limit, we assume that the receiver has indeed higher capabilities and move it to a better quality stream. Transition from one stream to another also means that the Client's corresponding Client Manager module will now interact with the new Stream Manager.

For a Client transition to occur, an additional rule is imposed in all cases: the Client must have sent a minimum number of reports (for our simulations this number was set to 5) since it joined the stream before leaves that stream. This rule tries to avoid very often transitions that cause unnecessary processing load and waste bandwidth. The Client is forced to stay in a stream for at least a certain amount of time (the time it takes to send the minimum number of RTCP reports) and so if it nevertheless decides to change stream after this time period, we are certain that this is a justified decision.

We declare as unsuccessful stream change the situation when a Client joins a stream with higher transmission rate (or a lower transmission rate) and after a sort time period (T_{change}) return to the previous stream. During our performance evaluation, we observe that the unsuccessful stream changes by the Clients cause instability to the operation of the proposed mechanism and must be avoided. In order to avoid unsuccessful stream changes by the Clients, when a Client makes an unsuccessful stream change we set the condition of the corresponding Client Manager to LOADED, we ignore the next 3 RTCP reports and we avert the Client to make the stream change which was unsuccessful for the next 2* T_{change} time.

We have to point out that Clients make transitions between streams synchronized at the end of each epoch. This helps us avoid possible problems that could be caused for example by two Clients sitting behind the same link and receiving different bit rates.

4. PERFORMANCE EVALUATION

In this section, we present a number of simulations that we made in order to analyze the behavior of the proposed mechanism during the multicast transmission of multimedia data with the use of simulcast approach. We implemented our mechanism and run simulations in the LBNL network simulator ns-2 ([10]).

We run two simulations: (1) Multicast transmission of adaptive multimedia in heterogeneous Clients and UDP traffic at the same time. (2) Multicast transmission of adaptive multimedia in heterogeneous Clients and TCP traffic at the same time. The Server transmitted three streams with the following limits: stream one: 100Kbps-600Kbps, stream two: 600Kbps-1100Kbps and stream three: 1100Kbps-1600Kbps. During the simulations the Server was using the following parameters in order to control the operation of the proposed mechanism: a=0.75, b=0.8, γ=2, LR_u=0.01, LR_c=0.055 and T_{change} = 20 sec (we obtain the above values through a number of experiments [2]). The AIMD algorithm of the Server was increasing the transmission rate of the stream one by 50Kbps, the transmission rate of the stream two by 70Kbps and transmission rate of the stream three by 100Kbps, during unloaded network periods and was decreasing the transmission rate of all the streams by 80% during network congestion periods.

Figure 2 shows the simulation topology. The bandwidth of each link is given to the simulation topology and varies from 0.7 Mbps to 4.0 Mbps. All the links in the simulation topology are full duplex, have delay 10 ms and they use the drop-tail (FIFO) policy to their queues. During the simulations, we have one Server (S) that multicast multimedia data to a group of 20 Clients (C1 to C20) with the use of the proposed mechanism. Clients C1 to C10 are connected to router n2 and Clients C11 to C20 are connected to router n3. The Clients can be divided in to three categories: (1) High capacity Clients with 1.7 Mbps available bandwidth, (2) Medium capacity Clients with 1.2 Mbps available bandwidth and (3) Low capacity Clients with 0.7 Mbps available bandwidth.

We execute the simulations for 300 seconds and the Server starts transmitting the stream one with transmission rate 100Kbps, the stream two with transmission rate 600Kbps and the stream three with transmission rate: 1100Kbps. In order to avoid synchronization, the Clients join randomly the stream one during the first 10 seconds of the simulation.

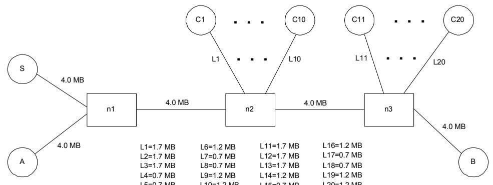

Figure 2. Simulation topology

4.1 First Simulation: Transmission with Background UDP Traffic

During this simulation, we investigate the behavior of the proposed mechanism during network congestion produced by a greedy UDP traffic. In order to produce UDP traffic, we attach to node A of the simulation topology, a CBR (Constant Bit Rate) traffic generator (CBR-Source), which transmits data to a CBR-Receiver attached to node B of the simulation topology. The CBR-Source produces UDP traffic with constant transmission rate of 2.5 Mbps. The CBR-Source starts the transmission of the data at 100th second, and stops the transmission of the data at 200th second.

Figure 3 shows the transmission rates of Server streams and the UDP traffic during the simulation three and Figure 4, Figure 5 and Figure 6 show the reception rate of a representative high capacity Client, a representative medium capacity Client and a representative low capacity Client respectively during the simulation three.

In this simulation the Server except of treat with fairness the group of Clients, it must share the bandwidth of the congested links between the router n1, n2 and between router n2, n3 with the CBR-Source, when the CBR-Source transmits data. As Figure 3 shows, the Server streams start from their minimum transmission rates and increase their transmission rates while Clients join them. Around 50th second all the three streams have reach their maximum transmission rates. When the transmission of UDP traffic starts (at 100th second), congestion occurs to links between the router n1, n2 and between router n2, n3. The Clients prefer smaller transmission rates due to congestion condition, and the Server reduces its transmission rate. When the transmission of UDP traffic stops (200th second), the Server gradually consumes again the available bandwidth.

As Figure 4, Figure 5 and Figure 6 show the Clients after some seconds join the stream that fulfils most their capabilities. When the transmission of CBR takes place, most of the Clients change stream and join the stream with smaller transmission rate due to the congestion condition. Low capacity Clients do not have the capability to go a stream with smaller transmission rate and keep receiving stream one. When the transmission of CBR stops, the Clients return to their initially stream and keep receiving this stream until the end of the simulation.

It is obvious from Figure 3 that the proposed mechanism has good behavior during network congestion condition produced by greedy UDP traffic. When the transmission of UDP traffic starts the Server reduces its transmission rate and when the transmission of UDP traffic stops the Server consumes again the available bandwidth.

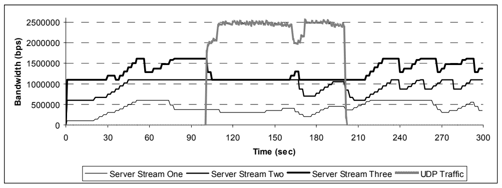

Figure 3. Server Streams and UDP traffic transmission rates during simulation three

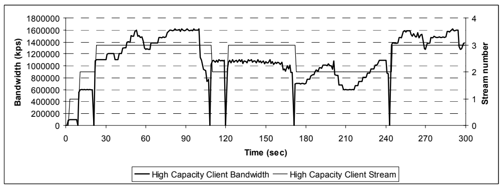

Figure 4. Bandwidth of high capacity Client during simulation three

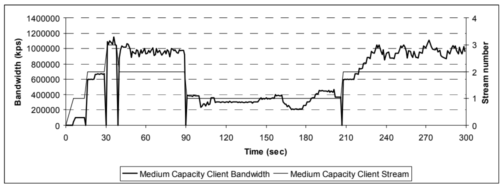

Figure 5. Bandwidth of medium capacity Client during simulation three

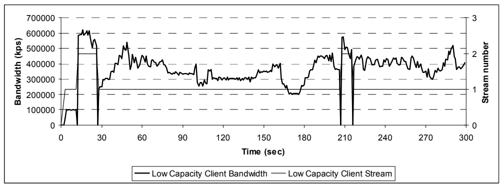

Figure 6. Bandwidth of low capacity Client during simulation three

4.2 Second Simulation: Transmission with Background TCP Traffic

During this simulation, we investigate the behavior of the implemented mechanism against TCP traffic. We use again the simulation topology of Figure 2 but we have set the bandwidth of links n1-n2 and n2-n3 to 3.3Mbps in order to ensure that heavy congestion will occur during the simulation. In order to produce TCP traffic, we connect to node A and B of the simulation topology, an FTP server and an FTP client respectively. The FTP server transmits a file to the FTP client using "4.3BSD Tahoe TCP" protocol [16]. The transmission of the file from the FTP server to the FTP client, starts at the 100th second and stops at the 200th second.

Figure 7 shows the transmission rates of Server streams and the TCP traffic during simulation two and Figure 8, Figure 9 and Figure 10 show the bandwidth of a representative high capacity

Client, a representative medium capacity Client and a representative low capacity Client respectively during simulation two.

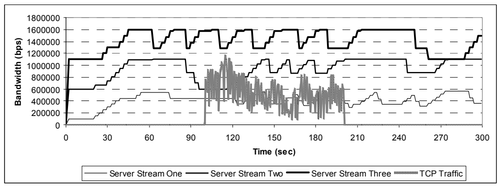

Figure 7. Server Streams and TCP traffic transmission rates during simulation two

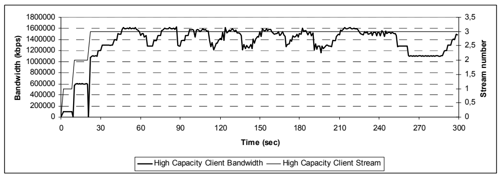

Figure 8. Bandwidth of high capacity Client during simulation two

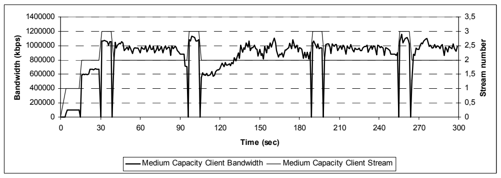

Figure 9. Bandwidth of medium capacity Client during simulation two

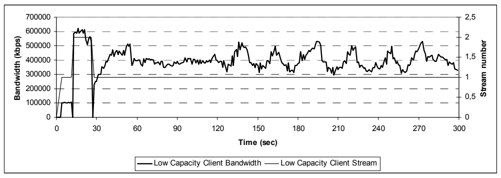

Figure 10. Bandwidth of low capacity Client during simulation two

In this simulation, the Server except of treating the group of Clients with fairness, it must share the bandwidth of the congested links between the router n1, n2 and between router n2, n3 with the TCP traffic when the FTP transmission of the file takes place. As Figure 7 shows, the Server streams start from their minimum transmission rate and increase their transmission rates while Clients join them. Around the 50th second all three streams have reached their maximum transmission rates. When the transmission of TCP source starts (at the 100th second), congestion occurs to links between the router n1, n2 and between router n2, n3. The Clients prefer smaller transmission rates due to congestion condition, and the Server releases bandwidth so that the TCP traffic can use it. When the transmission of the TCP traffic takes place, the Server releases some bandwidth (about 0.5 Mbps) for a while and reserves it again. When the transmission of TCP traffic stops (100th second) the Server gradually reserves again the available bandwidth.

As Figure 8, Figure 9 and Figure 10 show the Clients after some seconds have joined the stream that better fulfils their capabilities. When the transmission of TCP takes place most of the Clients do not change stream and keep receiving the same stream with reduced transmission rate due to the congestion condition. Medium capacity Client tries to join stream one but after some seconds returns again to Stream two, which better meets its capabilities.

It is obvious from Figure 7 that the behavior of our mechanism to TCP traffic is friendly. The TCP traffic has transmission rate of more than 0.4 Mbps many times and maximum transmission rate of 1.2Mbps during the simulation, which is good performance for TCP transmission. In addition, the Server many times releases bandwidth and provides it to TCP source and in one case (115th second) the Server releases 0.5 Mbps of its bandwidth. The Server has the following drawback: The Server's transmission rate during the transmission of TCP traffic is not stable. The Server would have ideal behavior if it reduced its transmission rate and kept it steady while the transmission of TCP traffic took place.

4.3 Comparison of simulation and experimental results

In this paragraph, we compare the simulation results, which are presented in this paper, with the experimental results, which have been obtained through the experimental evaluation of the proposed mechanism ([4]).

In outline, the proposed mechanism has similar behavior both in the simulation environment and in the real network environment where the experiments presented in [4] took place. In both cases the proposed mechanism behaves the same against TCP traffic: In addition, in both cases the proposed mechanism behaves the same during heavy network congestion: Figure 11 shows the transmission rate of TCP traffic and the transmission rate of the implemented prototype in the experimental testbed.

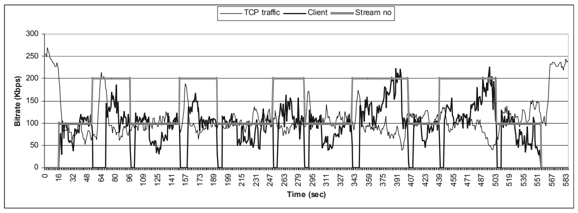

Figure 11. Transmission rate of TCP traffic and the implemented prototype in the experimental test-bed

We notice the following differences in the behavior of the proposed mechanism between the simulation environment and the real network environment:

- The transmission rate of the Server is not so stable in the real network environment as it is in the simulation environment.
- The Server needs more time to find the streams' transmission rates that satisfy most the heterogeneous group of receivers in the real network environment comparing with the simulation environment

The above differences derive from the following facts:

- During the simulation, we assume that the encoder of the Server has the capability to produce any transmission rate that the proposed mechanism suggests. This is not true during the experiments in a real network environment due to the fact that depending on the used compression scheme and the data content, the encoder might only be able to change its transmission rate in steps. When the proposed mechanism suggests a new transmission rate and the encoder cannot produce it, cause instability to the operation of the proposed mechanism. This is the reason why during the experiments in a real network environment the transmission rate of the Server is not stable.

- During the simulation, we assume that the CPU of Server is powerful enough to encode all the transmitted streams. This is not always true during the experiments in a real network environment. Many times the CPU can be overloaded, which has as result the instability of the Server operation. Due to this instability, the Server cannot keep the transmission rate that the proposed mechanism suggests. This lead to the above described behavior of Server.

In order to avoid the above describe undesirable behavior of the proposed mechanism during the experiments in a real network environment, we have to take in account constrains that the multimedia communication over the Internet has. This constrains are presented in [14].

5. CONCLUSIONS - FUTURE WORK

In this paper, we present the behaviour investigation of a mechanism for multicast transmission of adaptive multimedia data in a heterogeneous group of receivers with the use of replicated streams. We investigate the behaviour of the proposed mechanism through a number of simulations. In addition, we compare the simulation results presented in this paper with the experimental results, which we have been presented in paper [4]. Main conclusion of the simulation was that the proposed mechanism has friendly behaviour against the dominant traffic types (TCP traffic) of today's Internet and good behaviour during congestion condition. In addition the proposed mechanism treat with fairness a heterogeneous group of Clients.

Our future work includes the investigation of dynamically adding more streams instead of the static number of streams (specified during initialisation) that the proposed mechanism supports now. In addition we will enhance the proposed mechanism in order to support multicast of layered encoded video and we will evaluate this new version. Moreover we plan to increase the reliability and the efficiency of the implemented prototype in order to overcome the drawbacks that the implemented prototype has. In addition, our future work includes the improvement of the proposed mechanism's behavior against TCP traffic. In addition we will investigate the behavior of the proposed mechanism during the multicast transmission in very large group of receivers. The multicast transmission in very large group of receivers encounters the feedback implosion problem ([1]).

6. BIBLIOGRAPHY

[1] Bolot J.-C., Turletti T., Wakeman I., *Scalable feedback control for multicast video distribution in the Internet,* In Proceedings of SIGCOMM 1994, pp. 139-146, London, England, August 1994. ACM SIGCOMM.

[2] Bouras Ch., Gkamas A., *Streaming Multimedia Data With Adaptive QoS Characteristics*, Protocols for Multimedia Systems 2000, Cracow, Poland, October 22-25, 2000, pp 129-139.

[3] Bouras Ch., Gkamas A., Karaliotas An., Stamos K., *An Architecture for Redundant Multicast Transmission Supporting Adaptive QoS,* Workshop on Multimedia Information Systems, November 7-9, 2001 - Capri, Italy.

[4] Bouras Ch., Gkamas A., Karaliotas An., Stamos K., *Architecture And Performance Evaluation For Redundant Multicast Transmission Supporting Adaptive QoS,* 2001 International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2001) October 09-12, 2001 Split, Dubrovnik (Croatia) Ancona, Bari (Italy).

[5] Chang Y., Messerschmitt D., *Adaptive layered video coding for multi-time scale bandwidth fluctuations,* submitted to IEEE Journal on Selected Areas in Communications.

[6] Cheung S. Y., Ammar M., Li X., *On the Use of Destination Set Grouping to Improve Fariness in Multicast Video Distribution*, INFOCOM 96, March 1996, San Fransisco.

[7] Floyd S., Fall K., *Promoting the Use of End-to-End Congestion Control in the Internet,* IEEE/ACM Transactions on Networking, 1998.

[8] Jiang T., Zegura E. W., Ammar M., *Inter-receiver fair multicast communication over the Internet*, In Proceedings of the 9th International Workshopon Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), pp. 103-114, June 1999.

[9] Kim T., Ammar M. H., *A comparison of layering and stream replication video multicast schemes*, Proc. NOSSDAV'01, Port Jefferson, NY, June 25-26, 2001.

[10] McCanne S., Floyd S., *The UCB/LBNL network simulator, software online,* http://www.isi.edu/nsnam/ns/.

[11] McCanne S., Jacobson V., *Receiver-driven layered multicast*, 1996 ACM Sigcomm Conference, pp. 117-130, August 1996.

[12] Pandhye J., Kurose J., Towsley D., Koodli R., *A model based TCP-friendly rate control protocol*, Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Basking Ridge, NJ, June 1999.

[13] Shculzrinne H., Casner S., Frederick R., Jacobson V., *RTP: A Transport Protocol for Real-Time Applications*, RFC 1889, IETF, January 1996.

[14] Sisalem D., and Wolisz A., *Constrained TCP-friendly congestion control for multimedia communication,* tech. rep., GMD Fokus, Berlin Germany, Feb. 2000.

[15] Sisalem D., Wolisz A., *LDA+ TCP-Friendly Adaptation: A Measurement and Comparison Study,* in the 10th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'2000), June 25-28, 2000, Chapel Hill, NC, USA.

[16] Stevens W., *TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms*, RFC 2001, January 1997.

[17] Vickers B. J., Albuquerque C. V. N., Suda T., *Adaptive Multicast of Multi-Layered Video: Rate-Based and CreditBased Approaches,* Proc. of IEEE Infocom, March 1998.