Multimedia Tools and Applications, 25, 85–110, 2005
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Architecture and Performance Evaluation for Redundant Multicast Transmission Supporting Adaptive QoS

C. BOURAS bouras@cti.gr
A. GKAMAS gkamas@cti.gr
A. KARALIOTAS karaliot@cti.gr
K. STAMOS stamos@cti.gr
Computer Engineering and Informatics Department, University of Patras, GR-26500 Patras, Greece; Research Academic Computer Technology Institute, Riga Feraiou 61, GR-26221 Patras, Greece

Abstract. In this paper we describe the architecture of an application that was developed for the transmission of multimedia data, using the multicast mechanism, over the Internet. There are two major issues that have to be considered when designing and implementing such a service, the fairness and the adaptation schemes. The fairness problem results from the fact that Clients with different capabilities have to be served. In our application we use a mechanism that categorizes the Clients into a number of groups according to each Client's capabilities and (the mechanism) serves each group of Clients with a different multicast stream. With the term "capabilities" we do not only mean the processing power of the Client, but also the capacity and the condition of the network path towards that Client. Because of today's Internet heterogeneity and the lack of Quality of Service (QoS) support, the Server cannot assume that the Clients will permanently be able to handle a specific bit rate. We have therefore implemented an additional mechanism for the intra-stream bit rate adaptation. 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 experiments and a number of simulations in order to examine its behaviour to a heterogeneous group of Clients and its behaviour against TCP and UDP data streams.

Keywords: IP based networks and services, multimedia systems and services, multicast, Quality of Service, adaptation mechanisms, CORBA

1. Introduction—Related work

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. The decision whether the rate will increase or decrease is based on feedback information that the Clients send back to the Server. Many researchers believe that this end-to-end control scheme must be implemented in the endpoints because today's Internet architecture does not provide such a mechanism in the network layer [11, 22].

The implementation of adaptation mechanisms in the applications is often criticized. The main arguments that rise against it, are that the technologies that are used today for the implementation of the core networks provide capabilities to support QoS; as a result the network should offer to the applications QoS guarantees. This is generally true but there is a big problem about it: Today's Internet is divided into thousands of different administration domains. The QoS strategies that are implemented on each one are certainly different (for example QoS based on DiffServ Concept [21], QoS based on IntServ Concept [5], QoS based on IPv6 infrastructure [10]), and in many cases no QoS strategy is implemented at all. So the multimedia data flows that have to traverse many of these different domains in order to reach the end user don't have a sufficient QoS support. The proposed mechanism provides an adaptation service which does not require any QoS support from the network, and runs in any IP multicast network. Another idea widely supported among network administrators, is that the cost of exhaustive monitoring of the network as well as the upgrade of the links that constrain the entire network domain (bottlenecks and critical links) cost less than the deployment of QoS schemes (Research, testing and personnel training) [11].

In addition, any application that transmits data 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. We define as TCP friendly flow, a flow that consumes no more bandwidth than a TCP connection, which is traversing the same path with that flow [22].

The system we propose is based on multicast video transmission with the use of RTP/ RTCP [24]. The main perspectives we tried to fulfil are (1) each Client 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.

In order to achieve the first goal, we create *n* different streams (in most network conditions a small number of different streams is enough—typically 3 or 4 streams), each one within certain bandwidth limits. All the streams carry the same video information, each one of them having a different quality. Clients join in the appropriate stream depending on the condition of the network path towards them and the processing power of each one. If meanwhile the Client detects that the stream it has joined isn't suitable for it any more another implemented mechanism is used in order to provide the Clients with the capability of moving into another stream.

In order to achieve the second goal, we deploy the Additive Increase Multiple Decrease (AIMD) scheme in the inter-stream adaptation algorithm. The adaptation mechanism adapts the rate of each stream taking into account the number of the Clients that are congested or unloaded. In addition, if the capabilities of a Client aren't suitable for the stream it has joined, it moves to another stream with lower or higher bandwidth limits.

The most prominent feature of the proposed mechanism is its transmission rate estimation algorithm, which is based not only to packet loss rate estimations (as the other AIMD algorithms do) but also to the delay jitter estimations. As our experiments and simulation show, delay jitter can be used successfully as congestion indication. In addition we have implement a prototype based on the proposed mechanism and we have evaluate the proposed mechanism not only in a simulator environment but also in a real network.

The methods proposed for the multicast transmission of time sensitive data in the Internet can be generally divided in three main categories, depending on the number of multicast streams used:

- The Server uses a single multicast stream for all Clients [1, 24, 27, 28, 33]. This results to the most effective use of the network resources, but on the other hand the fairness problem arises.
- Simulcast: The Server 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 Clients with similar capabilities [9, 15]. 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 (MPEG, H263, JPEG). So each Client joins in the stream that carries the video quality, in terms of bit rate, that it is capable of receiving. The main disadvantage in this case is that the same video information is replicated over the network but recent research [16] has shown that under some condition simulcast behaves better than transmission of layered encoded video.
- The Server uses layered encoded video, which is video that can be reconstructed from a number of discrete data streams and transmits each layer into different multicast stream [6, 8, 19, 31]. The video is divided in to one basic stream and more additional streams. The basic stream provides the basic quality and the quality improves with each layer added. The Clients subscribe to one or more multicast streams depending on the available bandwidth into the network path to the Server.

This work is based on the simulcast approach [9] and it is an extension of the work, which has been presented in [2] and [4]. The discussion of the limitations of simulcast approach (compared, e.g., to transmission of layered encoded video) is beyond the scope of this paper. The rest of this paper is organised as follows: Section 2 presents the architecture of the implemented prototype. In Section 3, we give a detailed description of the operation of our prototype application. Section 4 presents some implementation issues. In Section 5, we present the performance evaluation of the implemented prototype. Finally, Section 6 concludes the paper and discusses some of our future work.

2. System architecture

The proposed mechanism is based on the simulcast transmission of video, 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 performance evaluation we use three multicast streams), (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 Client that has joined the stream controlled by this Stream Manager. The Synchronisation Server is

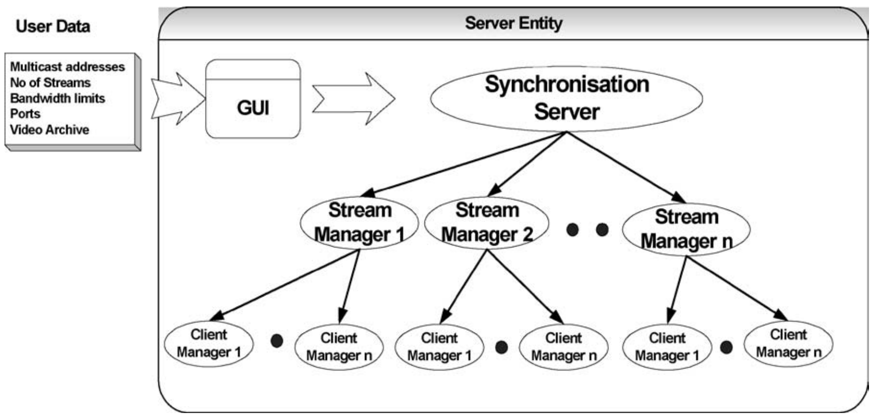


Figure 1. The architecture and the data flow of the Server.

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.

The Client architecture consists of the following modules:

- *Client buffer:* Multimedia data received are first stored in this module, and presentation does not begin unless there is a necessary amount of data stored in the Client buffer. In order to achieve smooth media presentation to the user, this buffer's capacity has to exceed the maximum delay jitter during data transmission.
- *Feedback:* This is the module that produces the information necessary for the Client Manager at the Server to estimate the Client's state. Control information is transmitted

with RTCP reports, which include, as mentioned earlier, information about packet loss rate and delay jitter.

- *Decoder:* This module takes the data packets from the Client buffer as input, decodes them and outputs them suitable for presentation. The quality of the presented video is higher when the receiving data rate is high. Video quality can also be affected by packet loss and delay. Presentation can come to a complete stop if data in the Client buffer drops below the required minimum.
- *User display:* The module responsible for the presentation of the video to the user, which can be a computer monitor.

3. Description of system operation and algorithms

The Server initially constructs a number of streams. When a Client joins a multicast stream, 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 Client Manager: For the packet loss rate:

$$\mathrm{LR}_{\mathrm{new}} = a * \mathrm{LR}_{\mathrm{old}} + (1 - a) * \mathrm{LR}_{\mathrm{net}}$$

Where: $\mathrm{LR}_{\mathrm{new}}$: The new filtered value of packet loss rate. $\mathrm{LR}_{\mathrm{old}}$: The previous filtered value of packet loss rate (for the first report after the start of transmission, this value is 0). $\mathrm{LR}_{\mathrm{net}}$: 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 and during our evaluation we set $a = 0.75$). For delay jitter:

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

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 and during our evaluation we set $b = 0.8$).

With the use of the above filters, we prevent a single spurious packet loss or packet delay having an excessive effect on the packet loss rate or delay jitter estimation. We have evaluated the good performance of the above filters through experiments presented in [2]. Instead of the above filters, other filters, which have good smoothing behavior, can be used.

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:

$$\begin{aligned}
\text{if } (\mathrm{LR_{new}} \geq \mathrm{LR_c})\ \text{unprocessedstate} &= \text{CONGESTED}\\
\text{if } (\mathrm{LR_u} < \mathrm{LR_{new}} < \mathrm{LR_c})\ \text{unprocessedstate} &= \text{LOADED}\\
\text{if } (\mathrm{LR_{new}} <= \mathrm{LR_u})\ \text{unprocessedstate} &= \text{UNLOADED}\\
\text{if } (J_{\text{new}} > \gamma * J_{\text{old}})\ \text{unprocessedstate} &= \text{CONGESTED}
\end{aligned}$$

We have defined LR_u as the maximum value of the unloaded packet loss rate and LR_c as the minimum value of the congested packet loss rate. Where γ is a parameter, which specifies how aggressive the Client Manager will be to the increase of delay jitter. The values of the above parameters depends on the network where the proposed mechanism is going to run and can be obtained through experiments.

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 Client Manager. 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:

$$\begin{aligned}
&\text{if (unprocessed state}_i == \textit{CONGESTED})\ \text{then USV}_i = -1\\
&\text{if (unprocessed state}_i == \textit{LOADED})\ \text{then USV}_i = 0\\
&\text{if (unprocessed state}_i == \textit{UNLOADED})\ \text{then USV}_i = 1
\end{aligned}$$

The processed state is then determined by the value of

$$\begin{aligned}
f(i) = {} & \mathrm{USV}_i * w_i + \mathrm{USV}_{i-1} * w_{i-1} + \cdots + \mathrm{USV}_{i-n+2}\\
& * w_{i-n+2} + \mathrm{USV}_{i-n+1} * w_{i-n+1}
\end{aligned}$$

where $w_1 < w_2 < \cdots < w_n$ are weights used to quantify the decreasing importance of old unprocessed states. We have chosen $1/w_i = (1/w_{i-1}) - 1$, with $w_1 = 1/n$, although any monotonous increasing sequence could have been used. During our evaluation, we observe 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. Then the processed state is computed based on the following equations:

$$\begin{aligned}
\text{if } (f(i) < 0)\ \text{then processed state}_i &= \text{CONGESTED}\\
\text{if } (f(i) == 0)\ \text{then processed state}_i &= \text{LOADED}\\
\text{if } (f(i) > 0)\ \text{then processed state}_i &= \text{UNLOADED}
\end{aligned}$$

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 Clients (the number n of the Clients can easily computed by the RTCP protocol) are in the UNLOADED state, video quality is improved. If more than a certain threshold of Clients is CONGESTED, video quality is degraded. In other cases, we keep the current video quality. The threshold used for our simulations was one-second of all Clients 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 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 Clients 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 Client was UNLOADED but failed to improve the video quality. When this number exceeds a certain limit (for our simulations this number was set to 4 which results a minimum of 20 seconds[1] between stream change which is a time space enough in order to take a justified decision), we assume that the Client 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.

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}) returns 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 avert the Client to make the stream change, which was unsuccessful for the next $2^k * T_{\text{change}}$ time (where k the number of continuant unsuccessful stream changes since the last successful stream change). Due to fact that T_{change} affects linearly the value $2^k * T_{\text{change}}$ time and the k affects the value of $2^k * T_{\text{change}}$ exponentially, during our evaluation we set T_{change} to 20 seconds but also other values of T_{change} can be used.

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.

The conservatism our protocol exhibits has two advantages: (1) We successfully avoid unnecessary stream changes through the tracking of Clients' unsuccessful stream changes. (2) Our protocol is TCP-friendly, because it only consumes excessive bandwidth when it is absolutely certain that this bandwidth can be handled, and furthermore uses the conservative AIMD algorithm.

4. Implementation issues

For the implementation of our system we used the Java Programming Language, and in particular the Java Media Framework API [13]. Java's object-oriented model fits our design and JMF offers a convenient level of abstraction, which allows the developer to concentrate on high-level issues, thus making it an ideal platform for experimental research. In particular, JMF provides support for RTP transmission and reception of real-time media streams across the network. It offers some very useful classes and interfaces, like the Session Manager, that encapsulates the creation, maintenance and closing of an RTP session, the Processor that encapsulates processing and control of time-based media data and the DataSource that encapsulates media protocol-handlers. Our JMF-based implementation is represented by figures 2 and 3. Figure 4 shows the Graphical User Interface of the Client.

All communication between the Server and the Clients is achieved using CORBA. This technology allows a module written in any language that supports CORBA to be integrated seamlessly in our system, as long as it implements a small number of functions necessary for remote communication.

CORBA communication between the Server and the Clients also requires a third entity, the Naming Service. It can be located on the same host as the Server or on any other host in

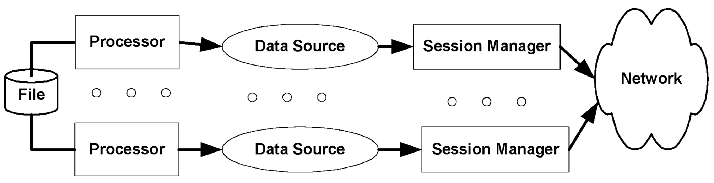


Figure 2. The Server operation.

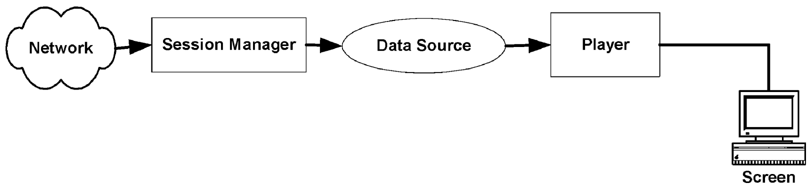


Figure 3. The client operation.

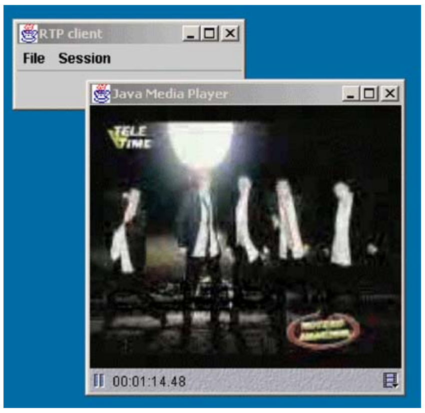


Figure 4. Client's GUI.

the network. All Clients and the Server, however, must know its location. When the Server is initialised, it registers itself and all the Stream Managers it creates to the Naming Service using a hierarchical representation similar to an operating system's file structure. When a Client is started it uses the Naming Service to request a reference to the Stream Manager it wishes to receive data from. Every time the Client makes a transition to a different stream, it uses the Naming Service to get a reference to the new Stream Manager. Since communication may also be directed from the Client to the Server, during initialisation every Client also registers itself to the Naming Service. This way the Client Manager module (which is part of the Server entity) can locate its corresponding Client and order it to move to a different stream whenever necessary.

These choices generally indicate our purpose for this implementation to be experiment- and flexibility-oriented, rather than performance-oriented and therefore it can be improved in terms of resource optimisation.

5. Performance evaluation

5.1. Performance evaluation through experiments

In order to evaluate the performance of the implemented prototype, we run three different experiments, each with a different configuration, over a controlled networking test-bed,

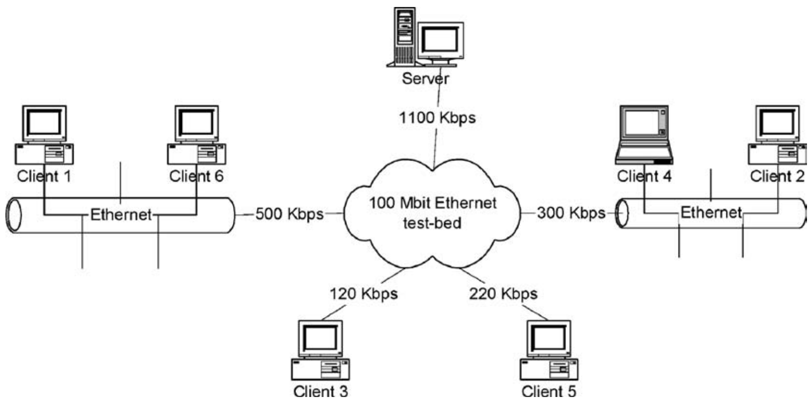


Figure 5. Experiments topology.

which we have implemented over the campus network of University of Patras in Greece. We connect each participant (the Server and the six Clients) with connections of different capacity to our network test-bed with the use of traffic policy on the access router of each participant in the test-bed. Figure 5 shows the experiments topology and Table 1 shows the experiments common parameters. Aim of the performance evaluation through experiments was the proven of the proposed concept and the evaluation of the proper operation of the implemented prototype. Our future work includes the detail validation of the implemented prototype through test over the Internet with use of large participants groups.

5.1.1. First experiment: Transmission into a heterogeneous group of clients. During this experiment we investigate the behaviour of the implemented prototype with a heterogeneous group of Clients. Table 2 provides the first experiment configuration and results.

The behaviour of the implemented prototype was as expected, for example Client 6 and Client 1, which share the same link with 500 Kbps capacity, join stream one, one after the other. After some time Client 6 and Client 1 join stream two almost at the same time point (145th second). Finally at 321st second Client 6 joins the third stream because of the high capacity of the link (500 Kbps) that connects Client 6 to the test-bed. As figure 7 shows, All the Clients, depending on the capacity of the link that connects them to the network test-bed, join the appropriate stream and the Server treats all Clients with fairness. An exception is

Table 1. Experiments common parameters.

Server	One Server transmitting three streams.
Clients	Six Clients all initially connected in stream one.
Server stream limits	Stream one 10–100 Kbps, Stream two 100–200 Kbps, Stream three 200–300 Kbps.
Server parameters	$a = 0.5$, $b = 0.8$, $\gamma = 2$, $\mathrm{LR}_u = 0.02$, $\mathrm{LR}_c = 0.05$ and $T_{\mathrm{change}} = 20$ sec, streams increasing transmission rate: 25 Kbps, streams decreasing transmission rate 50%.

Table 2. First experiment configuration and results.

Topology	Topology of figure 5.
Duration—Scenario	360 seconds—Server initially transmits only the stream. The Clients join the video transmission with the following order: Client 3, Client 2, Client 4, Client 6, Client 1, Client 5.
Results	In general, the behaviour of the implemented prototype was as expected. All the Clients, depending on the capacity of the link that connects them to the network test-bed, join the appropriate stream and the Server treats all Clients with fairness.
Figures	Figure 6 shows the transmission rates of Server streams and figure 7 shows the reception rates of all the Clients.

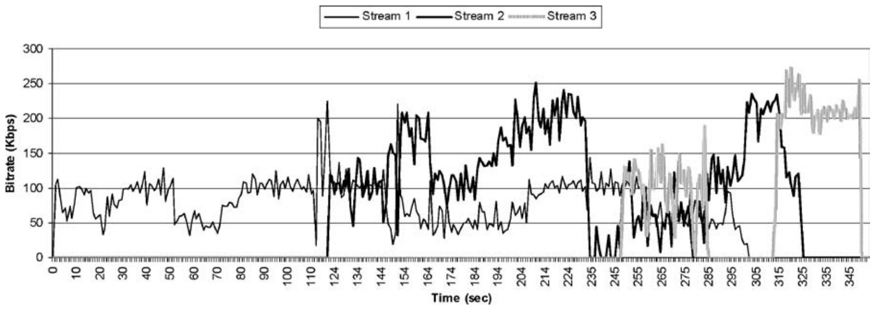


Figure 6. Transmission rates of Server streams during first experiment.

the behaviour of Client 1 between the 264th second and 307th second. During the above period, we expected that Client 1 would join stream three but Client 1 moved to stream one. We believe that the above behaviour of Client 1 is a result of low resources in the workstation that Client 1 was running on. The allocation of the Clients to the appropriate stream takes some time. This is because of the conservative operation of the implemented prototype in order to be TCP-friendly as the following experiment shows.

5.1.2. Second experiment: Transmission with background TCP traffic. In this experiment, we transmit at the same time multimedia data with the use of the implemented prototype and TCP traffic in the same link. During this experiment, we investigate the behaviour of the implemented prototype against TCP traffic. Table 3 shows second experiment configuration and results.

As figure 8 shows the implemented prototype has friendly behaviour towards TCP traffic: When the transmission of video starts, Client 5 joins stream one of the Server and TCP traffic reduces its transmission rate due to the congestion, which takes place to the link of Client 5. After some time Client 5 tries to join stream two with a higher transmission rate. This action of Client 5 produces congestion to the link and Client 5 backs off and returns to stream one in order to release bandwidth for the TCP traffic. The above described behaviour continues until the end of the experiment. During this experiment the TCP traffic

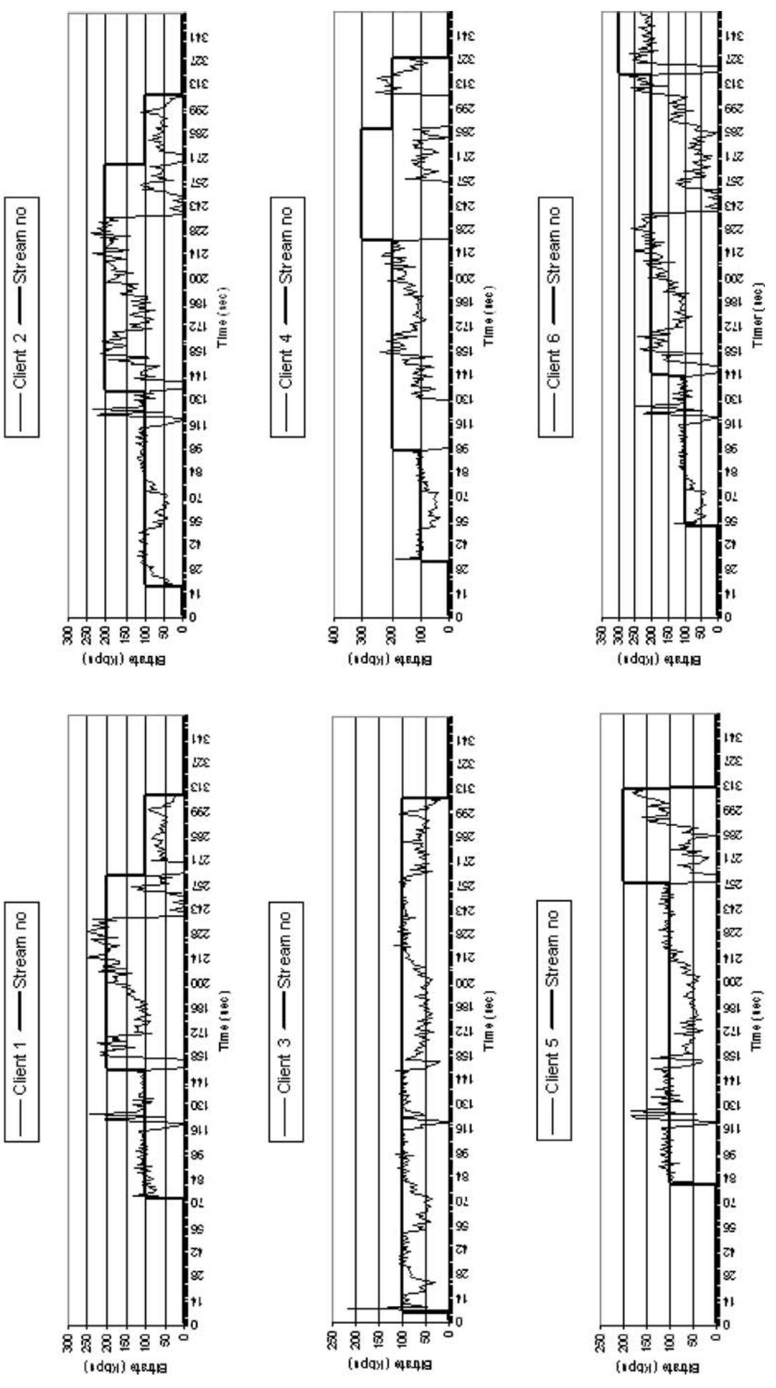


Figure 7. Clients reception rate during first experiment.

Table 3. Second experiment configuration and results.

Topology	The topology of figure 5, except that the capacity of the link, which connects Client 5 with the network test-bed, has been increased to 300 Kbps.
Duration—Scenario	600 seconds—Same scenario with first experiment except that we simultaneously transmit to the link of Client 5 TCP traffic with initial rate of 280 Kbps. (TCP traffic was generated by the LanTrafficV2[a] traffic generator, configured to send packets of 1436 bytes each (the Ethernet MTU) every 40 ms).
Results	During this experiment the TCP traffic has transmission rate of more than 100 Kbps and maximum transmission rate more than 200 Kbps, which is good performance for TCP connection.
Figures	Figure 8 shows the transmission rate of TCP traffic and the reception rate of Client 5 during the second experiment.

[a]http://www.zti.fr.

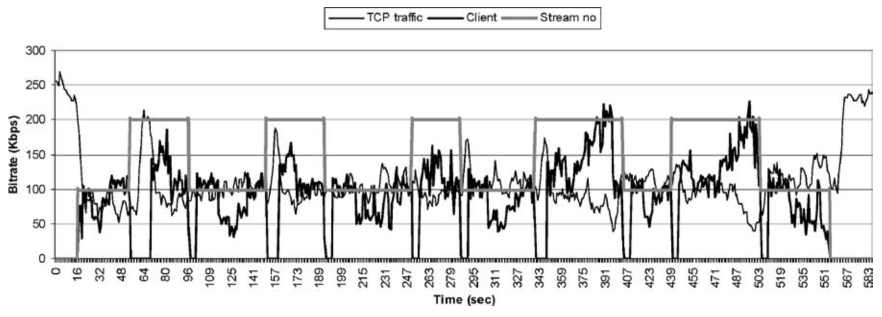


Figure 8. Transmission rate of TCP traffic and the implemented prototype.

has transmission rate of more than 100 Kbps and maximum transmission rate more than 200 Kbps, which is good performance for TCP connection.

5.1.3. Third experiment: Transmission with background UDP traffic. In this experiment, we transmit at the same time multimedia data with the use of the implemented prototype and UDP traffic in the same link. During this experiment, we investigate the behaviour of the implemented prototype against heavily congested conditions, which are produced by UDP traffic that does not implement any congestion control policy. Table 4 provides third experiment configuration and results.

As figure 9 shows, when the experiment starts, UDP traffic occupies all the available bandwidth. When the implemented prototype starts video transmission (70th second), Client 5 joins stream one of the Server and the UDP traffic reduces its transmission rate. Although UDP traffic reduces its transmission rate, this reduction is not sufficient and the UDP traffic continues to dominate the available bandwidth. At 204th, 262nd and 342nd seconds we changed the parameters of the traffic generator in order to reduce the transmission rate of UDP traffic (at the above time periods we briefly stopped the transmission rate of UDP traffic in order to change the parameters of the traffic generator). Gradually the video transmission

Table 4. Third experiment configuration and results.

Topology	The topology of figure 5, except that the capacity of the link, which connects Client 5 with the network test-bed, has been increased to 300 Kbps.
Duration—Scenario	420 seconds—Same scenario with first experiment except that we simultaneously transmit to the link of Client 5 UDP traffic with initial rate of 280 Kbps. UDP traffic began at a rate of 280 Kbps, and we later decreased it to 250, 200 and 175 Kbps. UDP traffic was generated by the LanTrafficV2 traffic generator, configured to send packets of 1436 bytes each (the Ethernet MTU) every 40, 45, 55 and 60 ms, according to the sending rate we wanted to achieve.
Results	UDP traffic dominates the link capacity due to the fact that the UDP traffic does not use any congestion control policy.
Figures	Figure 9 shows the transmission rate of UDP traffic and the reception rate of Client 5 during the third experiment.

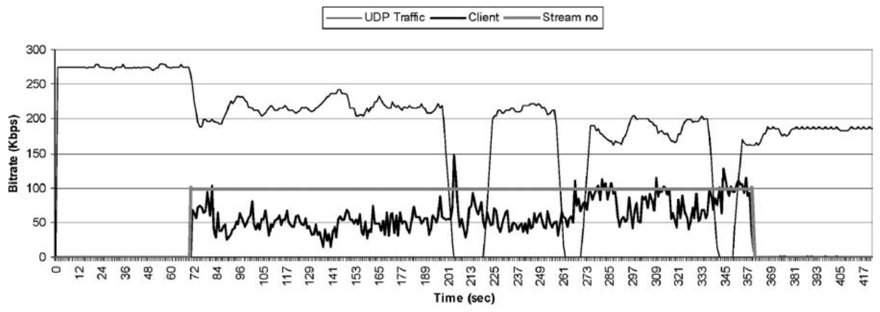


Figure 9. Transmission rate of UDP traffic and the implemented prototype.

reserves more bandwidth in the link but again the UDP traffic dominates the link capacity. The above described behaviour of the implemented prototype is as expected because during the design of the implemented prototype we focused on implementing a TCP friendly application. The fact that the UDP traffic does not use any congestion control policy and the implemented prototype reduces its transmission rate by 50% during congestion periods leads to the above described behaviour. We can improve the behaviour of the implemented prototype against UDP traffic by making the implemented prototype more aggressive into congestion (for example by decreasing the transmission rate only by 15% during congestion periods) but this will have a negative influence to the behaviour of the implemented prototype against TCP traffic, which is not desirable.

5.2. *Performance evaluation through simulations*

In this section, we present a number of simulations that we made in order to analyze the behavior of the implemented prototype during the multicast transmission of multimedia data with the use of simulcast approach. Primary aims of the simulations were the study

Table 5. Simulations common parameters.

Server	One Server transmitting three streams.
Clients	Five to twenty Clients all initially connected in stream one.
Server Stream Limits	Stream one: 100–600 Kbps, Stream two: 600–1100 Kbps and Stream three: 1100–1600 Kbps.
Server parameters	$a = 0.75$, $b = 0.8$, $\gamma = 2$, $\mathrm{LR}_u = 0.01$, $\mathrm{LR}_c = 0.055$ and $T_{\text{change}} = 20$ sec, stream one increasing transmission rate: 50 Kbps, stream two increasing transmission rate: 70 Kbps, stream three increasing transmission rate: 100 Kbps, decreasing transmission rate for all the streams: 50%.

of implemented mechanism fairness regarding the group of Clients and mechanism's behavior regarding the dominant traffic model of today's Internet (TCP and UDP traffic). We implemented our mechanism and run simulations in the LBNL network simulator ns-2 [19].

We choose to evaluate the multicast operation of the implemented prototype through simulation because it is not easy to implement a controlled network test bed with many users in today's Internet. In addition we have the opportunity to compare the operation of the implemented prototype in a network environment and in simulation environment. Table 5 shows the simulations common parameters.

5.2.1. First simulation: Transmission into a multicast distribution tree with shared links. In this simulation we investigate the performance of the proposed mechanism in a heterogeneous multicast environment with a multicast distribution tree that is shared among the Clients. With this approach, we investigate the behavior of the proposed mechanism, when the actions of one Client affect other Clients. Table 6 provides first simulation configuration and results.

Each router ($n2$, $n3$, $n4$) of the simulation topology is shared between the Server streams and an uncorrelated background traffic, which consumes maximally the 50% of the router capacity. In order to produce the uncorrelated background traffic, we use a traffic generator with active and idle periods. During the active periods the transmission rate of the traffic

Table 6. First simulation configuration and results.

Topology	Topology of figure 10: one Server (S), which transmits multimedia data to a group of 5 Clients (C1-C5). The routers of simulation topology are using the drop-tail[a] (FIFO) policy.
Duration—Scenario	1000 seconds—Server starts transmitting the stream one with transmission rate 100 Kbps, the stream two with transmission rate 600 Kbps and the stream three with transmission rate 1100 Kbps. Clients join randomly the stream one during the first 3 seconds of the simulation.
Results	After some seconds each Client joins the stream, which we expect and receives also a bandwidth share close to the bandwidth share, which we expect.
Figures	Figure 11 shows the bandwidth share of the Clients 1 to 5 and figure 12 shows the stream changes of the Clients 1 to 5.

[a]Drop-tail is the most common queue policy to Internet routers.

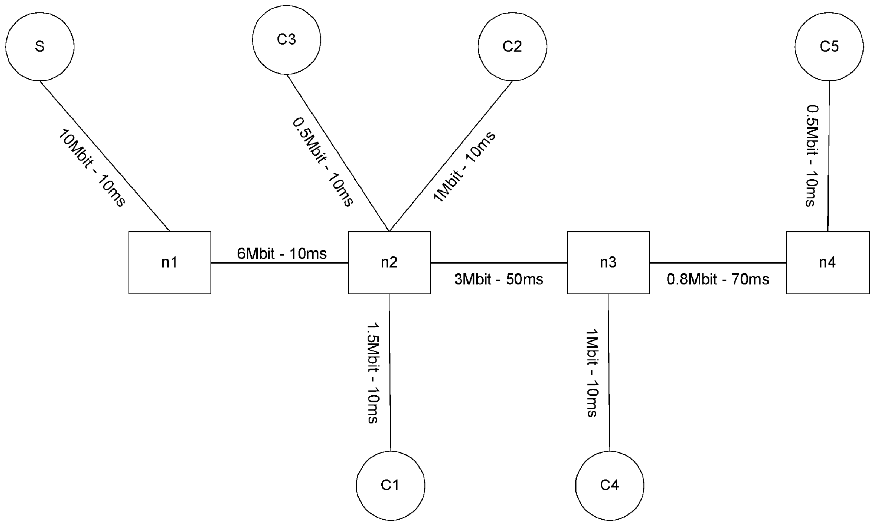


Figure 10. Simulation topology of first simulation.

generator follows a Pareto distribution with a scale factor of 1.1 and a mean of 20 packets. Active transfer phases are then followed by idle periods drawn by a Pareto distribution with a scale factor of 1.8 and a mean 0.5 seconds. As [23] suggests the above traffic generator models background web traffic.

With the above simulation topology, we expect that Client 1 will receive the stream three of the Server, Client 2 and Client 4 will receive the stream two of the Server and Client 3 and Client 5 will receive the stream one of the Server. Moreover the Clients, which we expect to receive the same Server stream, are connected to the Server through paths with different RTT delays and in addition the actions of some Client affect the bandwidth share of other Client (for example the actions of Client 4 affects the bandwidth share of Client 5).

As figures 11 and 12 suggest after some seconds each Client joins the stream, which we expect and receives also a bandwidth share close to the bandwidth share, which we expect. The Clients after some unsuccessful stream changes have join the Server stream which fulfills better their capabilities and stay at that stream until the end of the simulation (due to the tracing of unsuccessful stream changes that the proposed mechanism offers). In addition, due to the synchronization of stream changes the undesirable problems are minimal and in general the Clients actions does affect the bandwidth shares of the other Clients.

5.2.2. Second simulation: Transmission into a heterogeneous group of clients. In this simulation we investigate the behavior of the implemented mechanism and its capability to treat with fairness a bigger heterogeneous group of Clients during the multicast transmission of multimedia data. Table 7 provides second simulation configuration and results.

As the above figures show the operation of the implemented prototype is as expected: The Server streams most of the simulation time have their maximum transmission rate because

Table 7. Second simulation information.

Topology	Topology of figure 13: one Server (S), which transmits multimedia data to a group of 20 Clients (C1 to C20). The routers of simulation topology are using the drop-tail (FIFO) policy. 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.
Duration—Scenario	300 seconds—Server starts transmitting the stream one with transmission rate 100 Kbps, the stream two with transmission rate 600 Kbps and the stream three with transmission rate 1100 Kbps. Clients join randomly the stream one during the first 10 seconds of the simulation.
Results	Clients after some seconds have joined the stream that better fulfils their capabilities.
Figures	Figure 14 shows the transmission rates of Server streams and figure 15 show the bandwidth of three representative Clients during the second simulation.

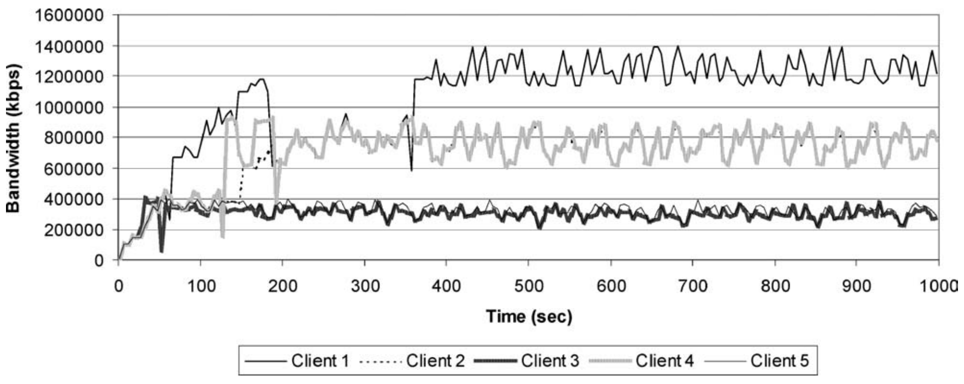


Figure 11. Bandwidth shares of Client 1 to Client 5 during first simulation.

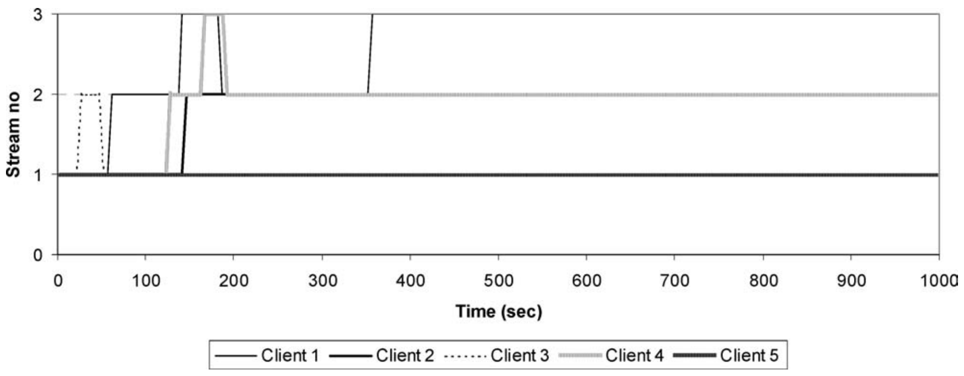


Figure 12. Stream changes of Client 1 to Client 5 during first simulation.

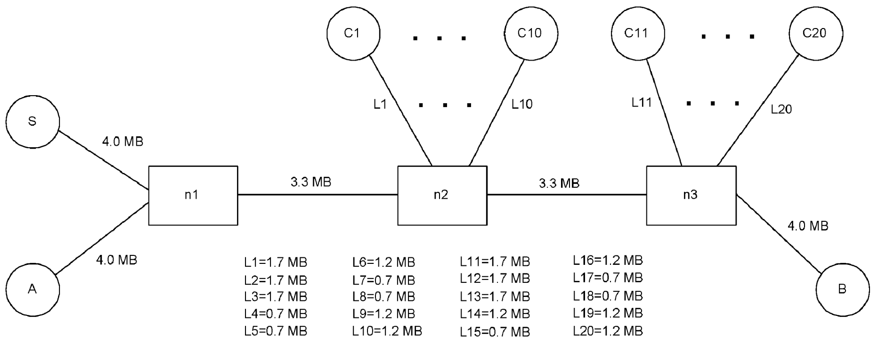


Figure 13. Simulation topology of second, third and fourth simulation.

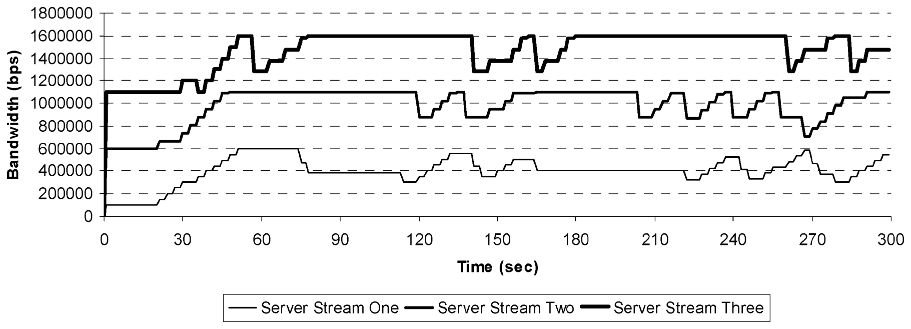


Figure 14. Server streams transmission rates during second simulation.

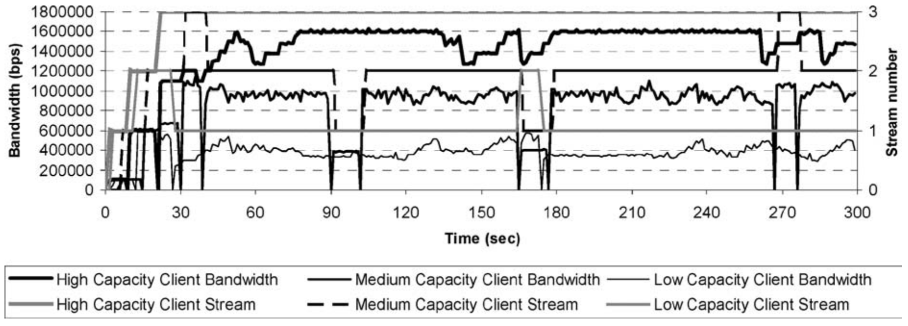


Figure 15. Clients bandwidth during second simulation.

the simulation topology does not have any bottleneck link and the Clients join the stream that better fulfils their capability. When a Client tries to join a steam with either a higher or a lower bandwidth than its available bandwidth, it returns to the initial stream after some seconds.

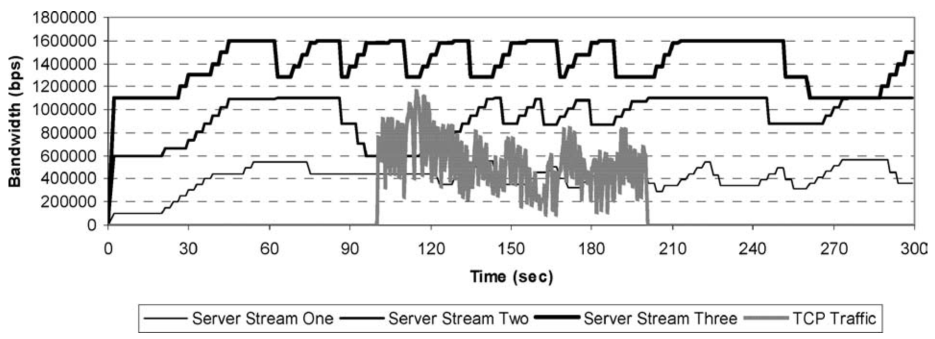


Figure 16. Server streams and TCP traffic transmission rates during third simulation.

5.2.3. Third simulation: Transmission with background TCP traffic. In this simulation, we multicast transmit at the same time multimedia data with the use of the implemented mechanism and TCP traffic. During this simulation, we investigate the behavior of the implemented mechanism against TCP traffic. In order to produce TCP traffic, we connect to node A and B of the simulation topology of figure 13, an FTP server and an FTP client respectively. The FTP server transmits a file to the FTP client using "4.3BSD Tahoe TCP" protocol [30]. The transmission of the file from the FTP server to the FTP client, starts at the 100th second and stops at the 200th second. Table 8 provides third simulation configuration and results.

As figure 16 shows, the Server streams start from their minimum transmission rate and increase their transmission rates while Clients join them. When the transmission of TCP source starts (at the 100th second), congestion occurs to links between the router $n1$, $n2$ and between router $n2$, $n3$ 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.

As figure 17 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.

Table 8. Third simulation configuration and results.

Topology	Topology of figure 13 (same with second simulation) except that we have set the bandwidth of links $n1$-$n2$ and $n2$-$n3$ to 3.3 Mbps.
Duration—Scenario	300 seconds—Same scenario with second simulation except that we transmit of TCP traffic from 100th second to 200th second between node A and B.
Results	Clients after some seconds have joined the stream that better fulfils their capabilities. TCP traffic has transmission rate of more than 0.4 Mbps many times and maximum transmission rate of 1.2 Mbps during the simulation, which is good performance for TCP transmission.
Figures	Figure 16 shows the transmission rates of Server streams and the TCP traffic and figure 17 show the bandwidth of three representative Clients during third simulation.

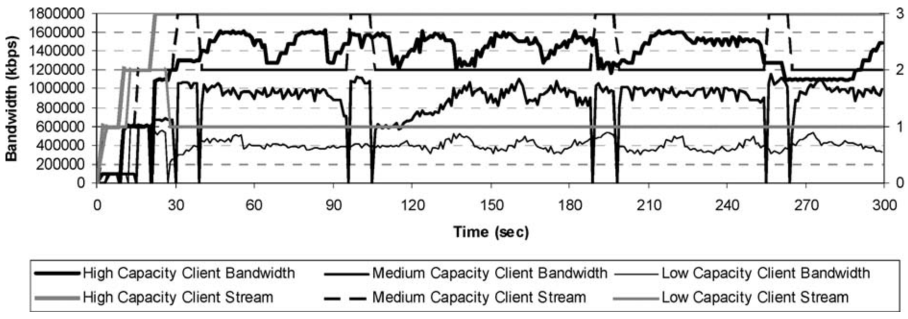


Figure 17. Clients bandwidth during third simulation.

It is obvious from figure 16 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 1.2 Mbps during the simulation, which is good performance for TCP transmission. 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.

5.2.4. Fourth simulation: Transmission with background UDP traffic. In this simulation, we multicast transmit at the same time multimedia data with the use of the implemented mechanism and UDP traffic. During this simulation, we investigate the behavior of the implemented 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 data at 100th second, and stops the transmission of data at 200th second. Table 9 provides fourth simulation configuration and results.

Table 9. Forth simulation configuration and results.

Topology	Topology of figure 13 (same with second simulation).
Duration—Scenario	300 seconds—Same scenario with second simulation except that we transmit UDP traffic from 100th second to 200th second between node A and B.
Results	When the transmission of UDP takes place, most of the Clients change stream and start receiving the stream with smaller transmission rate due to the congestion condition. When the transmission of UDP traffic stops (200th second), the Server gradually reserves again the available bandwidth.
Figures	Figure18 shows the transmission rates of Server streams and the TCP traffic and figure 19 the show the bandwidth of three representative Clients during fourth simulation.

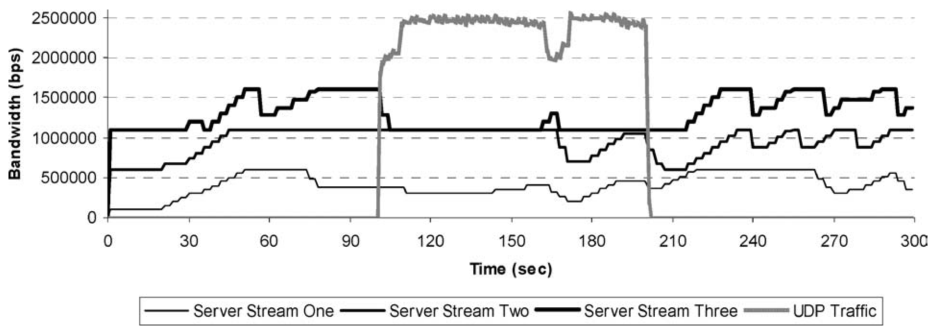


Figure 18. Server streams and UDP traffic transmission rates during fourth simulation.

As figure 18 shows, the Server streams start from their minimum transmission rate and increase their transmission rates while Clients join in. 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 near to 0.5 Mbps and keeps this transmission rate for the next 100 seconds (except from 170th second to 190th second, when the Server releases 1.0 Mbps), during which the transmission of UDP traffic takes place. When the transmission of UDP traffic stops (200th second), the Server gradually reserves again the available bandwidth.

As figure 19 shows, the Clients after some seconds have joined the stream that better fulfils their capabilities. When the transmission of CBR takes place, most of the Clients change stream and start receiving the stream with smaller transmission rate due to the congestion condition.

It is obvious from figure 18 that the proposed mechanism reduces the transmitting rate to the "maximum possible", meaning that it occupies only the bandwidth left by the UDP traffic. This is explained by the fact that our protocol implements an adaptation mechanism while UDP does not.

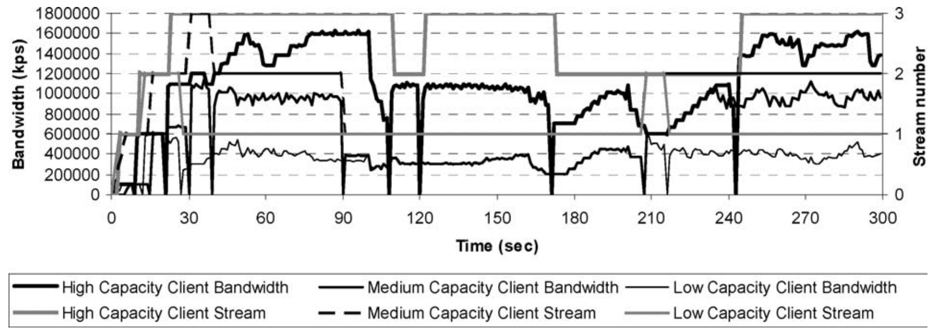


Figure 19. Bandwidth of high capacity Client during fourth simulation.

5.3. Comparison of simulation and experimental results

In outline, the proposed mechanism has similar behavior both in the simulation environment and in the real network environment. In both cases the proposed mechanism behaves the same against TCP traffic and heavy network congestion produced by greedy UDP traffic. 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 most satisfy the heterogeneous group of Clients 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, this causes 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 the 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 leads to the above described behavior of the Server.

In order to avoid the above described undesirable behavior of the proposed mechanism during the experiments in a real network environment, we have to take into account the following constrains that the multimedia communication over the Internet has:

- *Fixed limits*: The quality of a multimedia stream can usually be improved by increasing the bandwidth share of the stream. However, above a certain limit no noticeable improvements in the quality will be observed anymore.
- *Granular adaptation*: Depending on the used compression scheme and data content, a data source might only be able to change its transmission rate in steps.
- *Stable presentation*: To provide the user with a stable perceived quality, the adaptation mechanism needs to limit the maximum changes as well the rate of changes in the transmission rate of a multimedia stream.
- *Loss tolerance*: Depending on the transferred content, the user and the used compression scheme, some data losses might be tolerated during a multimedia communication.

6. Conclusion—Future work

In this paper, we present the architecture of a prototype for multicast transmission of adaptive multimedia data in a heterogeneous group of Clients with the use of replicated streams. We concentrate on the design of a mechanism for monitoring the network condition and estimate the appropriate rate for the transmission of the multimedia data in each stream in order to allocate each Client to the appropriate stream and treat the Clients with fairness. Moreover we implement a TCP-friendly application. We investigate the behaviour of the implemented prototype through a number of experiments and a number of simulations. Our future work includes the validation of the implemented prototype by using it for the multicast transmission of multimedia data in a heterogeneous group of Clients in the Internet. In addition we will perform a detailed validation of the implemented prototype through test over the Internet with use of large participants groups. Moreover, we will investigate the benefits of dynamically adding more streams instead of the static number of streams (specified during initialisation) that the implemented prototype supports now. Finally we intend to enhance the implementation by adding a mechanism in order to dynamically choose and modify the parameters that regulate the aggressiveness of the adaptation.

Note

1. The minimum RTCP retransmission timeout is 5 sec [26].

References

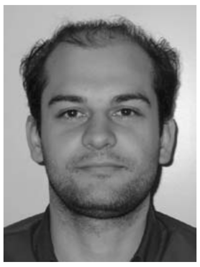
1. J. Bolot, T. Turletti, and I. Wakeman, “Scalable feedback control for multicast video distribution in the internet,” in Proceedings of SIGCOMM 1994, London, England, ACM SIGCOMM, Aug. 1994, pp. 139–146.
2. Ch. Bouras and A. Gkamas, “Streaming multimedia data with adaptive QoS characteristics,” in Protocols for Multimedia Systems 2000, Cracow, Poland, Oct. 22–25, 2000, pp. 129–139.
3. Ch. Bouras, A. Gkamas, A. Karaliotas, and K. Stamos, “Architecture and performance evaluation for redundant multicast transmission supporting adaptive QoS,” in 9th International Conference on Software, Telecommunications and Computer Networks (SoftCom 2001) Split, Dubrovnik(Croatia), Ancona, Bari (Italy), Vol. II, Oct. 09-12 2001, pp. 585–592.
4. Ch. Bouras, A. Gkamas, A. Karaliotas, and K. Stamos, “An architecture for redundant multicast transmission supporting adaptive QoS,” 7th International Workshop on Multimedia Systems, Capri, Italy, Nov. 7–9, 2001, pp. 133–142.
5. R. Braden, D. Clark, and S. Shenker, “Integrated services in the internet architecture: An overview,” RFC 1633.
6. J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shaver, “FLID-DL: Congestion control for layered multicast,” in Proceedings of NGC 2000, Nov. 2000, pp. 71–81.
7. S. Cen, C. Pu, and J. Walpole, “Flow and congestion control for internet media streaming applications,” in Proceedings of Multimedia Computing and Networking, 1998.
8. Y. Chang, C. Li, and D.G. Messerschmitt, “Adapting network video to multi-time scale bandwidth fluctuations,” in Proceedings 2000 IEEE International Conference on Multimedia and Expo, New York, NY, USA, July 2–August 2000.
9. S. Cheung, M.H. Ammar, and X. Li, “On the use of destination set grouping to improve fariness in multicast video distribution,” INFOCOM 1996: 553–560, March 24–28, 1996, San Francisco, California.
10. S. Deering and R. Hinden, Internet Protocol, Version 6 (IPv6) Specification, RFC 2460.

11. C. Diot, “On QoS & traffic engineering and SLS-related work by sprint,” in workshop on Internet Design for SLS Delivery, Tulip Inn Tropen, Amsterdam, The Netherlands, 25–26 Jan. 2001.
12. S. Floyd and K. Fall, “Promoting the use of end-to-end congestion control in the internet,” IEEE/ACM Transactions on Networking, 1998, Submitted.
13. Java Media Framework: http://java.sun.com/products/java-media/jmf/index.html.
14. T. Jiang, M.H. Ammar, and E.W. Zegura, “Inter-receiver fairness: A novel performance measure for multicast ABR sessions,” SIGMETRICS 1998, pp. 202–211.
15. T. Jiang, E.W. Zegura, and M. Ammar, “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), June 1999, pp. 103–114.
16. T. Kim and M.H. Ammar, “A comparison of layering and stream replication video multicast schemes,” in Proc. NOSSDAV’01, Port Jefferson, NY, June 25–26, 2001.
17. X. Li, M. Ammar, and S. Paul “Video multicast over the internet,” IEEE Network Magazine, April 1999.
18. S. McCanne and S. Floyd, “The UCB/LBNL network simulator,” Software Online. http://www.isi.edu/nsnam/ns/
19. S. McCanne and V. Jacobson, “Receiver-driven layered multicast,” 1996 ACM Sigcomm Conference, Aug. 1996, pp. 117–130.
20. P. Mundur, A. Sood, and R. Simon, “Network delay Jitter and client buffer requirements in distributed video-on-demand systems,” Department of Computer Science George Mason University Fairfax, VA 22030.
21. K. Nichols, S. Blake, F. Baker, and D. Black, “Definition of the differentiated services field (DS field) in the IPv4 and IPv6 Headers,” RFC 2474.
22. J. Pandhye, J. Kurose, D. Towsley, and R. Koodli, “A model based TCP-friendly rate control protocol,” in Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Basking Ridge, NJ, June 1999.
23. K. Park, G. Kim, and M. Crovella, “On the relationship between file sizes, transport protocols, and self-similar network traffic,” in Proceedings of the International Conference on Network Protocols, Oct. 1996, pp. 171–180.
24. L. Rizzo, “pgmcc: A TCP-friendly single-rate multicast congestion control scheme,” in Proceedings of SIGCOMM’2000, Stockholm, Aug. 2000.
25. H. Schulzrinne and S. Casner, “RTP profile for audio and video conferences with minimal control,” RFC 3551, IETF, July 2003.
26. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A transport protocol for real-time applications,” RFC 3550, IETF, July 2003.
27. D. Sisalem, “Fairness of adaptive multimedia applications,” in ICC ’98. 1998 IEEE International Conference on Communications. Conference Record. Affiliated with SUPERCOMM’98 IEEE, 1998, pp. 891-5, Vol. 2. 3 Vol. xxxvii+1838 pp.
28. D. Sisalem and A. Wolisz, “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), Chapel Hill, NC, USA, June 25–28, 2000.
29. H. Smith, M. Mutka, and D. Rover, “A feedback based rate control algorithm for multicast transmitted video vonferencing,” Journal of High Speed Networks, Accepted.
30. M. Allman, V. Paxson, and W. Stevens, “TCP congestion control,” RFC 2581, April 1999.
31. B.J. Vickers, C.V.N. Albuquerque, and T. Suda, “Adaptive multicast of multi-layered video: Rate-based and creditBased approaches,” in Proc. of IEEE Infocom, March 1998.
32. J. Walpole, R. Koster, S. Cen, C. Cowan, D. Maier, D. McNamee, C. Pu, D. Steere, and L. Yu, “A player for adaptive mpeg video streaming over the internet,” in Proceedings of the 26th Applied Imagery Pattern Recognition Workshop AIPR-97, SPIE, Washington, DC, Oct. 1997.
33. J. Widmer and M. Handley, “Extending equation-based congestion control to multicast applications,” in Proc. ACM SIGCOMM, San Diego, CA, Aug. 2001.

Christos Bouras obtained his Diploma and Ph.D. from the Computer Science and Engineering Department of Patras University (Greece). He is currently an Associate Professor in the above department. Also he is a scientific advisor of Research Unit 6 in Research Academic Computer Technology Institute (CTI), Patras, Greece. His research interests include Analysis of Performance of Networking and Computer Systems, Computer Networks and Protocols, Telematics and New Services, QoS and Pricing for Networks and Services, e-Learning Networked Virtual Environments and WWW Issues.

Apostolos Gkamas obtained his Diploma, Master Degree and Ph.D. from the Computer Engineering and Informatics Department of Patras University (Greece). He is currently an R&D Computer Engineer at the Research Unit 6 of the Computer Technology Institute, Patras, Greece. His research interests include Computer Networks, Telematics, Distributed Systems, Multimedia and Hypermedia.

Anastasios Karaliotas obtained his Diploma and Master Degree from the Computer Engineering and Informatics Department of Patras University (Greece). He works in the Network Technologies Sector of Research Academic Computer Technology Institute (CTI) since November 1999. His research interests are focused on the design, implementation and operation of computer networks.



Kostas Stamos obtained his Diploma and Master Degree from the Computer Engineering and Informatics Department of Patras University (Greece). He has worked for the Networking Technologies Sector of Research Academic Computer Technology Institute (CTI), Patras, Greece from the end of 1999 until December 2000. Since July 2001 he works with Research Unit 6 of CTI.