University of Cyprus
Department of Computer Science
Networks Research Laboratory

Adaptive Methods for the Transmission of Video Streams in Wireless Networks

Deliverable 2.2

Feedback Algorithms for the Increase of the Objective Quality

Project Title	: Adaptive Methods for the Transmission of Video Streams in Wireless Networks
Deliverable Type	: Final WP report (M18)

Deliverable Number	: D2.2
Title of Deliverable	: Feedback Algorithms for the Increase of the Objective Quality Techniques
Work Package Title	: Network Adaptation Techniques
Work Package Number	: WP2
Internal Document Number	:
Contractual Delivery Date	:
Actual Delivery Date	:
Author(s)	: Pavlos Antoniou Marios Lestas Andreas Pitsillides Vasos Vassiliou
Email(s)	: paul.antoniou@cs.ucy.ac.cy lestas@cs.ucy.ac.cy Andreas.Pitsillides@ucy.ac.cy vasosv@cs.ucy.ac.cy

Abstract

Nowadays the need for adaptive delivery of media streams over the Internet is of high importance because of the unpredictable nature of the heterogeneous environments. Compressed video streams exhibit large variations in their data rates something which makes their management in a packet-based best-effort network like IP extremely difficult. The problem is extenuated when we consider mobile users connecting with wireless terminals due to the erroneous and time variant conditions of the wireless links. Thus, applications of real-time video streaming in heterogeneous networks and computing environments like the Internet need to implement highly scalable and adaptive techniques in terms of content encoding and transmission rates. Taking all these into consideration it is apparent that designing adaptive mechanisms for Internet video transmission poses many challenges. Under these circumstances a combination of Content Adaptation Techniques and Network Adaptation Techniques is an imperative need. In this deliverable we propose two novel feedback algorithms for the increase of the objective quality of the transmitted video that involve the aforementioned adaptation techniques. We also propose two new algorithms for the control of the congestion in high speed networks. The performance of these algorithms will be evaluated through simulations in the next deliverable.

Keywords: Network Adaptation Techniques, Content Adaptation Techniques, Scalable video, congestion control.

Table of Contents

List of Figures

List of Tables

1. Introduction

Needless to say that video transmission over the wireless networks is considered to be the prime candidate for being the next killer application of the Internet. The overwhelming majority of today's mobile devices like mobile phones, PDAs and laptops are capable of reproducing video streams either through 3G-enabled networks like UMTS or over the Internet using dedicated protocols like 802.11. In this report we are going to focus primarily on video transmission over the Internet.

One of the most significant problems that video communications face is the unpredictable nature of the heterogeneous networks like the Internet primarily in terms of bandwidth, latency and loss variation. It is beyond any doubt that video transmission applications need to implement highly scalable and adaptive techniques in terms of content encoding and transmission rates in order to cope with erroneous and time variant conditions of such networks.

For these reasons some adaptive schemes have to be introduced from both the network as well as content encoding perspective. Content Adaptive Techniques (CATs) deal with adaptation of content to the desirable transmission rate using primarily scalable video approaches whereas Network Adaptation Techniques (NATs) deal with the end-to-end adaptation of network parameters to the needs of a real time multimedia application using algorithms which take into account the state and/or load of the network and the type of errors. Many problems in this area remain open research issues. Thus in this deliverable initially we investigate, analyze and understand the inefficiencies that are associated with the existing adaptation techniques in order to be able to contribute to this effort. Section 2 discusses some transmission requirements for Internet video streaming while Section 3 describes video compression techniques. Moreover Section 4 deals with Content Adaptation Techniques whereas in Section 5 we are investigating Network Adaptation Techniques based on adaptive control schemes either sender-driven or receiver-driven while providing an in depth comparison of the aforementioned schemes. Lastly in Section 6 we present and analyze our novel approaches for adaptive video transmission and for the control of congestion in high speed networks.

2. Internet Video Transmission Requirements

Delivering real-time video over the Internet is an important component of many Internet multimedia applications. With respect to the real-time nature of video streaming, instable bandwidth, latency (or end to end delay) and packet loss, are all problems that can affect video streaming over the Internet.

It is apparent that the current Internet does not offer any Quality of Service (QoS) guarantees to video transmission over the Internet. The Differentiated Services (DiffServ) and the Integrated Services (IntServ) as well were models proposed by the Internet Engineering Task Force (IETF) to provide QoS guarantees. IntServ is a framework developed to provide individualized QoS guarantees to individual application sessions. The goal of DiffServ is to provide the ability to handle different classes of traffic in different ways within the Internet. However neither IntServ nor DiffServ have taken off and found widespread adoption in today's Internet.

The important goal of adaptive video streaming is to perform the streaming in a manner that a sequence of constraints should be met. Basically the most important constraint is the successful delivery of every single frame prior its playback time due to the fact that any frame which is either lost in transmission or partially delivered to the receiver or even delivered beyond its playback time is considered lost by the

decoder. The three main factors that influence to a large extend the transmission of a video stream over the Internet as well as the quality of the received sequence, are thoroughly described below.
Firstly, the per-node bandwidth constraint is critical for bandwidth-demanding applications such as video transmission. In addition there are various bandwidth requirements especially if a single sender multicasts video streams to several receivers. Although the estimation of the available bandwidth and the adjustment of the transmitted video bit rate to the available bandwidth is a time consuming process, the estimation of the available bandwidth is an imperative need and it should be done in a real time manner either by the receiver or by the sender because traditional routers typically do not actively participate in this procedure. Available bandwidth has to be carefully estimated due to the fact that an excessive traffic load caused by wrong estimations can cause congestion collapse which can further degrade the quality of the transmitted video stream. Needless to say that in order to be able to achieve an acceptable presentation quality for any user (who is connected either by modem or by broadband connection), transmission of real time video typically should have a minimum bandwidth requirement of 28Kbps.
Secondly, end to end delay as well as the variation in end-to-end delay (jitter) can affect the quality of streaming video. Even though real time video requires bounded end to end delay that is every frame must arrive at the decoder in time to be decoded and displayed, the current Internet does not offer such a delay guarantee. If a video packet arrives beyond the time constraint it will be considered lost and the user-perceived quality will degrade. Buffering can partially solve those problems but again if video packets do not arrive on time the playback process will pause causing annoying interruptions of the video sequence.
Thirdly, wired network may be affected by entire packet loss; wireless channels are afflicted by both bit errors and burst errors. In this way, the bandwidth constraint should care about not only the amount of bandwidth but also the consistency and quality of this bandwidth. It is beyond any doubt that packet loss ratio is required to be kept below a threshold but on the other hand Internet does not provide any loss guarantee.
Sometimes, it is not all about the quality of bandwidth; content creation, serving, usability and availability are also challenges that need to be overcome. Quality of service (QoS) mechanisms have been the focus because today's Internet lacks support for QoS assurance, which makes the transmission of video more challenging. Furthermore, the heterogeneity of the Internet's transmission resources and end-systems makes it to support different traffic characteristics among multiple receivers of the same video stream.

3. Video Compression

To achieve efficiency and to meet bandwidth and delay constraints as well, raw video must be compressed before transmission. In other words video compression is done at sender's side in a scalable manner using a predefined standardized coding/decoding system. Then at the client side the encoded video layers would be decoded and played in a proper way. In this survey we only consider MPEG standards.
MPEG stands for the Moving Picture Experts Group. MPEG is an ISO/IEC working group, established in 1988 to develop standards for digital audio and video formats. There are five MPEG standards being used or in development. Each compression standard was designed with a specific application and bit rate in mind, although MPEG compression scales well with increased bit rates. They include:

1) MPEG-1. It was designed for up to 1.5 Mbps. It is a standard for the compression of moving pictures and audio. This was based on CD-ROM video applications, and is a popular standard for video on the Internet, transmitted as .mpg files. In addition, level 3 of MPEG-1 is the most popular standard for digital compression of audio, known as MP3. MPEG-1 is the standard of compression for VideoCD, the most popular video distribution format throughout much of Europe.

2) MPEG-2. It was designed for between 1.5 and 15 Mbps. It is a standard on which Digital Television set top boxes and DVD compression is based. It is based on MPEG-1, but designed for the compression and transmission of digital broadcast television. The most significant enhancement from MPEG-1 is its ability to efficiently compress interlaced video. MPEG-2 scales well to HDTV resolution and bit rates, obviating the need for an MPEG-3.

3) MPEG-4. It is a standard for multimedia and Web compression. MPEG-4 is based on object-based compression, similar in nature to the Virtual Reality Modelling Language. Individual objects within a scene are tracked separately and compressed together to create an MPEG4 file. This results in very efficient compression that is very scalable; from low bit rates to very high. It also allows developers to control objects independently in a scene, and therefore introduce interactivity.

4) MPEG-7. This standard, currently under development, is also called the Multimedia Content Description Interface. When released, the group hopes the standard will provide a framework for multimedia content that will include information on content manipulation, filtering and personalization, as well as the integrity and security of the content. Contrary to the previous MPEG standards, which described actual content, MPEG-7 will represent information about the content.

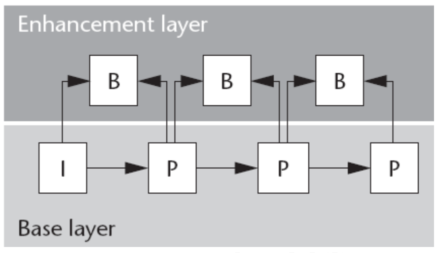
5) MPEG-21. Work on this standard, also called the Multimedia Framework, has just begun. MPEG-21 will attempt to describe the elements needed to build an infrastructure for the delivery and consumption of multimedia content, and how they will relate to each other.

4. Content Adaptation Techniques

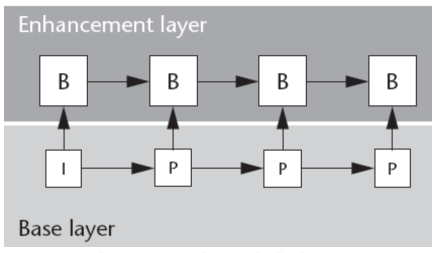
Content Adaptation Techniques (CATs) are methods commonly used for the adaptation of content to the desirable rate. Recent studies on CATs reveal that the transmission of video streams in multiple layers is feasible without the need for re-encoding or regeneration of the content.

Common CATs are primarily based on rate-adaptive video encoding [6] that has been studied extensively for various standards and applications, such as video conferencing with H.261 and H.263, storage media with MPEG-1 and MPEG-2, real-time transmission with MPEG-1 and MPEG-2, and the recent object-based coding with MPEG-4. The main objective of a rate-adaptive encoding algorithm is to maximize the perceptual quality under a given encoding rate. Such adaptive encoding can be achieved by the alteration of the encoder's quantization parameter (QP) and/or the alteration of the video frame rate which are described in more detail below. Traditional video encoders (e.g., H.261, MPEG-1/2) typically rely on altering the QP of the encoder to achieve rate adaptation. These encoding schemes must perform coding with constant frame rates. This is because even a slight reduction in frame rate can substantially degrade the perceptual quality at the receiver, especially during a dynamic scene change. Since altering the QP is not enough to achieve very low bit rate, these encoding schemes may not be suitable for very low bit-rate video applications. On the contrary, MPEG-4 and H.263 coding schemes are suitable for very low bit-rate video applications since they allow the alteration of the frame rate.

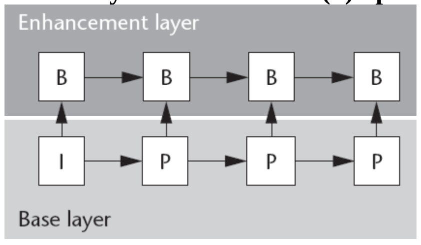
In fact, the alteration of the frame rate is achieved by frame-skipping. In addition, MPEG-4 is the first international standard addressing the coding of video objects (VOs). With the flexibility and efficiency provided by coding video objects, MPEG-4 is capable of addressing interactive content-based video services as well as conventional stored and live video.
From a video source point of view, video compression can be classified into two approaches: scalable and non-scalable video coding [1]. CATs are primarily based on scalable video approaches. Scalable video encoder compresses a raw video into multiple substreams or layers using QP or frame rate alteration techniques. One of them is the base substream/layer and others are enhancement substreams/layers. The base substream can be independently decoded and provide coarse visual quality; enhancement substream is decoded with base substream together to provide enhanced video quality. Below we provide some scalable coding schemes that are integrated in MPEG-4 standard.
There are several scalable coding schemes in MPEG-4 as mentioned in [2]: SNR scalability, spatial scalability, temporal scalability, fine granularity scalability and object-based spatial scalability. The three first aforementioned schemes are illustrated in Fig. 1.



(a) Temporal Scalability.



(b) Spatial Scalability.



(c) SNR Scalability.

Fig. 1. Scalable video encoding.

1) **Signal to Noise Ratio (SNR) Scalability** is a technique to code a video sequence into two layers at the same frame rate and the same spatial resolution, but different quantization accuracy by altering the quantization parameter.
2) **Spatial Scalability** is a technique to code a video sequence into two layers at the same frame rate, but different spatial resolutions. The base layer is coded at a lower spatial resolution. The reconstructed base-layer picture is up-sampled to form the prediction for the high-resolution picture in the enhancement layer.
3) **Temporal Scalability** is a technique to code a video sequence into two layers at the same spatial resolution, but different frame rates (frame rate alteration). The base layer is coded at a lower frame rate. The enhancement layer provides the missing frames to form a video with a higher frame rate. Coding efficiency of temporal scalability is high and very close to non-scalable coding. Fig. 1(a) shows a structure of temporal scalability. Only P-type prediction is used in the base layer. The enhancement-layer prediction can be either P-type or B-type from the base layer or -type from the enhancement layer.

4) **Object-based Spatial Scalability** extends the 'conventional' types of scalability towards arbitrary shape objects, so that it can be used in conjunction with other object-based capabilities. Thus, a very flexible content-based scaling of video information can be achieved. This makes it possible to enhance SNR, spatial resolution, shape accuracy, etc, only for objects of interest or for a particular region, which can be done dynamically at play-time.
5) **Fine Granularity Scalability (FGS)** was developed in response to the growing need on a video coding standard for streaming video over the Internet as is analysed in [3] and [4]. FGS and its combination with temporal scalability address a variety of challenging problems in delivering video over the Internet. FGS allows the content creator to code a video sequence once and to be delivered through channels with a wide range of bitrates. It provides the best user experience under varying channel conditions. It overcomes the "digital cut-off" problem associated with digital video. In other words, it makes compressed digital video behave similarly to analogue video in terms of robustness while maintaining all the advantages of digital video.
Layered information needs to be adapted for a number of transmission rates. The techniques for reducing the transmitted information are primarily based on dropping or adding layers. To achieve optimal adaptation to any transmission rate we believe that it is necessary to have a large number of layers. However, we need to be aware of possible drawbacks of having a large layer number, such as difficulty in separating/generating the layers at the source, and find the optimal number for them.
According to different network conditions, the sender or the receiver selects to send or receive respectively different quality levels of video. Scalable video encoding provides an alternative solution to meet with heterogeneous demands of clients.

5. Network Adaptation Techniques

Designing network adaptation techniques for Internet video transmission poses many challenges. In this section, we take a holistic approach to these challenges and present solutions primarily from transport perspective focusing on congestion control. In particular we classify approaches and summarize representative research work.
The basic requirements of network adaptation techniques are (1) to provide as accurate information as possible for the network load, (2) to distinguish between errors originating from congestion at the core routers and errors stemming from the wireless medium, and (3) to properly adapt the transmission rate. The qualitative differentiation of errors is required so that we reduce the transmission rate only when it is justifiable (e.g. when we have heavy core congestion). In order for the estimations of network load and the type and quality of errors to be precise we need proper feedback from the receiver end of the stream.
Congestion control combines techniques like rate control and rate-adaptive encoding. Bursty losses, excessive end to end delay and delay variation have devastating effects on video perceived quality primarily due to network congestion. Thus, congestion control is required to reduce packet loss, delay and delay variation as well. One congestion control mechanism is rate control [5]. Rate control attempts to minimize network congestion and the amount of packet loss by matching the rate of the video stream to the available network bandwidth. Therefore, without rate control, the traffic exceeding the available bandwidth would be discarded in the network. To force the source to send the video stream at the rate dictated by the available bandwidth, rate control algorithms should be used in conjunction with rate-adaptive video encoding [6] techniques. The main objective of a rate-adaptive encoding technique is to maximize the perceptual quality under a given encoding rate. Note that rate control is

from the transport perspective and belongs to NATs, while rate-adaptive video encoding is from the compression perspective and belongs to CATs.
Except for the aforementioned classification, adaptation to network parameters may be sender driven or receiver driven. In a sender-driven algorithm, the source adapts its transmission rate in response to congestion feedback from the network and/or the receivers. In a receiver-driven algorithm, the source transmits several sessions of data, and the receivers adapt to congestion by changing the selection of sessions to which they listen.
The two algorithms that attempt to alleviate the problem of congestion within the Internet namely rate control and rate-adaptive video encoding are illustrated in Fig. 2.

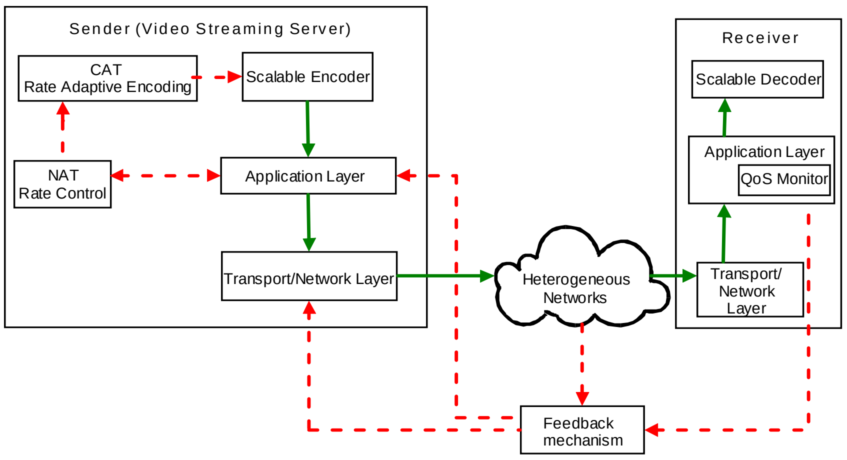


Fig. 2. Real-time video streaming architecture.

As can be seen from Fig. 2 scalable video encoder compresses the video stream (which is depicted with solid arrows) according to rate adaptive encoding algorithm. The output bitrate of the scalable encoder based on the rate control algorithm should be equal or less than the estimated available network bandwidth. After this stage, the compressed video bit stream passes through the application/transport/network layers before entering the Internet. The most prevalent protocol used for video streaming over the Internet is the Real-time Transport Protocol (RTP) which runs on top of the UDP/IP. Video packets may be either dropped by core network routers due to congestion or afflicted by both bit errors and burst errors in the wireless last hop link of the video stream path. Moreover packets can be considered lost by the receiver if the arrive beyond the playback time due to excess delay. Packets that are successfully delivered to the receiver first pass through the network/transport/application layers before being decoded at the video decoder.
The quality of service monitor located at the receiver side may reside inside application layer or even inside transport layer. For example QoS monitoring in video streaming applications regarding today's Internet can be done with RTCP protocol which is part of the RTP protocol. RTCP stands for Real-time Transport Control Protocol and maintains network congestion status based on the behaviour of the arriving packets, e.g. packet loss, end to end delay and jitter. Such information is used by the feedback mechanism (which is either incorporated in the RTP/RTCP protocol or it is part of a dedicated feedback protocol) which sends information back to the sender (control information is depicted with dashed arrows). Based on such feedback information, the rate control algorithm estimates the available network bandwidth and conveys the estimated available network bandwidth to the rate adaptive encoder. Afterwards the rate adaptive encoder regulates the output rate of the video stream

according to the estimated available network bandwidth using techniques that will be mentioned in Section V. In the rest of this section we are investigating existing approaches concerning rate control.

Rate Control

It is beyond any doubt that the most prevalent transport protocols used in the today's video streaming over the Internet are either TCP (Transmission Control Protocol) or UDP (User Datagram Protocol). However the overwhelming majority of streaming applications predominantly use TCP instead of UDP according to recent research on this topic [8].

Actually TCP is one of the most popular transport protocols for video streaming, even though the rate variability of TCP makes it difficult to provide good video quality. On the other hand UDP is the alternative to TCP. UDP forsakes TCP's error correction and allows packets to drop out if they're late or damaged. Despite the prospect of dropouts, this approach is arguably better for continuous media delivery. If broadcasting live events, everyone will get the same information simultaneously. One disadvantage to the UDP approach is that many network firewalls block UDP information. Furthermore UDP is not able to provide congestion control and overcome the lack of service guarantees in the Internet. So the existence of a mechanism to prevent congestion in a higher layer than UDP is necessary.

Nevertheless most of the protocols and mechanisms that are presented below which are used for real-time streaming, do not address the problem of wireless losses. They were designed mostly for wired networks and lack a rate control mechanism that handles wireless losses efficiently.

To sum up there are two types of congestion control used in the Internet. These are window-based and rate-based. The first type of control slowly increase the congestion window until congestion is detected (losses packets). After that, the protocol reduces the congestion window greatly. The latter type of control sets the sending rate based on the estimated available bandwidth in the network.

Moreover three alternative categories of the rate control schemes exist: source-based, receiver-based and hybrid which are described in Sections IV-A, IV-B and IV-C respectively. We end this Section by summarizing, evaluating and comparing all the aforementioned techniques in section IV-D.

A. Source-Based Rate Control

In this approach the source (sender) is responsible for adapting the video transmission rate. Source-based rate control mechanisms provide feedback information about the network state so the sender could adjust the rate of the video stream. The source-based rate control can be implied to both unicast and multicast approaches.

As far as the unicast transmission is concerned two approaches exist that are probe-based and equation-based. The first approach is based on probing experiments. The sending rate is regulated by additive increase and multiplicative decrease (AIMD) or by multiplicative increase and multiplicative decrease (MIMD) strategies. The increasing or decreasing of the rate depends on whether the packet loss ratio is below or above a certain threshold. On the equation-based approach, the sending rate is regulated by a throughput equation that takes into consideration the state of the transmission path within the network. Usually the sending rate is determined by a predefined formula like the one proposed in [13]:

$$\lambda = \frac{1{,}22 \times MTU}{RTT \times \sqrt{p}} \quad (1),$$

where λ is the throughput, p is the packet loss ratio, RTT is the round trip time and MTU is the maximum transfer unit of the path.
Regarding multicast transmission, the sender uses single-channel multicast where only the probe-based rate control can be employed. Single-channel multicast is efficient since all the receivers share one channel. If multicast video were to be delivered through individual unicast streams, the bandwidth efficiency is low but the services could be differentiated since each receiver can negotiate the parameters of the services with the source.
In the rest of this section existing source-based protocols are being investigated.

Transmission Control Protocol (TCP)
TCP congestion control constitutes the most prevalent probe-based rate control approach. The idea of TCP congestion control is for each source to determine how much capacity is available in the network, so it knows how many packets it can safely have in transit. The TCP sender paces data transmissions based on a sliding window that depends on both the available buffer space at the receiver and the available bandwidth in the network, represented by receiver window and congestion window, respectively. A strategy called AIMD (Additive Increase/Multiplicative Decrease) regulates the number of packets that are sent at one time. This strategy makes TCP inefficient for video streaming applications due to the aperiodic tooth-saw transmission rate. Needless to say that TCP assumes every packet loss as congestion-induced and reacts by cutting the sending rate. To accommodate the variability, video streaming applications require receiver-side buffering. TCP will effectively stop traffic until either the original packets or backup packets arrive. Hence it's unsuitable for video and audio transmission because TCP imposes its own flow control and windowing schemes on the data stream, effectively destroying temporal relations between video frames and audio packets. Also reliable message delivery is unnecessary for video and audio applications due to the fact that losses are tolerable and TCP retransmission causes further jitter and skews. In wireless networks were packet loss can also be due to signal attenuation, fading, scattering, interference or mobility, TCP is known to have reduced efficiency.

Multimedia Transport Protocol (MTP) [10]
MTP is a probe-based TCP modification that gracefully disables retransmissions, while preserving the transmission timings and congestion responsiveness characteristics of TCP. MTP performs slow start, congestion avoidance, fast retransmission and fast recovery as does TCP, yet offers a UDP-like transparent API that enables streaming media applications to make informed media scaling decisions and provides UDP packet delivery semantics. In addition MTP does not offer guaranteed or in-order packet delivery. Basically MTP keeps the same loss detection and recovery mechanisms of TCP but reduces the high delay and jitter characteristics.
When encountering duplicate acknowledgements, the MTP sender performs congestion avoidance, fast retransmission and fast recovery as TCP does, and yields identical congestion window movement and packet transmission timings. On reception of a triple duplicate acknowledgement instead of a retransmission, the MTP sender inflates its transmission window and sends a new packet. When it receives an acknowledgment for this new packet it deflates back the inflated transmission window. This is done in order not to count the retransmission-replacement packet as a new transmission when making the next new packet transmission decision while preserving the same transmission behaviour at the network layer as that of TCP.

On a retransmission timeout, the MTP sender acts in a similar manner as a TCP sender performing slow start but it restarts by transmitting a new packet unlike TCP which restarts the transmission from one below the highest consecutively acknowledged packet sequence number.
Simulation results as done in [10] showed that MTP video streams inherit the good characteristics of both TCP and UDP streams. Also results showed that MTP streams are also TCP-Friendly while they can effectively perform media scaling and adapt to the available TCP-Friendly bitrate. Furthermore MTP dramatically reduces media frame reception jitter from TCP's reliable in-order packet delivery mechanism, and illustrates the potential of MTP as a streaming transport protocol for interactive as well as non-interactive applications.
On the other hand MTP provides a fluctuating transmission rate due to the TCP-like congestion avoidance mechanism something that is inefficient for streaming applications. MTP is also incapable of transmitting video streams over wireless networks because it assumes every packet loss as congestion-induced as TCP does, and reacts by cutting the sending rate. For a better evaluation of this protocol, researchers in [10] should provide results concerning transmission of video streams that correspond to real video traces.

Video Transport Protocol (VTP) [11]
VTP is a transport protocol based on probing with a new end-to-end rate control mechanism which was designed specifically for real-time streaming in wireless networks. Its prime goal was to behave well in the wired Internet, to be robust to random errors, to provide TCP friendliness and to be deployed with wireless links as well.
VTP combines rate estimation with loss discrimination techniques and consists of two important components namely the Achieved Rate (AR) estimation and the Loss Discrimination Algorithm (LDA). VTP measures the AR and adapts its sending rate accordingly when congestion is detected by the LDA. AR is the rate that the sender has successfully pushed through the bottleneck. More specifically is the rate that the receiver can measure, plus the fraction corresponding to packet loss at the exit of the bottleneck due to random errors. LDA allows VTP to distinguish congestion losses from error losses.
VTP rate control is based on the analysis of TCP instantaneous sending rate. Similar to the Additive Increase strategy in TCP, VTP linearly probes the available bandwidth until congestion is detected. On congestion detection, the VTP sender reduces its sending rate and the video encoding rate to a level the network can accommodate by mimicking the TCP behaviour without involving Multiplicative Decrease. This enables VTP to deliver a larger portion of the overall video stream and to achieve inter-protocol fairness with competing TCP traffic.
As shown below in Fig. 3 VTP avoids the drastic rate reductions (like cutting the rate by half as TCP does) but maintains the same average rate. In contrast to the highly fluctuating TCP rate, VTP reduces on average its rate by less but keep it longer providing a TCP friendly behaviour. The two shaded areas A1 and A2 represent the amount of extra data that the two protocols would be able to transmit if the loss did not happen. These areas should be equal in order to make VTP friendly to TCP.

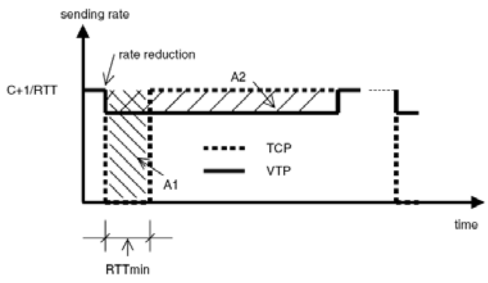


Fig. 3. Comparison of the instantaneous sending rate between TCP and VTP.

When VTP has given up the same amount of data transmission as TCP would in the same situation, it enters congestion avoidance phase. During this phase VTP matches the TCP behaviour.
An important aspect of VTP is that it is completely end-to-end that performs well in the wireless environment without requiring support from lower layer feedback and AQM mechanisms. VTP aims to be adaptive and flexible by making minimal assumptions about the network and using network feedback as a rough indicator, not as rigorous set of input parameters. These principles encompass the motivating factors of the VTP design.
Simulations conducted in [11] reveal that VTP manages to utilize the bandwidth efficiently, maintains a smooth rate and reacts to bandwidth changes under different error rates very quickly. Moreover VTP seems to be opportunistically friendly to TCP but fair enough with flows of the same protocol. Although the results presented in [11] reveal that VTP is as robust to random errors as TFRC [9] is, they do not testify its superiority and its robustness to wireless losses compared to the TFRC when used in wireless environments.

TCP Friendly Rate Control (TFRC) [9]
TCP Friendly Rate Control is one of the most popular equation-based end-to-end streaming algorithms and often used as the reference and benchmark. TFRC is meant to serve as a congestion control framework for any application that do not require the full reliability of TCP and would benefit form low variation in sending rate.
In TFRC the data sender sends a stream of data packets to the data receiver at a controlled rate given by the following formula:

$$X = \frac{s}{R * \sqrt{\frac{2*b*p}{3}} + (t\_RTO * (3 * \sqrt{\frac{3*b*p}{8}}) * p * (1 + 32 * p^2)))} \quad (2),$$

where X is the transmit rate in bytes/second, s is the packet size in bytes, R is the round trip time in seconds, p is the loss event rate, between 0 and 1.0, of the number of loss events as a fraction of the number of packets transmitted, t_RTO is the TCP retransmission timeout value in seconds and b is the number of packets acknowledged by a single TCP acknowledgement.
When a feedback packet is received from the data receiver, the data sender changes its sending rate, based on the information contained in the feedback report. If the sender does not receive a feedback report for two round trip times, it cuts its sending rate in half. This is achieved by means of a timer called the nofeedback timer. When a feedback packet is received the following actions should be performed: Calculate a new round trip sample, Update the round trip time estimate, Update the timeout interval, Update the sending rate and Reset the nofeedback timer.
The receiver periodically sends feedback messages to the sender. Feedback packets should normally be sent at least once per RTT. A feedback packet should also be sent

whenever a new loss event is detected without waiting for the end of an RTT, and whenever an out-of-order data packet is received that removes a loss event from the history. If the sender is transmitting many packets per RTT there may be some advantages to sending periodic feedback messages more than once per RTT as this allows faster response to changing RTT measurements, and more resilience to feedback packet loss. However, there is little gain from sending a large number of feedback messages per RTT.
TFRC is designed to respond to a loss event (which may include several packet drops) instead of a packet loss. However, with the increasing popularity of wireless Internet terminals and the demand for delivering video streaming to mobile users, it is necessary for streaming protocols to work efficiently also on wireless links, withstanding the high random wireless errors. Legacy TCP does not work well in this case; it tends to cut its window by half, leading to a severely degraded performance. Since TFRC attempts to faithfully match the throughput of TCP, it suffers the same low efficiency in the presence of moderate to high random transient errors.

Stream Control Transport Protocol (SCTP) [16]
Driven by industry interest and general agreement on the unsuitability of either TCP or UDP, the IETF Signalling Transport (SIGTRAN) group was formed in 1999 to standardize a suitable transport protocol for signalling traffic over IP. Stream Control Transmission Protocol (SCTP) is the result of this work, recently published as RFC 2960 by The Internet Society. SCTP is the fundamental member of a family of protocols being designed by the SIGTRAN group to allow SS7 messages to be transported over an IP infrastructure. Furthermore the enhanced capabilities of SCTP when compared with traditional Internet transport protocols such as TCP and UDP may make it attractive as a transport for a wide range of traditional Internet services such as those based on HTTP and SIP.
The basis of SCTP congestion control is an amalgamation of current best practice for TCP implementations with extensions to deal with the multi-homing aspect of SCTP and modifications due to the message rather than stream-based nature of the protocol. The standard specifies an adaptive sliding window control with adapted versions of the well known TCP slow-start, congestion avoidance, fast retransmit and fast recovery mechanisms. In addition recent work on TCP, such as congestion window validation, is also incorporated. One currently optional mechanism for TCP, the use of selective acknowledgements (SACKs) to report out of sequence data arriving at the receiver, has been incorporated into SCTP as the baseline for congestion control implementation. This is due to the demonstrated superiority of this mechanism to earlier TCP congestion control options. Although the possibility to support IP Explicit congestion Notification (ECN) has been incorporated into SCTP, this mechanism is optional and (as with TCP) a packet loss is the normal method of congestion indication.
The SCTP is designed to accommodate real-time streaming and supports multi-streaming, where a sender can multiplex several outgoing streams into one connection. This can potentially be very advantageous for compressed video formats since packets belonging to different parts of the video stream can be treated differently with respect to retransmission and order of delivery. The congestion control mechanism in SCTP is identical to TCP, where the congestion window is reduced by half in the event of packet loss. Like TCP, SCTP employs slow start to initially seek out available bandwidth and congestion avoidance to adapt to changing path conditions. This results in perfect fairness with TCP, but leads to high variability

in throughput at the receiver. An investigation of the applicability of SCTP to MPEG-4 streaming is the subject of [19].
Taking all these into consideration we may conclude that this protocol is incapable of transmitting video streams over wireless links since it matches TCP congestion control and congestion avoidance mechanisms. Even though SCTP offers acknowledged error-free non-duplicated transfer of datagrams, these features are not required by real-time streaming applications.

Datagram Congestion Control Protocol (DCCP) [17]

DCCP, the Datagram Congestion Control Protocol, is a new transport protocol in the TCP/UDP family that provides a congestion-controlled flow of unreliable datagrams. Delay-sensitive applications, such as streaming media and telephony, prefer timeliness to reliability. These applications have historically used UDP and implemented their own congestion control mechanisms or no congestion control at all. DCCP makes it easy to deploy these applications without risking congestion collapse. It aims to add to a UDP-like foundation the minimum mechanisms necessary to support congestion control, such as possibly-reliable transmission of acknowledgement information. This minimal design should make DCCP suitable as a building block for more advanced application semantics, such as selective reliability. Protocol’s design principles particularly shed light on the ways TCP's reliable byte-stream semantics influence its implementation of congestion control.
DCCP provides the following features, among others: (a) an unreliable flow of datagrams, with acknowledgements, (b) a reliable handshake for connection setup and teardown, (c) reliable negotiation of features, (d) a choice of TCP-friendly congestion control mechanisms, including, initially, TCP-like congestion control (CCID 2) and TCP-Friendly Rate Control [9] (CCID 3). CCID 2 uses a version of TCP’s congestion control mechanisms, and is appropriate for flows that want to quickly take advantage of available bandwidth, and can cope with quickly changing send rates; CCID 3 is appropriate for flows that require a steadier send rate, (e) options that tell the sender, with high reliability, which packets reached the receiver, and whether those packets were ECN marked, corrupted, or dropped in the receive buffer, (f) congestion control incorporating Explicit Congestion Notification (ECN) and the ECN Nonce, (g) mechanisms allowing a server to avoid holding any state for unacknowledged connection attempts or already-finished connections and (h) path MTU discovery.
DCCP is intended for applications which require the flow-based semantics of TCP, but have a preference for delivery of timely data over in-order delivery or reliability, or which would like different congestion control dynamics than TCP. To date most such applications have used either TCP, whose reliability and in-order semantics can introduce arbitrary delay, or used UDP and implemented their own congestion control mechanisms (or no congestion control at all). DCCP will provide a standard way to implement congestion control and congestion control negotiation for such applications, and enable the use of ECN, along with conformant end-to-end congestion control, for applications that would otherwise be using UDP. Similarly, DCCP is intended for applications that do not require features of SCTP [16] such as sequenced delivery within multiple streams.
DCCP like all the aforementioned protocols except for VTP can not be efficiently used in video transmission over wireless links due to the fact that it performs poorly over links with bit errors. In particular it takes the loss as an indication of congestion and it does not include mechanism that handles wireless losses efficiently.

Scalable Streaming Video Protocol (SSVP) [18]

SSVP is a new probe-based transport protocol which relies on a basic yet efficient end-to-end congestion control mechanism. SSVP, in a complementary role, operates on top of UDP and is specifically designed to support unicast video streaming applications. The main objective of SSVP is to provide efficient and smooth rate control while maintaining fairness and friendliness with corporate flows. Moreover it incorporates end-to-end congestion control and does not rely on QoS functionality in routers, such as Random Early Drop (RED), ECN or other Active Queue Management (AQM) mechanisms.

SSVP is further augmented by a layered adaptation mechanism, where additional layers are allocated based on explicit criteria in order to prevent wasteful layer transitions that impair perceived video quality. Quantifying the interactions of SSVP protocol with the specific adaptation scheme, it was identified that SSVP maintains a regular transmission rate, while layered encoding adapts video quality along with long-term variations in the available bandwidth.

SSVP adjusts the sending rate in a TCP-friendly manner, exploiting the feedback of reception statistics (control packets). Both binomial and AIMD congestion control are implied to achieve TCP-friendliness. Apart from link capacity, the selection of increase rate and decrease ratio composes another influencing parameter. Along these lines, in order to attain TCP-friendliness, SSVP incorporates AIMD congestion control. Let α, β the specific values of additive increase and multiplicative decrease rate, respectively. The choice of α and β has a direct impact on protocol responsiveness to conditions of increasing contention or bandwidth availability.

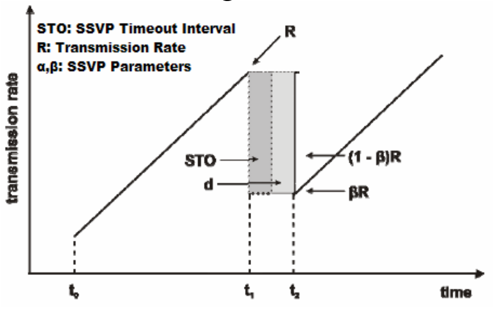


Fig. 4. SSVP transmission rate evolution.

Transmission rate is controlled by properly adjusting the inter-packet-gap. In the absence of congestion, the transmission rate is periodically increased, until it matches the rate intimated by the optimal video quality. Upon detecting congestion, the video coder is immediately notified to progressively reduce the coding rate. Based on analysis shown in [18] and with respect to the user perception of video quality, the final selection of the aforementioned parameters was α=0.2 and β=0.875. This selection results in oscillations of a smaller magnitude than standard TCP (having α=1 and β=0.5), while per-RTT rate adjustments enforce a relatively responsive behaviour.

SSVP also implements a layered adaptation mechanism that is based on information decomposition in one base layer plus one or more enhancement layers which can be combined to render the stream high quality. Layered adaptation is performed by adding or dropping layers depending on the network dynamics. In order to sustain smooth video transmission under awkward conditions, a quality adaptation mechanism that resides on server side coarsely adjusts video quality without need to implement transcoding. SSVP focus on defining a priori whether a new layer should be added under the properties of AIMD congestion control. More precisely a new layer is being allocated as soon as the available bandwidth R exceeds the total consumption rate of all currently active layers (n_a) plus the new one, so $R>(1+n_a)C$ where C is the constant consumption rate of each layer.

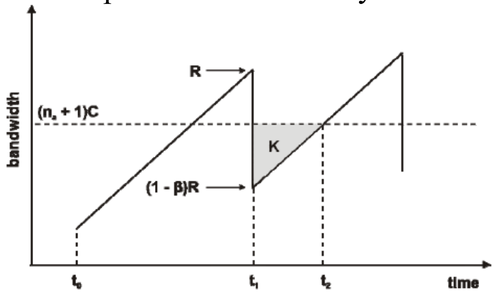


Fig. 5. Layered adaptation under AIMD.

Furthermore a second rule associated with the amount of buffering required at the receiver is applied in order to enable the adaptation mechanism to trade short-term improvements for long-term quality smoothing, preventing buffer overflows and eventually rapid fluctuations in quality which frustrate the end-users. In addition, a more efficient utilization of the available network resources is achieved by properly allocating the bandwidth among the active layers.

We would like to point out that this proposal composes a protocol which is not directly comparable with Datagram Congestion Control Protocol (DCCP) [17] mentioned in the previous section. Although both DCCP and SSVP emerge from a common incentive, SSVP primarily addresses real-time video transmission. For example, SSVP segments include header information enabling the manipulation of a transmitted video stream (e.g. frame prioritization).

Since SSVP uses the same congestion avoidance and control scheme as TCP (i.e. AIMD), it is insufficient for transporting video streams over wireless environments but it provides a simple and yet efficient and-to-end congestion control management scheme on top of the lightweight UDP. Thus with a careful selection of protocol parameters will be able to deliver smooth video over wired networks in a wide range of network dynamics.

Source-Adaptive Multilayered Multicast Algorithms (SAMM) [12]

In the SAMM algorithms that were proposed for multicast video transmission, the source adjusts its encoding parameters, including the number of video and their respective rates in response to a continuous flow of congestion feedback from the receivers. SAMM algorithms may be network-based or end-to-end based. At the network-based congestion is monitored and indicated by network's intermediate nodes. On the other hand, at the end-to-end algorithm the responsibility for congestion control resides exclusively at the source and receivers.

In order to support the above algorithms, network architecture must be defined. Four basic components are necessary for the implementation of a video SAMM algorithm. These are: adaptive layered video sources, layered video receivers, multicast-capable routers, and nodes with feedback merging capability. The adaptive layered video source referred to the ability of the source to generate layered video data and the layered video receiver referred to the ability of the receiver to receive layered video data. Multicast-capable routers must be capable of performing multicast forwarding and routing, priority drop preference, flow isolation and congestion control. Router-based priority dropping and flow isolation is to ensure a lower drop priority to the base layer compared to subsequent enhancement layers. To prevent implosion,

feedback aggregation is done by feedback mergers distributed in the network and organized in an overlay network. A receiver sends a feedback to the nearest feedback merger after reception of a given number of video packets; feedback mergers then convey to the source a list of rates requested by receivers with the associated numbers of receivers per rate.
In case of network-based SAMM algorithm, as mentioned above, the source periodically generates and multicasts to receivers a forward feedback packet. When the receiver gets the forward feedback packet copies the packet's contents into a backward feedback packet and returns it to the source. As forward feedback packets travel from the source to the receivers, routers mark them in order to explicitly indicate the amount of bandwidth available in the network for the transmission of a SAMM video flow. Routers also have to track the number of video multicast flows attempting to share the available bandwidth and calculate the fair share of the available bandwidth for each video multicast flow competing for the outgoing link
Network-based SAMM algorithm has some implementation difficulties such as the routers involvement and the overhead created by the mark feedback packets. To face up these difficulties of network-based algorithm, an end-to-end SAMM algorithm is proposed. Congestion control is performed exclusively at the source, the receivers and the feedback mergers. The receiver estimates the available bandwidth on the path from the source by monitoring its received video rate and periodically returns feedback packets toward the source. When congestion is happened, the available bandwidth decreases and the arrival rate of video packets at downstream receivers changes accordingly. Due to this fact, an estimate of the bandwidth available on the path from the source can be obtained by monitoring how fast video packets arrive at the receiver. The receiver assumes the available bandwidth is equal to the received video rate but the actual available bandwidth may be higher. In order to exploit the available bandwidth, the receiver may occasionally report a rate that is higher, by an increment, than the observed arrival rate of video packets. The receiver reports a higher rate whenever there is a change in the observed arrival rate and no packet losses have been recorded in a given interval of time. This allows the source to capture newly available bandwidth in an incremental, and therefore, stable manner.

RTP/RTCP [14]
All the aforementioned protocols and algorithms refer to the transport layer. On the contrary Real Time Protocol (RTP) is an Internet-standard protocol that lies on the application layer. It deals with end-to-end transport of real-time data, including audio and video. It can be used for media-on-demand as well as interactive services such as Internet telephony in unicast communication as well as in single-stream video multicast approach.
RTP consists of a data and a control part and is usually implemented within the application. RTP can be used over unicast as well as multicast. It typically runs on top of UDP to make use of its multiplexing and checksum functions. RTP doesn't guarantee timely delivery of packets and it doesn’t keep the packets in sequence and gives the responsibility for recovering lost segments and resequencing of the packets to the application layer. Therefore RTP does not offer any form if reliability or flow/congestion control. It provides timestamps, sequence numbers as hooks for adding reliability and flow/congestion control, but their implementation is totally left to the application.
The Real Time Control Protocol (RTCP) is a companion protocol to RTP for gathering statistics on a media connection and information such as bytes sent, packets

sent, lost packets, jitter and round trip delay. Sender, receivers, and third-party monitors can use this information to judge the quality of their connections and make adjustments as required such as changing from low compression codec to a high compression codec. For example, the sender may modify its transmission rate based on the feedback (sender-driven-approach); receivers can determine whether problems are local, regional or global (receiver-driven approach); network managers may use information in the RTCP packets to evaluate the performance of their networks for multicast distribution. RTCP provides support for real-time conferencing for large groups within an internet, including source identification and support for gateways and multicast-to-unicast translators. It also offers quality-of-service feedback from receivers to the multicast group as well as support for the synchronization of different media streams.

For security, the RTP/RTCP data can be encrypted to enable improved privacy against eavesdropping.

RTP/RTCP provides functionality and control mechanisms necessary for carrying real-time content. But RTP/RTCP itself is not responsible for the higher level tasks like assembly and synchronization which have to be done in the application level. RTP is a protocol framework that is deliberately not complete. It is open to new payload formats and new multimedia software. By adding new profile and payload format specifications, one can tailor RTP to new data formats and new applications.

B. Receiver-Based Rate Control

The source-based rate-adaptation suffers from poor performance because of the different bandwidth requirements of the receivers. The heterogeneity of the network environment averts the source to transmit with a single rate. Apart from that there are no widely acceptable techniques on how to determine network capacity and how to decide the rate. Different approaches have been proposed using the receiver-based rate control but the most popular is the approach that combines a layered compression algorithm in conjunction with a multicast transmission scheme. The different layers of the hierarchical signal are striped across multiple multicast groups and receivers adapt to congestion by adding and dropping layers. Receivers implicitly define the multicast distribution trees simply by expressing their interest in receiving flows. Thus there is no explicit signalling between the receivers and routers or between the receivers and source.

Like the source-based rate control, we categorize the existing receiver-based rate control protocols and mechanisms into probe-based approaches and equation-based approaches. In probe-based approach the receiver chooses the optimal level of subscription and adds layers by joining the corresponding multicast groups until congestion occurs. Then it backs off to an operating point below bottleneck. On the other hand in equation-based approach the receiver estimates the available bandwidth and then decides whether to join or to disjoin a multicast group.

Receiver-driven Layered Multicast (RLM) [15]

The RLM protocol works within the existing IP model and is assumed only best-effort multipoint packet delivery, the delivery efficiency of IP Multicast and group-oriented communication. In the RLM protocol the source encodes its signal into layers and transmits each layer on a distinct multicast group. Receivers may join or leave groups according to the capacity: if congestion happens then drop a layer or if spare capacity exists then add a layer. The receiver searches for the optimal level of subscription and according to this scheme the receiver adds layers until congestion

occurs and backs off to an operating point below bottleneck. The receiver must determine its current level of subscription. By definition, the subscription is too high if it causes congestion. This is easy to detect because congestion is expressed explicitly in the data stream through lost packets and degraded quality. On the other hand, when the subscription is too low, there is no equivalent signal. The way to carry out this problem is by spontaneously adding layers at “well chosen” times (join-experiments). If the addition of the layers causes congestion the receiver quickly drops the offering layers. If no congestion occurs then the receiver is one step closer to the optimal operating point.
Over the time, through a learning algorithm, each receiver determines the level of subscription that causes congestion. There is separate join-timer for each level of subscription and applying exponential backoff to problematic layers. In addition to join-timer we must know and the detection time which the time that it takes for a local layer change to be fully established in the network and for the resulting impact to be detected back at the receiver. So if a join-experiment lasts longer than the detection-time without congestion occurring, then we deem the experiment successful else the experiment failed and increase the join-timer for that layer. The problem with the above is that the aggregate frequency of such experiments increases with the number of receivers. Since a failed join-experiment could incur congestion to the network, an increase of join-experiments could aggravate network congestion.
To face that problem a shared learning algorithm was proposed. In shared learning algorithm each receiver know about the experiments done by others and their results because receivers share knowledge of what is going on in the network. The mechanism used in the RLM protocol to achieve this goal is multicasting the beginning of experiments and the results of experiments to every member of the group but this neither scalable nor efficient because (a) every receiver need not know about every experiment and/or its result (that is just too much state information) and (b) using multicast to distribute control information such as beginning of experiments and their results, beyond certain scope is inefficient, because it consumes additional bandwidth particularly if every receiver need not know about every experiment and/or its results.
Researchers evaluated RLM for simple scenarios and considered only inter-RLM interaction. They found that RLM can result in high inter-RLM unfairness. Also there was an investigation of the relative merits of uniform versus priority dropping for the transmission of layered video in [22]. It was revealed that RLM performs reasonably well over a broad range of conditions, but performs poorly in extreme conditions like bursty traffic. Apart from that in [23] the behavior of RLM for VBR traffic was further explored and it was shown that RLM exhibits high instability for VBR, has very poor fairness properties in most of the cases and achieves low link utilization under VBR traffic scenarios.

Layered Video Multicast Retransmission (LVMR) [20]

Layered video multicast with retransmissions (LVMR) is a system for distributing video using layered coding over the Internet. LVMR addresses the network congestion and heterogeneity problem using layered video coding techniques by allowing each receiver to subscribe to a subnet of the video layers according to its processing power and network bandwidth availability. The two key contributions of the system are: (1) improving the quality of reception within each layer by retransmitting lost packets given an upper bound on recovery time and applying an adaptive playback point scheme to help achieve more successful retransmission, and

(2) adapting to network congestion and heterogeneity using hierarchical rate control mechanism.
In contrast to the existing sender-based and receiver-based rate control in which the entire information about network congestion is either available at the sender (in sender-based approach) or replicated at the receivers (in receiver-based approach), the hierarchical rate control mechanism distributes the information between the sender, receivers, and some agents in the network (for example receivers can ask neighbours – designated receivers DR – for lost packets) in such a way that each entity maintains only the information relevant to itself. In addition to that, the hierarchical approach enables intelligent decisions to be made in terms of conducting concurrent experiments and choosing one of several possible experiments at any instant of time based on minimal state information at the agents in the network.
The key to scalability in layered multicast is for the receivers to make a decision on their own regarding adding or dropping a layer. However, if these decisions are made independent of the results of join/leave experiments done by others, the results can be disastrous. Thus it is fundamental for each receiver to know about the experiments and their results. LVMR proposes intelligent partitioning of the knowledge base and distributes relevant information to the members in an efficient way while trying to avoid the drawbacks of the shared learning algorithm proposed by RLM [15]. The idea of shared learning, although an improvement to adding and dropping layers indiscriminately, requires each receiver to maintain a variety of state information which it may or may not require. In addition the use of multicast to exchange control information may decrease usable bandwidth on low speed links and lead to lower quality for receivers on these links. In LVMR the hierarchical approach used, allows receivers to maintain minimal state information and decrease control traffic on the multicast session.
Needless to say that due to the fact that LVMR does not depend on any QoS mechanism or other components in the network it can be immediately deployed. On the other hand LVMR shows limited scalability. Neighbours may not be necessarily available all the time so as to share information between the receivers. Furthermore statistically designated receivers and agents makes this approach difficult to deploy and quality adaptation and bandwidth adaptation depend on the number of video layers.

C. Hybrid Rate Control

According to hybrid rate control the receivers adjust the receiving rate of video by adding/dropping channels/layers while the sender also adjust the transmission rate of the channel based on feedback from the receiver. This scheme combines the advantages of sender-driven and receiver-driven approaches resulting in a more effective but more complicated adaptation scheme.

Destination Set Group (DSG) [21]

In order to address the fairness issue in feedback-controlled multicast video distribution a protocol called Destination Set Grouping (DSG) was implemented. DSG is a replicated stream video multicast scheme where the source keeps a small number of video streams carrying the same video but each targeted at receivers with different capabilities. Each stream is feedback-controlled within prescribed limits by its group of receivers. Receivers may move among the streams as their capabilities or the network capabilities change.

The fundamental design goals of the DSG video multicast protocol are (a) improved fairness over a single-group feedback-controlled video multicast scheme and (b) the ability to operate efficiently when the number of receivers is large. Fairness is achieved in DSG by transmitting video of differing quality and differing data rates on different multicast channels and allowing receivers to select the most appropriate one. The DSG protocol is highly scalable because the stream change decisions are made by receivers. Receivers are provided with the necessary information to make the correct stream change decisions. Researchers also use a slightly modified version of the probabilistic feedback technique to avoid feedback implosion.
The DSG protocol has two main components. Firstly, an intra-stream protocol is used by receivers listening to the same stream to adjust the data rate of the stream within its prescribed limits. An independent feedback control mechanism is used within each stream. Each receiver estimates its video reception quality using its packet loss rate. Secondly, an inter-stream or stream change protocol used by receivers to change to a higher or lower quality stream as their needs change. The inter-stream protocol allows a receiver to change to a different stream in situations where they cannot adjust the rate of the stream they are currently receiving to their satisfaction.
The main advantage of this approach is that it is a very scalable solution which deals with heterogeneity but on the other hand the network carries redundant information because replicated streams are being transmitted. On the other hand receiver-based layered multicast techniques such as LVMR [20] and RLM [15] achieve better bandwidth efficiency at the cost of complexity.

D. Summary and Comments

Table 1 summarizes all the aforementioned rate control techniques used in today's Internet while categorizes the characteristics of each protocol.
As can be seen, the overwhelming majority of source-based protocols and techniques operate under unicast transmission scheme whereas only SAMM and RTP/RTCP enable multicast transmission. This is primarily because the source-based rate control works reasonably well for unicast video. On the contrary receiver-based rate control systems like RLM and LVMR are targeted at solving the heterogeneity problem in the multicast case.
Moreover we observe that some of the source-based protocols like SCTP, MTP and SSVP implement the same congestion avoidance mechanisms (namely AIMD) as TCP. Therefore they suffer the same low efficiency in the presence of moderate to high random transient errors due to the fact that they take the loss as an indication of congestion.
It is beyond any doubt that most of the protocols and techniques presented in this survey, do not address the problem of wireless losses because they were designed mostly for wired networks and lack a rate control mechanism that handles wireless losses efficiently. As mentioned in [24] current transport protocols perform poorly over links with bit errors and the recommended approach has been for link layers to incorporate link-level mechanisms like FEC or link-level retransmission to deal with corrupted packets. If mechanisms were available for explicit corruption notification from the link layer to the transport end nodes, transport protocols could correctly interpret the loss as corruption instead of congestion and respond appropriately.
We would like to mention that sender-based protocols and techniques are more flexible and can be deployed with less effort than receiver-based ones. Needless to say that in a sender-base scheme only the sender has the authority to alter the rate (and as a consequence the quality) of the transmitted stream according to network

dynamics. Apparently receiver-based techniques maximize the delivered user perceived quality as every receiver has the opportunity to adjust the quality of the received video stream according to its needs and capabilities. However the main issue in the deployment of receiver-base rate control protocols is the effect of misbehaving receivers. To cope with these issues senders ought to develop mechanisms to detect misbehaving receivers.

		Equation-based / Probe-based	Rate-based / Window-based	Unicast / Multicast	Transport Layer / Application Layer	Wireless Support
Source Based	UDP	-	-	Unicast	Transport	No
	TCP	Probe	Window	Unicast	Transport	No
	TFRC	Equation	Rate	Unicast	Transport	No
	SCTP	Probe	Window	Unicast	Transport	No
	DCCP	Equation/Probe	Rate/Window	Unicast	Transport	No
	SSVP	Probe	Rate	Unicast	Transport	No
	MTP	Probe	Window	Unicast	Transport	No
	VTP	Probe	Window	Unicast	Transport	Yes
	SAMM	-	Rate	Multicast	-	No
	RTP/RTCP	-	-	Uni./Multi.	Application	No
Receiver Based	RLM	Probe	-	Multicast	Transport/ Application	No
	LVMR	-	-	Multicast	Transport/ Application	No
Hybrid	DSG	-	-	Multicast	Transport/ Application	No

Table 1. Comparison amongst the different rate control techniques.

6. Proposed Feedback Algorithms

6.1 Adaptive Feedback Algorithm for Internet Video Streaming based on Fuzzy Control (ADIVIS)

In this Section we propose a new feedback technique in conjunction with a decision algorithm for improving the quality of wireless Internet video streaming applications.
One of the most significant problems that video communications face is the unpredictable nature of the Internet primarily in terms of bandwidth, end-to-end delay and loss variation. Video streaming applications need to implement highly scalable and adaptive techniques in terms of content encoding and transmission rates in order to cope with the erroneous and time variant conditions of the network.
Content Adaptation Techniques (CATs) deal with adaptation of content to the desirable transmission rate using primarily scalable video approaches whereas Network Adaptation Techniques (NATs) deal with the end-to-end adaptation of real time multimedia application needs to the network parameters using algorithms which take into account the state and/or load of the network and the type of errors.
Our approach aims at combining Network Adaptation Techniques with Content Adaptation Techniques in order to finely adapt the video stream bitrate to the

dynamically changing network parameters. In this section, we propose a new feedback mechanism that works in co-operation with an adaptation decision algorithm. From content adaptation point of view, our system involves video streams encoded in a layered manner.
ADIVIS involves an adaptive feedback mechanism for Internet video streaming and a fuzzy decision algorithm. We assume that each video stream is encoded in multiple layers stored at the sender side. The layered video content is transmitted over an RTP connection.
The feedback mechanism combines receiver's critical information on the perceived quality as well as measurements obtained by the core network in order to evaluate the available bandwidth of the network path. The estimated available bandwidth is then fed into the decision algorithm which decides in a fuzzy manner the optimal number of layers that should be sent by adding or dropping layers.
We present the implementation of our rate adaptation heuristic in an MPEG-4 streaming system depicted in the following Fig. 6.

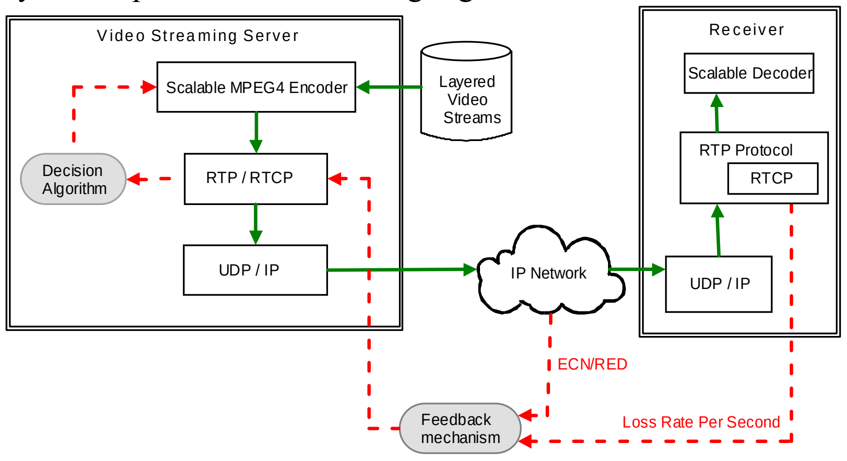


Fig. 6. ADIVIS scheme.

The above diagram illustrates a unicast-oriented ADIVIS-based system. The two outlined components, namely, feedback mechanism and decision algorithm, focus on the adaptation of the layered video content to the available network bandwidth. Dashed arrows track the path of control packets (RTCP) whereas solid arrows track the path of video data packets (RTP).
The feedback mechanism collects QoS information like loss rate and jitter from both the core network and the receiver that will be used for the evaluation of the available bandwidth of the path between the sender and a receiver. The decision algorithm which is implemented at the sender side processes the feedback information and decides the optimum number of layers that will be sent.
The role of the feedback and adaptation components is to link the quality demand of video-enabled applications to the underlying network leading to network adaptation. Network adaptation should be assisted by a proper content adaptation technique which is carried out by layered video encoding.
The rest of this Section is organized as follows. Section 6.1.1 deals with the layered encoding whereas in Section 6.1.2 the feedback mechanism is further analyzed. Section 6.1.3 mentions the fuzzy decision algorithm.

6.1.1 Layered Encoding

Layered encoding is suitable for adapting the quantity of data transmitted by a video server to the capacity of a given network path. Video streams are encoded in a layered manner in a way that every additional layer increases the perceived quality of the stream. Usually a layered video stream consists of a base layer and several additional enhancement layers. Base layers should be encoded in a very low rate so as to accommodate for a large variety of mobile handheld devices as well as terminals connected to the Internet through low bandwidth modem connections. Additional enhancement layers are added, or dropped, in order to adapt the content rate to the desirable transmission rate derived from the feedback mechanism.

6.1.2 Feedback Mechanism

As mentioned before, the feedback mechanism collects information from both the core network and the receiver in order to evaluate the available bandwidth of the network path from the video streaming server to the wireless receiver.

Each receiver sends reception statistics using RTCP packets. In particular special RTCP packets called RR (Receiver Report) packets are being sent carrying a variety of control information. RR packets are primarily used for reception statistics from participants that are not active senders. RRs provide (a) loss fraction (since the previous Sender Report or Receiver Report packet was sent) which denotes the recent quality of the distribution, (b) cumulative number of packet lost (CNPL), (c) highest sequence number received (EHSR) and (d) inter-arrival jitter (which gives an estimation of playout buffer delay at the receiver). As mentioned in [25] using RTCP-RR feedback reports we can evaluate:

a) **Number of lost packets** during an interval (between 2 successive RTCP feedback reports). The difference in the cumulative number of packet lost gives the number lost during that interval. (loss rate can be used to select amount of FEC to employ),
b) **Number of packets expected during the interval**. The difference in the extended last sequence number received gives the number of packets expected during that interval,
c) The ratio of the two aforementioned metrics is the **packet loss fraction** over the interval,
d) The **loss rate per second** can be obtained by dividing the loss fraction by the difference in NTP timestamps, expressed in seconds and
e) Round Trip Time.

The difference between the last two reports received can be used to estimate the recent quality of the distribution. In particular the difference between two consecutive values of loss rate per second (LRPS) can be used in order to track the increasing or decreasing trend of packet loss percentage.

In addition to the notifications provided by the receiver, important information regarding the current status of the congestion within the core network can be efficiently used for the evaluation of the available bandwidth. The Explicit Congestion Notification (ECN) [26], [27] mechanism is used for the notification of congestion to the end nodes in order to prevent unnecessary packet drops. ECN option allows active queue management (AQM) mechanisms such as RED [28] to probabilistically mark (rather than drop) packets. The number of marked packets within a given period may derive a meaningful deduction about the congestion status. This proposal supports ECN scheme in conjunction with UDP transport protocol. Instead of having ACK packets for providing feedback information, RTCP packets can inform the data sender when a congestion experienced (CE) packet has been received at the receiver. These data can be incorporated inside a dedicate field of the RR packet.

6.1.3 Fuzzy Decision Algorithm

Fuzzy control may be viewed as a way of designing feedback controllers in situations where rigorous control theoretic approaches can not be applied due to difficulties in obtaining formal analytical models.

Linguistic variables are a key concept of fuzzy logic control. They take on linguistic values which are words (linguistic terms) that are used to describe characteristics of the variables. Our fuzzy control system is based on two linguistic input variables and one linguistic output variable. All quantities in our system are considered at the discrete instant kT, with T the decision period.

Our first linguistic input variable involves the LRPS parameter. LRPS(kT) is the loss rate per second at each decision period and LRPS(kT-T) is the loss rate per second with a delay T. The linguistic variable $D_{LRPS}(kT)$ gives the increasing or decreasing trend of the LRPS and can be evaluated by:

$$D_{LRPS}(kT) = LRPS(kT) - LRPS(kT-T) \qquad (3)$$

The LRPS parameter is lower and upper bounded by 0 and 1 respectively. Consequently, $D_{LRPS}(kT)$ ranges from -1 to +1.

For the second input linguistic variable we use the number of packets that have the ECN bit set within a period, as a strong indication for congestion. The receiver calculates periodically this number called $N_{ECN}(kT)$. The sender extracts this value from an RR packet and calculates a scaled parameter, $N_{ECNsc}(kT)$, which ranges from 0 to +1, and represents the percentage of packets marked within this period. Eq. 4 is used to obtain the scaled parameter $N_{ECNsc}(kT)$:

$$N_{ECNsc}(kT) = \frac{N_{ECN}(kT)}{N_{ps}(kT)} \qquad (4)$$

Where Nps(kT) is the number of packets sent within the same period. Therefore, we calculate the parameter $DN_{ECNsc}(kT)$, which gives the increasing or decreasing trend of the number of marked packets. The $DN_{ECNsc}(kT)$ is upper and lower bounded by +1 and -1 respectively, and can be evaluated by:

$$DN_{ECNsc}(kT) = N_{ECNsc}(kT) - N_{ECNsc}(kT-T) \qquad (5)$$

Our linguistic output variable, a(kT), is defined for every possible combination of inputs. The defuzzified crisp values of a(kT) can be used by the decision algorithm for the evaluation of the available bandwidth using the following formula:

$$Available\_bandwidth(kT) = \alpha(kT) * Available\_bandwidth(kT-T) \qquad (6)$$

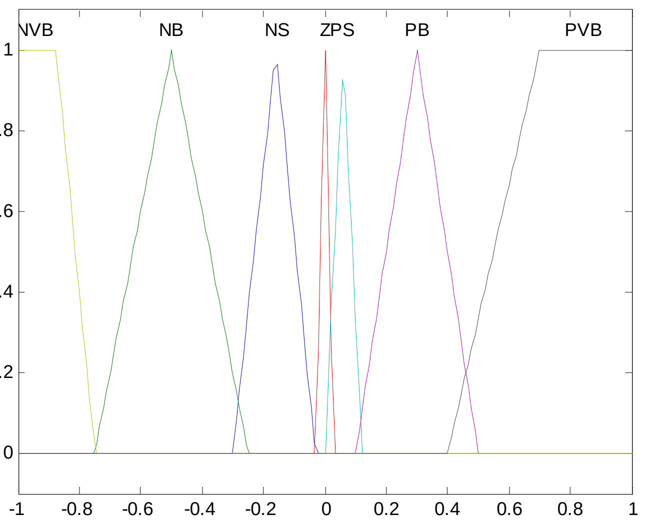
The defuzzified output value is selected to range from 0.5 to 1.5. Thus a 'gradual' increase is allowed when there is available bandwidth and reduced congestion, whereas quick action is taken to reduce the rate to half in case of severe congestion. The output of the fuzzy system could have been a discrete value indicating directly the number of layers that should be sent. Instead, we chose to obtain a crisp value because we wanted our algorithm to be applicable not only in cases where the video streams are not encoded in a coarse grained manner but also when fine grained scalability encoding techniques are applied.

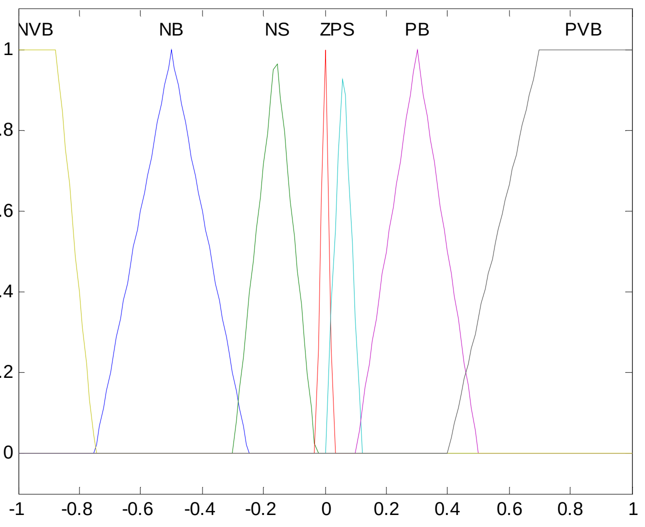
The following table involves if-then rule statements which are used to formulate the conditional statements that comprise fuzzy logic.

$\alpha(kT)$		$\Delta N_{ECNsc}(kT)$						
		NVB	NB	NS	Z	PS	PB	PVB
$\Delta_{LRPS}(kT)$	NVB	**H**	**H**	**B**	**B**	**Z**	**S**	**VS**
	NB	**H**	**VB**	**Z**	**Z**	**Z**	**S**	**VS**
	NS	**B**	**Z**	**B**	**Z**	**Z**	**S**	**VS**
	Z	**B**	**Z**	**Z**	**B**	**Z**	**S**	**VS**
	PS	**Z**	**Z**	**Z**	**Z**	**S**	**S**	**VS**
	PB	**Z**	**Z**	**Z**	**Z**	**S**	**S**	**VS**
	PVB	**VS**	**VS**	**VS**	**VS**	**VS**	**VS**	**VS**

Table 2. Table content notations: Negative/Positive Very Big (NVB, PVB), Negative/Positive Big (NB, PB), Negative/Positive Small (NS, PS), Zero (Z), Very Small/Big (VS, VB), Small/Big (S, B), Medium (M), Huge (H).

The derived membership functions are illustrated in the figures above.





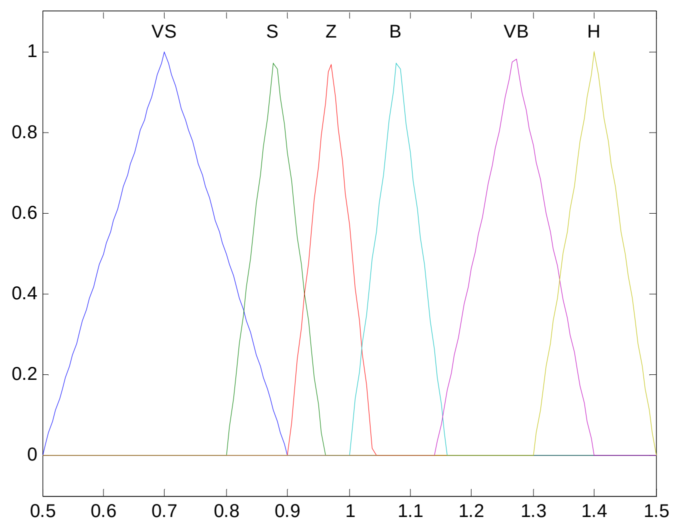


Fig. 7. Membership functions.

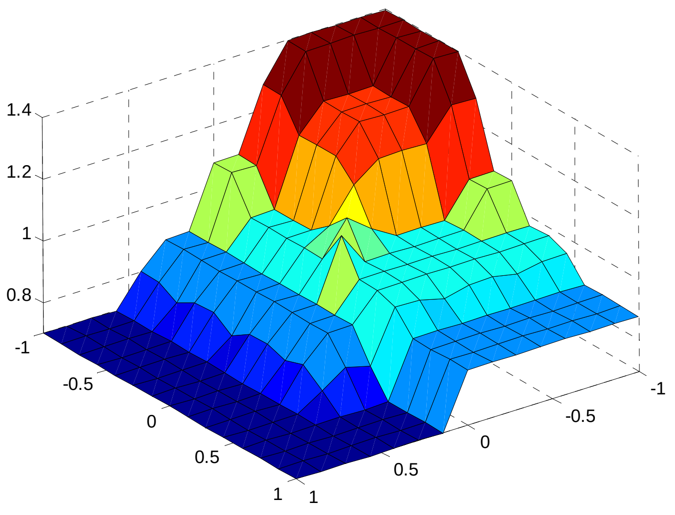


Fig. 8. Decision surface of the fuzzy inference engine shaped by the rule base and the linguistic variables.

Our algorithm has to decide which layers should be sent according to the available network bandwidth, based on a non aggressive layer selection approach. The server will host an appropriate number of layers where each layer corresponds to a different transmission rate. To avoid ping-pong effects there should not be a transition to an upper level layer every time the available bandwidth exceeds the threshold of a specific transmission rate that corresponds to a higher video layer. Instead, a time hysteresis is introduced in order to avoid frequent transitions from one layer to another which may cause instability. In the case of a transition to a lower layer, the effect is immediate, as we seek quick relief from possible congestion. This is shown in the pseudo-code of the decision algorithm below:

```
For all layers j up to MAX_LAYER
    if Available_bandwidth ≤ BitRate(j) and hysteresis = false
    begin
        If layer(j) = current_layer
            break;
        hysteresis = true;
        selected_layer = layer(j);
    end
    else if Available_bandwidth ≤ BitRate(j)
    begin
        hysteresis = false;
        if selected_layer < layer(j)
        begin
            current_layer = selected_layer;
            break;
        end
        else
        begin
            current_layer = layer(j);
            break;
        end
    end
```

Fig. 9. Pseudocode of the fuzzy decision algorithm.

The time hysteresis is equal to the time interval between the reception of two successive RR packets. If the available bandwidth exceeds the threshold of a specific transmission rate that corresponds to an upper level layer, then the hysteresis variable is set. When a new RR packet arrives, if the available bandwidth is still at the same levels, a transition occurs.

6.2 Receiver-driven Adaptive Feedback algorithm (RAF)

In this section we analyze a novel receiver-driven feedback algorithm for the increase of the objective as well as user-perceived quality of the transmitted video stream. The proposed algorithm tries to identify and analyze the different parameters that influence the quality of the video streams while are being transmitted over the Internet. In particular this algorithm adapts these parameters accordingly in order to provide a sufficient user-perceived video quality.

These parameters are adjusted according to their significance and the decision of each individual user. The proposed algorithm adjusts the possible values or the combination of these parameters in such a way in order to guarantee an adequate video quality.

Moreover the objective quality which is adjusted by these parameters is associated with the subjective (user-perceived) quality. In this way we can assure that our algorithm can be successfully applied in several streaming applications.

In addition our algorithm uses the Goddard Streaming Application that was designed in [29] for layered video transmission. We modified this software to evaluate the performance of our algorithm as will be presented in the next deliverable. Goddard will be further analyzed in the next deliverable too.

The rest of this Section is organized as follows. Section 6.2.1 deals with the aforementioned parameters that influence the quality of the transmitted video stream whereas in Section 6.2.2 the algorithm is further analyzed.

6.2.1 Parameters

In this section we will investigate the basic parameters that can be adjusted according to user needs and the unpredictable conditions of the Internet.

Bits Per Pixel (Bpp)

Colour depth is a computer graphics term describing the number of bits used to represent the colour of a single pixel in a bitmapped image or video frame buffer. This concept is also known as bits per pixel (Bpp), particularly when specified along with the number of bits used. Higher colour depth gives a broader range of distinct colours. For example in an 8-bpp video stream, each pixel is depicted using 8 bits, thus we can have 2^8 different colours for each pixel. As the number of bpp is getting larger the perceived quality and the size of the video stream are increased too.
In this project we use a specific scale for the bpp parameter and every step corresponds to a different user perceived quality. This scale is calibrated in 4 different categories as shown in Table 3. The values shown in Table 3 are given by the helixcommunitty.org [30] and refer to high mobility video streams. The values shown in this table are negative because they refer to compressed video streams.

Bpp	Description
>= 0.225	Very Good Quality
>= 0.175	Good Quality
>= 0.125	Medium Quality
<= 0.125	Low Quality

Table 3. Values of Bpp parameter in conjunction with the user perceived quality.

As we can see the bigger the bpp value the better the perceived quality of the video stream.

max_scale

This parameter represents the maximum number of encoding layers that a video streaming server can support. It can be combined with the parameter bitrate_#_ which is presented below. This parameter can be adjusted by the user who defines the maximum number of layers that can be transmitted during a session. According to user decision and based on the conditions of the network, our algorithm dynamically adapts the number of layers that should be transmitted every time.

bitrate_#_

This parameter shows the bit rate of every layer that the video streaming server transmits and depends on the max_scale parameter. The symbol # takes several different values from 0 up to the max_scale minus 1. For example parameter bitrate_3_ corresponds to the bit rate of the third layer.

colour_standard_NTSC_PAL

This parameter represents the colour prototype of the video transmission that can be set by the user. We have two different prototypes; the American standard NTSC operates in 30 frames per second (fps) and the European standard PAL operates in 25 fps. In our algorithm this parameter takes the value 0 for NTSC transmission and 1 for PAL transmission.

fps (frames per second)

Frame rate, or frame frequency, is the measurement of how quickly an imaging device produces unique consecutive images/frames. Frame rate is most often expressed in frames per second or simply, hertz (Hz). This parameter can be set by the user and it is independent of the colour standard that will be used during the transmission. If the user does not set this parameter then it takes its default value according to the colour standard.

aspect_ratio
The aspect ratio of a two-dimensional frame is the ratio of its longer dimension to its shorter dimension. Our algorithm supports only two values for these parameters; 16:9 and 4:3.

frm_width
This parameter corresponds to the width of the frame quoted in pixels. The height of the frame can be evaluated using the aspect_ratio parameter mentioned above.

max_bitrate
A user is connected to the network via a dedicated link. Thus the max_bitrate parameter is equal to the bandwidth of this connection. This value can be altered during the video transmission. Our algorithm has the ability to adapt the bit rate of the transmitted video stream according to the value of this parameter.

avail_bandwidth
The available bandwidth of the link between the user (video client) and the video streaming server is very fluctuating as well as dynamic changing parameter that depends on the variable environment of the network. It is an important parameter that influences the dynamic adaptation of the transmitted video bit rate.

Synopsis
All the aforementioned parameters are depicted in Table 4, where we present the initial values of each parameter. Any user may set anyone of them according to his/her needs or based on the capabilities of his/her (fixed/mobile) terminal equipment in the quest for a better perceived quality. The values of these parameters are dynamically set by the algorithm during the transmission according to the changing conditions of the network.

Parameter	Short Description
max_scale_ = 5	*Represents the maximum number of layers that a video streaming server can handle i.e 5.*

bitrate_0_ = 56000
bitrate_1_ = 128000
bitrate_2_ = 256000
bitrate_3_ = 512000
bitrate_4_ = 768000
	Represent the bit rate of each individual layer. Values are measured in bits per second (bps).

colour_standard_NTSC_PAL_ = 0	*The colour standard of the video stream that will be transmitted. NTSC (30fps) corresponds to 0 whereas PAL (25fps) corresponds to 1.*
fps_ = 10	*The number of frames per second that will be used for the encoding of the video stream.*
aspect_ratio_ = 1.33333333	*The aspect ratio of the image/frame. The values*

	that this parameter can take are shown below: *1.33333333 for aspect ratio 4:3* *1.77777777 for aspect ratio 16:9*
frm_width_ = 180	*This parameter refers to the width of the frame measured in pixels. Our algorithm can evaluate the height of the frame using the aspect_ratio_ parameter.*
bpp_quality_ = 0.2	*The quality of a frame is measured in bits per pixels as shown below:* *>=0.225 Very Good quality* *>=0.175 Good quality* *>=0.125 Medium quality* *< 0.125 Low quality*
max_bitrate_ = 56000	*** This parameter declares the connection bandwidth of the user. The values are measured in bits per second (bps). For example the max_bitrate_ parameter for a 56K-modem user takes the value of 56000.*
avail_bandwidth_ = 56000	*** This parameter declares the available bandwidth in bits per second (bps).*
** This parameter can be dynamically altered by our algorithm during the transmission of the video stream. The values shown above are initially declared prior the establishment of the connection.	

Table 4. Initial values of the aforementioned parameters.

6.2.2 Description of the algorithm

The aforementioned initial values of the parameter are fed into the proposed algorithm which is trying to adapt the transmitted video stream according to them. Prior the adaptation many issues must be taken into consideration in order to end up with the best calibration that results to a high quality perceived video stream at the end user. The operations that take place prior the transmission are thoroughly described below while the key points that influence the decisions taken by the algorithm.

Maximum possible transmission rate

The available bandwidth of a connection fluctuates due to the unpredictable conditions within the network. Our algorithm must decide whether the number of the transmitted layer will be increased or decreased. Moreover some parameters like the bitrate, the size and the quality of the frame have to be set a priori.

Based on the available bandwidth of the network, the algorithm must decide the bit rate of the transmitted video stream according to the capabilities of the video streaming server and the access bandwidth of the user. These three parameters depend on each other. Thus the transmission bit rate takes the lowest value amongst the three parameters (available network bandwidth, access bandwidth, and the maximum bandwidth that the video streaming server can support).

Quality Table and Mean Opinion Score (MOS)

After evaluating the maximum transmission bit rate of the video stream, our algorithm creates a table that includes all the possible combinations between the parameters frames per second (fps), and bits per second (bpp). We consider that the

dimensions of a frame (W x H) remain constant and that there is no loss of frames that may deteriorate the quality.
This table takes into consideration the colour standards NTSC and PAL. It has been mentioned that NTSC runs on 30 fps (precisely on 29.97fps) and the corresponding value for PAL is 25 fps. Beyond those values the perceived quality of the video stream remains unchangeable. Furthermore [30] suggests that for 0.225 bpp the quality of video stream is high. So if we multiply these values (bpp x fps) we get an indicative value (6.75) for high quality video. This product does not strictly describe the maximum perceived quality as we may have higher values for bpp.
Nevertheless in order to make a reduction to the 100% we multiply this indicative value by 14 resulting to 94.5%. The resulting number represents the relative perceived quality or Mean Opinion Score (MOS). The same procedure is followed by our algorithm in order to create the Table 5. As can be seen the table is sorted by the product fps x bpp x 14.

fps x bpp x 14	fps	bpp	MOS
94.5	30	0.225	Very Good Quality
84	30	0.2	
78.75	25	0.225	Good Quality, without artefacts.
73.5	30	0.175	
70	25	0.2	
63	20	0.225	
61.25	25	0.175	
56	20	0.2	Medium Quality, with little artefacts.
52.5	30	0.125	
49	20	0.175	
47.25	15	0.225	
43.75	25	0.125	
42	15	0.2	
42	30	0.1	
36.75	15	0.175	Poor Quality, with perceived artefacts.
35	25	0.1	
35	20	0.125	
31.5	10	0.225	
28	10	0.2	
28	20	0.1	
26.25	15	0.125	
24.5	10	0.175	
21	15	0.1	
17.5	10	0.125	Bad Quality with many artefacts.
14	10	0.1	

Table 5. Quality Table.

We could possibly correlate the product shown in the first column (fps x bpp x 14) with the MOS measurements that are presented in [31] and in Table 6. This correlation is shown in the last column of the Table 5.
We have not confirmed the validity of the relationships that are shown in Table 5. The relationship MOS = bpp x fps x 14 is based on the assumption that given a perfect video stream transmission the only parameters that influence the perceived quality are the bits per pixel and the frames per second. We would like to mention that the delay, delay variation and packet loss are not taken into consideration in perceived quality evaluations. If we consider these parameters in a forthcoming stage, we may have some changes in the values shown in Table 5 but they will not influence the operation of the generic algorithm we present here.

PSNR (dB)	MOS	Quality
> 37	81 – 100	Very Good
31 – 37	61 – 80	Good
25 – 31	41 – 60	Medium
20 – 25	21 – 40	Poor
< 20	0 – 20	Bad

Table 6. PSNR versus MOS.

Relationships between parameters

Using the following mathematical equations we can evaluate the value of a missing parameter. A basic equation that extracted from [30] is shown below:

$$BPP = BitRate \times 1/FPS \times 1/(W \times H) \qquad (7)$$

This equation can be solved accordingly in order to determine anyone of these parameters as shown below:

$$BitRate = BPP \times FPS \times W \times H \qquad (8)$$

$$W \times H = BitRate / (BPP \times FPS) \qquad (9)$$

$$FPS = BitRate / (BPP \times W \times H) \qquad (10)$$

We can also use the Aspect Ratio (AR) notation:

$$AR = W / H \qquad (11)$$

If we import this equation in the previous (7)-(10) we get:

$$H = Sqrt\ (\ BitRate / (BPP \times FPS \times AR)\ ) \qquad (12)$$

$$W = AR \times H \qquad (13)$$

where AR has a decimal value that comes out from equation (11).

Frame Loss Rate

Our algorithm guarantees sufficient user perceived quality under variable network conditions where the core network suffers from congestion. In particular, our system collects statistical data concerning the number of received or lost frames. The algorithm uses the Goddard [29] and the equation (13) in order to evaluate the frame loss rate.

Frame Loss Rate = # of lost frames / # of total number of frames (14)

where:

# of total number of frames = # of lost frames + # of received frames

If the evaluated percentage of lost frames is over 10% then the algorithm uses Table 5 in order to determine which combination uses more fps. The main aim of the algorithm is to send more frames per second than the requested fps. In this way the user will receive the requested quality. The following equation calculates the number of frames per second that should be sent:

# of frames should be sent =
(100 x # of frames requested) / (100 – percentage of frames lost) (15)

For example if a user requests for 20 fps and the percentage of frames lost due to congestion is 30% then the algorithm decides that should transmits (100 x 20) / (100 – 30) = 28.57 fps. In this way the user will receive 20 fps (28.57 x 30% = 20). It is beyond any doubt that the streaming server will not transmit 28.57 fps but it will round this value to the nearest upper multiple of 5; that is 30. In case the colour standard is not NTSC but PAL, the video streaming server will transmit 25 fps.

6.3 Adaptive Congestion Control (ACP)

6.3.1 Introduction

We develop ACP (Adaptive Congestion Protocol) which is a new congestion control protocol with learning capability. This learning capability enables the protocol to adapt to dynamically changing network conditions and maintain stability. ACP can be characterized as a dual protocol where intelligent decisions are taken within the network. The main control architecture is in the same spirit as the one used by the ABR service in ATM networks. Each link calculates at regular time intervals a value which represents the sending rate it desires from all users traversing the link. A packet as it traverses from source to destination it accumulates, in a designated field in the packet header, the minimum of the desired sending rates it encounters in its path. This information is communicated to the user which has generated the packet through an acknowledgement mechanism. The user side algorithm then gradually modifies its congestion window in order to match its sending rate with the value received from the network. The user side algorithm also incorporates a delayed increase policy in the presence of congestion to avoid excessive queue sizes and reduce packet drops.

At each link, the desired sending rate is calculated using both queue length and rate information. The algorithm does not require maintenance of per flow states within the network. Previous experience in the design of such link algorithms ([32] ,[33], [34], [35], [36]) has shown that in order to maintain stability in the presence of delays, the control parameters of the algorithm need to be normalized with the number of users utilizing the network. This is an unknown time varying parameter. Algorithms which have been proposed to estimate this parameter are based on point wise division in

time ([37], [38], [39]). This approach, however, is known to lack robustness and lead to erroneous estimates. In this work, we use on-line parameter identification techniques to derive an estimation algorithm which is shown through analysis and simulations to work effectively.
We will evaluate the performance of the proposed protocol in the next deliverable using simulations. Extensive simulations indicate that the proposed protocol satisfies all the design objectives. The scheme guides the network to a stable equilibrium which is characterized by high network utilization, max-min fairness, small queue sizes and almost no packet drops. It is scalable with respect to changing delays, bandwidths and number of users utilizing the network. It also exhibits nice dynamical properties such as smooth responses and fast convergence. In our simulations we will use realistic traffic patterns which include both bulk data transfers and short lived flows. Finally, we address stability issues of the ACP protocol by using phase plane analysis.
We use a non-linear network model to generate phase portraits which demonstrate that ACP is stable for all delays.

6.3.2 Design guidelines

Our objective has been to design an effective window based congestion control protocol assuming availability of a feedback mechanism which allows the explicit exchange of information between the end users and the network. An effective congestion control protocol must satisfy some basic requirements. It must guide the system to a stable equilibrium point which is characterized by high network utilization, max-min fairness, small queue sizes and no packet drops ([41]). It also needs to be scalable with respect to changing bandwidths, delays, and number of users. Finally, it must exhibit nice dynamical properties such as smooth responses with fast convergence and no overshoot.
XCP [40] constitutes the most notable attempt to fulfil the above objectives. The protocol achieves most of the specifications but fails to achieve max-min fairness at equilibrium in the case of multiple congested links ([42]). XCP has been shown in [40] to outperform all other TCP proposals so our objective has been to develop a protocol which outperforms XCP.
At each link in the network, XCP tries to match the input data rate to the link capacity and at the same time maintain small queue sizes. It achieves this by explicitly dictating to each user utilizing the link the amount by which the congestion window must be increased. However, this does not guarantee that all users utilizing the link will share the same sending rate. XCP addresses this problem by implementing a fairness controller which, however, proves to be ineffective in the case of multiple congested nodes. In order to fix this problem ACP adopts a different design approach. Each link in the network, instead of calculating desired increments of the sending rate of the users traversing the link, it calculates the desired sending rate of the users directly. In a way similar to XCP, the desired sending rate is made available to the end users through an explicit feedback mechanism. A packet as it traverses from source to destination it accumulates, in a designated field in the packet header, the minimum of the desired sending rates it encounters in its path. This information is communicated to the user which has generated the packet through an acknowledgement mechanism. The user then gradually modifies its congestion window in order to match its sending rate with the value received from the network. Since the above mechanism guarantees that all users bottlenecked at a particular link share the same sending rate, fairness is achieved automatically.

Having decided on the control architecture the next step is to design the algorithm which calculates the desired sending at each link. At each link, the objective is to match the input data rate to the link capacity and to maintain small queue sizes. So, the algorithm which updates the desired sending rate must use both queue length and rate information. Pure rate information does not guarantee bounded queue sizes and pure queue length information offers a limited control space and thus leads to oscillations in environments with high bandwidth delay products. Ignoring a projection operator which imposes hard bounds on the desired sending rate a continuous time version of the link algorithm is the following:

$$\dot{p} = \frac{1}{\hat{N}}\left[\frac{k_i}{d}(0.99*C - y) - \frac{k_q}{d^2}q\right] \text{ , } p(0) = p_o \qquad (1)$$

where p is the desired sending rate, C is the link capacity, y is the input data rate, q is the queue size, k_i and k_q are design parameters, d is the calculated average propagation delay and N is an estimate of the number of users traversing the link. The main idea is to integrate the excess capacity and add a queue length factor in order to ensure that at equilibrium the queue size converges to zero. Similar ideas have been used in previous attempts to design congestion control schemes ([43], [40], [44]). The main novelty of our approach is the way with which we estimate the number of flows online in order to maintain stability. We use online parameter identification techniques to derive the estimation algorithm and we implement a certainty equivalent controller ([26]).

Equation (1) is a continuous time representation of the algorithm. In practice, each link updates the desired sending rate every control period. The choice of this period needs careful consideration as it affects the stability and transient properties of the congestion control protocol. ACP chooses its control period using the same mechanism as XCP. Each user utilizes a designated field in the packet header to inform the routers of its current round trip time estimate. Each router then calculates the average round trip time over all packets traversing the link and sets the control period equal to this value. We choose this particular control period to counter the destabilizing effect of delays. Based on fundamental control theory principles, as delays increase within the network we slow down the response time of our controllers in order to maintain stability.

However, simply by choosing the control period as described above, we cannot guarantee stability in the presence of delays. Previous work ([40]) has shown that in order to maintain stability the excess capacity term must be divided with the time delay and the queue size term must be divided with the square of the propagation delay as shown in equation (1). Note that, in a discrete time implementation of the algorithm since we use a varying control period which is equal to the delay we only need to divide the queue term with the delay to maintain stability.

Previous experience in the design of explicit rate based schemes has also shown that in order to maintain stability in the presence of delays, the control parameters need to be normalized with the number of users utilizing the network. This is an unknown time varying parameter which needs to be estimated. Many estimation algorithms have been proposed in literature with different degrees of implementation complexity. Algorithms which do not require maintenance of per flow states within the network are based on point-wise division in time. Assuming that all users traversing the link, send data with the desired sending rate p, the input data rate y satisfies the following relationship:

$$y = Np \qquad (2)$$

where N is the number of flows traversing the link. Since both y and p are known at the link, N can be estimated by dividing y with p. However, such a point-wise division in time is known to lack robustness and can lead to erroneous estimates. In this work, we treat (2) as a linear parametric model of the unknown parameter N and we use online parameter identification techniques to derive an estimation algorithm which is shown through analysis and simulations to work effectively.

As described earlier, each user in the network receives, through explicit feedback, the minimum of the desired sending rates a packet encounters in its path. Since we aim at implementing a window based protocol the rate information must be transformed into window information. We do this by multiplying the desired sending rate with a measure of the propagation delay. Such a measure is obtained by calculating the minimum of the round trip time estimates observed throughout the session. In order to avoid the generation of bursty traffic we do not immediately change the congestion window to the desired value calculated. Instead, we make this change gradually in one round trip time by means of a first order filter.

Even with such a gradual increase policy, excessive queue sizes and packet losses may be observed during transients. The problem arises when a particular link is congested (the input data rate is close to the link capacity) and new users traversing the link enter the network. In this case, the new users adopt the desired sending rate of the link in one round trip time. If the desired sending rate is large, the net effect is a sudden increase in the input data rate at the link. Since, the link is already congested, this increase can lead to large instantaneous queue sizes and packet drops. To alleviate this problem we apply a delayed increase policy at the source in the case of congestion. When a link is congested it sets a designated bit in the header of all incoming packets. In this way the users traversing the link are notified of the presence of congestion and react by applying a delayed increase policy. When they have to increase their congestion window they decrease the smoothing gain of the first order filter by a factor of 10 so that they increase at a much slower rate, thus giving time to the link to detect the new users, adapt its desired sending rate and avoid packet losses.

6.3.3 The protocol

A. The packet header

H_rtt (sender's rtt estimate
H_feedback (desired sending rate)
H_congestion (congestion bit)

Fig. 10. ACP Header.

In a way similar to XCP the ACP packet carries a congestion header which consists of 3 fields as shown in Fig. 10. The H_rtt field carries the current round trip time estimate of the source which has generated the packet. The field is set by the user and is never modified in transit. It is read by each router and is used to calculate the control period. The H_feedback field carries the sending rate which the network requests from the user which has generated the packet. This field is initiated with the user's desired rate and is then updated by each link the packet encounters in its path. At each link, the value in the field is compared with the desired sending rate value and the smallest value is stored in the H_feedback field. In this way, a packet as it traverses from source to destination it accumulates the minimum sending rate it encounters in its path. The H_congestion bit is a single bit which is initialized by the user with a zero value and is set by a link if the input data rate at that link is more that 95% of the link capacity. In this way, the link informs its users that it is on the verge of becoming congested so that they can apply a delayed increase policy and avoid excessive instantaneous queue sizes and packet losses.

B. The ACP Sender

As in TCP, ACP maintains a congestion window cwnd which represents the number of outstanding packets and an estimate of the current round trip time rtt. In addition to these variables ACP calculates the minimum of the round trip time estimates which have been recorded, mrtt. This is a good measure of the propagation delay of the source destination path and is used to transform the rate information reaching the sender to window information.

The initial congestion window value is set to 1 and is never allowed to become less than this value because this would cause the source to stop sending data. On packet departure, the H_feedback field in the packet header is initialized with the desired sending rate of the application and the H_rtt field stores the current estimate of the round trip time. If the source does not have a valid estimate of the round trip time the H_rtt field is set to zero.

The congestion window is updated every time the sender receives an acknowledgement. When a new acknowledgement is received, the value in the H_feedback field, which represents the sending rate requested by the network in bytes per second, is read and is used to calculate the desired congestion window as follows:

$$desired\_window = \frac{H\_feedback \times mrtt}{size} \qquad (3)$$

where size is the packet size in bytes. We multiply with the mrtt to transform the rate information into window information and we divide by the packet size to change the units from bytes to packets. The desired window is the new congestion window requested by the network. We do not immediately set the cwnd equal to the desired congestion window because this abrupt change may lead to bursty traffic.

Instead we choose to gradually make this change by means of a first order filter. The smoothing gain of this filter depends on the state of the H congestion bit in the acknowledgement received. If this is equal to 1, which indicates congestion in the source destination path, we apply a less aggressive increase policy. The congestion window is updated according to the following equation:

$$cwnd = \begin{cases} cwnd + \frac{0.1}{cwnd}(desired\_window - cwnd) & if\_desired\_window > cwnd, H\_congestion = 1 \\ \Pr\left[cwnd + \frac{1}{cwnd}(desired\_window - cwnd)\right] & otherwise \end{cases} \qquad (4)$$

where the projection operator Pr[.] is defined as follows:

$$\Pr[x] = \begin{cases} x & x > 1 \\ 1 & otherwise \end{cases} \qquad (5)$$

The projection operator guarantees that the congestion window does not become less than 1.

C. The ACP Receiver

The ACP receiver is identical to the XCP receiver. When it receives a packet it generates an acknowledgement in which it copies the congestion header of the packet.

D. The ACP router

At each output queue of the router, the objective is to match the input data rate y to the link capacity C and at the same time maintain small queue sizes. To achieve this objective the router maintains for each link, a value which represents the sending rate it desires from all users traversing the link. The desired sending rate is denoted by p and is updated every control period. The router implements a per link control timer. The desired sending rate and other statistics are updated every time the timer expires. The control period is set equal to the average round trip time d. The average round trip time is initialized with a value of 0.05 and is updated every control period. On packet arrival the router reads the H rtt field in the packet header and updates the variables which are used to calculate the average round trip time.

The router calculates at each output queue the input data rate y. For each link, the router maintains a variable which denotes the number of received bytes. This variable is incremented with the packet size every time the queue associated with the link receives a packet. When the control timer expires, the link calculates the input data rate by dividing the received number of bytes with the control period. It then resets, the received number of bytes.

The router also maintains at each output queue the persistent queue size q. The q is computed by taking the minimum queue seen by the arriving packets during the last propagation delay. The propagation delay is unknown at the router and is thus estimated by subtracting the local queueing delay from the average RTT. The local queueing delay is calculated by dividing the instantaneous queue size with the link capacity.

The above variables are used to calculate the desired rate p every control period using the following iterative algorithm:

$$p(k+1) = \Pr\left[p(k) + \frac{1}{\hat{N}(k)} \left[k_i (0.99 * C - y(k)) - \frac{1}{d(k)} k_q q(k) \right] \right], \quad p(0) = 0 \quad (6)$$

where k_i and k_q are design parameters, $\hat{N}$ represents an estimate of the number of users utilizing the link and the projection operator is defined as follows:

$$\Pr[x] = \begin{cases} 0 & x < 0 \\ C & x > C \\ x \; otherwise & \end{cases} \qquad (7)$$

The projection operator guarantees that the desired sending rate is non-negative and smaller than the link capacity. Values outside this range are not feasible. The design parameters k_i and k_q are chosen to be 0.1587 and 0.3175 respectively. There are several things to note about the link algorithm (6). The basic idea is to integrate the excess capacity and to add a queue size term to guarantee that at equilibrium the queue size converges to zero. Previous work has shown that in a continuous time representation of the algorithm, in order to maintain stability, the excess capacity term must be divided with the time delay and the queue size term must be divided with the square of the propagation delay. However, when transforming the continuous time representation to the discrete time representation of equation (6), we multiply both terms with the time delay and so we end up dividing only the queue term with the delay to maintain stability. Note also that we slightly underutilize the link at equilibrium by setting the virtual capacity equal to 99% of the true link capacity. We do this to reserve bandwidth resources which can be used to accommodate statistical fluctuations of the bursty network traffic. This prevents excessive instantaneous queue sizes.

Previous experience in the design of link algorithms for congestion control has shown that to maintain stability we need to normalize the control parameters with the number of users utilizing the network. A novel part of this work is that we use online parameter identification techniques to derive an algorithm which estimates the unknown parameter online. The derivation is based on a fluid flow model of the network and is presented in Appendix A together with the properties of the algorithm. Here we present a discrete time implementation of the algorithm.

$$\hat{N}(k+1)\_=\Pr\left[\hat{N}(k)+\frac{\hat{\gamma}\left(y(k)-\hat{N}(k)\,p(k)\right)p(k)}{1+p^2(k)}\right],\ \hat{N}(0)=10, \quad (8)$$

where the projection operator Pr[.], is defined as follows:

$$\Pr[x]=\begin{cases} x & x>1 \\ 1 & otherwise \end{cases}, \quad (9)$$

The projection operator guarantees that the number of flows traversing the link is never allowed to be less than 1. Values less than one are obviously not feasible. γ is a design parameter which affects the convergence properties of the algorithm. We choose γ to be equal to 0.1. Note that the initial value of the estimated number of flows $\hat{N}$ is equal to 10. We choose this value to ensure a relatively conservative policy when initially updating the desired sending rate.

The desired sending rate calculated at each link is used to update the H_feedback field in the packet header. On packet departure, the router compares the desired sending rate with the value stored in the H_feedback field and updates the field with the minimum value. In this way, a packet as it traverses from source to destination it accumulates the minimum of the desired sending rates it encounters in its path.

The last function performed by the router at each link is to notify the users traversing the link of the presence of congestion so that they can apply a delayed increase policy. On packet departure the link checks whether the input data rate is larger than 0.95 the link capacity. In this case it deduces that the link is congested and sets the H_congestion bit in the packet header.

6.4 Queue Length Based Internet Congestion Control protocol

In this section, we present a new, queue length based Internet congestion control protocol. The proposed protocol utilizes an explicit multi-bit feedback scheme similar to the one in [40] and does not require maintenance of per flow states within the network. It implements at each link a certainty equivalent, proportional controller which utilizes estimates of the effective number of users utilizing the link. These estimates are generated online using a novel estimation algorithm which is presented in detail in [45] and is based on online parameter identification techniques.

6.4.1 The Protocol

In this section, we describe the main features of the protocol. Some of the functionalities of the protocol are inspired from the work in [40].

A. The packet header

The packet carries a congestion header which consists of 3 fields as shown in Fig. 11. The H_rtt field carries the current round trip time estimate of the source which has generated the packet. The field is set by the user and is never modified in transit. It is read by each router and is used to calculate the control period. The H_feedback field carries the sending rate which the network requests from the user which has generated the packet. This field is initiated with the user’s desired rate and is then updated by each link the packet encounters in its path. At each link, the value in the field is compared with the desired sending rate value and the smallest value is stored in the H_feedback field. In this way, a packet as it traverses from source to destination it accumulates the minimum sending rate it encounters in its path. The H_congestion bit is a single bit which is initialized by the user with a zero value and is set by a link if the input data rate at that link is more that 95% of the link capacity. In this way, the link informs its users that it is on the verge of becoming congested so that they can apply a delayed increase policy and avoid excessive instantaneous queue sizes and packet losses.

H_rtt (sender’s rtt estimate
H_feedback (desired sending rate)
H_congestion (congestion bit)

Fig. 11. Congestion Header.

B. The sender

As in TCP, each link maintains a congestion window cwnd which represents the number of outstanding packets and a smoothed estimate of the current round trip time srtt which is calculated using an exponentially weighted moving average filter.

The initial congestion window value is set to 1 and is never allowed to become less than this value because this would cause the source to stop sending data. On packet departure, the H_feedback field in the packet header is initialized with the desired sending rate of the application and the H_rtt field stores the current estimate of the round trip time. If the source does not have a valid estimate of the round trip time the H_rtt field is set to zero.

The congestion window is updated every time the sender receives an acknowledgement. When a new acknowledgement is received, the value in the H_feedback field, which represents the sending rate requested by the network in bytes per second, is read and is used to calculate the desired congestion window as follows:

$$desired\_window = \frac{H\_feedback \times mrtt}{size} \qquad (10)$$

where size is the packet size in bytes. We multiply with the srtt to transform the rate information into window information and we divide by the packet size to change the units from bytes to packets. The desired window is the new congestion window requested by the network. We do not immediately set the cwnd equal to the desired congestion window because this abrupt change may lead to bursty traffic. Instead we choose to gradually make this change by means of a first order filter. The smoothing gain of this filter depends on the state of the H_congestion bit in the acknowledgement received. If this is equal to 1, which indicates congestion in the source destination path, we apply a less aggressive increase policy. The congestion window is updated according to the following equation:

$$cwnd = \begin{cases} cwnd + \dfrac{0.1}{cwnd}(desired\_window - cwnd) & if\_desired\_window > cwnd, H\_congestion = 1 \\ \Pr\left[cwnd + \dfrac{1}{cwnd}(desired\_window - cwnd)\right] & otherwise \end{cases} \qquad (11)$$

where the projection operator Pr[.] is defined as follows:

$$\Pr[x] = \begin{cases} x & x > 1 \\ 1 & otherwise \end{cases} \qquad (12)$$

The projection operator guarantees that the congestion window does not become less than 1.

C. The receiver

The receiver is identical to the XCP receiver. When it receives a packet it generates an acknowledgement in which it copies the congestion header of the packet.

D. The router

The router maintains at each output queue a value p, which represents the sending rate it desires from all users traversing the link. The desired rate is updated every time a control timer expires. The timer is set to to expire every average round trip time d. The average round trip time is initiated with a value of 0.05 and is also updated every time the control timer expires. To calculate the average round trip time, the router records upon packet arrival, the value stored in the H_rtt field of the packet header and divides the sum of the values recorded in one control period with the period.
In order to calculate the desired sending rate, the router requires two more variables at each output queue: the persistent queue size q and the rate of incoming packets y. The persistent queue size q is computed by taking the minimum queue seen by the arriving packets during the last propagation delay. The propagation delay is unknown at the router and is thus estimated by subtracting the local queueing delay from the average RTT. The local queueing delay is calculated by dividing the instantaneous queue size with the link capacity.
The rate of incoming packets y, is equal to the number of packets entering the queue in one control period divided by the control period. To calculate the number of received packets, the router maintains a variable which is incremented with the packet

size every time the queue receives a packet. When the control timer expires, the link calculates y by adding the received number of packets and then dividing with the control period. It then resets the received number of bytes. The above variables are used to calculate the desired rate p every control period. At each link, the objective is to regulate the queue size q so that it tracks a reference queue size q_{ref} chosen by the designer. To achieve the latter, we calculate the desired sending rate p using a modified version of the algorithm proposed in ([46]). The algorithm is as follows:

$$p(k) = \Pr\left[\frac{C - \frac{1}{d(k)}\left(q(k) - 2q_{ref}\right)}{\hat{N}(k)}\right], (13)$$

where $\hat{N}$ represents an estimate of the number of users utilizing the link and the projection operator is defined as follows:

$$\Pr[x] = \begin{cases} 1 & x < 1 \\ C & x > C \\ x \; otherwise & \end{cases} \qquad (14)$$

It guarantees that the desired sending rate is greater than 1 and smaller than the link capacity. The lower bound is imposed since we know priori that the protocol will send a packet, only if it has at least one byte to send. Values greater than the link capacity are not feasible. The control algorithm (13) basically applies proportional action. The delay term d(k) is added to maintain stability in the presence of delays. A novel part of the proposed scheme is that at each link, the estimate of the number of users utilizing the link $\hat{N}(k)$ is generated online using an algorithm which is presented in detail in [45] and is based on online parameter identification techniques. The estimation algorithm is as follows:

$$\hat{N}(k+1) = \Pr\left[\hat{N}(k) + \delta\hat{N}(k)\right], \; \hat{N}(0)=10, \quad (15)$$

where,

$$\delta\hat{N}(k) = \frac{\gamma\left[y(k) + \frac{q(k-1)}{d(k)} - \hat{N}(k)p(k-1)p(k-1)\right]}{1 + p^2(k-1)}, \; (16)$$

The projection operator Pr[.] is defined in (12). The projection operator guarantees that the number of flows traversing the link is never allowed to be less than 1. Values less than one are obviously not feasible. γ is a design parameter which affects the convergence properties of the algorithm. We choose γ to be equal to 0.1. Note that the initial value of the estimated number of flows $\hat{N}$ is equal to 10. We choose this value to ensure a relatively conservative policy when initially updating the desired

sending rate. The desired sending rate calculated at each link is used to update the H_feedback field in the packet header. On packet departure, the router compares the desired sending rate with the value stored in the H_feedback field and updates the field with the minimum value. In this way, a packet as it traverses from source to destination it accumulates the minimum of the desired sending rates it encounters in its path.
The last function performed by the router at each link is to notify the users traversing the link of the presence of congestion so that they can apply a delayed increase policy. On packet departure the link checks whether the input data rate is larger than 0.95 the link capacity. In this case it deduces that the link is congested and sets the H_congestion bit in the packet header.

7. Conclusions

In this deliverable we showed that designing Network Adaptation Techniques for Internet video transmission poses many challenges. In addition we took a holistic approach to these challenges namely Internet video transmission requirements and we presented some of the existing solutions regarding network adaptation techniques primarily from transport perspective (focusing on congestion control) that are used for the improvement of the perceived quality.
The content adaptation techniques which are methods commonly used for the adaptation of content to the desirable transmission rate working in cooperation with the network adaptation techniques must be able to adapt the content to the necessary transmission rate without having to regenerate the information. Transmitting the information in multiple layers which are decoded together at the receiving end is deemed the most appropriate for cases of extreme conditions.
Moreover we presented two novel feedback techniques for the increase of the user perceived quality. ADIVIS is a sender-driven approach. The video streaming server first codes the video sequence into multiple data streams (called layers) and then sends each stream/layer to each receiver. Each receiver establishes a new connection with the streaming server. The server uses its feedback mechanism and its decision algorithm in order to calibrate the transmission bit rate through this connection. During the transmission the user cannot change any parameter of the video stream; only the server has the right to do this.
On the other hand the second algorithm (RAF) is merely uses a receiver-driven technique as the user has the ability to set some transmission parameters according to his/her needs and the capabilities of his/her terminal equipment.
Furthermore we proposed two novel congestion control protocols for high speed networks. Adaptive Congestion control Protocol (ACP) is a window based protocol which does not require maintenance pf per flow states within the network. It utilizes an explicit multi-bit feedback signalling scheme to convey congestion information from the network to the end users and vice versa. A distinct feature of the protocol is the implementation at each link of an estimation algorithm which is derived using on line parameter identification techniques. The algorithm generates estimates of the number of users utilizing the link which are used to tune the control parameters in order to maintain stability. This feature, enables the protocol to adapt to dynamically changing network conditions.
We also present a new Internet congestion control protocol whose objective is to regulate the queue size at each link so that it tracks a reference queue size chosen by the designer.

The performance evaluation of all the algorithms as well as the pros and the cons of each approach will be presented through simulations that will be conducted and presented in the next deliverable.

References

[1] D. Wu, Y. T. Hou, W. Zhu, Y. Zhang and J. M. Peha, "Streaming over the Internet: Approaches and Directions," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, March 2001.

[2] H. Radha, Y. Chen, K. Parthasarathy and R. Cohen, "Scalable Internet Video using MPEG-4," in Signal Processing: Image Communication pp. 95-126, 1999.

[3] H. Radha, M. Van Der Schaar and Y. Chen, "The MPEG-4 Fine-Grained Scalable Video Coding Method for Multimedia Streaming Over IP," IEEE Transactions on Multimedia, Vol. 3, No. 1, March 2001.

[4] W. Li, "Overview of Fine Granularity Scalability in MPEG-4 Video Standard," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, March 2001.

[5] J. C. Bolot and T. Turletti "Experience with Control Mechanisms for Packet Video in the Internet".

[6] D. Wu, Y. T. Hou, W. Zhu, H.-J. Lee, T. Chiang, Y.-Q. Zhang, and H. J. Chao, "On end-to-end architecture for transporting MPEG-4 video over the Internet," IEEE Trans. Circuits Systems Video Technology, Vol. 10, pp. 923–941, Sept. 2000.

[7] A. Eleftheriadis and D. Anastassiou, "Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video," in Proc. IEEE Int.Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV'95), Apr. 1995, pp. 95–106.

[8] J. Van Der Merwe, S. Sen, and C. Kalmanek, "Streaming Video Traffic: Characterization and Network Impact," in Proceedings of the 7th International Workshop on Web Content Caching and Distribution, Boulder CO, USA, August 2002.

[9] M. Handley, S. Floyd, J. Padhye and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," RFC3448, Internet Engineering Task Force.

[10] J. Chung, M. Claypool and Robert Kinicki "MTP: A Streaming-Friendly Transport Protocol," Technical Report WPI-CS-TR-05-10, Computer Science, WPI, May 2005.

[11] G. Yang, M. Gerla and M. Y. Sanadidi, "Adaptive Video Streaming in presence of wireless errors," in Proceedings of the IPIF/IEEE MMNS, San Diego, CA, October 2004, Springer-Verlag.

[12] B. J. Vickers, C. Albuquerque and T. Suda, "Source Adaptive Multi-Layered Multicast Algorithms for Real-Time Video Distribution," IEEE/ACM Transactions on Networking, Vol 8, No. 6, pp. 720-733, December 2000.

[13] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," IEEE/ACM Transactions on Networking, Vol. 7, pp. 458-472, August 1999.

[14] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003.

[15] S. McCanne, V. Jacobson and M. Vitterli, "Receiver-driven Layered Multicast," in Proceedings of ACM SIGCOMM'96, pp. 117-130, Stanford, CA, August 1996.

[16] R. Steward, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang and V. Paxson, “Stream Control Transmission Protocol,” RFC 2960.

[17] E. Kohler, M. Handley, S. Floyd and J. Padhye, “Datagram Congestion Control Protocol,” draft-ietf-dccp-spec-04.txt, work in progress, June 2003.

[18] P. Papadimitriou and V. Tsaousidis, “End-to-end Congestion Management for Real-Time Streaming Video over the Internet,”.

[19] A. Balk, M. Sigler, M. Gerla, and M. Y. Sanadidi, “Investigation of MPEG-4 Video Streaming over SCTP,” In Proceedings of SCI ’02, July 2002.

[20] X. Li, S. Paul, and M. Ammar, “Layered video multicast with retransmission (lvmr): Evaluation of hierarchical rate control,” In Proceedings of IEEE INFOCOM, Mar. 1998.

[21] Cheung, M. H. Ammar, and X. li, "On the Use of Destination Set Grouping to Improve Fairness in a Multicast Video Distribution,” In Proceedings of INFOCOM ’96, pp. 553-560, San Francisco, CA, March 1996.

[22] S. Bajaj, L. Breslau and S. Shenker, “Uniform versus Priority Dropping for Layered Video,” In SIGCOMM ’98, Vancouver, British Columbia, Canada, September 1998.

[23] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson and C. J. Sreenan, “Stability and Fairness Issues In Layered Multicast,” In NOSSDAV ’99, 1999.

[24] A. Gurtov and S. Floyd, “Modelling Wireless Links for Transport Protocols,” ACM Computer Communication Review, pp. 85-96, April 2004.

[25] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” RFC 3550, July 2003.

[26] S. Floyd, “TCP and explicit congestion notification,” ACM Computer Communication Review, 24(5):8–23, October 1994. http://www.aciri.org/floyd/.

[27] K. Ramakrishnan and S. Floyd, “A proposal to add explicit congestion notification (ECN) to IP,” RFC 2481, January 1999.

[28] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” IEEE/ACM Transactions on Networking, vol. 1, pp. 397 – 413, August 1993. ftp://ftp.ee.lbl.gov/papers/early.ps.gz.

[29] J. Chung, M. Claypool and R. Kinicki, “MTP: A Streaming-Friendly Transport Protocol,” Technical Report WPI-CS-TR-05-10, Computer Science, Worcester Polytechnic Institute. May 2005.

[30] “Encoding Settings Calculator,” https://helixcommunity.org/project/shownotes.php?release_id=204

[31] J.-R. Ohm, Bildsignalverarbeitung fuer Multimedia-Systeme, http://bs.hhi.de/users/ohm/download/bvm-kap1&2.pdf, (1999).

[32] L. Benmohamed and S. M. Meerkov. Feedback control of congestion in packet switching networks: The case of a single congested node. IEEE/ACM Transactions on Networking, 1(6):693-708, December 1993.

[33] C. E. Rohrs and R.A. Berry. A linear control approach to explicit rate feedback in ATM networks. In Proc. IEEE INFOCOM’97, volume 1, pages 277-282, March 1997.

[34] T. Basar, E. Altman, and R. Srikant. A distributed globally convergent algorithm for fair, queue-length-based congestion control. In Proc. IEEE Conf. Decision and Control, volume 1, pages 622-627, December 2001.

[35] Y. Zhao, S. Q. Li, and S. Sigarto. A linear dynamic model for design of stable explicit-rate ABR control systems. In Proc. IEEE INFOCOM’97, volume 1, pages 283-292, March 1997.

[36] E. Altman, T. Basar, and R. Srikant. Robust rate control for ABR sources. In Proc. IEEE INFOCOM'98, volume 1, pages 166-173, March 1998.

[37] C. Fulton, S. Li, and C. S. Lim. An ABR feedback control scheme with tracking. In Proc. IEEE INFOCOM'97, volume 2, pages 805-814, April 1997.

[38] M. K. Wong and F. Bonomi. A novel explicit rate congestion cotnrol algortithm. In Proc. IEEE GLOBECOM'98, volume 4, pages 2432-2439, November 1998.

[39] Y. Zhang, D. Leonard, and D. Loguinov. Jetmax: Scalable max-min congestion control for high-speed heterogeneous networks. In Proc. IEEE INFOCOM, April 2006.

[40] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for high-bandwidth-delay products. In Proc. ACM SIGCOMM, August 2002.

[41] Paganini F., Z. Wan, Doyle J.C, and Low S. H. Congestion control for high performance stability and fairness in general networks. IEEE/ACM Transactions on Networking, 13(1):43-56, February 2005.

[42] S. H. Low, L. L. H. Andrew, and B. P. Wydrowski. Understanding XCP: Equilibrium and fairness. In Proc. IEEE INFOCOM, volume 2, pages 1025-1036, March 2005.

[43] F. Paganini. On the stability of optimization-based flow control. In Proceedings of 2001 American Control Conference., volume 6, pages 4689 -4694, 2001.

[44] N. Dokkipati, M. Kobayashi, Rui Zhang-Shen, and Nick McKeown. Processor sharing flows in the internet. In Proc. Thirteenth Intenrational Workshop on Quality of Service 2005,, June 2005.

[45] M. Lestas, A. Pitsillides, P. Ioannou, and G. Hadjipollas. A new estimation scheme for the effective number of users in internet congestion control. IEEE/ACM Transactions on Networking, Submitted for publication.

[46] G. Veciana C. F. Su and J. Walrand. Explicit rate flow control for ABR services in ATM networks. IEEE/ACM Transactions on Networking, Vol. 8, No. 3, June 2000, pp. 350–361.