

Running head: USER QOE OPTIMIZATION

Optimizing User Quality of Experience through Overlay Routing, Bandwidth
Management and Dynamic Transcoding

Maarten Wijnants* and Wim Lamotte

Expertise Centre for Digital Media, Hasselt University – IBBT

Wetenschapspark 2, BE-3590 Diepenbeek, Belgium

Tel: +32 (0)11 26 84 11 – Fax: +32 (0)11 26 84 99

{maarten.wijnants,wim.lamotte}@uhasselt.be

Bart De Vleeschauwer, Filip De Turck, Bart Dhoedt and Piet Demeester

IBCN, Department of Information Technology, Ghent University – IBBT

Gaston Crommenlaan 8/201, BE-9050 Ledeberg-Ghent, Belgium

Tel: +32 (0)9 331 49 00 – Fax: +32 (0)9 331 48 99

{bart.devleeschauwer,filip.deturck,bart.dhoedt,piet.demeester}@intec.ugent.be

Peter Lambert, Dieter Van de Walle, Jan De Cock, Stijn Notebaert and Rik Van de Walle

Multimedia Lab, Department of Electronics and Information Systems, Ghent University –

IBBT

Gaston Crommenlaan 8/201, BE-9050 Ledeberg-Ghent, Belgium

Tel: +32 (0)9 331 49 14 – Fax: +32 (0)9 331 48 96

{peter.lambert,dieter.vandewalle,jan.decock,stijn.notebaert,rik.vandewalle}@ugent.be

Abstract

Accessing multimedia services via fixed and wireless networks has become common practice. These services are typically much more sensitive to packet loss, delay and/or congestion than traditional services. In particular, multimedia data is often time critical and, as a result, network issues are not well tolerated and significantly deteriorate the user's Quality of Experience (QoE). We therefore propose a QoE optimization platform that is able to mitigate problems that might occur at any location in the delivery path from service provider to customer. More specifically, the distributed architecture supports overlay routing to circumvent erratic parts of the network core. In addition, it comprises proxy components that realize last mile optimization through automatic bandwidth management and the application of processing on multimedia flows. In this paper we introduce a transcoding service for this proxy component which enables the transformation of H.264/AVC video flows to an arbitrary bitrate. Through representative experimental results, we illustrate how this addition enhances the QoE optimization capabilities of the proposed platform by allowing the proxy component to compute more flexible and effective bandwidth distributions.

Keywords – QoE optimization, overlay routing, network traffic shaping, NIProxy, H.264/AVC video transcoding, dynamic rate shaping

Optimizing User Quality of Experience through Overlay Routing, Bandwidth Management and Dynamic Transcoding

In recent years, a popularization of the networked access of multimedia services has occurred. Compared to traditional services like web browsing and e-mail, these services impose much stricter requirements on the transportation network. For instance, interactive applications such as VoIP and online gaming demand a low delay to guarantee a fluid operation. As another example, packet loss negatively impacts video streaming services since it will rapidly degrade playback at receiver-side due to the introduction of perceptual distortions. Complicating matters even further is the fact that, due to the recent trend towards mobile computing, service providers are increasingly targeting not only fixed but also mobile customers. Since fixed and mobile devices as well as networks have largely divergent capabilities, a highly heterogeneous usage environment is created, which in turn results in growing service dependability as well as adaptation requirements.

Empirical experience has proven that current generation networks are not always capable of guaranteeing that the requirements imposed by multimedia services are satisfied. For instance, the Internet only provides best-effort routing, meaning no guarantees are given regarding the level of service that will be experienced by network packets. The access part of a client's network connection is another possible source of complications, mainly due to its bandwidth capacity constraints. Insufficient last mile bandwidth may be available to support all the user's active services (or even to receive all content that is being exchanged as part of a single multimedia service). This will likely give rise to congestion and hence also an increase in packet loss and delay in case adequate techniques for the adaptation of network traffic are lacking.

Based on these observations, we argue that current networks frequently fail to provide customers of multimedia services with an acceptable usage experience or, more

formally, Quality of Experience (QoE). In our previous work we therefore introduced a two-tier overlay platform which supports full end-to-end user QoE optimization (De Vleeschauwer et al., 2008). On the one hand, the proposed architecture enhances data dissemination in the network core by providing an overlay routing service which improves resilience to issues like failing or congested network links. On the other hand, proxy servers deployed close to end-users provide functionality to deliberately apportion bandwidth among network flows on the last mile and simultaneously act as service provision platform since they are also capable of applying processing on (multimedia) network flows.

In this paper, we present a novel transcoding service for the proxy components of our proposed overlay architecture which enables them to dynamically adapt the bitrate of H.264/AVC-encoded video bitstreams. Given the complexity of the H.264/AVC specification, cascading a full decode and subsequent re-encode step would limit the applicability of the transcoder to off-line scenarios. The transcoding service therefore operates entirely in the compressed domain to enable real-time video transformation. As a secondary contribution, we illustrate how the capabilities of the proxy component's network traffic shaping functionality and the novel H.264/AVC transcoding module can be bundled to produce highly dynamic and flexible bandwidth management results. As will be validated using representative experimental results, the availability of the H.264/AVC service considerably extends the QoE optimization performance of our platform.

The outline of the remainder of this paper is as follows. The next Section provides an overview of the architecture of the proposed overlay platform and harbors a detailed description of each of the platform's constituting components. In the following Section a thorough evaluation of our platform and, in particular, of its novel H.264/AVC video transcoding capabilities is presented. Through representative experimental results, this Section will demonstrate the platform's ability to beneficially impact user QoE. Next, an overview of related work is given. The final Section draws our conclusions.

End-to-End Architecture

Our platform for QoE enhancement adheres to the separation of concerns paradigm and as such comprises three distinct types of components located at key locations in the Internet. The encompassed component types are overlay servers which act as overlay layer routers in the architecture, overlay access components that regulate the access to the overlay infrastructure and NIProxy instances which enable the execution of QoE improving operations on the last mile of the network connection. A schematic overview of the proposed end-to-end platform is depicted in Figure 1.

Overlay Routing Components

The essential function of the overlay routing network is providing a backup route when there is a problem with the quality of the direct Internet connection (De Vleeschauwer et al., 2008 ; De Vleeschauwer, De Turck, Dhoedt, & Demeester, 2009).

Overlay Server. A central component in the overlay routing network is the Overlay Server (OS). These components are present in several autonomous systems and organize into an overlay network. They maintain overlay routing tables which map target overlay server IP addresses to the next hop overlay server IP address. An overlay packet consists of a dedicated overlay header and a payload (see Figure 2). When such a packet is received by an overlay server, the server checks the target overlay server address of the packet and consults its routing table to determine the next overlay hop IP address. Subsequently, the packet is forwarded to this next hop. In addition to information on the target overlay server, the overlay header contains a field for the IP address of the overlay access component that the packet needs to be sent to and a field for the type of QoS the packet expects. This latter field allows to specify which type of treatment a specific packet should receive. For instance, some service may be very delay sensitive (e.g., VoIP), while others are highly intolerant to packet loss (e.g., streaming video). When the final overlay

server receives the packet, it will forward it to the access component at the target site which makes sure it will be received by the client.

The overlay servers maintain an overlay topology which contains information regarding the connectivity between pairs of overlay servers. Overlay server inter-connection quality is analyzed and estimated through active network probing. In particular, the overlay servers exchange ICMP echo messages with their neighbors in the topology. Based on the outcome of this probing, the quality of the connection is determined and values for delay and packet loss are deduced. This information is disseminated among overlay servers to make sure they all share an identical view on the connectivity in the entire overlay network. Based on the topology, the overlay routing tables are calculated. When a problem is detected in the topology, the overlay routing tables are updated and as a result one or more intermediate overlay hops will be used to relay the packets around the problematic parts of the network.

In Figure 3 the architecture of an overlay routing server is shown. It consists of a control and data plane. The control plane maintains the overlay topology and overlay routing tables. In addition, it coordinates the monitoring of the overlay edges and performs overlay management tasks such as adding new overlay servers to the network. The data plane on the other hand is responsible for actual packet handling and for forwarding packets to the next hop. By decoupling the control plane from the data plane it is possible to adjust the operation of the data plane without impacting the control plane. The implementation of the data plane is based on Java UDP sockets and draws from the Click modular router software (Kohler, Morris, Chen, Jannotti, & Kaashoek, 2000).

Overlay Access Component. The overlay Access Components (AC) detect QoS issues on the direct end-to-end IP path and enable applications to access the overlay routing service. The availability of these components eliminates the necessity to modify legacy applications to use the overlay routing service. Each end point of a connection is

associated with one AC. The AC can be deployed on or close to the actual end-device. For instance, an AC could be deployed on a residential gateway or could take advantage of the recent evolution of the access node towards an intelligent platform with the ability to host a variety of services (Gilon - de Lumley et al., 2007). The AC detects when a new connection becomes active and decides whether packets belonging to this connection should be sent via the direct Internet route or should be pushed to an overlay server for overlay dissemination. It is important to note that the overlay route will only be used in case of a problem with the direct path. This is done to prevent needless encumbrance of overlay resources. QoS problems are again detected by performing active probing. However, the connection quality could also be determined through passive monitoring using knowledge about the transport protocol. For instance, by sniffing packets of RTP/RTCP- or TCP-based services on an intermediate node, it is possible to deduce connection quality parameters (De Vleeschauwer et al., 2007 ; Simoens et al., 2007). When a QoS problem such as packet loss or high delay is detected on the direct IP path, the AC places a new overlay header in front of the packets belonging to the affected connection and pushes the packet to a nearby overlay server. This overlay server will then transmit the packet via an Internet route with better QoS properties, using a number of intermediate overlay hops, to the AC located near the target of the connection. As a final step, that AC will then remove the overlay header and transmit the packet to the actual target node. In this way, the connectivity between source and sink is preserved, while the usage of the overlay network remains transparent for the benefiting application.

Figure 4 depicts the architecture of the overlay access component. The prototype implementation makes use of the netfilter framework (Netfilter/iptables project homepage, 2009) to intercept packets and reinsert them in the network. It also consists of components which take care of the active monitoring of the individual connections and of the detection of new connections that could be candidates for overlay treatment. Finally,

a management module is also present. This module is responsible for selecting the overlay server which the AC will forward intercepted packets to in case they require overlay routing. Performance testing has shown that on a commodity 2 MHz AMD Opteron 2212 a total of more than 40.000 packets per second can be treated for 256 byte packets, corresponding to a bandwidth exceeding 80 Megabits per second (Mbps) (De Vleeschauwer et al., 2009). For a packet size of 1460 bytes, a bandwidth of more than 500 Mbps could be achieved. Because the AC can be located on the client device or on other hardware such as residential gateways and access nodes, the economical cost of using this component can be quite low. Due to the recent move towards providing more intelligence in the network (Gilon - de Lumley et al., 2007), no essential hardware changes should be made to the equipment and instead new services could be developed that use the access node service plane to provide the AC functionality.

In Figure 5, a scenario is illustrated where an edge in the network exhibits a degraded performance. More specifically, an IP link that is used on the direct path between overlay servers OS1 and OS4 suffers from congestion. This is responded to by the overlay servers by adjusting their routing tables accordingly. The problematic edge will also impact the communication session between clients 1 and 2. As a result, the ACs will transmit the traffic of these clients via the alternative overlay path. In particular, the failing IP link is circumvented by relaying network traffic between OS1 and OS4 through OS3 as an intermediate hop.

Network Intelligence Proxy

Besides overlay routing components, the overlay network also encompasses Network Intelligence Proxy (NIProxy) instances. These components are deployed at the edge of the network and aim to improve user QoE by deliberately managing last mile content delivery to clients (Wijnants & Lamotte, 2007). The NIProxy collects *intelligence* and exerts this

contextual knowledge to enhance the multimedia handling capabilities of transportation networks by outfitting them with two complementary traffic engineering techniques.

Context Introduction in the Network. The NIProxy is a context-aware proxy server which queries two distinct sources to fill up its context repository (Wijnants & Lamotte, 2007). The first source is the transportation network and the contextual knowledge in this case takes the form of quantitative network-related measurements and statistics such as the throughput, latency and error characteristics of communication links. This type of context is accumulated through performing active network probing and it makes the NIProxy aware of the currently prevailing access channel conditions. The second context source is the end-user application. Applications can provide the NIProxy with any application-related knowledge they deem relevant. As a result, the NIProxy's application-related context might vary depending on the kind of application under consideration. The importance or significance assigned by the user to different network flows (or flow aggregates, e.g., audio or video) incorporated by the application is an example of knowledge which could contribute to the NIProxy's application context.

As could have already been deduced from the discussion thus far, the NIProxy is not a transparent network intermediary. Clients need to explicitly connect to a NIProxy instance. In addition, the client software is responsible for responding to network probes and for providing application-related context. To facilitate these processes and to minimize the amount of required modification effort, a generic support library called the Network Intelligence Layer (NILayer) has been developed for inclusion in the application software. As is illustrated in Figure 6, the support library conceptually positions itself between the application and transport layers of the TCP/IP protocol stack and implements all aspects related to client-NIProxy communication. The NILayer for example exports an API allowing developers to easily provide the NIProxy with context pertaining to their application. Due to its generic design, the applicability and reusability

of the NILayer support library is maximized.

Automatic Client Downstream Bandwidth Management. The first QoE-improving mechanism supported by the NIProxy is automatic and dynamic client downstream bandwidth management (Wijnants & Lamotte, 2008). In particular, the NIProxy is capable of orchestrating the last mile bandwidth consumption of networked applications. Generally speaking, based on its network awareness the NIProxy prevents over-encumbrance of the client's access network connection, while its application-related context is exploited to create an intelligent allocation of the downstream bandwidth that is actually available. From an implementational point of view, the NIProxy's bandwidth management mechanism operates by arranging flows in a *stream hierarchy*, a tree-like structure which expresses the relationships that exist between network traffic introduced by distributed applications. In this stream hierarchy, actual network streams are represented by leaf nodes. Internal nodes on the other hand each implement a specific bandwidth distribution strategy and hence dictate the bandwidth brokering process by allocating bandwidth to their children in a particular manner.

The NIProxy's network traffic shaping process is controlled entirely by the choice of types of internal nodes to use and the way these nodes are composed to model the general layout of the stream hierarchy. Once this layout has been designed and assuming the NI stream hierarchy is kept up-to-date (e.g., newly initiated network flows are adequately incorporated), performing network traffic shaping simply amounts to appointing the correct bandwidth amount to the hierarchy root node. The internal nodes, commencing with the root node, will automatically start distributing the bandwidth value they have been assigned among their children according to the bandwidth distribution approach they implement. Eventually, portions of the total bandwidth capacity will reach one or more leaf nodes, at which point this bandwidth amount is reserved for the transmission of the network stream that is associated with the leaf node.

Different types of internal nodes are available to structure the stream hierarchy, each having distinct characteristics and capabilities. The experimental results presented later on in this paper have been generated using only the `WeightStream` internal node type. The `WeightStream` node implements a two-phase bandwidth distribution process and apportions bandwidth among its children as a function of their *weight value* and maximal bandwidth consumption. In mathematical notation, in the initial phase, each child c is assigned a bandwidth amount of

$$BW_c = w_c \times MaxBW_c \times \frac{BW}{\sum_{children} (w_{child} \times MaxBW_{child})}$$

where $w_c \in [0, 1]$ represents the child's weight value, $MaxBW_c$ indicates the amount of bandwidth which the child can maximally consume and BW denotes the total amount of bandwidth available to the `WeightStream` node. After executing this initial phase, a portion of BW might be left unallocated due to children not fully consuming their assigned bandwidth amount. If so, the `WeightStream` node starts the second phase of its bandwidth distribution process and assigns the excess bandwidth to child nodes on a one-by-one basis, in order of decreasing weight value. The second phase in other words allows children to upgrade their bandwidth consumption based on unused bandwidth inherited from phase 1, hereby favoring children with a higher weight value. Other types of internal nodes that have been implemented include `Mutex`, `Priority` and `Percentage` (Wijnants & Lamotte, 2008).

As is the case for internal stream hierarchy nodes, there also exist multiple leaf node types. The classification of leaf nodes is based on their capabilities to control the bandwidth consumption of the network stream they represent. Leaf nodes belonging to the *discrete* category are limited to toggling the bandwidth usage of their associated flow between a discrete number of levels. In its most rudimentary form, a discrete leaf defines only two such levels, corresponding to respectively a zero and maximal stream bandwidth

consumption. Such nodes are confined to turning their associated flow off or forwarding it at its maximal bitrate. In contrast, the class of *continuous* leaf nodes is able to modify their associated stream's bandwidth consumption in a continuous manner. Stated differently, these nodes define a bandwidth consumption range and provide functionality to set their corresponding flow's bandwidth usage to an arbitrary value within this range. Continuous leaf nodes are more powerful than discrete leaves as they enable additional flexibility and dynamism to be introduced in the network traffic shaping process. This however comes at the expense of increased resource consumption and processing requirements. In particular, contrary to discrete leaf nodes, the operation of continuous leaves typically requires the adoption of resource-intensive and possibly computationally complex techniques like, for instance, mid-stream buffering, real-time transcoding and/or dynamic rate control.

Multimedia Service Provisioning. Complementary to client downstream bandwidth management, the NIProxy also supports last mile QoE optimization through multimedia service provisioning (Wijnants & Lamotte, 2007). The NIProxy is in other words capable of applying processing on network flows containing multimedia content on behalf of its connected clients. Analogous to the network traffic shaping mechanism, services can query and exploit the NIProxy's contextual knowledge. The range of services that can be provided is theoretically limitless. Examples include the transcoding and transmoding of multimedia data, network flow encryption to increase security and privacy and services which increase the resilience of network traffic to transmission errors. Thanks to its service provision mechanism, the NIProxy is able to satisfy the growing adaptability and dependability requirements of emerging networked applications, since it allows the NIProxy to transform multimedia content according to, for instance, current channel conditions or hardware limitations of the user's terminal.

To guarantee maximal flexibility and to achieve run-time extensibility, the

NIProxy's multimedia service provision mechanism is implemented using a plug-in approach. Each service corresponds to a NIProxy plug-in that can be (un)loaded dynamically during NIProxy execution as needed. By opting for a plug-in based design, services become standalone entities that are conceptually separated not only from each other but also from the NIProxy's general software architecture. This separation confers several advantages. As an example, it facilitates third party service development as it eliminates the need for service implementers to have extensive knowledge of the NIProxy internals. Despite their isolation from each other, services are still able to cooperate. Multiple services, each implementing a well-delineated function, can namely be combined to form a service chain. Services belonging to a chain are applied consecutively so that the output produced by each service serves as input for the next service in the chain. Supporting service chaining in other words enables collaborative processing of single network streams by multiple independent NIProxy services. Finally, service chains can be personalized on a per-client client, yielding an increased amount of flexibility and enabling the NIProxy to efficiently satisfy each user's specific requirements and constraints.

Improved Last Mile QoE Optimization through Interoperation. A distinctive feature of the NIProxy is that interoperability between its both QoE-increasing mechanisms is supported. Instead of implementing the network traffic shaping and multimedia service provision mechanisms as isolated entities, the NIProxy allows them to interact and collaborate with each other. As an example, it is possible for NIProxy services to consult and even influence the bandwidth distribution strategies which the NIProxy draws up for clients. This in turn allows for the development of services having a very high level of performance as well as efficiency. More importantly, supporting interoperation unlocks end-user QoE optimization possibilities and results that could not be attained by applying both mechanisms independently.

H.264/AVC Transcoding Plug-in

As video is presumably the most challenging type of multimedia traffic, especially in terms of bandwidth requirements, the ability to alter the bitrate of video streams plays an essential role in our architecture to deliver end-to-end QoE. Many different approaches exist to actively change the bitrate of a coded video fragment. A straightforward solution is to first entirely decode and subsequently re-encode the video flow so that the output bitstream exhibits the desired properties (i.e., conforms to a certain target bitrate). The major drawback of this approach is that it is computationally very expensive as many calculations are performed unnecessarily (e.g., the re-calculation of the motion field). This is particularly true in the context of the complex H.264/AVC specification. A second class of techniques directly alter the properties of video streams in the compressed domain (i.e., without first performing a decoding step). Such techniques are generally referred to as transcoding algorithms.

In this paper we used an H.264/AVC transcoding algorithm specifically targeting bitrate reduction. The primary aim was to minimize the computational complexity and achieve a low delay, while maintaining a high visual quality. The transcoding algorithm operates entirely in the compressed domain and performs its calculations directly on the transformed coefficients. This way, only entropy decoding and encoding have to be performed. In order to lower the bitrate of the video stream, one or more transformed coefficients of a transformed and quantized 4×4 block are set to zero. This can be done based on a dynamically changing cut-off frequency or by setting a transform coefficient level threshold (also called *constrained* dynamic rate shaping or DRS (Eleftheriadis & Batra, 2006)). In the context of this paper, these are respectively called frequency-dependent DRS (FD-DRS) and level-dependent DRS (LD-DRS).

FD-DRS In (Sun, Kwok, & Zdepski, 1996), an architecture for bitrate adaptation is

achieved by removing the high-frequency coefficients in MPEG-2 bitstreams. This technique shows resemblance to a low-pass filter. All coefficients corresponding to frequencies lower than the defined cut-off frequency (f_c) are passed, whereas higher frequency coefficients are dropped. In MPEG-2, the variable-length decoder does not require any decoding capability and simply determines the codeword lengths. This parsing alone no longer suffices for FD-DRS in H.264/AVC. In order to parse the coefficients, it is necessary to decode the number of levels (and trailing ones) according to one of the predefined tables, which is selected in a context-adaptive way.

LD-DRS Besides the frequency-dependent DRS, an alternative method was analyzed which discards coefficients based on their magnitude. Because smaller coefficients have less impact on the reconstructed residual information, the error introduced by dropping these coefficients is limited. LD-DRS also avoids that large high-frequency coefficients are dropped in favour of smaller low-frequency coefficients. The LD-DRS method eliminates the coefficients in the 4×4 block of residual information which have an absolute value smaller than the defined threshold, t . Both the trailing ones and the other coefficients are removed by applying this selective transmission method. This results in bitrate savings caused by the elimination of trailing ones and regular coefficients, and obviously in a change of the number of coefficients and zeros.

Because of the many coding dependencies in H.264/AVC bitstreams, the transcoding modifications cause mismatches between transcoder and decoder which results in drift artifacts due to intra-prediction and motion compensation. As it turns out, removing coefficients in intra-coded pictures has a seriously negative impact on the quality of the entire Group Of Pictures (GOP). This effect is illustrated in Figure 7, where LD-DRS is applied once on P pictures only, and once on both I and P pictures. As can be seen, intra-coded pictures suffer severely from selective transmission, resulting in a

significantly lower quality of the entire GOP. We therefore do not apply transcoding on intra-coded frames to prevent the propagation of drift effects in time.

The configuration parameters for the FD-DRS and LD-DRS schemes have an impact on the output bitrate and the resulting video quality. In what follows, we discuss the results for the well-known Stefan test sequence (CIF resolution at 30Hz).

FD-DRS has proven to yield acceptable results for MPEG-1 and MPEG-2 bitstreams. The results for the H.264/AVC bitstream of the Stefan sequence are shown in Figure 8 for a wide range of quantization parameters. Dropping only the highest-frequency coefficients ($f_c = 12$) has little effect on the bitrate of the outgoing bitstream. A larger reduction of the bitrate is only possible for low QPs. However, this results in a quality loss of 1 to 2 dB for every additional skipped frequency. It is clear that FD-DRS is not able to lower the bitrate sufficiently for higher quantization parameters. This results from the fact that at larger quantization parameters, the residual coefficients become more and more concentrated at the lower frequencies. Hence, by using a frequency-dependent technique, the rate reduction becomes less effective when the QP increases. The only solution is to drop low-frequency coefficients, which are typically larger, resulting in a considerable quality loss. Another argument against FD-DRS is that all information is concentrated in 16 values, whereas in MPEG-2, 64 values were used. Hence, the loss of one coefficient results in a larger degradation than is the case in MPEG-2.

The results for LD-DRS are shown in Figure 9, again for the Stefan sequence. These show that dropping levels with an absolute value below a certain threshold results in a more explicit decrease of the bitrate, while the quality does not drop significantly faster than FD-DRS. The special case of dropping trailing ones is a good technique for lowering the bitrate by up to 30% at a minimal loss of quality. The advantage of LD-DRS is that the bitrate reduction remains large for a wide range of quantization parameters, whereas the rate reduction for FD-DRS becomes marginal at larger QPs.

When examining the measurement results in more detail, it becomes clear that for almost every targeted rate reduction, an LD-DRS threshold can be selected that results in a better PSNR than the nearest FD-DRS competitor. For example, if a rate reduction of 40 percent is desired for the Stefan sequence with quantization parameter 22, LD-DRS with $t = 1$ results in an increase of 3 dB when compared to FD-DRS with $f_c = 2$. Based on these observations, the LD-DRS method was implemented for the overall architecture.

In our implementation, the LD-DRS transcoding algorithm is steered by a rate control algorithm to ensure a desired target bitrate is achieved. Based on the current buffer occupancy, a bit budget for the current frame is estimated. For each frame, this target bit budget is approximated by dynamically adjusting the cut-off frequency. To determine the cut-off frequency, a proportional-integral-derivative (PID) controller is used. The parameters of this PID controller were empirically determined based on a test set of video sequences.

The transcoding algorithm with rate control was implemented and integrated in the overlay network as a plug-in for the NIProxy components. Consequently, these components are now able to dynamically set the desired bitrate of H.264/AVC bitstreams on the last mile. This in turn enables the management of H.264/AVC flows in the client's stream hierarchy using continuous leaf nodes, since these nodes require an accurate enforcing of the bandwidth budget they calculate for their associated network stream. As continuous leaf nodes add a considerable amount of flexibility to the NIProxy's bandwidth management capabilities, the H.264/AVC transcoding plug-in allows for the calculation of highly dynamic bandwidth distributions that can react both rapidly and adequately to context changes. The likely outcome is an improvement in the QoE provided to the user. Consequently, the H.264/AVC transcoding plug-in exemplifies the interoperation possibilities between services and the NIProxy's network traffic shaping mechanism and the resulting positive influence such collaboration can have on the user QoE.

Evaluation

Experimental Setup

To evaluate the overlay network's added value in terms of user QoE optimization, a testbed was constructed. Figure 10 displays the physical layout of this testbed as well as the network topology it conceptually translated to. In total, the testbed consisted of 10 PCs running the GNU/Linux operating system and encompassed three overlay servers, two nodes on which both an overlay access component and a NIPROXY instance were deployed, two multimedia clients and one streaming video server. To emulate varying network conditions, the testbed in addition included two Click impairment nodes (Kohler et al., 2000). These nodes served a dual purpose as they were employed to artificially introduce random packet loss in the core network as well as to enforce bandwidth restrictions on the last mile. To evaluate the overlay platform, communication sessions between the server and both multimedia clients were set up. However, the connection of only one of the clients was protected by the overlay and proxy components.

Experiment 1: Mitigating Network Core Impairments

In a first test, an arbitrary video sequence was streamed from the multimedia server to each of the two clients. Approximately 43 seconds after the start of the experiment, random packet loss was activated in the core network using the Click impairment node. This resulted in the communication session of both clients suffering from lost packets, which in turn yielded video decoding issues and hence a degraded video playback on the clients' screens. However, after a few seconds, the overlay network detected the problem and automatically began bypassing the lossy network link by routing via an alternative overlay path. In Figure 11, the packet loss per second that was experienced by both clients is shown. One can see that as soon as the overlay network started rerouting, the packet loss no longer occurred for the protected connection. The unprotected client on the

other hand continued to suffer from a degraded performance, as its packets were not rerouted and hence continued to pass through the problematic part of the network core.

Experiment 2: Last Mile QoE Optimization

To evaluate the NIProxy component and its H.264/AVC transcoding plug-in, a second experiment was performed. This experiment involved the simultaneous streaming of two H.264/AVC video flows to both multimedia clients while their last mile network connection was artificially impaired in terms of available downstream bandwidth. The goal of this experiment was hence to investigate how the NIProxy reacted to this bandwidth impairment and whether it was able to improve the user QoE by exploiting its H.264/AVC transcoding plug-in to dynamically and intelligently adapt the involved video flows so that their bitrate matched the last mile bandwidth capacity. The principal characteristics of the employed video sequences are listed in Table 1.

The experiment itself can conceptually be divided in 5 consecutive intervals, where each interval transition was triggered by a certain state change. In particular, the switch from the first to the second interval was caused by the introduction of the second H.264/AVC flow in the experiment, whereas the transitions from interval 2 to 3 and from interval 3 to 4 were initiated by shifts in H.264/AVC stream importance (in the experiment, users were able to alter stream importance through the GUI of the client software). Finally, in the last interval, the last mile impairment was moderated so that the clients suddenly disposed of an increased amount of downstream bandwidth.

A network trace, illustrating all H.264/AVC network traffic received by the overlay-enhanced client during the experiment, is depicted in Figure 12, whereas the stream hierarchy which formed the basis for the NIProxy to calculate this bandwidth distribution is provided in Figure 13. Due to the constrained nature of the experiment, the stream hierarchy consisted of only 3 nodes and had a very straightforward structure.

In particular, the root of the hierarchy consisted of an internal node of type `WeightStream`, which was used to differentiate between the two H.264/AVC flows involved in the experiment. Both these flows were represented by a continuous leaf node. The figure also illustrates the weight values that were associated with both leaf nodes during the different experiment intervals. In this case, the weight values were determined so that they directly reflected the relative importance of the involved H.264/AVC flows, as specified by the user. Finally, as explained in the previous Section, the H.264/AVC transcoding plug-in was exploited to dynamically transcode the video streams to the target bitrates calculated by their associated continuous leaf node.

Looking at the network trace, we see that during the initial interval only one H.264/AVC flow was present and that there was no need for transcoding since sufficient last mile bandwidth was available to forward the flow to the client at its maximal quality. This situation changed with the introduction of the second H.264/AVC flow in the second interval. Since both flows had identical weight values and comparable maximal bitrates, they were assigned a comparable bandwidth budget by the NIProxy's bandwidth distribution mechanism (i.e., approximately 62 KiloBytes per second (KBps)). At the beginning of the two subsequent intervals, the user incremented the importance value of the first H.264/AVC flow. This resulted in this stream receiving an increased share of the available downstream bandwidth, at the expense of the second H.264/AVC flow which now needed to be transcoded to a lower bitrate. Finally, despite its lower importance value, the additional last mile bandwidth that became available during the last interval was exploited to upgrade the quality of the second H.264/AVC stream, since the first flow was already being forwarded to the client at its maximal bitrate.

A number of important findings can be deduced from the presented experimental results. A first observation is that the NIProxy, by leveraging its network awareness, ensured that the downstream capacity of the last mile was roughly respected throughout

the entire experiment. Consequently, last mile congestion was avoided, resulting in a minimization of packet loss and delay and hence an optimal reception of the forwarded flows at client-side. Secondly, the produced bandwidth distribution correctly captured the relative importance of the involved H.264/AVC streams (i.e., more bandwidth was assigned to more significant streams). This result can be attributed to the NIProxy's application-related contextual knowledge. Finally, as is demonstrated in the network trace in Figure 12, the H.264/AVC transcoder succeeded in fairly accurately enforcing the bandwidth amounts allocated to the video flows involved in the experiment. The availability of the H.264/AVC transcoding service consequently allowed the NIProxy to adapt video streams to a continuous range of bitrates, this way enabling an optimal and full exploitation of the bandwidth available on the last mile.

The findings described in the previous paragraph did not apply for the non overlay-enhanced client. Since this client lacked management of its last mile downstream bandwidth and also could not profit from H.264/AVC transcoding functionality, its user witnessed a usage experience that was far from optimal. In particular, the user did not dispose of any mechanisms to differentiate between network streams. In addition, a substantial amount of packet loss was detected due to the client's last mile connection being flooded with more H.264/AVC data than it could transfer. This in turn resulted in perceptual artifacts caused by decoding issues and hence a severely deteriorated video playback at client-side.

Discussion

The results produced during the described experiments comprehensively demonstrate the advantages of the proposed overlay network. In particular, they exemplify that the overlay platform is capable of (i) successfully countering impairments in the core of the network (in this case packet loss) and (ii) intelligently allocating the bandwidth

available on the last mile. For reasons of clarity and intelligibility, these features were presented using separate experiments. They are however not mutually exclusive and can hence just as well be achieved simultaneously, meaning the proposed overlay network supports user QoE optimization on the entire delivery path from content source to sink.

Related Work

In previous research, the usage of overlay network technology for enhancing network routing has already been looked at. The RON project (Andersen, Balakrishnan, Kaashoek, & Morris, 2001) describes a system for routing around network failures in which all involved parties deploy an overlay server. The main difference with our work is that we offer an overlay solution that is able to give end-users access to the overlay network without requiring them to run overlay servers themselves. This results in a more scalable approach than the RON, which is limited to 50 overlay sites. In (De Vleeschauwer, De Turck, Dhoedt, & Demeester, 2004), we have described algorithms for determining the optimal locations for the overlay servers and (De Vleeschauwer, De Turck, Dhoedt, & Demeester, 2006) introduces a number of algorithms for the management of the overlay topology.

Both mechanisms provided by the NIProxy to optimize user QoE on the last mile are also topics of active research. Interesting work in the context of client bandwidth management includes the GARA architecture (Foster, Fidler, Roy, Sander, & Winkler, 2004), the Exact Bandwidth Distribution Scheme (X-BDS) (Hnatyshin & Sethi, 2006) and the bandwidth sharing algorithm presented in (Anjum & Tassiulas, 1999). The NIProxy's multimedia service provision mechanism on the other hand is largely related to the Service Oriented Architecture (SOA) paradigm and hence shares many of its underlying principles and features (Web Services and Service-Oriented Architectures, 2009). Examples of platforms and frameworks adhering to this paradigm abound in the literature; see, for

instance, the research by Klara Nahrstedt (e.g., (Nahrstedt, Yu, Liang, & Cui, 2005)) and the MobiGATE system (Zheng, Chan, & Ngai, 2006). Finally, the NIProxy also shows substantial interfaces with architectures that are concerned with QoS provision like, for instance, the Congestion Manager architecture (Andersen, Bansal, Curtis, Seshan, & Balakrishnan, 2000). As a result, the NIProxy does not innovate in terms of the QoE-improving mechanisms it provides. What does distinguish the NIProxy from other approaches is that it integrates these mechanisms in a single system and in addition does so in an interoperable and collaboration-enabled manner. Also, the NIProxy's awareness includes application-related context, a type of knowledge that is often left unconsidered in related systems.

Techniques altering the properties of video streams without performing full decoding and subsequent re-encoding are generally called transcoding algorithms. For single-layer video flows, transcoding can be applied to adapt the bitrate, resolution or frame rate. In this paper, we have focused on bitrate reduction (SNR transcoding) of H.264/AVC streaming video. Two main classes of transcoding methods for bitrate reduction exist: dynamic rate shaping (DRS) and requantization transcoding. For MPEG-2, the former was investigated in (Eleftheriadis & Batra, 2006), whereas the latter is discussed in (Sun et al., 1996). However, the techniques developed in these previous works cannot readily be used for H.264/AVC. As low complexity and low delay introduction were crucial in the context of this paper, we opted to use a slightly modified version of DRS suited for real-time H.264/AVC transcoding.

Conclusions and Future Work

We have presented an overlay network supporting full end-to-end QoE management. The proposed platform consists of components in the network core, which improve the standard network routing service, as well as components at the edge which achieve an

optimization of the bandwidth consumption on the last mile. We discussed how the overlay network was extended with real-time H.264/AVC transcoding functionality and we have investigated the impact hereof on the platform's QoE optimization capabilities. In particular, using experimental results we have demonstrated that this addition enables the overlay network to more optimally and fully exploit the bandwidth available on the last mile, which in turn results in a further improvement of the QoE provided to the end-user.

Our primary topic of future work is an investigation of the scalability of the proposed system. Secondly, we will extend the work to generic overlay networks and will investigate the integration of service hosting and resource discovery mechanisms in the presented platform. Thirdly, we plan to look into the possibility to also apply the NIProxy's network traffic shaping and service provisioning functionality in the upstream direction and to examine the implications hereof on our platform's QoE optimization options. Finally, organizing user studies might yield valuable qualitative feedback regarding the QoE optimization operations executed by the overlay network.

Références

- Andersen, D., Balakrishnan, H., Kaashoek, F., & Morris, R. (2001, October). Resilient Overlay Networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP 2001)* (p. 131-145). Banff, Canada.
- Andersen, D., Bansal, D., Curtis, D., Seshan, S., & Balakrishnan, H. (2000, October). System Support for Bandwidth Management and Content Adaptation in Internet Applications. In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI 2000)* (p. 213-226). San Diego, California, USA.
- Anjum, F. M., & Tassiulas, L. (1999, March). Fair Bandwidth Sharing among Adaptive and Non-Adaptive Flows in the Internet. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 1999)* (p. 1412-1420). New York, USA.
- De Vleeschauwer, B., De Turck, F., Dhoedt, B., & Demeester, P. (2004, September). On the Construction of QoS Enabled Overlay Networks. In *Quality of Future Internet Services (QofIS 2004)* (Vol. 3266, p. 164-173). Barcelona, Spain.
- De Vleeschauwer, B., De Turck, F., Dhoedt, B., & Demeester, P. (2006). Dynamic Algorithms to Provide a Robust and Scalable Overlay Routing Service. In *Proceedings of the 20th IEEE International Conference on Information Networking (ICOIN 2006)* (pp. 945–954). Sendai, Japan.
- De Vleeschauwer, B., De Turck, F., Dhoedt, B., & Demeester, P. (2009). Dynamic Overlay Networks for Robust and Scalable Routing. In *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications (Accepted)*.
- De Vleeschauwer, B., De Turck, F., Dhoedt, B., Demeester, P., Wijnants, M., & Lamotte, W. (2008, January). End-to-end QoE Optimization Through Overlay Network Deployment. In *Proceedings of the 22nd IEEE International Conference on Information Networking (ICOIN 2008)*. Pusan, Korea.

- De Vleeschauwer, B., Simoens, P., Van de Meerssche, W., De Turck, F., Dhoedt, B., Demeester, P., et al. (2007, May). Enabling Autonomic Access Network QoE Management Through TCP Connection Monitoring. In *Proceedings of the 1st IEEE Workshop on Autonomic Communications and Network Management (ACNM 2007)*. Munich, Germany.
- Eleftheriadis, A., & Batra, P. (2006, April). Dynamic Rate Shaping of Compressed Digital Video. *IEEE Transactions on Multimedia*, 8(2), 297-314.
- Foster, I., Fidler, M., Roy, A., Sander, V., & Winkler, L. (2004). End-to-End Quality of Service for High-End Applications. *Computer Communications*, 27(14), 1375-1388.
- Gilon - de Lumley, E., Hoet, J., Dequeker, H., Huysegems, R., Six, E., Van de Meerssche, W., et al. (2007, December). Service Rich Access Networks: The Service Plane Solution. In *Proceedings of BroadBand Europe*. Antwerp, Belgium.
- Hnatyshin, V., & Sethi, A. S. (2006, September/October). Architecture for Dynamic and Fair Distribution of Bandwidth. *International Journal of Network Management*, 16(5), 317-336.
- Kohler, E., Morris, R., Chen, B., Jannotti, J., & Kaashoek, M. F. (2000, August). The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3), 263-297.
- Nahrstedt, K., Yu, B., Liang, J., & Cui, Y. (2005, March). Hourglass Multimedia Content and Service Composition Framework for Smart Room Environments. *Elsevier Journal on Pervasive and Mobile Computing*, 1(1), 43-75.
- Netfilter/iptables project homepage. (2009, July). (<http://www.netfilter.org/>)
- Simoens, P., De Vleeschauwer, B., Van de Meerssche, W., De Turck, F., Dhoedt, B., Demeester, P., et al. (2007, July). RTP Connection Monitoring for Enabling Autonomous Access Network QoS Management. In *Proceedings of the 11th European Conference on Networks and Optical Communications (NOC 2007)*. Stockholm, Sweden.

- Sun, H., Kwok, W., & Zdepski, J. (1996, April). Architectures for MPEG Compressed Bitstream Scaling. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(2), 191-199.
- Web Services and Service-Oriented Architectures. (2009, July).
(<http://www.service-architecture.com/>)
- Wijnants, M., & Lamotte, W. (2007, June). The NIProxy: a Flexible Proxy Server Supporting Client Bandwidth Management and Multimedia Service Provision. In *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007)*. Helsinki, Finland.
- Wijnants, M., & Lamotte, W. (2008, January). Managing Client Bandwidth in the Presence of Both Real-Time and non Real-Time Network Traffic. In *Proceedings of the 3rd IEEE International Conference on COMMunication System softWARE and MiddlewaRE (COMSWARE 2008)*. Bangalore, India.
- Zheng, Y., Chan, A. T. S., & Ngai, G. (2006, September-October). Applying Coordination for Service Adaptation in Mobile Computing. *IEEE Internet Computing*, 10(5), 61-67.

Author Note

Part of the presented work is funded through the IBBT QoE and Gr@sp projects and the FWO Videostreaming project.

	Video1 (V1)	Video2 (V2)
resolution (pixels)	CIF (352x288)	CIF (352x288)
framerate (FPS)	25	25
GOP-size	25	25
bitrate (KBps)	82	90

Table 1

Video sequence parameters for the second experiment.

Figure Captions

Figure 1. Architectural overview of the proposed overlay network.

Figure 2. Overlay header format.

Figure 3. Overlay routing server architecture.

Figure 4. Overlay access component architecture.

Figure 5. Example overlay routing scenario. Both the routing tables of the overlay servers and the routing information maintained by the ACs is shown. The former relate target OS to next hop OS, while the latter maps destination IP address to overlay AC and OS at the remote side.

Figure 6. Client-NIProxy communication through the NILayer support library.

Figure 7. Impact of DRS on intra-coded pictures (LD-DRS, $t=1$, $QP = 22$).

Figure 8. Bitrates and quality for FD-DRS (Stefan sequence).

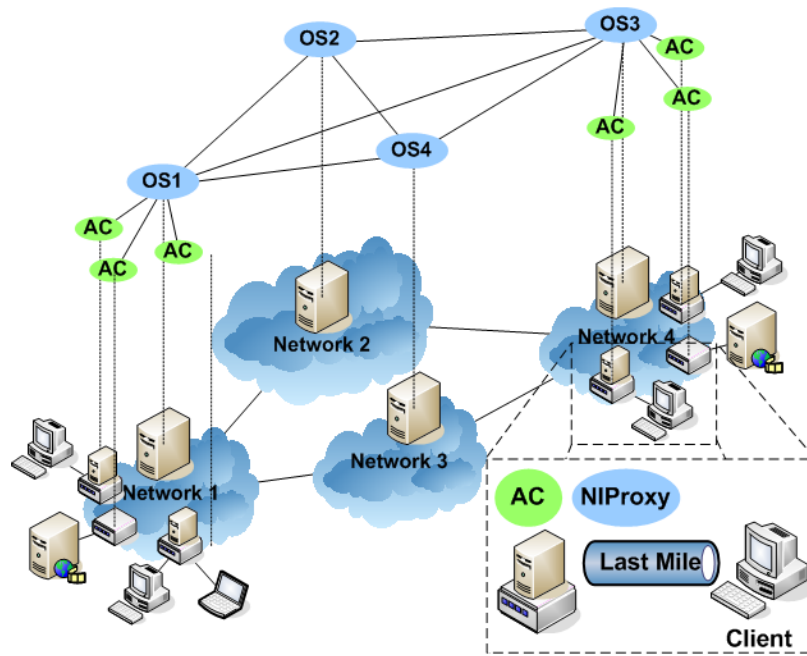
Figure 9. Bitrates and quality for LD-DRS (Stefan sequence).

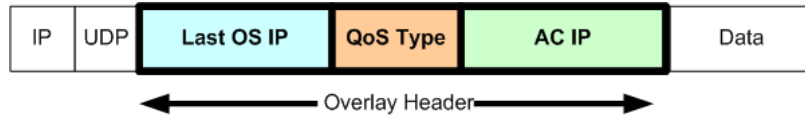
Figure 10. Experimental setup: Physical layout of the overlay platform testbed and the network topology it emulated.

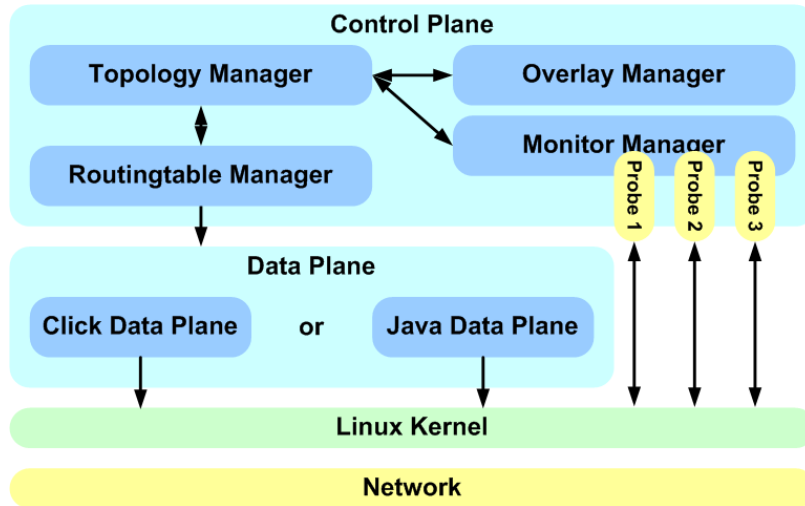
Figure 11. Experiment 1: Packet loss ratio, with and without overlay routing.

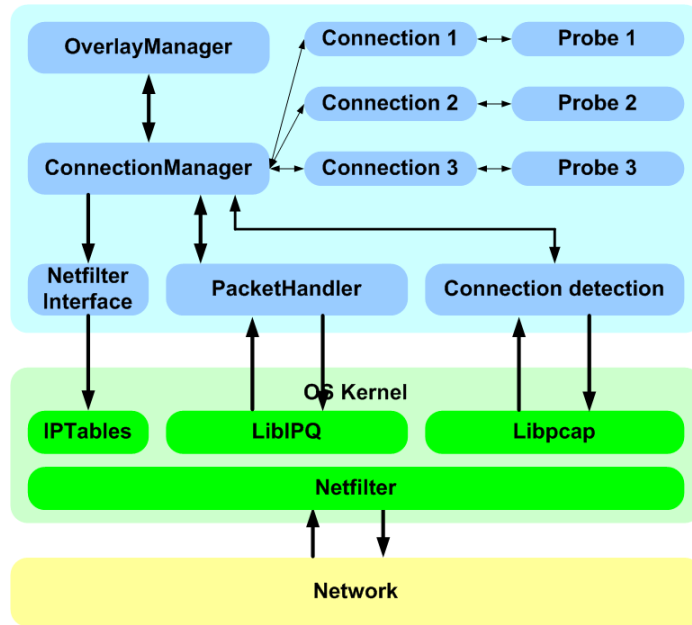
Figure 12. Experiment 2: Stacked graph illustrating all H.264/AVC data received by the overlay-enhanced client, in KBps.

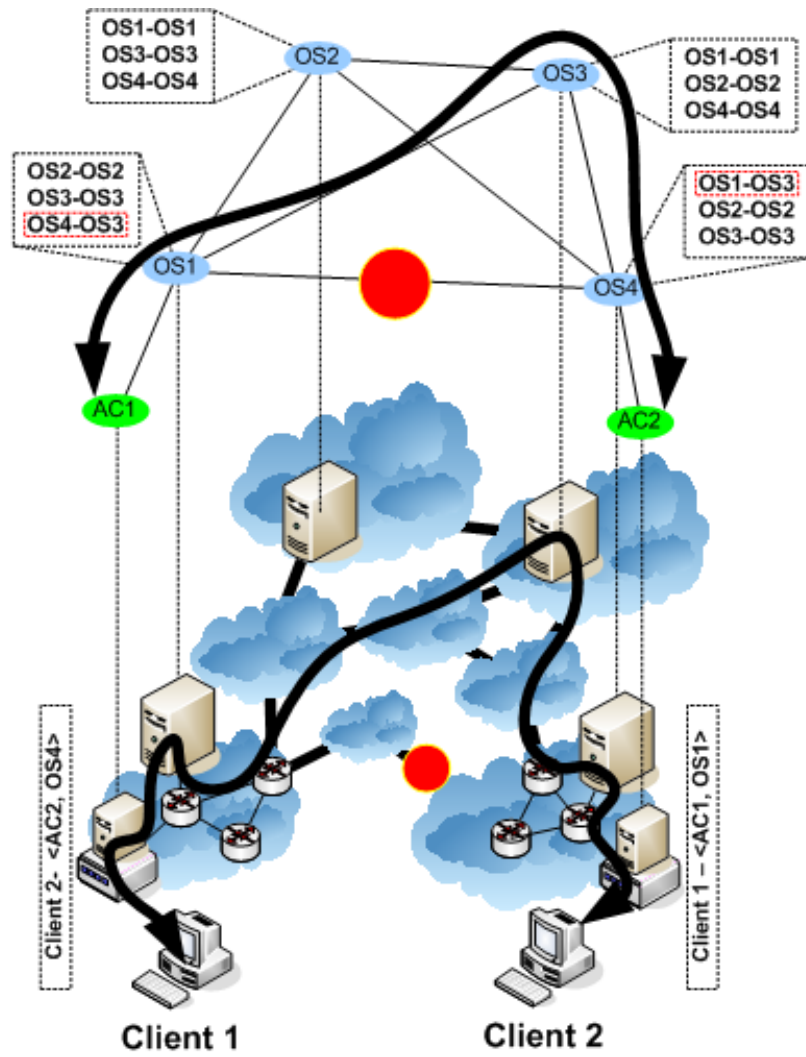
Figure 13. Experiment 2: Stream hierarchy which directed the bandwidth distribution.










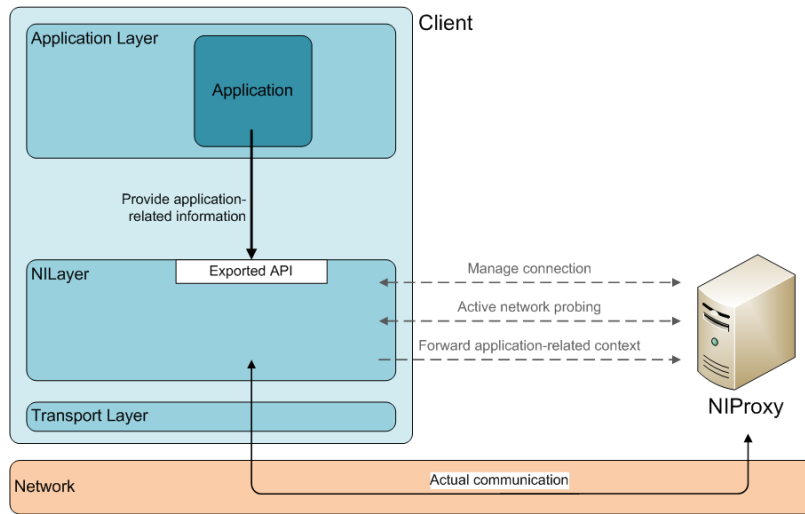


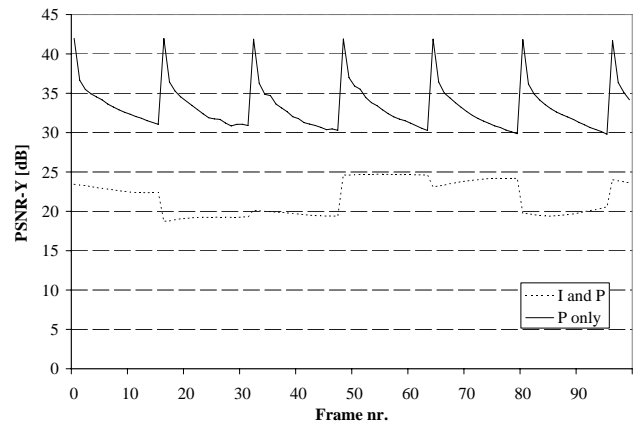


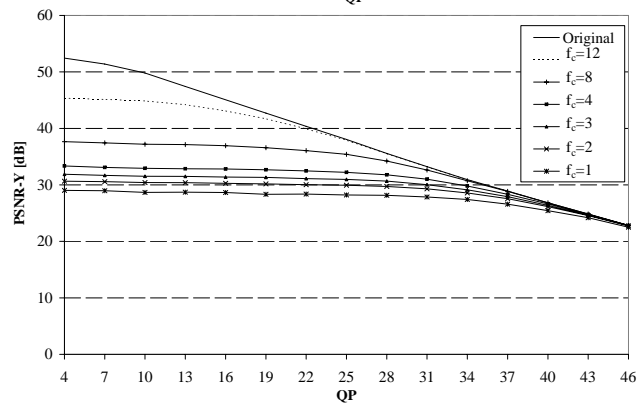
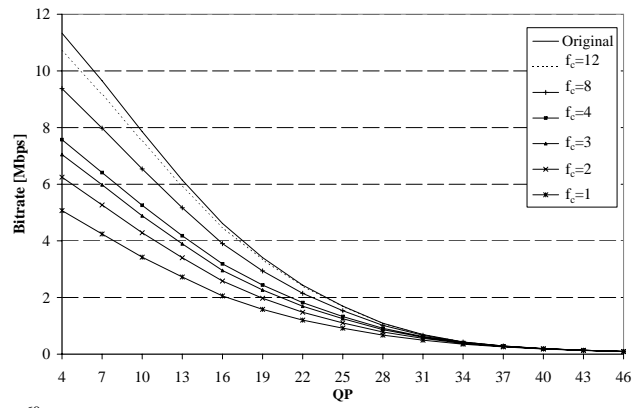
-  Overlay Server
-  Overlay Access Component
-  Network Problem

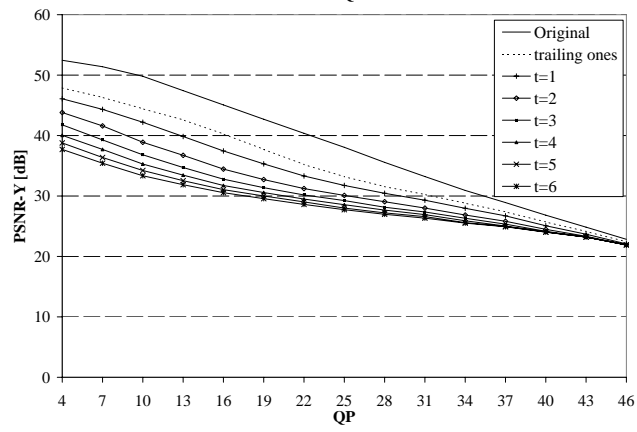
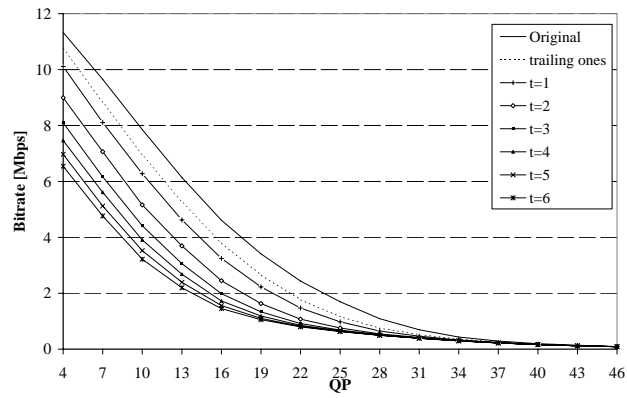
Tar. OS - Next Hop OS
 ...
 Tar. Client - <AC, Last OS>


OS
 Routing Table
 AC
 Routing Info






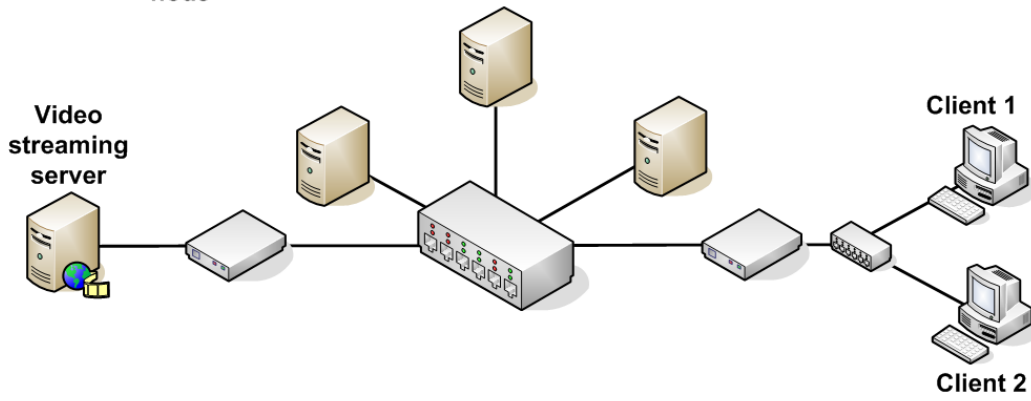




 Overlay access component and NIProxy

 Overlay server

 Click impairment node



 Overlay access component and NIProxy

 Overlay server

 Network router

