Optimizing User QoE 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
{maarten.wijnants,wim.lamotte}@uhasselt.be

Bart De Vleeschauwer, Filip De Turck, Bart Dhoedt and Piet Demeester
IBCN, Department of Information Technoglogy, Ghent University – IBBT
Gaston Crommenlaan 8/201, BE-9050 Ledeberg-Ghent, Belgium
{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
{peter.lambert,dieter.vandewalle,jan.decock,stijn.notebaert,rik.vandewalle}@ugent.be

Abstract

More and more, multimedia services are being accessed via fixed and mobile networks. 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 bit rate. 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.

1. Introduction

In recent years, we have witnessed an expansive growth in the networked access of multimedia services. Compared to traditional services, like web browsing and e-mail, these services impose much stricter requirements on the transportation network in terms of packet loss, congestion, delay and jitter. For instance, interactive applications such as VoIP and online gaming require a low delay to guarantee a fluid operation. As another example, packet loss has severe consequences for video streaming services since it will rapidly degrade playback at receiver-side due to the introduction of visual artifacts. Complicating matters even further is the fact that, due to the recent popularization of 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.

Unfortunately, the current generation of networks is 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 throughput, packet loss and delay that will be experienced by network packets. The access part of a client's network connection can also cause severe problems, mainly due to its limited bandwidth capacity. In particular, compared to the core of the network, the so-called last mile is usually much less capacitated. As a result, insufficient last mile bandwidth may be available to support all services which a client is currently using (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 believe current networks are often unable to provide users of multimedia services with an acceptable usage experience or, more formally, Quality of Experience (QoE). In our previous work we therefore proposed an overlay platform which supports full end-to-end user QoE optimization by employing a two-tier approach [4]. On the one hand, the proposed architecture enhances data dissemination in the network core by providing an overlay routing service which ensures re-

978-1-4244-2100-8/08/$25.00 2008 IEEE

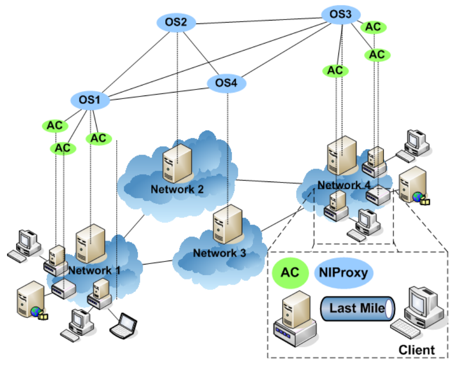

Figure 1. Architectural overview of the proposed overlay network.

silience to possible issues like failing or congested network links. On the other hand, proxy servers deployed close to end-users provide functionality to intelligently apportion bandwidth among network flows on the last mile and in addition act as a multimedia 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 bit rate 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 application possibilities of the transcoder to off-line scenarios. The transcoding service therefore operates entirely in the compressed domain to enable real-time video transformation. A secondary contribution of this paper is an illustration of how the capabilities of the proxy component's bandwidth management functionality and the novel H.264/AVC transcoding service can be bundled to produce highly dynamic and flexible bandwidth management results. As will be illustrated using representative experimental results, the outcome of this collaboration is an even further optimization of the QoE provided to the user.

The outline of the remainder of this paper is as follows. In section 2 the full architecture of the proposed overlay network is described. Section 3 harbors a thorough evaluation of our platform and demonstrates its ability to improve user QoE. In section 4 a brief overview of related work is presented. Finally, conclusions are drawn in section 5.

2. End-to-end architecture

A schematic representation of the proposed overlay network is depicted in figure 1. The end-to-end infrastructure consists of three main components: overlay servers, overlay access components and NIProxy instances.

2.1. Overlay routing components

The overlay servers (OSs) are located in several Autonomous Systems in the Internet and sustain connectivity between each other. To do this, they maintain overlay routing tables that map target OS IP addresses to the next hop OS IP address. When an overlay packet arrives at an overlay server, it consults its routing table to determine the next overlay hop IP address and sends the packet to the next OS. This process is repeated until the target overlay server is reached. The overlay packets contain an overlay header with information on the target overlay server, the overlay access component that a packet needs to be sent to and a field for the type of QoS the packet expects. This header is inserted between the UDP header and the packet payload. If the final overlay server receives a packet, it will forward it to the access component which is responsible for further handling it (see next paragraph). To construct the routing tables, the OSs maintain an overlay topology that contains information on the connectivity between pairs of overlay servers. This information is obtained by performing active monitoring and the results of this monitoring are exchanged between the servers.

While the overlay servers maintain a resilient overlay routing network, extra components are required to give end-users access to this service, since we cannot assume all end-users deploy overlay servers themselves. Therefore, we designed and implemented an overlay network access component (AC). These components are deployed on or close to the end-device and are responsible for determining when traffic is actually sent to the overlay servers. The rationale behind this is that it is not always necessary to use the overlay resources. More specifically, when there is no problem on the direct path between source and destination, the overlay network should not be used. The AC therefore monitors the quality of the connection. In case a problem with QoS is detected or if the AC notices that the connectivity to the destination is lost, it will send the packets to the nearest overlay server, which will subsequently forward it to an AC close to the target node. In this way, the connectivity between source and sink is maintained, while access to the overlay network remains transparent for the service itself. For a more detailed overview of the overlay routing network, the reader is referred to [4].

2.2. Network Intelligence Proxy

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 intelligently managing content delivery to clients over the last mile of their network connection [12]. As its name implies, the NIProxy attempts to accomplish this objective by incorporating different types of *awareness* or *context* in the transportation network. Based on this contextual knowledge, the NIProxy performs last mile QoE optimization in two complementary ways, namely through automatic client downstream bandwidth management as well as multimedia service provisioning.

2.2.1. Awareness introduction in the network

The NIProxy is a context-aware proxy server whose contextual information is at the moment twofold and comprises both network- and application-related knowledge [12]. The NIProxy's *network awareness* encompasses information regarding the state of the transportation network and, in particular, the current access channel conditions. Its *application awareness* on the other hand consists of knowledge of the application(s) clients are currently running and can hence vary depending on the kind of application(s) under consideration. In contrast to the network context, which the NIProxy collects through active network probing, the application awareness needs to be provided by the client software. To facilitate this process and to reduce the amount of modification required to the client software, a generic and hence reusable support library is provided which enables developers to provide the NIProxy with context pertaining to their application with minimal effort. The importance or significance assigned by the client to different network flows (or types of flows, e.g. audio or video) is an example of knowledge which could contribute to the NIProxy's application awareness.

2.2.2. Automatic client downstream bandwidth management

The first QoE-improving mechanism supported by the NIProxy is automatic and dynamic client downstream bandwidth management [13]. 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 awareness 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 constructing and maintaining a *stream hierarchy*, a tree-like structure that is composed of both internal and leaf nodes. The internal hierarchy nodes implement a certain bandwidth distribution strategy, whereas the leaf nodes always correspond to an actual network flow.

Different types of internal stream hierarchy nodes are available, each having distinct characteristics and capabilities. The `WeightStream` internal node, for instance, apportions bandwidth among its children in two consecutive phases. More specifically, in the initial phase, each child c_i receives

$$BW_i = w_i * MaxBW_i * \frac{BW}{\sum_i (w_i * MaxBW_i)}$$

bandwidth, where $w_i \in [0, 1]$ represents the child's current weight value, $MaxBW_i$ indicates the amount of bandwidth which the child can maximally consume and BW denotes the total amount of bandwidth available to the `WeightStream` node. In case any bandwidth remains unused after the execution of the first phase, this excess bandwidth is exploited in the second phase to allow children, in order of decreasing weight value, to switch to a higher bandwidth consumption level.

As is the case for the internal nodes, there also exist different categories of hierarchy leaf nodes. In particular, both *discrete* and *continuous* leaf nodes are available. Whereas discrete leaf nodes are limited to toggling the bandwidth consumption of their associated network flow between a discrete number of values (i.e. they can turn the stream off or can forward it to the client at maximal quality, or at any intermediate bandwidth consumption level supported by the stream), their continuous counterparts are capable of forwarding network flows to the client at any rate in the continuous interval [0, maximal stream bandwidth usage]. As a result, continuous hierarchy leaf nodes allow for a higher degree of flexibility in managing the client's downstream bandwidth, however at the expense of increased computational complexity: in contrast to discrete leaf nodes, the operation of continuous leaf nodes typically requires the adoption of advanced and possibly computationally intensive techniques like, for instance, mid-stream buffering, real-time transcoding and/or dynamic rate control.

By adequately constructing the stream hierarchy, it is possible to express relationships between network flows or, conversely, to differentiate between them (or between collections of flows, e.g. audio versus video). Once the stream hierarchy has been composed and presuming it is updated correctly over time, managing client downstream bandwidth simply amounts to allocating the correct amount of bandwidth to the hierarchy root node. The stream hierarchy's internal nodes, starting with the root node, will subsequently commence apportioning this bandwidth according to the particular bandwidth distribution technique each implements. Eventually, portions of the client's available downstream bandwidth will reach one or more hierarchy leaf nodes, which will use the bandwidth they are allocated to forward their associated network flow to the client.

2.2.3. Multimedia service provisioning

Complementary to client downstream bandwidth management, the NIProxy also supports last mile QoE optimization through multimedia service provisioning [12]. The

NIProxy is in other words capable of applying processing on network flows containing multimedia content on behalf of its connected clients. Like the bandwidth management mechanism, services can query and exploit the NIProxy's dual awareness. Example services that can be provided include the transcoding and transmoding of multimedia data, network flow encryption to increase security and privacy, services which increase the resilience of network traffic to transmission errors, and so on. 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 terminal hardware limitations.

The NIProxy's multimedia service provision functionality is implemented using a plug-in approach: each service corresponds to a NIProxy plug-in that can be (un)loaded dynamically as it becomes needed (obsolete). The NIProxy supports the notion of service chaining, meaning consecutive application of different services on the same network flow is possible. As a result, collaborative processing of a single network stream by multiple, independent services is enabled. Furthermore, per-client personalization of the service chain is also supported, yielding an increased amount of flexibility and enabling the NIProxy to efficiently fulfill each user's specific requirements and constraints.

2.2.4. Improved last mile QoE optimization through interoperation

An important feature of the NIProxy is that interoperability between its both QoE-increasing mechanisms is supported. Instead of implementing the bandwidth management and 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. In addition, the supported interoperation enables QoE optimization to an extent that could not be attained by applying the two mechanisms independently.

2.3. 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 bit rate of video streams plays an essential role in our architecture to deliver end-to-end QoE. Many different approaches exist to actively change the bit rate of a coded video stream. A straightforward solution is to entirely decode and re-encode the video flow so that the output bitstream has the desired properties (i.e. conforms to a certain target bit rate). However, this approach is computationally very expensive as many unnecessary calculations are performed (e.g. the re-calculation of the motion field). This is particularly true in the context of the complex H.264/AVC specification.

In this paper we used an H.264/AVC transcoding algorithm specifically targeting bit rate reduction. The primary aim was to minimize the computational complexity and to achieve low delay, while maintaining high visual quality (compared to the cascade of decoder and encoder). The transcoding algorithm operates entirely in the compressed domain and performs the calculations directly on the transformed coefficients. This way, only entropy decoding and encoding have to be performed. In order to lower the bit rate of the video stream, one or more transformed coefficients of a transformed and quantized $4{\times}4$ block are set to zero. This can be done based on a dynamically changing cut-off frequency (also called *constrained* dynamic rate shaping). Because of the many coding dependencies in H.264/AVC bitstreams, these modifications cause mismatches between transcoder and decoder which results in drift artifacts due to intra prediction and motion compensation. The transcoding algorithm is therefore not applied to intra-coded frames in order to limit drift effects in the course of time.

The transcoder is steered by a rate control algorithm to ensure a desired target bit rate 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 bit rate 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. Since 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 is an example illustration of how services and the NIProxy's bandwidth management mechanism are able to interoperate and the resulting positive influence such collaboration can have on the user QoE.

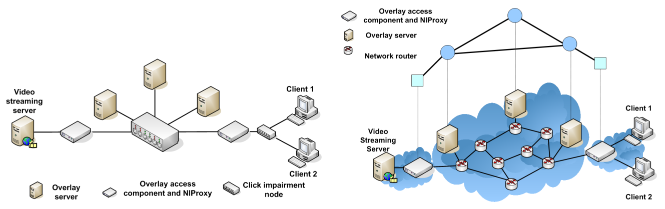

Figure 2. (left) Physical layout of the overlay platform testbed; (right) Network topology it conceptually translates to.

3. Evaluation

3.1. Setup

To evaluate how the overlay network can optimize the QoE, a testbed was constructed (see figure 2). This testbed consists of 10 Linux PCs and contains three overlay servers, two nodes running both an overlay access component and a NIProxy instance, two multimedia clients and one streaming video server. To emulate varying network conditions, Click impairment nodes [8] were used. In particular, we used these nodes 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.

3.2. 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. After some time, random packet loss was activated in the core network. This resulted in the communication session of both clients suffering from lost packets, which in turn yielded visual artifacts on the clients' screens. However, after a few seconds, the overlay network detected the problem and automatically decided to avoid the lossy network link by routing via an alternative overlay path. In figure 3, the packet loss per second that was experienced by the two clients is shown. One can see that as soon as the overlay network started rerouting, the packet loss no longer occurred. The unprotected client on the other hand continued to suffer from a degraded performance.

3.3. 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

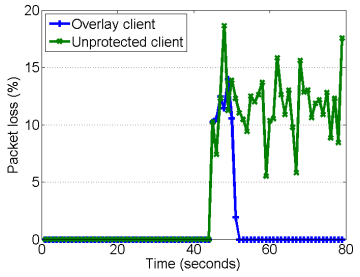

Figure 3. Experiment 1: Packet loss ratio, with and without overlay routing.

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

Table 1. Video sequence parameters for the second experiment.

two H.264/AVC video flows to the 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 bit rate 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[1]. 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 4; the stream hierarchy which formed the basis for the NIProxy to calculate this bandwidth distribution is also shown. 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

[1]Users can alter stream importance through the GUI of the client software.

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 section 2.3, the H.264/AVC transcoding plug-in was exploited to dynamically transcode the video streams to the target bit rates 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 bit rates, 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 bit rate. 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 bit rate.

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, the availability of the H.264/AVC transcoding service allowed the NIProxy to adapt video streams to a continuous range of bit rates, 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

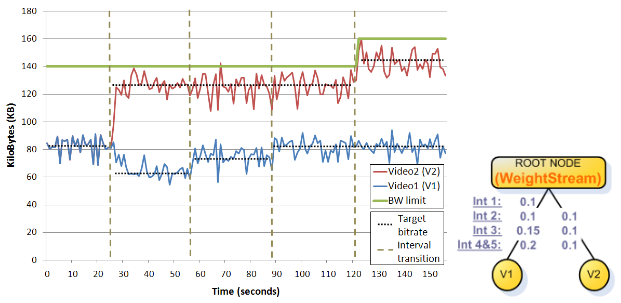

Figure 4. Experiment 2: (left) Stacked graph illustrating all H.264/AVC data received by the overlay-enhanced client, in KBps; (right) Stream hierarchy which directed the bandwidth distribution.

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 decoding issues and hence a severely deteriorated video playback at client-side.

3.4. 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.

4. Related Work

In previous research, the usage of overlay network technology for enhancing network routing has already been looked at. The RON project [1] 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 can only have 50 overlay sites. In [2], we have described algorithms for determining the optimal locations for the overlay servers and [3] 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 [6] and the Exact Bandwidth Distribution Scheme (X-BDS) [7]. The NIProxy's multimedia service provision mechanism on the other hand is largely related to the Service Oriented Architecture (SOA) paradigm [10]. Many examples of systems and frameworks adhering to this paradigm can be found in the literature; see, for instance, the research performed by Klara Nahrstedt (e.g. [9]). 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 bit rate, resolution or frame rate. In this paper, we have focused on bit rate reduction (SNR transcoding) of H.264/AVC streaming video. Two main classes of transcoding methods for bit rate reduction exist: dynamic rate shaping (DRS) and requantization transcoding. For MPEG-2, the former was investigated in [5], whereas the latter is discussed in [11]. 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.

5. Conclusions

We have presented an overlay network supporting true 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.

Acknowledgments

Part of the presented work is funded through the IBBT ISBO QoE project and the FWO Videostreaming project.

References

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP 2001)*, pages 131–145, Banff, Canada, October 2001.

[2] B. De Vleeschauwer, F. De Turck, B. Dhoedt, and P. Demeester. On the Construction of QoS Enabled Overlay Networks. In *Quality of Future Internet Services (QofIS 2004)*, volume 3266 of *Lecture Notes in Computer Science*, pages 164–173, Barcelona, Spain, September 2004.

[3] B. De Vleeschauwer, F. De Turck, B. Dhoedt, and P. Demeester. Dynamic Algorithms to Provide a Robust and Scalable Overlay Routing Service. In *International Conference on Information Networking (ICOIN 2006)*, volume 3961 of *Lecture Notes in Computer Science*, pages 945–954, Sendai, Japan, 2006.

[4] B. De Vleeschauwer, F. De Turck, B. Dhoedt, P. Demeester, M. Wijnants, and W. Lamotte. End-to-end QoE Optimization Through Overlay Network Deployment. In *Proceedings of the 22nd IEEE International Conference on Information Networking (ICOIN 2008)*, Pusan, Korea, January 2008.

[5] A. Eleftheriadis and P. Batra. Dynamic Rate Shaping of Compressed Digital Video. *IEEE Transactions on Multimedia*, 8(2):297–314, April 2006.

[6] I. Foster, M. Fidler, A. Roy, V. Sander, and L. Winkler. End-to-End Quality of Service for High-End Applications. *Computer Communications*, 27(14):1375–1388, 2004.

[7] V. Hnatyshin and A. S. Sethi. Architecture for Dynamic and Fair Distribution of Bandwidth. *International Journal of Network Management*, 16(5):317–336, September/October 2006.

[8] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[9] K. Nahrstedt, B. Yu, J. Liang, and Y. Cui. Hourglass Multimedia Content and Service Composition Framework for Smart Room Environments. *Elsevier Journal on Pervasive and Mobile Computing*, 1(1):43–75, March 2005.

[10] Service Oriented Architecture. `http://www.serviceoriented.org/service_oriented_architecture.html`, April 2008.

[11] H. Sun, W. Kwok, and J. Zdepski. Architectures for MPEG Compressed Bitstream Scaling. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(2):191–199, April 1996.

[12] M. Wijnants and W. Lamotte. 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, June 2007.

[13] M. Wijnants and W. Lamotte. 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, January 2008.