End-to-end QoE Optimization Through Overlay Network Deployment

Bart De Vleeschauwer, Filip De Turck, Bart Dhoedt and Piet Demeester
Ghent University - IBBT - IMEC, Department of Information Technology
Gaston Crommenlaan 8 bus 201, 9050 Gent, Belgium
Email: bart.devleeschauwer@intec.ugent.be

Maarten Wijnants, Wim Lamotte
Hasselt University and Interdisciplinary institute for BroadBand Technology (IBBT)
Expertise Centre for Digital Media and transnationale Universiteit Limburg
Wetenschapspark 2, BE-3590 Diepenbeek, Belgium
Email: maarten.wijnants,wim.lamotte@uhasselt.be

***Abstract*— In this paper an overlay network for end-to-end QoE management is presented. The goal of this infrastructure is QoE optimization by routing around failures in the IP network and optimizing the bandwidth usage on the last mile to the client. The overlay network consists of components that are located both in the core and at the edge of the network. A number of overlay servers perform end-to-end QoS monitoring and maintain an overlay topology, allowing them to route around link failures and congestion. Overlay access components situated at the edge of the network are responsible for determining whether packets are sent to the overlay network, while proxy components manage the bandwidth on the last mile. This paper gives a detailed overview of the end-to-end architecture together with representative experimental results which comprehensively demonstrate the overlay network's ability to optimize the QoE.**

I. Introduction

Originally, the Internet was mainly used for services like web surfing, file transfer and email. However, the last years have seen a continuous rise in the usage of the Internet for multimedia applications. Examples include video services such as IPTV, VoIP services such as Skype and online gaming applications. Since these services are much more sensitive to packet loss, bandwidth restrictions, delay and jitter than traditional services such as web browsing, these anomalies will have a great impact on the Quality of Experience (QoE). For instance, even a small amount of packet loss will result in visual artifacts for an IPTV service, while a high delay will greatly impact the fluidity of interactive applications.

Packet loss, increased delay and a restricted amount of bandwidth can have several causes. One problem that can occur is suboptimal routing between Internet hosts. The Internet routing service is Best-Effort, meaning that no guarantees are given on the available bandwidth and the delay. While some solutions are available to provide Quality of Service (QoS) in one network, like IntServ and DiffServ, these approaches were never able to offer a full end-to-end solution, due to scalability issues and the requirement of implementing these technologies in all the Autonomous Systems (AS) of the Internet. The problem of defining and implementing Service Level Agreements between the ASs has not yet been solved either. As a result, routing in the Internet is not always able to find the best path in terms of QoS. Instead, Internet routing will find a path that has a low number of intermediate hops but might have a large delay or an amount of packet loss. Finally, when a problem occurs in the AS graph, the routers may take considerable time to route around this problem, resulting in periods of connectivity loss that can even be in the order of minutes [1]. The last mile can also impose a bandwidth bottleneck to the end device. In particular, the amount of downstream bandwidth that is available on the last mile might not suffice to support all services which a client is currently using. This may result in congestion on the last mile, resulting in packet loss and an increase in delay.

To provide a solution to these problems, we propose to use an overlay network that is able to optimize the end-to-end QoE. This overlay network is responsible for routing around problems in the core network and for managing the bandwidth on the last mile. It consists of overlay servers that are located in the core network and that offer a resilient overlay routing infrastructure. In addition to this, the overlay network contains components that are located close to the multimedia clients and servers. These overlay access components are responsible for determining whether packets can benefit from overlay network routing and they provide transparent access to the overlay routing infrastructure. The overlay network also contains Network Intelligence Proxies that are situated at the network edge. These NIProxies are responsible for client bandwidth management and in addition have the ability to perform processing on network flows containing multimedia content. As an example, the proxies can be equipped with real-time video transcoding functionality, which would enable them to reduce the bandwidth requirements of video flows on-the-fly. In this paper we give an overview of the architecture of the overlay network and show it is able to maintain a high QoE when packet loss occurs in the network and when the bandwidth on the access line is restricted.

This paper is structured as follows: In section II a brief overview is given of related work. Section III describes the full architecture of the proposed overlay network while section IV harbors a thorough evaluation of the overlay network and demonstrates its ability to improve the QoE. Finally, conclusions are drawn in section V.

II. Related Work

In previous research, the usage of overlay network technology for enhancing network routing has also been looked at. The RON project [2] 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 [3], we have described algorithms for determining the optimal locations for the overlay servers and [4] introduces a number of algorithms for the management of the overlay topology. Overlay networks can also be used for overlay multicasting, when no network layer multicasting support is available. In [5] we developed an algorithm for the bounded diameter minimal cost Steiner tree problem.

Managing client downstream bandwidth is another topic of active research; see, for instance, the work presented in [6], [7] and [8]. The NIProxy distinguishes itself from these approaches in that the latter are concerned with QoS provision (i.e. guaranteeing that the requirements of data flows are satisfied), whereas the NIProxy's bandwidth distribution mechanism pursues the more high-level goal of maximizing the multimedia experience provided to users of networked applications. To achieve this objective, the NIProxy extensively exploits its application awareness, a feature that is lacking in many related systems. The NIProxy's application awareness also differentiates it from related work on multimedia service provision like, for instance, [9] and [10]. Finally, another unique characteristic of the NIProxy is that it integrates bandwidth management and multimedia service provision in a single system in such a manner that interoperation between both mechanisms is supported.

III. Overlay Network Architecture

This sections contains an overview of the overlay architecture for end-to-end QoE management. In Fig. 1 the end-to-end architecture is shown. The abbreviations OS, AC and NIProxy are used for the Overlay Servers, the Overlay Access Components and the Network Intelligence Proxies respectively.

A. Overlay Server

The overlay servers (OS) are located in several Autonomous Systems in the Internet. They sustain the 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

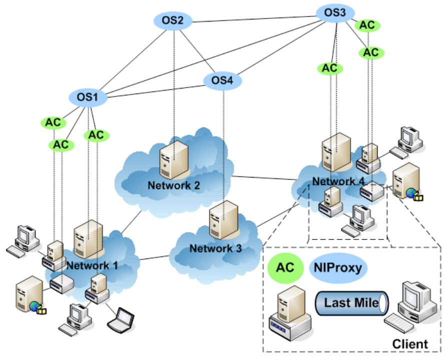

Fig. 1. The overlay network architecture

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 (Fig. 2). 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 who is responsible for further handling it.

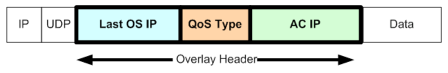

Fig. 2. The overlay header format

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 retrieved by performing active monitoring. The overlay servers send ICMP echo messages to their neighbours in the overlay topology and by analysing these probe messages, they deduce values for the delay and packet loss on the overlay edges. This information is subsequently exchanged between overlay servers, allowing them to share a view on the connectivity in the whole overlay network. It is stored in an overlay topology that is used to construct the overlay routing tables. In [4] a number of algorithms are described to construct and manage the overlay topology. When a connectivity problem is detected, the overlay servers will update their routing tables. This will cause them to use one or more intermediate overlay hops to forward the packets around the compromised link(s).

B. Overlay Network Access Component

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 will deploy overlay servers themselves. Therefore, we designed and implemented an overlay network access component. These components are deployed on or close to the end device. They are responsible for determining when traffic is sent to the actual overlay servers. The rationale behind this is that it is not always necessary to use the overlay network resources. More specifically, when there is no congestion or other problem on the path between source and destination, the overlay network should not be used. The AC monitors the quality of the connection. If a problem with QoS is detected or if the AC detects that the connectivity to the destination is lost, it will send the packets to its closest overlay server, which will forward it to an access component close to the target node. In this way, the AC is able to maintain the connectivity between source and target node. The access to the overlay network is transparent for the service itself. To provide its service, the AC peforms the following tasks:

- The AC maintains a connection to at least one overlay server. All the overlay traffic of the AC will be sent to this overlay server.
- When the AC detects that the device is communicating with another IP host, it determines whether the other party is also a client of the overlay network. If this is the case, it will start a probing thread that will periodically send packets to the AC at the other side, which will send an immediate reply. Based on these probe packets, the AC will analyse the connectivity and maintain values for the delay and the packet loss that is occurring on the path.
- If the QoS of a connection falls below a certain threshold (e.g. a delay that is above 200 milliseconds or packet loss), the AC decides that the packets of the connection need to be sent via the overlay network. Any subsequent packet belonging to that connection that arrives at the AC is encapsulated in an overlay header and is sent via the overlay network to the AC at the other side. The last overlay hop address that is filled in in the overlay packet header is the address of the overlay server to which the other AC is connected, the field for the access component is completed with the address of the AC at the other side.
- When an AC receives an overlay packet, it removes the overlay header and forwards the packet further to its destination, as if no overlay handling was done.
- When a connection is no longer active, this is detected via a time-out. When this happens, the AC stops probing the QoS of the path and removes the connection from its connection table.

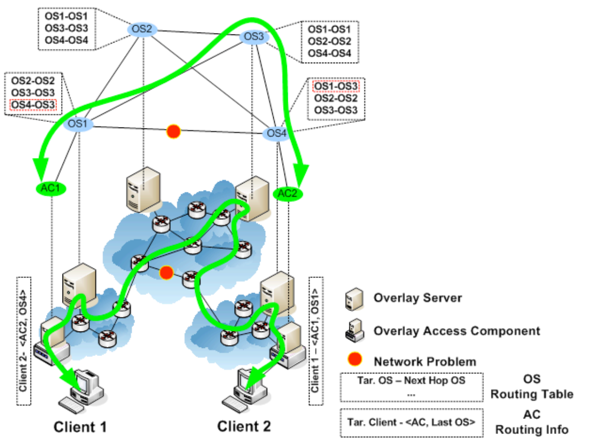

Fig. 3. An example overlay scenario. The routing tables of the overlay servers and the routing info that is maintained at the ACs, mapping the target IP addresses to the overlay AC and the OS at the other side, are shown.

In Fig. 3, a scenario is shown where an edge in the network has a degraded performance. More specifically, an IP link that is used on the direct path between overlay servers OS1 and OS4 suffers from congestion. Therefore, the overlay servers will change their routing tables accordingly. The problematic link will also impact the communication session between clients 1 and 2. As a result, the ACs will send the traffic via the alternative overlay path.

The advantage of using ACs that are located close to the clients is that we are able to offer an overlay routing service to end users, located anywhere in the Internet. It suffices that an AC is deployed close to the end device, to be able to offer the overlay routing capabilities to all their services. The AC can be deployed on the end device itself, but can also be situated on an intermediate device like a residential gateway, an access node or a proxy close to the end user.

C. *The Network Intelligence Proxy*

Using the components discussed thus far, the proposed overlay network is able to provide resilient routing in the core of the network. However, to create a true end-to-end solution which spans the entire delivery path from content source to sink, the last mile of this path should also be managed. The overlay architecture therefore also contains Network Intelligence Proxy instances, network intermediaries which aim to optimize last mile content delivery to clients [11]. As its name implies, the NIProxy attempts to accomplish this high-level objective by incorporating different types of *awareness* or *context* in the transportation network. More specifically, the NIProxy is at the moment network- as well as application-aware. The NIProxy's network awareness comprises knowledge of the current state of the network and, in particular, the capacity of client network connections, whereas its application awareness encompasses knowledge of the application(s) clients are currently running. In contrast to the network awareness, which consists of objective measurements indicating, for instance, the current throughput, latency and packet loss rate of a client's access link, the information constituting the NIProxy's application awareness is application-specific and can hence vary depending on the kind of application under consideration. The importance assigned by the client to the different network flows (or types of flows, e.g. audio or video) that are exchanged as part of a networked application is an example of knowledge which could contribute to the NIProxy's application awareness.

Based on its dual awareness, the NIProxy attempts to improve the user QoE in two complementary ways. First of all, the NIProxy provides automatic client downstream bandwidth management, meaning it is capable of dynamically distributing the downstream bandwidth available to a client over all network flows in which the client is currently interested. More specifically, based on its network awareness the NIProxy prevents over-encumbrance of the client's network connection, while its application awareness is exploited to create an intelligent allocation of the downstream bandwidth that is actually available. Secondly, the NIProxy is capable of applying processing on network flows containing multimedia content and hence also acts as multimedia service provision platform for its clients. As stated in our previous work [11], a major feature of the NIProxy is that interoperability between both its QoE-increasing mechanisms is supported. This enables a level of QoE maximization that could not be achieved by any of the two mechanisms separately.

From a more technical point of view, the NIProxy's bandwidth management mechanism operates by organizing all network streams in which a client is interested in a *stream hierarchy*. Such a stream hierarchy has 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 correspond to an actual network flow (e.g. a specific video stream). Different types of internal nodes are available, each with their distinct characteristics and capabilities (see [11] for more information). 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). The NIProxy's multimedia service provision functionality on the other hand is implemented using a plug-in approach. In particular, each provided service corresponds to a NIProxy plug-in that can be (un)loaded dynamically as it becomes needed (or obsolete). In our previous work [11], we described the implementation of an example plug-in which extends the NIProxy's functionality with real-time video transcoding capabilities. This service enables the NIProxy to reduce the bandwidth requirements of video streams by on-the-fly transcoding them to a lower quality.

Although NIProxy instances could theoretically be introduced nearly anywhere in the network topology, in practice they should be deployed relatively close to end users to be able to produce meaningful results (i.e. to be able to significantly improve user QoE). More specifically, optimal results will be produced in case the NIProxy is incorporated at locations where network performance degrades significantly. A typical example of such a location is the junction point separating the core of the network from the access network. Thanks to its ability to manage and possibly adapt network traffic, the NIProxy is capable of mitigating the mismatch in network performance that exists at such junction points.

The NIProxy concentrates on the last network hop(s) in the content delivery path. In particular, based on the current limitations of the last mile, NIProxy instances attempt to manage the transmission of content over it in such a manner that the user's QoE is optimized. This however immediately also uncovers an important hiatus in the NIProxy's current implementation: the NIProxy expects to receive the content which it needs to forward over these last hops flawlessly (i.e. in time and free of errors). For instance, in terms of the example given in the previous paragraph, the NIProxy assumes the network core to operate impeccably. Of course, this will not necessarily always be the case; as an example, packet loss could be present in the core of the network due to congestion. Integrating the NIProxy with the previously discussed overlay server components resolves this hiatus and results in an overlay network supporting true end-to-end QoE optimization, as will be demonstrated in the next section.

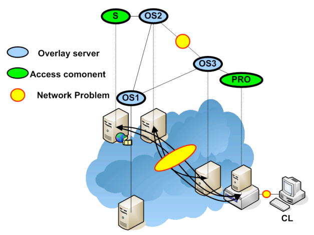

Fig. 4. The overlay network testbed

IV. Evaluation

A testbed was constructed, containing three overlay servers, a NIProxy (PRO), a streaming server (S) and a multimedia client (CL). The NIProxy is located between the network and the client and fulfills the role of access node. Access components are deployed on the NIProxy and on the multimedia server. The topology of the testbed is shown in Fig. 4. For the test, the server sends 4 video streams to the multimedia client. The network is impaired in two ways. Impairment nodes are used to artificially introduce random packet loss in the core network. At the same time, a bandwidth restriction is enforced on the last mile.

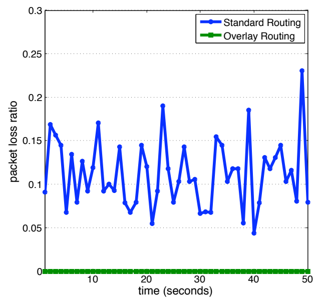

Fig. 5. The packet loss ratio between the server and the proxy with and without overlay routing.

Overlay link	Packets
OS2-OS1	7545
OS2-OS3	0
OS1-OS3	7545

TABLE I
Overlay Link Usage

The graph shown in Fig. 5 shows the packet loss ratio per second for a 50 second period with and without overlay routing. Throughout the test, an average packet loss of 10 % was present in the network, between (S, OS2) and (OS3, PRO, CL). One can see that when the overlay network is deployed, no packet loss occurs. The reason for this is that the overlay servers detect that there is packet loss and offer an alternative route that has better QoS characteristics. Table I shows the number of packets that were routed via the overlay edges during the 50 second interval; here we clearly see that the OS2-OS3 link is avoided. Without the overlay network, the packets are sent through the lossy part of the network, resulting in packets being lost. On the client's screen, the packet loss in the core network resulted in visual artifacts, which have a great impact on the QoE. By optimizing the route that is followed, the overlay servers and access components are able to deliver the video packets reliably to the NIProxy, which is responsible for further managing the bandwidth on the last mile.

Once the packets have been received by the NIProxy, it manages their final delivery to the client. The performed test consisted of 5 consecutive intervals, where each interval transition was caused by a change in one or more conditions. In particular, the transition from the first to the second and from the fourth to the fifth interval was initiated by a change in the downstream bandwidth available on the client's last mile network connection, whereas the other interval transitions were the result of client-initiated shifts in stream importance. The bandwidth fluctuations were automatically discovered by the NIProxy thanks to its network awareness. In contrast, the stream importance information was provided to the NIProxy in the form of application awareness and served as basis to construct and maintain the client's stream hierarchy during the experiment, which is illustrated in Fig. 6. As can be seen, an internal node of type *priority* was used to differentiate between the different video flows[1]. Also note from this figure that not one but two leaf nodes were incorporated in the stream hierarchy for each video stream. One node corresponded to the original version (OV) of the video stream (i.e. the video stream as it was transmitted by the streaming server), whereas the other node represented the transcoded version (TV) of this stream (i.e. the version generated by the NIProxy's video transcoding service). Both qualities were grouped together using an internal *mutex* node to guarantee that at most one version was assigned bandwidth.

Based on the just described stream hierarchy, the NIProxy managed the client's downstream bandwidth as illustrated by the network trace depicted in Fig. 7. A first important observation is that the downstream capacity of the last mile of the client's network connection was at all times respected. This result can be attributed to the NIProxy's network awareness and the outcome was an optimal reception of the forwarded flows at client-side (i.e. the video streams that were actually forwarded to the client were delivered with minimal delay and packet loss). Secondly, the network trace also comprehensively demonstrates the influence of the NIProxy's application awareness on the produced bandwidth distribution. In particular, as can be seen in Fig. 7, the bandwidth requirements of the least important video streams were reduced by transcoding them to a lower quality, this way preserving client downstream bandwidth for the forwarding of more important video flows. As a result, at any time during the experiment, the client received the video streams it deemed most significant at that moment at the highest quality possible.

V. Conclusion

In this paper, we have described an overlay network for end-to-end QoE management. It consists of components in the network core and at the edge. The overlay network achieves an optimization of the standard network routing service and optimizes the bandwidth usage on the last mile. In the evaluation section, we have shown that the overlay network is able to enhance the QoE. By providing resilient overlay routing in the network core, packet loss was eliminated, while congestion of the last mile was avoided by intelligently apportioning bandwidth to individual network flows at the edge.

[1]A priority node partitions bandwidth among its children according to their current priority value: bandwidth is first assigned to the child with the highest priority value, any remaining bandwidth is subsequently assigned to the child with the second highest priority, etcetera. In the experiment, the steam importance values specified by the client were mapped to priorities.

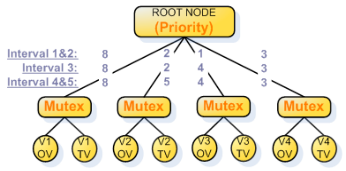

Fig. 6. Stream hierarchy based on which the NIProxy managed client downstream bandwidth. The priority values assigned to the different video flows are grouped horizontally per experiment interval(s).

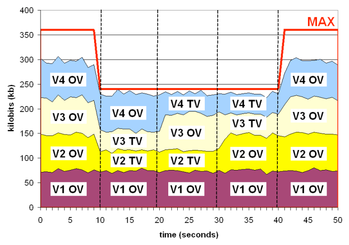

Fig. 7. Stacked graph illustrating all video traffic received by the client. The dashed vertical lines separate the different experiment intervals.

References

[1] N. Feamster, D. G. Andersen, H. Balakrishnan, and F. Kaashoek, "Measuring the effects of internet path faults on reactive routing," in *ACM Sigmetrics - Performance*, San Diego, USA, 2003.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Symposium on Operating Systems Principles*, 2001, pp. 131–145.

[3] 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)*, ser. Lecture Notes in Computer Science, vol. 3266, Barcelona, Spain, september 2004, pp. 164–173.

[4] ——, "Dynamic algorithms to provide a robust and scalable overlay routing service," in *International Conference on Information Networing (ICOIN)*, ser. Lecture Notes in Computer Science, vol. 3961, Sendai, Japan, 2006, pp. 945–954.

[5] ——, "Online management of QoS enabled overlay multicast services," in *Proceedings of The IEEE Global Telecommunications Conference (GLOBECOM)*, San Fransisco, USA, November 2007.

[6] E. Kusmierek, B.-Y. Choi, Z. Duan, and Z.-L. Zhang, "An Integrated Network Resource and QoS Management Framework," in *Proceedings of the IEEE Workshop on IP Operations and Management (IPOM)*, Dallas, USA, October 2002, pp. 68–72.

[7] M. Furini and D. Towsley, "Real-Time Traffic Transmission over the Internet," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 33–40, March 2001.

[8] V. Hnatyshin and A. S. Sethi, "Architecture for Dynamic and Fair Distribution of Bandwidth," *International Journal of Network Management*, vol. 16, no. 5, pp. 317–336, September/October 2006.

[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*, vol. 1, no. 1, pp. 43–75, March 2005.

[10] R. Mohan, J. R. Smith, and C.-S. Li, "Adapting Multimedia Internet Content for Universal Access," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 104–114, March 1999.

[11] 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.