An Autonomic Delivery Framework for HTTP Adaptive Streaming in Multicast-enabled Multimedia Access Networks

Niels Bouten*, Steven Latré*, Wim Van De Meerssche*,
Koen De Schepper†, Bart De Vleeschauwer†, Werner Van Leekwijck† and Filip De Turck*
*Ghent University - IBBT - IBCN - Department of Information Technology
Gaston Crommenlaan 8/201, B-9050 Gent, Belgium, e-mail: niels.bouten@intec.ugent.be
†Alcatel-Lucent Bell Labs, Copernicuslaan 50, B-2018 Antwerpen, Belgium

***Abstract*—The consumption of multimedia services over HTTP-based delivery mechanisms has recently gained popularity due to their increased flexibility and reliability. Traditional broadcast TV channels are now offered over the Internet, in order to support Live TV for a broad range of consumer devices. Moreover, service providers can greatly benefit from offering external live content (e.g., YouTube, Hulu) in a managed way. Recently, HTTP Adaptive Streaming (HAS) techniques have been proposed in which video clients dynamically adapt their requested video quality level based on the current network and device state. Unlike linear TV, traditional HTTP- and HAS-based video streaming services depend on unicast sessions, leading to a network traffic load proportional to the number of multimedia consumers. In this paper we propose a novel HAS-based video delivery architecture, which features intelligent multicasting and caching in order to decrease the required bandwidth considerably in a Live TV scenario. Furthermore we discuss the autonomic selection of multicasted content to support Video on Demand (VoD) sessions. Experiments were conducted on a large scale and realistic emulation environment and compared with a traditional HAS-based media delivery setup using only unicast connections.**

I. Introduction

During the last decade, the consumption of multimedia services over the Internet has witnessed an enormous increase. In terms of bandwidth, these services now have the largest share in the network [1]. Additionally, multimedia services such as video streaming, require stringent quality guarantees in order to meet the customers' demands in terms of Quality of Experience (QoE). These video services can be divided into two main categories: Internet Protocol Television (IPTV) and Over-The-Top (OTT) video services. IPTV services are offered by a network provider as part of its Triple Play service and typically consist of the broadcast of the live television signal, as well as additional services such as Video on Demand (VoD) in which the QoE is managed through resource reservation. OTT video services provide similar services, but the video is delivered over the traditional best-effort Internet. YouTube is probably one of the most popular OTT video providers, offering both VoD and live streaming (e.g., of selected sport events). Other service providers are now also offering their broadcasting content as an OTT video service. For example, Hulu was launched in March 2008 and offers VoD, live streaming and additional interactivity as an OTT service.

978-1-4673-0269-2/12/$31.00

The diversification of the services and devices has also led to a diversification of service delivery requirements. Traditional services offered by IPTV cannot be accessed by every device, since they require the use of a custom set-top box. In an OTT scenario the requirements are dependent on the network characteristics of the end-user connection. Although UDP-based connections were first envisioned to be a perfect candidate for the real-time consumption of video, the increased importance of QoE has resulted in a widespread adoption of the more traditional HTTP-based techniques, where the video is downloaded reliably over HTTP. Currently, the major providers of video content (e.g., YouTube, Hulu, BBC iPlayer) stream their content over HTTP. Another advantage of HTTP streaming is the smooth interaction with firewalls and NAT mechanisms. More recently, HTTP Adaptive Streaming techniques (HAS) have been proposed as an evolution of the initial progressive download techniques. However, these techniques are prone to congestion in the network, since they use the best-effort Internet for content delivery. Managing the delivery of such video services would be beneficial for both the end-user, by ensuring a decent level of QoE, and the network providers, diminishing the need to over-dimension the network.

To some extent, HTTP Adaptive streaming can provide a solution to network congestion, since it allows seamless degradation of the quality level. A more advisable solution however would be to reduce the number of unicast connections congesting the access network without reducing the perceived quality, by applying a technique similar to a managed Broadcast TV service. In a Live TV scenario, each unicast connection will transfer the same live content at the same time. Since these transfers are redundant, they can be grouped into a single transfer. In this paper, we discuss the design of a novel delivery framework that tackles the aforementioned issues. More specifically, the contributions of the paper are: a distributed architecture, enabling the scalable delivery of live streaming, an autonomic management algorithm for on demand video streaming and an emulation environment for the evaluation of the proposed architecture. The framework

features a combination of multicasting and caching to transparently transform a set of HAS connections into a single multicast connection for delivery over a network. Furthermore, an autonomic mechanism is proposed to select the content and qualities to be multicasted in a Video on Demand scenario, so to optimise the use of the available bandwidth on the bottleneck and the perceived QoE by the end-user.

The remainder of this paper is structured as follows: first, an overview of related technologies is presented in Section II. The components of the architecture are presented in Section III, followed by a discussion of the autonomic multicast management in Section IV. In Section V, we discuss how the proposed components of the delivery framework interact with each other and the HAS video streamer and client, even under packet loss scenarios. The performance of the delivery framework is evaluated in Section VI. Finally, Section VII concludes this paper and provides some directions for future work.

II. Related work

A. HTTP Adaptive Streaming

Several HTTP Adaptive Streaming protocols have been proposed by industrial players, such as ISS Smooth Streaming (Microsoft) [2], HTTP Live Streaming (Apple) [3] and HTTP Dynamic Streaming (Adobe) [4]. Although differences exist in their details, all protocols exhibit the same set of architectural components. At the encoding side, the video content is first encoded in several different quality levels and resolutions. This is followed by a division of the content into segments (typically several seconds worth of video) by a stream segmenter. These segments can be transported as a single file over HTTP. For each quality level, the most recently generated video segments are documented in a manifest file. This file also holds additional segment information such as segment location, size and quality. As such, the various segment files are linked into one video sequence through the meta-data contained in the manifest files. The segments and manifest files are hosted on one or more media distribution servers, typically HTTP web servers. Based on the information contained in the manifest files, the clients request the appropriate media segments through HTTP GET-methods. The client can then decide to download higher or lower quality segments to ensure a seamless rate adaptation. A video selection heuristic contained in the video client is responsible for deciding which quality levels are to be downloaded.

B. Multicast streaming of multimedia services

In a typical multimedia architecture, multicast techniques are used for streaming live video as a broadcast signal to multiple video clients (e.g., for live streaming). However, the use of multicast has also been investigated for other types of services such as Peer-to-peer (P2P) video multicast streaming using Scalable Video Coding (SVC) [5] and Multiple Description Video Coding (MDC) [6]. The solution presented in [6] uses a meshed P2P network where peers are connected to each other by UDP links to transmit streaming video. By using SVC, different clients receive different video qualities depending on their link capacity. While their solution focuses on the rate adaptation decision, we use multicast to decrease the load on the network.

Also for VoD services, multicast streaming solutions have been proposed [7]. Here, the described multicast-like transfer mechanism uses additional intelligence in the router in order to tackle two problems: synchronisation of video delivery requests and buffering of video packets. If a requested video packet is received at a router, it is stored into the video request synchronisation buffer and only released when enough video data is available in the buffer. While this approach successfully employs multicast for VoD services it results in long delays when requesting a video. We specifically evaluate the response times of our architecture to ensure that the additional delay is limited. P2PVR [8] proposes a P2P-based VoD architecture were peers are organised into a playback offset aware tree-based overlay. On-demand streaming data is shared among other peers with similar playback offset. Also a directory assists peers, who are searching for nodes that possess the expected streaming data. In our approach, we use a similar tree-based overlay with a fixed number of proxies closer to the video clients.

In the proposed architecture, a reliable HTTP connection is partly transformed (i.e. between two points in the network) into a UDP-based multicast connection. As UDP is an unreliable protocol, it is important to protect these multicast connections sufficiently. Many multicast resilience solutions focus on path reconstruction as part of a Carrier Ethernet solution [9]. In this case, additional multicast trees are constructed that can be used as a backup path once data losses are detected. This solution is however not feasible in a tree-based topology such as today's access networks, where only a single path exists between sender and receiver. Other multicast resilience solutions are often derived from resilience methods in wireless networks and focus on retransmission techniques [10], adaptive redundancy techniques [11] or codec-specific transcoding [12]. In our solution we use such retransmission techniques to handle packet loss.

III. A scalable architecture for video delivery

The approach presented in this paper seamlessly introduces multicast content delivery into an existing HAS-aware network. The goal of the proposed architecture is to decrease the required peak bandwidth of a Live TV service between HAS server and client through a combination of multicast streaming and content caching. To accomplish this, two component types are added to the original architecture: a distribution server and multiple delivery servers. The distribution server connects to the HAS server acting as a client, downloading manifests and segments on regular basis and forwarding them over a UDP-based multicast connection to several connected delivery servers. These delivery servers now act as a proxy for the original HAS server and clients can connect and interact with them in the same way as they would with the HAS server. The first time a video becomes available, it is pushed via multicast to several delivery servers, containing caches, closer to the

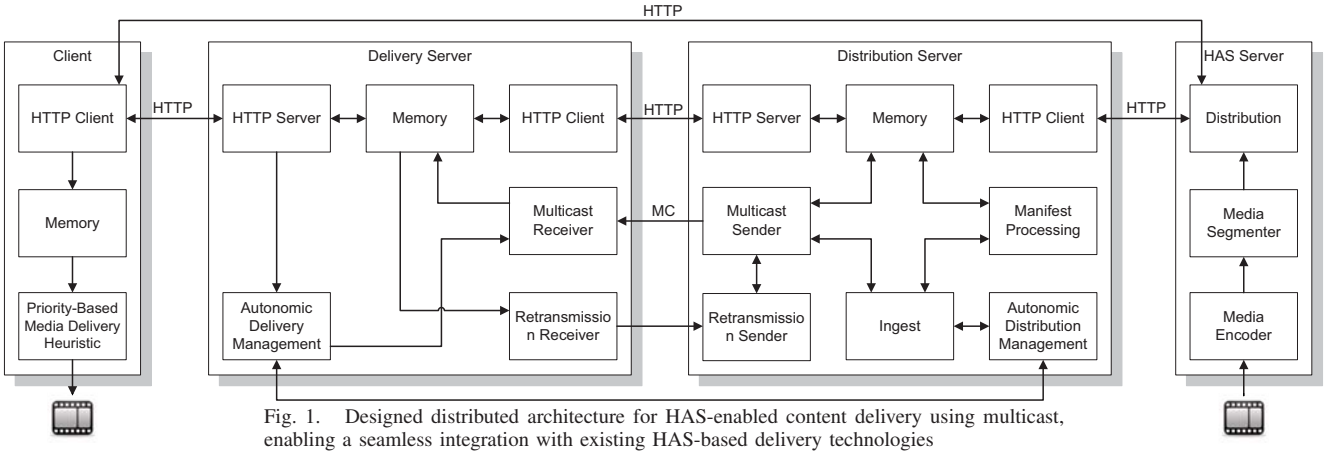

Fig. 1. Designed distributed architecture for HAS-enabled content delivery using multicast, enabling a seamless integration with existing HAS-based delivery technologies

video clients. When clients now request this video, it can be downloaded from the caching servers, resulting in a considerable decrease in bandwidth consumption. In the subsequent sections, the two novel component types are discussed in more detail. The proposed architecture is illustrated in Figure 1.

A. Distribution server

The distribution server is responsible for distributing the video content pro-actively to the connected delivery servers over multicast. The distribution server connects to the HAS server through the *HTTP Client* and serves incoming requests via the *HTTP Server* component. Video data is provided by the *Memory* component, which acts as a cache. The *Manifest Processing* continuously polls the HAS server for new manifests at a configurable rate (e.g., every second). When new segments are available and the channel is currently multicasted, the *Ingest* component will order the *Multicast Sender* to start multicasting the content in the selected qualities. Within one multicast tree, segments are pushed onto the tree in a chronological order as they appear in the video sequence. Whenever a manifest becomes available, it is pushed upon its associated multicast tree, taking into account the condition that the first packets of all selected qualities of a segment have to be sent before sending a manifest first mentioning that segment. This ensures that every segment is available at the cache of the delivery server before the content can be requested by a video client. Since multicast over UDP is an unreliable transport mechanism, resilience measures need to be taken. The *Retransmission Sender* is responsible for handling retransmission requests from the delivery servers. Both unicast and multicast retransmissions are supported, as well as a HTTP fallback mechanism. Each packet sent by the *Multicast Sender* is forwarded to the *Retransmission Sender* history, allowing the retransmission of a single packet.

B. Delivery server

The content received over multicast from the distribution server through the *Multicast Receiver* is stored locally by the *Memory* component. From then on the delivery server fulfils the role of local proxy for the HAS server towards the clients. Requests from the connected clients are received through the *HTTP Server* component. First the *Memory* is checked for the requested files. When these are not available, the *HTTP Client* is contacted to retrieve them from the distribution server over HTTP. On the detection of packet loss (i.e. a gap in sequence numbers between two consecutive packets), an error message concerning the packet loss is reported. The *Memory* then contacts the *Retransmission Receiver* to request the retransmission of the reported packets. A more in-depth discussion of the retransmission mechanism can be found in Section V-B.

IV. Autonomic multicast management for Video on Demand

The architecture discussed earlier provides a decrease of the consumed bandwidth between distribution and delivery servers through a combination of caching and multicasting in a HAS-based Live TV scenario. However, in a realistic scenario, a mixture of Video on Demand and Live TV services will occur. In this mix, several unicast VoD connections can also be mapped on a single multicast connection. Which unicast connections are suited for such a multicast grouping depends on the status of the caches and the future requests at the delivery servers. As such, the mapping of unicast VoD connections is far less trivial than that of Live TV. In this section, we propose a novel management algorithm for autonomously selecting which unicast flows need to be mapped to a multicast connection in order to benefit from a reduced bandwidth consumption in a HAS-based Video on Demand scenario. This scenario requires management of the multicasted content since the clients no longer request the video segments in a synchronised way. As multicasting video from the distribution server to the delivery servers is only useful when the video is served by multiple delivery servers, the *Autonomic Distribution Management* requires regular reports from the various delivery servers, to assess the popularity of the video items in the VoD catalogue. The caches located at the delivery server allow temporarily storing the video flows and thus relax the need for the video flows to be fully synchronised. Hence, the requests for video content do not need to occur simultaneously in order to be mapped to a multicast connection, but can occur within a predefined time

window of W segments.

The size of W depends on the cache size and replacement algorithm used at the delivery servers. The algorithm at the distribution server groups all requested segments for a certain video within the time window W starting from the highest segment number (suppose H) to the lowest segment number larger than $H - W$ (suppose L). If the number of requesting delivery servers (suppose D) for that range is larger than a certain threshold, the content for that video is now multicasted starting with segment H. Since this segment will at least be available for a time of W segments at the delivery servers, the clients that were requesting segment L, will be able to request segment H from the cache. This grouping of unicast flows will result in a reduced bandwidth consumption proportional to D. The multicast trees that were created this way are communicated to the *Autonomic Delivery Management*, which can now autonomously decide which multicast trees to join or leave depending on the number of clients requesting that content. Each interval, the created multicast trees are re-evaluated, so to optimise the number of served delivery servers. The algorithm will be further optimised by sharing knowledge of the cache state of the delivery server to the distribution server.

V. Component Interaction

A. Impact on delay

The proposed architecture modifies the delivery of HAS video, therefore there are some important consequences on the delay for the retrieval of manifest files on the Client. A first additional delay is caused by the fact that a manifest is updated at a constant rate at the HAS server, but the Manifest Processing component only periodically polls the server for new manifests. A second delay is caused by the time needed to download a segment, since a manifest is only forwarded after the first packets of the first segment are present at the delivery servers. This delay is equal to the time needed to send a segment from HAS to distribution server. Thirdly, the time needed to multicast the first packet of the first segment mentioned in the manifest causes extra delay. The last source of additional delay is the time needed to multicast the first packet of a manifest. When retrieving segments over HTTP there is no extra delay apart from the download delay. However since the delivery servers are closer to the clients in the access network, link delays for requesting and downloading video are much smaller than in the original HAS delivery framework, where clients are further removed from the HAS-server.

Figure 2 illustrates the additional delay with respect to the live moment a client experiences, relative to the offset of the client start-up time with respect to the manifest generation at the HAS server. Initially, clients in the multicast scenario are experiencing additional delay, since new segments and manifests need some time to be multicasted to the delivery servers first. This additional delay is equal to the segment duration (i.e. 10 seconds for Apple HLS) since those clients are downloading the manifest and segments generated one segment time earlier. However, when manifests and segments are already multicasted to the delivery server, clients connected

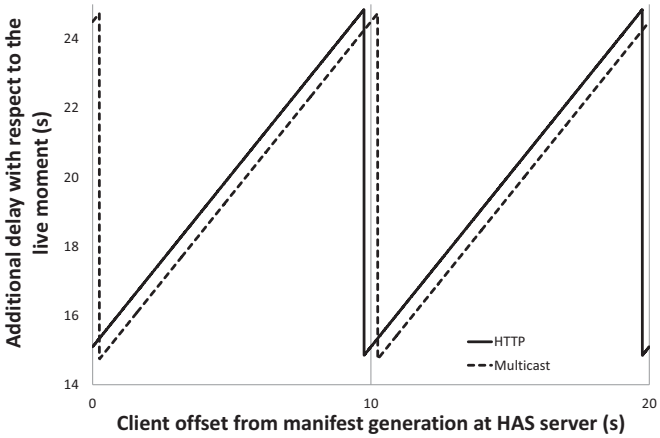

Fig. 2. Comparison of additional delay in both the HTTP and multicast scenario

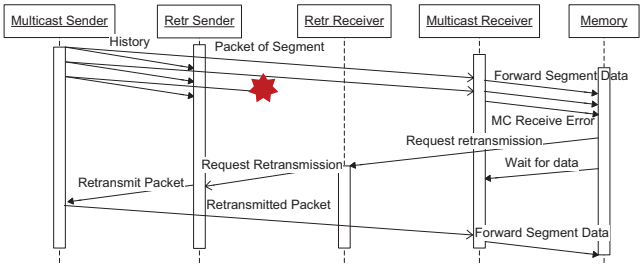

Fig. 3. Illustration of the retransmission mechanism at the Delivery Server

in the multicast scenario are experiencing lower download delays, since they are closer to the delivery servers than the clients connected over HTTP with the HAS server.

B. Handling of packet loss

As part of the data is multicasted over unreliable UDP-connections, it is crucial to implement resilience measures to account for packet loss. To address this, a retransmission mechanism was implemented supporting three different modes: HTTP fallback, UDP unicast and multicast retransmissions. Multicast retransmissions are requested by the *Multicast Receiver* component at the delivery server and served by the *Retransmission Sender*, the advantage is of course that each delivery server will receive the packets that were marked as lost. The HTTP fallback mechanism is triggered when bursty packet loss occurs, where it is more beneficial to retransmit the data over a reliable transport mechanism. Figure 3 illustrates the handling of packet loss by the multicast retransmission mechanism.

VI. Performance Evaluation

A. Implementation details

A prototype of the architecture was implemented using the Apple Live Streaming protocol as an underlying HAS technique. Since only Apple-based devices are able to playback this video and the number of available devices was limited, we implemented our own HAS client, allowing us to accurately emulate the client behaviour on a large scale. The quality level selection heuristic used in our prototype is based upon an existing heuristic called Priority-Based Media Delivery [13] and decides which quality to download by considering several

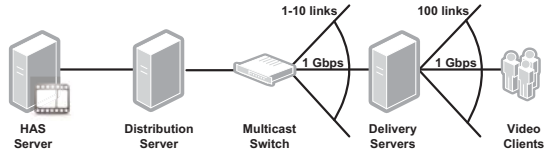

Fig. 4. Emulated network topology

previously downloaded fragments through a weighted moving average.

In the proposed architecture, caches are used in the *Memory* components of both distribution and delivery server. Since we test our prototype using a Live TV scenario, the cache replacement strategy best suited is Least Recently Used (LRU). Since clients play back the video in the same chronological order manifests and segments are created, LRU leads to the least possible cache misses.

One-to-many IP multicast, which is often used for streaming media applications over an IP infrastructure, is used as a multicast protocol. This technique does not require prior knowledge of the number of receivers, allowing a large number of dynamically connected receivers. Since the bandwidth consumed by multicast over UDP can be very bursty, it needs to be shaped. Without shaping, each segment would be multicasted at the maximum possible line speed, smothering other traffic on that link. There are two options: shaping the total outgoing rate or shaping each file separately. In our implementation a hybrid approach is applied combining both shaping mechanisms into two levels of token buckets.

B. Experimental setup

The network model illustrated in Figure 4 depicts a typical tree based access network of 1012 nodes consisting of 1 distribution server, 10 delivery servers and 1000 virtual video clients (mapped onto 10 physical clients) connected by gigabit links. The distribution server offers 5 live channels, each available in six qualities: 8Mbit/s (Full HD), 4Mbit/s (HD Ready), 2Mbit/s (SD), 900kbit/s (High quality web video), 500kbit/s (Moderate quality web video), 200kbit/s (Low quality web video). Each delivery server has a cache size of 500MB, which enables it to cache enough segments of the live stream so that no entry discarded by LRU is ever requested again. This is justified since we mainly focus on the influence of the system on bandwidth consumption. The distribution of viewers over the five channels is set according to values measured on a real network and follows a Zipf distribution with parameter β equal to 1.7. The distribution of viewers over the different delivery servers is uniformly random, which implies there are no significant local popularity differences.

C. Impact of multicasting on consumed bandwidth

Figure 5 shows the impact of the number of delivery servers and the multicasted channels on the consumed bandwidth between distribution and delivery servers. The effect of multicasting all channels and all qualities is a reduced bandwidth when more than three delivery servers request the video content. This is because all clients are downloading the highest quality of the requested channel and the multicasting of the

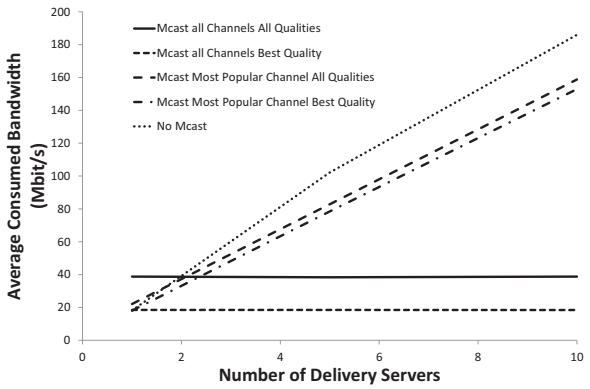

Fig. 5. Impact on bandwidth of the different multicast strategies

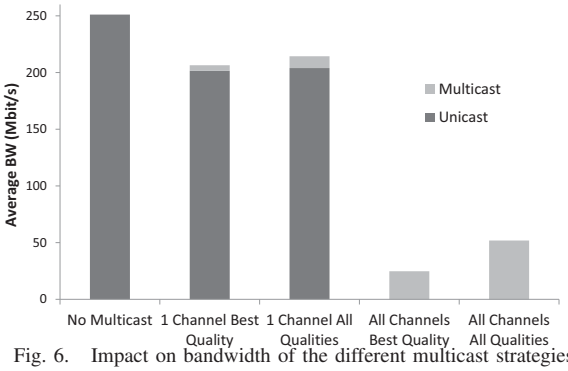

Fig. 6. Impact on bandwidth of the different multicast strategies

lower quality channels is thus causing unnecessary bandwidth consumption. In case only the highest quality of all channels is multicasted, this bandwidth reduction already occurs with only two delivery servers. When only the most popular channel is multicasted, with the other channels still being served through unicast, the bandwidth reduction is obviously less significant. Figure 6 shows the average used bandwidth for a variety of multicasting strategies in a network with 10 delivery servers. Multicasting the best quality of each channel uses 10 times less bandwidth than when nothing is multicasted. Multicasting the best quality of 1 channel only results in a 20% bandwidth reduction.

D. Impact of the retransmission strategy on bandwidth

For this experiment, we introduce the terms multicast loss and unicast loss, respectively for the loss introduced on the link between the distribution server and the multicast switch and on the links between the multicast switch and the delivery servers. Multicast loss will cause all delivery servers to experience the same amount of packet loss, while unicast loss will affect a specific delivery server. The amount of loss that is introduced across all experiments is set at 1%. The consumed bandwidth on the link between distribution server and multicast switch is displayed in Figure 7. For a single delivery server, all 4 scenarios are similar, as there is no difference between introducing loss on both links and between retransmission strategies. Unicast retransmits and multicast retransmits for unicast loss show a linear correlation between the increase in consumed bandwidth and the number of delivery servers. Multicast retransmits for multicast loss use

Fig. 7. Impact on bandwidth of the different retransmission strategies on consumed bandwidth at distribution server

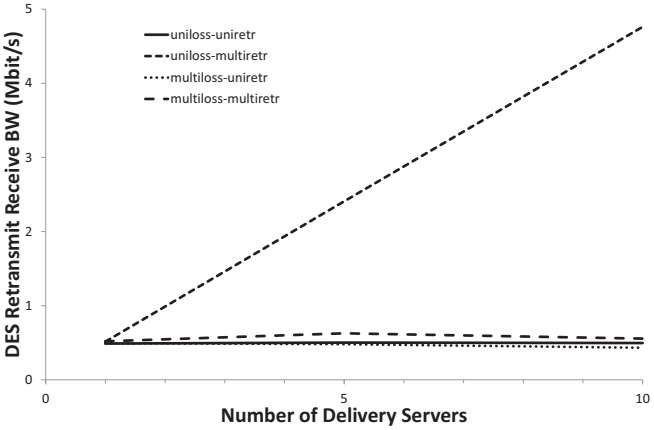

Fig. 8. Impact on bandwidth of the different retransmission strategies on consumed bandwidth at delivery servers

a constant bandwidth, independent of the number of delivery servers, as duplicate requests are ignored. The bandwidth used for receiving retransmissions on the link of the first delivery server is shown in Figure 8. In this case, multicast retransmits for unicast loss result in an additional bandwidth use on this link, as retransmits requested by other delivery servers are received here as well. These test results show that it would be beneficial to adapt the retransmission strategy according to the type of loss, in order to optimise the consumed bandwidth.

VII. Conclusion

In this paper, we characterised the merits of a novel HAS-based multimedia architecture, allowing a decrease of the consumed bandwidth, through a combination of caching and multicast streaming. Two additional component types were added to a traditional HAS-based architecture: a single distribution and multiple delivery servers. We compared our novel multicast-enabled architecture with a traditional HAS set-up, which uses only unicast connections. The experiments show that the obtained bandwidth reduction factor, when using multicasting and caching, is proportional to the number of connected delivery servers, even when multiple HAS qualities are multicasted. For example, our novel architecture requires 4 times less bandwidth than traditional HAS-based approaches for a moderate network size consisting of 8 delivery servers; the larger the network, the bigger the obtained advantage is. Additionally, redundancy tests showed that the implemented multicast retransmission mechanism also provides a retransmission bandwidth reduction compared to unicast retransmission, both for multicast and unicast loss. As such, we have shown how our multicast-enabled architecture is more scalable without losing robustness in delivering HAS-based Live TV. In future work, we plan to improve the autonomic management algorithm that decides what content to multicast in a Video on Demand scenario. The goal is to combine segment popularity statistics with an assessment of the status of each cache to decide which content to multicast. This will enable the scalable delivery of HAS-based video for Live TV as well as for Video on Demand services.

Acknowledgment

Steven Latré is funded by grant of the Fund for Scientific Research, Flanders (FWO-V).

References

[1] "Cisco visual networking index: Forecast and methodology: 2010-2015," pp. 1–16, February 2011. [Online]. Available: http://bit.ly/ciscoforecast

[2] Microsoft, "Smooth streaming: The official microsoft iis site," http://www.iis.net/download/SmoothStreaming - Last accessed on 1 September, 2011.

[3] R. Pantos and W. May, "HTTP Live Streaming," 2011. [Online]. Available: http://tools.ietf.org/html/draft-pantos-http-live-streaming-07

[4] Adobe, "Http dynamic streaming: Flexible delivery of on-demand and live video streaming," http://www.adobe.com/products/httpdynamicstreaming/ - Last accessed on 1 September, 2011.

[5] H. Schwarz, D. Marpe, and T. Wieg, "Overview of the scalable video coding extension of the h.264/avc standard," in *IEEE Transactions on Circuits and Systems for Video Technology In Circuits and Systems for Video Technology*, 2007, pp. 1103–1120.

[6] F. de Asís López-Fuentes, "P2p video streaming combining svc and mdc," *Applied Mathematics and Computer Science*, vol. 21, no. 2, pp. 295–306, 2011.

[7] T. Miyoshi and K. Sekiya, "Efficient transfer method for on-demand video delivery based on streaming packet analysis," in *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on*, may 2011, pp. 141–146.

[8] Y.-S. Yu, C.-K. Shieh, C.-H. Lin, and S.-Y. Wang, "P2pvr: A playback offset aware multicast tree for on-demand video streaming with vcr functions," *J. Syst. Archit.*, vol. 57, pp. 392–403, April 2011. [Online]. Available: http://dx.doi.org/10.1016/j.sysarc.2011.03.001

[9] S. Ruepp, H. Wessing, J. Zhang, A. V. Manolova, A. Rasmussen, L. Dittmann, and M. Berger, "Evaluating multicast resilience in carrier ethernet," *WSEAS Trans. Cir. and Sys.*, vol. 9, pp. 101–110, February 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1852308.1852312

[10] N. Choi, Y. Seok, T. Kwon, and Y. Choi, "Multicasting multimedia streams in ieee 802.11 networks: a focus on reliability and rate adaptation," *Wirel. Netw.*, vol. 17, pp. 119–131, January 2011. [Online]. Available: http://dx.doi.org/10.1007/s11276-010-0268-9

[11] R. Vaishampayan, J. J. Garcia-Luna-Aceves, and K. Obraczka, "An adaptive redundancy protocol for mesh based multicasting," *Comput. Commun.*, vol. 30, pp. 1015–1028, March 2007. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2006.08.031

[12] C.-M. Chen, C.-W. Lin, and Y.-C. Chen, "Adaptive error-resilience transcoding using prioritized intra-refresh for video multicast over wireless networks," *Signal Processing: Image Communication*, vol. 22, no. 3, pp. 277 – 297, 2007, special issue on Mobile Video. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596506001421

[13] T. Schierl, Y. Sanchez de la Fuente, R. Globisch, C. Hellge, and T. Wiegand, "Priority-based media delivery using svc with rtp and http streaming," *Multimedia Tools and Applications*, vol. 55, pp. 227–246, 2011, 10.1007/s11042-010-0572-5. [Online]. Available: http://dx.doi.org/10.1007/s11042-010-0572-5