Towards Intelligent Scheduling of Multimedia Content in Future Access Networks

Jeroen Famaey, Wim Van de Meerssche, Steven Latré, Stijn Melis, Tim Wauters, Filip De Turck
Department of Information Technology
Ghent University – IBBT
Gaston Crommenlaan 8/201, B-9050 Gent, Belgium
Email: jeroen.famaey@intec.ugent.be

Koen De Schepper, Bart De Vleeschauwer, Rafael Huysegems
Alcatel-Lucent Bell Labs
Copernicuslaan 50, B-2018 Antwerpen, Belgium

***Abstract*—The popularity of streaming multimedia services has greatly increased in recent years. Telco- and cable-providers have started offering a plethora of multimedia services in the access and aggregation network, including video on demand, interactive digital television, and time-shifted TV. However, these services introduce additional challenges, such as stringent time constraints, and high bandwidth requirements. To overcome these problems, we explore the advantages of delivering such multimedia content using deadline-aware scheduling and caching algorithms. These algorithms decide when to send and store which content. This enables the network to optimize bandwidth consumption and satisfy deadline constraints.**

The designed algorithm was evaluated and compared to classical deadline-unaware delivery protocols. This allows us to study the efficiency of the new algorithm, and identify the scenarios in which deadline-aware scheduling improves delivery of multimedia content.

I. Introduction

A plethora of multimedia services have popped up in recent years, giving end-users access to a wide range of digital content. This content is offered both on the Internet (e.g. YouTube, P2P-TV) and in access and aggregation networks (e.g. video-on-demand, digital television, and time-shifted TV). The delivery of multimedia content introduces new challenges and difficulties, including stringent delivery deadlines and high bandwidth demands. Existing best-effort networks and protocols are not designed to overcome these challenges.

As a solution, we propose a novel approach for delivering content with strict Quality of Service requirements in access and aggregation networks. By introducing intelligent scheduling nodes inside the network, content can be delivered more timely and bandwidth can be used more efficiently. Additionally, popular content can be temporarily stored in intermediary nodes and re-used for later requests. This paper describes a generic content delivery protocol and a deadline-aware scheduling algorithm. In contrast to most existing scheduling and caching strategies, our approach uses a generic model, independent of the type of multimedia service.

In addition to a description of the protocol and algorithm, this paper contains an in-depth evaluation using simulation results. The goal of this evaluation is to answer several pertinent questions:

1) Does deadline-aware scheduling increase the number of satisfied deadlines?
2) Under what circumstances is deadline-aware scheduling useful?
3) What is the effect of the amount of available client-side memory on the efficiency of the algorithms?

The rest of this paper is structured as follows. Section II gives an overview of existing work on the topic of scheduling and caching multimedia content and summarizes the main contributions of our work compared to them. The concepts and terminology used throughout this paper are described in Section III. Subsequently, Sections IV and V elaborate upon the delivery protocol and scheduling algorithms respectively. An evaluation of the algorithms is given and discussed in Section VI. Finally, this paper is concluded in Section VII.

II. Related Work

The large bandwidth requirements of multimedia services have triggered the research for caching solutions in access networks. In [1], [2], the authors argue that the evolution of IPTV services from VoD to TimeShiftedTV services causes an explosion in the requested resources which can only be tackled by effectively caching requested multimedia content. Several caching algorithms have been proposed for streaming multimedia services [3], [4]. The work presented there focuses on cache replacement techniques and how caches can cooperate. While we use these cache replacement techniques to organise each cache individually, our work focuses on scheduling of the transmission of content to the clients.

Research in the scheduling of multimedia content has primarily focused on scheduling in shared broadcast environments such as cellular [5] and wireless [6] networks. In these scenarios, the prime focus is on the prioritization and selection of traffic in contention situations. In [5], an earliest deadline first algorithm for scheduling in HSDPA networks is proposed. The deadline is calculated implicitly based on the measured delay. The scheduling algorithm proposed in [6] is video codec dependent and chooses to prioritize I and P frames when congestion is imminent to minimize video quality drops. Our work differs from this as we concentrate on the bandwidth consumption optimization of the network itself.

In an IPTV setting, scheduling algorithms in combination with P2P live streaming systems have been suggested both in [7] and [8]. In [7] each peer that has content available tries to schedule the streaming of this content in such a way that play-back freeze ups at the clients are minimized. The proposed solution suggested in [8] follows a similar approach but relies on the distributed clients to schedule requests on which the serving peers can respond. In [7], the scheduling occurs completely decentralized, which is a viable assumption in a P2P system where the content is distributed orthogonally. However, in an IPTV scenario, where services are streamed from one central point at the edge of the network, the scheduling algorithm needs to incorporate these hierarchical dependencies, which is assumed in our network model.

III. Concepts & Terminology

This section gives a description of the concepts and terminology used throughout this paper. The first part of this section focuses on the network model. Then, we elaborate on how memory and multimedia content are represented.

A. Network Model

The network consists of a set of content servers, scheduler nodes, and clients. The *content servers* host the multimedia content. Every server is responsible for one or more content items, and it is assumed that these items are entirely available on the server at all times. Every content server has an outgoing link. The *scheduler nodes* are intermediate nodes capable of caching parts of content items, and scheduling requests. Every scheduler node has a set of incoming and outgoing links. Finally, the *clients* represent the end-users that generate requests for the hosted content. Every client has an incoming link. Additionally, every link has an associated *bandwidth limit* and every client and scheduler node has a *memory limit*.

For every node, its *parent* is defined as the node one step closer to the content server, while its *children* are the neighboring nodes towards the client-side.

B. Memory & Streaming Model

The content hosted by the content servers, is represented by way of *content items*. A content item has an associated *first* and *last byte*, and an *application bit-rate*. This is the bit-rate by which the requesting client-side application processes the content (e.g. the play bit-rate of a video). A node requests (part of) a content item by sending a *request* to its parent on the path to the content server hosting this content. This request identifies the content item and the range of bytes, consisting of the *first* and *last requested byte*, the node wants to receive. Additionally it specifies the *deadline* by which the content must arrive at the child. A request may be sent by the client before its first byte deadline. The corresponding time-interval is called the *known time* of the request.

Upon receiving a request from a child node, a node sets up an outgoing connection. This connection consists of a *next byte*, which specifies the next byte that will be sent. Additionally, it has a *current bit-rate*.

The bytes of a content item that are currently in the node's memory are represented as a set of *blocks*. A block has a *first* and *last byte*. Additionally, it may have an associated *incoming connection*, via which it receives data from the parent node. The last byte of the request associated with the incoming connection, is called the *last requested byte* of the block. The block also has a set of *current* and *future outgoing connections*. Finally, to allow data to be replaced, blocks may have a *shrink rate*. This is the bit-rate by which data is removed from the block.

IV. Content Delivery Protocol

The content delivery protocol is responsible for creating, maintaining and removing memory blocks, and setting up connections when new requests arrive, or existing ones are canceled.

When a new request arrives at the node from one of its children, the node must request, to its parent, the bytes of this request. To facilitate content re-use, only bytes that are not already in its memory are requested from the parent. Additionally, bytes that have already been requested with an earlier deadline, but that have not yet been received, are not requested either.

A request may end for two reasons. First, the client might no longer be interested in the data, and explicitly cancel the request. Second, all the data associated with the request may have been sent to the child node from which the request originated. In the case all data of a request has been sent to the child node, nothing needs to be done, except removing the request and closing the associated connection. However, in the second case, the deadline and last requested byte of incoming connections associated with blocks of which the outgoing connection associated with the canceled request is a current or future connection may need to be adjusted. If this connection dominates the deadline or last requested byte of a block, they need to be recalculated based on the remaining current and future outgoing connections of that block.

Currently, the delivery protocol caches only requested content. This means that when a block has no future outgoing connections, the shrink-rate is set equal to the bit-rate of the lowest current outgoing connection of the block (i.e. the connection with the lowest next byte). Otherwise, the shrink-rate of a block is set to 0. In future work, we plan to study the effect of using existing caching strategies in combination with the delivery protocol.

V. Scheduling Algorithms

Scheduling algorithms are responsible for setting the bit-rates of outgoing connections. The algorithm is fully decentralized, and every node is responsible for setting its own outgoing rates. Additionally, the node can influence the bit-rates of its incoming connections by setting maximum incoming rates for those connections, and setting a global maximum incoming rate per link. The algorithm is executed periodically on every node.

Three scheduling approaches are proposed in this paper. Fair-share (FS), and weighted fair-share (WFS) are classical deadline-unaware algorithms, used as a basis for comparison. Earliest deadline first (EDF) is a more intelligent deadline-aware scheduling algorithm.

A. Outgoing Connections

The scheduling algorithms divide the available bandwidth per link over all outgoing connections on that link. They differ in the amount of bandwidth that is given to every connection.

1) Fair-Share (FS): The available link-bandwidth is divided equally among all connections on the link, independent of their requested bit-rate, or deadline. If the maximum incoming bit-rate of a connection is lower than its share, the left-over bandwidth is divided among the other connections. How this maximum bit-rate is determined, is explained in Section V-B. In an end-to-end scenario, FS behaves similarly to end-to-end TCP.

2) Weighted Fair-Share (WFS): Identical to FS, except that bandwidth is divided proportionally to the requested bit-rate of the connections.

3) Earliest Deadline First (EDF): A deadline-aware algorithm that gives all available bandwidth on a link to the connection with the earliest next byte deadline. If multiple connections share the same deadline, bandwidth is divided among them in WFS-fashion. If any bandwidth is left-over, it is given to the connection(s) with the next-earliest deadline.

B. Incoming Connections

If a node is incapable of limiting the bit-rate of incoming connections, it might receive more data than it can store in memory or send to its children. Therefore, a mechanism is in place that provides a node with the ability to set the maximum bit-rate per incoming connection. The scheduling algorithms make sure the bandwidth share of an outgoing connection does not exceed this maximum bit-rate. Note that the maximum bit-rate of incoming connection is only limited when the cache memory of the node is full.

A heuristic was devised that sets the maximum incoming bit-rate of a connection to the maximum achievable shrink-rate of its target block. The maximum achievable shrink-rate of a block equals 0 if the block has future outgoing connections, otherwise it equals the maximum bit-rate of its lowest current outgoing connection. the maximum bit-rate of an outgoing connection equals the bandwidth-share that would have been given to this connection by the scheduling algorithm if it were not limited by the incoming connection of its source block.

VI. Evaluation & Discussion

The goal of the evaluation is to determine in what types of scenarios deadline-aware scheduling is useful compared to classical deadline-unaware scheduling approaches. To this end, the deadline-aware EDF algorithm is compared to the deadline-unaware FS and WFS approaches.

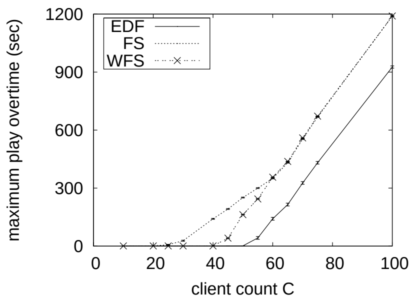


Figure 1. The **maximum** play overtime (in seconds) as a function of C ($M = 250$, $K = 360$), for EDF, FS and WFS

A. Methodology & Simulation Setup

As an evaluation metric, the *play overtime* of requests is used. This is the actual playtime minus the expected playtime of the request. The actual playtime is defined as the time between the first byte deadline and the last byte arriving at the requesting application. The expected playtime is calculated by dividing the size of the requested content (in bits) by its requested bit-rate (in bits per second). This metric is thus a direct measure for overall deadline satisfaction. If the play overtime of all requests is 0, all deadlines have been met.

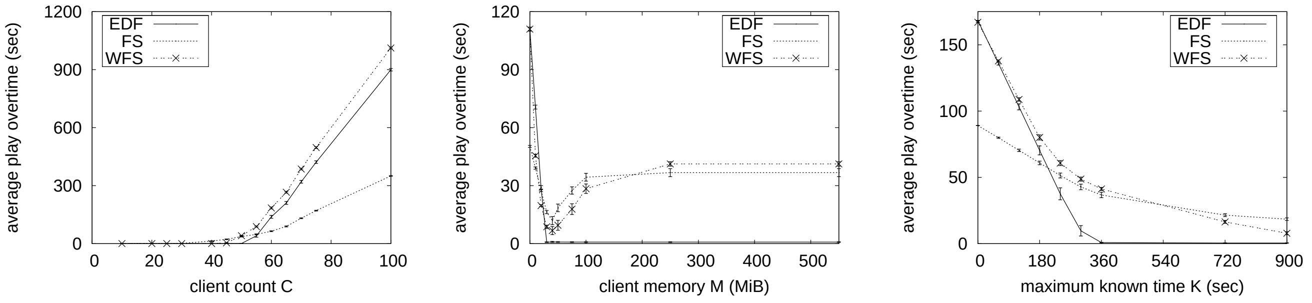
The simulated network topology consists of 1 content server, 1 scheduler node, and C scheduling capable clients. The link between the server and scheduler has a bandwidth of 100 Mbps[1], and the links to the clients 10 Mbps. As only an end-to-end scenario was evaluated, the cache memory size of the scheduler node was set to 0 MiB[2]. The client cache memory was set to a variable M MiB. During a simulation run, all clients send a single request for a 15 minute video of 5 (HD), 1.5 (SD) or 0.5 (LD) Mbps. It is assumed that one third of all requests arrive for each quality type, and that applications never consume data faster than this bit-rate (not even when they have missed their deadline). All clients request a personalized video, so no content re-use occurs. Finally, all requests are known at time 0, while their first byte deadline is chosen uniformly at random from the interval $[0, K]$ seconds. The parameter K thus represents the maximum known time for any request. Simulations were performed for the parameters $C \in [0, 100]$, $M \in [0, 550]$ and $K \in [0, 900]$. A subset of the obtained results is discussed in the rest of this section.

All simulation results are averaged over 10 iterations. The error bars in the graphs represent the standard error of this average.

B. Results

Fig. 1 shows maximum play overtime over all requests for $M = 250$ and $K = 360$. The point where the maximum play overtime no longer remains 0, denotes the number of deadlines that can be successfully met by the algorithms. FS can handle only 25 requests, while WFS can handle 40 and

[1] 1 Mbps = 1 megabit per second = 1000×1000 bit per second
[2] 1 MiB = 1 mebibyte = 1024×1024 byte



(a) $M = 250$, $K = 360$ (b) $C = 50$, $K = 360$ (c) $C = 50$, $M = 250$

Figure 2. The **average** play overtime (in seconds) as a function of C, M and K, for EDF, FS and WFS

EDF 50. When K is increased to 900 seconds (not depicted) this increases even further for EDF to 70. This means that EDF can successfully handle up to 2.8 times as many requests as FS and 2.5 times as many as WFS.

A comparison between Fig. 1 and Fig. 2a, reveals that for EDF the average and maximum play overtime are very similar (at most a 3% difference). On the other hand, for FS and WFS this difference is up to 340% and 17% respectively. This poorly balanced behavior between the average and worst-case playtime of requests can be attributed to a bias towards requests with a large known time, as they will be served long before their first byte deadline is reached. EDF on the other hand will first serve only requests with a nearby first byte deadline. Additionally, FS gives the same amount of bandwidth to low bit-rate requests as to high bit-rate requests. Therefore, FS will perform well for the LD-type requests at the cost of HD-type requests.

Fig. 2b and Fig. 2c show that EDF only performs significantly better than the other scheduling algorithms if M and K are large enough. Consequently, in order for deadline-aware scheduling to be useful, the amount of available memory for buffering and the known time variation between requests should be large enough.

Finally, it is expected that if M increases, so will the efficiency of scheduling algorithms. However, Fig. 2b clearly shows that for FS and WFS this assumption does not hold. Their performance increases up to a certain point, but then decreases again. This is because, when memory is low, they cannot send much content for requests with a large known time, as the associated client buffer will quickly run full. However, when this is not the case, the fair-share algorithms will be able to serve large portions of large known time requests, at the cost of requests with an early deadline. FS and WFS thus become unknowingly deadline-aware when available memory is low.

VII. Conclusion

In this paper, we proposed a novel approach to delivering multimedia content in access networks. By way of intelligent nodes placed throughout the network, content can be strategically scheduled and cached in order to satisfy stringent Quality of Service demands. This paper proposes a generic memory management model, usable for many kinds of multimedia services, and a novel deadline-aware scheduling algorithm.

The algorithm was evaluated and compared to deadline-unaware approaches in an end-to-end scenario. This lead us to conclude that deadline-aware scheduling is mostly useful when enough client-side memory is available and the variation in known time between requests is large enough. Additionally, it was shown that, in the simulated scenario, deadline-aware scheduling could satisfy the deadline of up to 2.5 as many requests compared to the best performing deadline-unaware algorithm.

Acknowledgment

Jeroen Famaey is funded by the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT); Steven Latré and Tim Wauters are funded by the Fund for Scientific Research Flanders (FWO); This research was performed partially within the framework of the EUREKA CELTIC RUBENS project.

References

[1] M. Verhoeyen, D. De Vleeschauwer, and D. Robinson, "Content storage architectures for boosted IPTV service," *Bell Lab. Tech. J.*, vol. 13, no. 3, pp. 29–43, 2008.

[2] B. Krogfoss, L. Sofman, and A. Agrawal, "Caching architectures and optimization strategies for IPTV networks," *Bell Labs Technical Journal*, vol. 13, no. 3, pp. 13–28, 2008.

[3] J. Liu and J. Xu, "Chapter 1 proxy caching for media streaming over the internet ."

[4] T. Wauters, W. Van de Meerssche, F. De Turck, B. Dhoedt, and P. Demeester, "Co-operative proxy caching algorithms for time-shifted IPTV services," in *Software Engineering and Advanced Applications, 2006. SEAA '06. 32nd EUROMICRO Conference on*, 2006, pp. 379–386.

[5] W. Wang, W. Wang, J. Du, and T. Peng, "A fast packet scheduling algorithm to provide QoS for streaming service in shared channels," in *Communication Systems, 2008. ICCS 2008. 11th IEEE Singapore International Conference on*, 2008, pp. 901–905.

[6] S. Kozlov, P. van der Stok, and J. Lukkien, "Adaptive scheduling of MPEG video frames during real-time wireless video streaming," in *WOWMOM '05: Proceedings of the Sixth IEEE International Symposium on World of Wireless Mobile and Multimedia Networks*, 2005, pp. 460–462.

[7] Y. Li, Z. Li, M. Chiang, and A. R. Calderbank, "Video transmission scheduling for peer-to-peer live streaming systems," in *Multimedia and Expo, 2008 IEEE International Conference on*, 2008, pp. 653–656.

[8] M.-T. Lu, J.-C. Wu, K.-J. Peng, P. Huang, J. J. Yao, and H. H. Chen, "Design and evaluation of a P2P IPTV system for heterogeneous networks," *Multimedia, IEEE Transactions on*, vol. 9, no. 8, pp. 1568–1579, 2007.