Dynamic Cache Optimization for DASH Clients in Content Delivery Networks

Antti Heikkinen, Tiia Ojanperä and Janne Vehkaperä
VTT Technical Research Centre of Finland Ltd.
Kaitoväylä 1, Oulu, Finland
Email: {firstname.lastname@vtt.fi}

***Abstract*—Content Delivery Networks (CDN) play a key role in multimedia distribution in today's Internet in order to achieve more efficient bandwidth management, reliability, and Quality of Service (QoS). This paper proposes an advanced CDN solution for enhancing the delivery and caching of adaptive HTTP-based video streaming that dominates consumer video service distribution today. The paper also presents a testbed implementation of the proposed system, utilizing open source components and a standardized MPEG-DASH based video streaming solution. The proposed CDN management features and the associated signaling are inspired by the upcoming MPEG standard for Server and Network assisted DASH (SAND). The paper also presents selected results obtained from an experimental evaluation. The results attest the benefits of the proposed solution in enhancing MPEG-DASH delivery in CDNs for the selected test cases.**

Keywords*—Network-assisted video streaming, MPEG-DASH, SAND, CDN, testbed.

I. Introduction

Traffic growth in the Internet [1] is already causing difficulties for ISPs in maintaining high availability, reliability, and Quality of Service (QoS) for their customers. The situation is aggravated by the fact that the Internet traffic growth is increasingly dominated by bandwidth-consuming and QoS-sensitive video services [1]. End-users are also demanding all the time better Quality of Experience (QoE) for video services.Yet, the revenues of network operators are not increasing at the same pace, causing the available network bandwidth to lag behind the demand and resulting in poor QoE for the users. This unsustainable situation has forced network operators and service providers to look for cost-efficient means for managing the increasing traffic loads in networks in order to ensure customer satisfaction.

Content Delivery Networks (CDN) [2] are vastly used for improving the speed, accuracy, and availability of network-delivered content in the Internet. Nowadays, CDNs play a key role also in video service provisioning, as HTTP has become a de-facto standard in delivering Internet-based video. The popularity of video services and the need for service providers to ensure high-quality services for their customers are reflecting on the demand for CDNs, which is rapidly increasing [1]. Nevertheless, supporting video services efficiently and reliably in CDNs leaves places for optimization. Internet-based video today is largely of streaming type. Due to the heterogeneous nature of networks, varying network conditions, and diversity of end-user devices, many services implement adaptive video streaming technologies in order to provide good QoE for the end-users. Adaptive HTTP video streaming in particular causes CDNs to maintain multiple representations of the same video content which may easily create storage problems in caches. In addition, without any knowledge of the CDN and cache status, a video client's adaptation algorithm may perform inefficiently and even cause QoE degradations for the user [3]. Furthermore, CDNs need mechanisms for recovering from server failures as well as supporting dynamic scaling of resources in virtualized environments.

This paper presents a solution and testbed implementation for optimizing CDN operation for adaptive HTTP video streaming services in terms of caching and network resource usage as well as response time. This is accomplished by the advanced CDN and cache management features and associated signalling proposed in the paper for a video streaming system. The proposed solution utilizes the Dynamic Adaptive Streaming over HTTP (DASH) standard [4], and follows the specifications outlined by MPEG for the currently unfinished DASH-amendment titled Server and Network Assisted DASH (SAND) [5], while extending them in places. The rest of the paper is organized as follows. Section II presents the advanced CDN architecture. Section III introduces the testbed implementation and its experimental evaluation. Section IV concludes the paper.

II. Advanced CDN Architecture

The advanced CDN architecture proposed in this paper for video delivery is presented in Fig. 1. The main improvements compared to a traditional CDN are the content- and service-aware network elements and the content-aware caching mechanism. The architecture follows the upcoming MPEG standard for Server and Network assisted DASH (SAND) [5]. SAND introduces messages between DASH clients and (various) network elements that could improve content delivery in networks in terms of more accurate adaptation and reaction time. The network-side elements participating in the message exchange are called DASH assisting network elements (DANE). DANEs have at least a minimum intelligence about DASH but may also perform more complex operations that influence DASH content delivery. The advanced CDN architecture consists of the following components:

1) Origin server: is the source of the content. The video content is in MPEG-DASH format and it is delivered from the origin server to edge servers on a request.

2) Routing server: is responsible for the initial routing by directing clients to an edge server (ES) at the beginning of the session. The routing server recognizes the clients' IP addresses and returns the address of the ES which is closest to the client.

978-1-4673-9292-1/16/$31.00 ©2016 IEEE

3) Edge servers (ES): are located near the end-users and are in charge of receiving and handling the client requests. When a client requests data from the CDN, the selected ES distributes data to the client. If the requested data is not available in the ES, it requests the data from the origin server. The ES stores the requested data for a certain time. In the proposed architecture, ESs may contain content-aware network element and DANE (referred to as ES-DANE). A content-aware network element has at least minimum knowledge about the content it is processing or delivering.

4) Monitoring and management server (MMS): contains a DANE and collects data related to ESs, for instance, cache status, congestion or failure in the CDN. It is a SAND signaling and control point between the connected DASH clients and ESs. Based on the information received from the clients and ESs, MMS makes decisions regarding CDN optimization (e.g. dynamically redirects clients to the optimal ES). Depending on the CDN structure, the management may be realized hierarchically but this is not considered in detail in this paper.

5) DASH clients: may be either SAND-capable when it can receive optimized service from the CDN or a normal DASH client without any additional network-side support.

The advanced CDN architecture includes regular network elements and content- or service-aware network elements. The regular network elements deliver video services without any knowledge of the content. The content aware network elements have at least some knowledge about the content. The management signaling in the proposed architecture is depicted in Fig. 2. SAND or SAND-like signaling mechanism is used for communicating the Parameters Enhancing Delivery (PED), Parameters Enhancing Reception (PER), and Metrics messages between DANEs as well as DANEs and SAND-capable DASH clients. We adopt the main principles of SAND in our proposed solution in addition to using the same terminology. However, the paper also defines new messages in respect of the current SAND specification [5] in order to support the envisioned CDN optimizations. Devising a fully standard-compliant solution is left for future work once the SAND standardization is finalized.

A SAND-capable DASH client initiates the signaling by sending a Metrics message to the DANE running in the Monitoring and management server (MMS). MMS communicates with edge servers (ES) by sending PED messages based on the client's Metrics message. The MMS can, for instance, inquiry the cache status from an ES or the status of the ES, including for example its congestion, failure or network conditions. In addition, MMS can request pre-fetching of segments into the ES's cache based on the client requests and prevailing cache status. By using a single control point in the CDN, the ESs can be dynamically added or removed in the CDN without the need to update media presentation description (MPD) files. The DANE component in the ES executes the requested tasks based on the received control messages and sends a response to MMS. The ES's DANE has knowledge of the cache status of the corresponding ES and it can pre-fetch and delete segments in the cache depending on whether they are needed or not. MMS can make decisions based on the ESs' feedback and send PER messages to the SAND-capable DASH client(s), when needed.

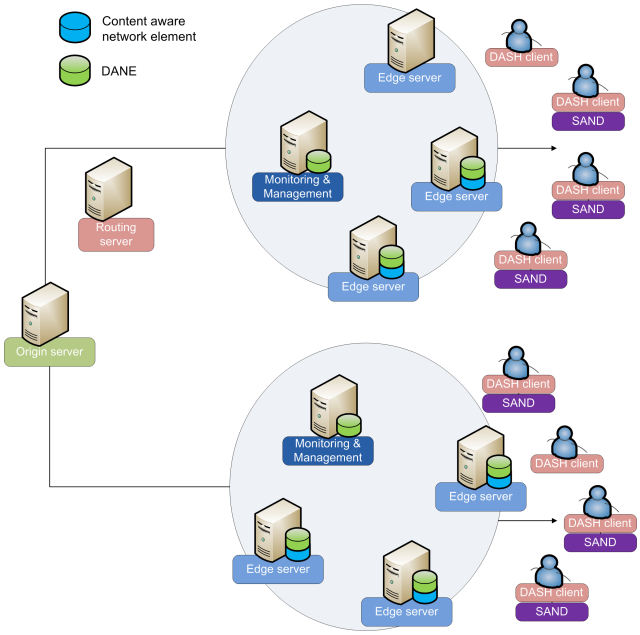

Fig. 1. The advanced CDN architecture.

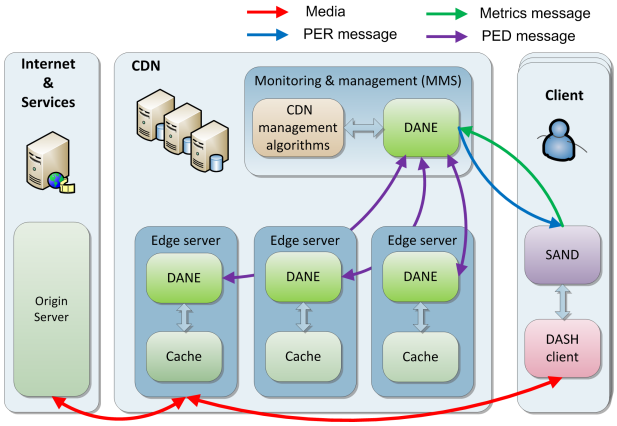

Fig. 2. Signaling in the advanced CDN architecture.

III. Experimental Evaluation

The proposed advanced CDN architecture is validated with an experimental evaluation conducted with a testbed implementation in this paper. This section describes the testbed implementation as well as the test cases considered in the evaluation. In addition selected results are presented.

A. Advanced CDN Testbed

We have built an advanced CDN testbed, by using off-the-shelf servers and open source software. All the servers run Ubuntu 14.04 OS. The origin and the two edge servers (ES1, ES2) run Nginx as the HTTP server. In the ESs, Nginx is configured to cache MPEG-DASH content from the origin server. In the single file case, where segments are distinguished by byte range in DASH, the ES caches each byte range into a single file. And in the case where a video file is partitioned into multiple files, each file is cached into a single file in the ES. Both ESs contain DANE component. The initial routing done

at the Routing server is based on DNS redirection. We have geo-aware DNS with BIND9 and GeoIP database. In addition we have dedicated MMS which includes DANE functionalities. All the servers are connected to a laboratory network and the ESs also to the Internet.

The communication is based on SAND-like messages and the WebSocket protocol. For the purposes of the implementation and testing, we have defined a generic format for the messages, that is, each message contains a Message ID, Sender, Receiver, and Payload fields. The Message ID is used for identifying different messages, Sender describes the message sender, Receiver describes the message receiver and Payload contains the actual information. The messages supported in the current testbed are listed in Table I. The WebSocket server address of the MMS is configured into the DASH clients.

The proposed content-aware features were implemented into ESs. The ESs can parse the MPD file format and are aware of which representations and segments belong to a video file. The ESs' DANE component contains a WebSocket server and it can receive messages from the MMS. By using information parsed from the MPD file, an ES can seek defined segments in the cache and pre-fetch segments from the origin server at the MMS's request. The MMS can also inquiry cache or server status from the ESs.

For testing the client-side operation, we implemented an application which simulates a DASH client (DashSimu). DashSimu does not decode or play the video but otherwise includes the same functionality as a normal DASH client. DashSimu contains a two-phase buffering and an adaptation algorithm. DashSimu supports the defined SAND-like messaging and runs a WebSocket client for sending Metrics messages to the MMS. MMS receives the Metrics message, and depending on the message, sends PED messages to the ES(s). The ES executes certain tasks based on the received message and sends a response to the MMS. The MMS includes a management algorithm which makes decisions based on the ES responses. The MMS sends PER message to DashSimu, if necessary. DashSimu reacts to the messages, for instance, by taking a new baseURL into use.

B. Test Cases

In order to demonstrate the different usages of the advanced CDN architecture and validate the testbed, we selected three test case for evaluation. For the tests, we used the "Tears of Steel" video sequence (12 min 14 s) in the MPEG-DASH format. The stream was encoded using the HEVC format into 4 different representations at 2, 4, 6, and 8 Mbps. All the representations were divided into 10 s segments. In the tests, we used the DashSimu application as a client where the initial buffer size was 10 s and total buffer 60 s. During the tests, we measured the network traffic on the CDN using Zabbix, which is an open source monitoring tool for networks and servers.

1) Test Case "Optimal ES Selection": focuses on directing DASH clients dynamically to the most optimal cache based on the prevailing cache contents. 20 new users want to start watching a video stream from the CDN. The CDN has 4 different bitrate representations (R) of the video available, and their availability at the ES caches depend on the client requests. When a new DASH client requests for the desired video URL, the request is directed to the ES1 by the routing server. The client then receives a MPD from ES1. The client sends a SAND Metrics message (SegmentRequest) informing that it will be using all the representations R1-R4 to MMS. MMS checks edge servers cache status by sending CacheStatusRequest to ES1 and ES2. ESs responds (CacheResponse) that the ES1 cache contains only half of the segments (percentage in CacheResponse message is 50), whereas ES2 contains the segments for all the representations (percentage in CacheResponse message is 100). MMS sends NewBaseURL messages to the client. The new baseURL indicates a location in ES2, from where client starts ordering segments. Fig. 3 shows the average incoming and outgoing traffic for each server on the CDN. We can see that some of the clients start ordering segments from ES1 before they receive a new baseURL. After one minute, all the clients order segments from ES2. We measured the network traffic on the CDN when the optimal edge server selection was disabled. Traffic between the origin server and ESs was only 0.09 Mbps with the optimal ES selection and 6 Mbps without it. Optimal ES selection thus improves cache hit ratio and reduces network traffic between origin and ES.

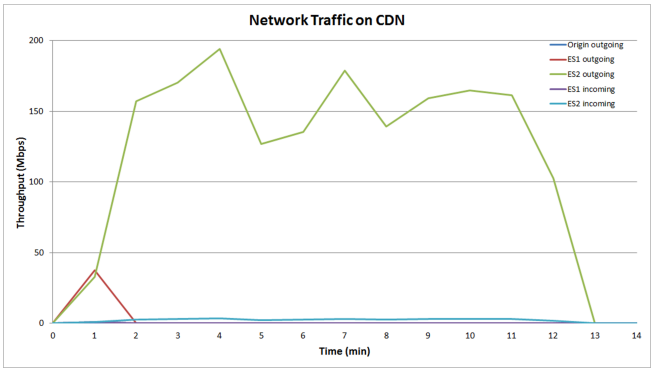

Fig. 3. Network traffic of 20 users on the CDN in the test case 1.

2) Test Case "Server Failure Recovery": 20 users is watching a video stream in R4 from ES1. After 5 minutes, ES1 experiences a severe network failure making it impossible for the DASH clients to retrieve any further segments from it. When a client notices a problem in the video streaming, it sends a ServerFailure message to MMS. MMS checks the status of each available ES by sending ServerStatusRequests to ES1 and ES2. MMS notices that ES1 is not responding whereas ES2 is in operation. After a timeout, the MMS concludes that ES1 is not OK and sends a NewBaseURL message to the clients, which start ordering segments from ES2 instead of ES1. We can see from Fig. 4 that the clients start ordering segments from ES1. When ES1 faces a severe failure, the amount of outgoing traffic drops. Almost at the same time, the clients start ordering segments from ES2 and all the clients take the new baseURL into use and change to ES2. After the change we can see a traffic peak caused by rebuffering. All the clients manage to take the new baseURL into use before their buffers are empty and they recover from the server failure without stalls in the video playout. Without the dynamic baseURL change capability, the clients would have stopped playing after the buffers run out.

3) Test Case "Next Segment Signaling": studies how a CDN can optimize the delivery of DASH content by pre-

TABLE I. SAND-LIKE CONTROL MESSAGES USED IN THE ADVANCED CDN TESTBED

Type	Name	Sender	Receiver	Payload	Description
Metrics	SegmentRequest	DashClient	MMS	mpdURL, repIDs, segmentRange	The list of segments the client would need
Metrics	ServerFailure	DashClient	MMS	mpdURL	A notification about detected server problems
PED	CacheStatusRequest	MMS	EdgeServer	mpdURL, repIDs, segmentRange	A query if the segments are in the cache
PED	CacheUpdateRequest	MMS	EdgeServer	mpdURL, repIDs, segmentRange	A request to pre-fetch segments into the cache
PED	ServerStatusRequest	MMS	EdgeServer	-	A query about a server's status
PED	CacheResponse	EdgeServer	MMS	mpdURL, repIDs, segmentRange, percentage	The percentage of the segments in the cache
PED	ServerStatusResponse	EdgeServer	MMS	status	The server's status, e.g. in operation, congestion
PER	NewBaseURL	MMS	DashClient	baseURL	The new baseURL for ordering segments

fetching segments into caches. 40 users start watching a video from the CDN. For 10 clients, adaptation is enabled and they are using all the 4 representations. 10 other clients use only R1, 10 only R2, and the remaining 10 clients only R3. When a DASH client requests for the desired video URL, the request is directed to the ES1 from where it receives the MPD. The client sends SegmentRequest message stating that it plans to requests for R1-R4 for the first n segments (n=12) to MMS. MMS checks edge servers cache status by sending CacheStatusRequest to ES1 and ES2. Both the ESs seek the segments in their cache without a match and respond with a CacheReponse where the percentage is 0. The MMS sends a CacheUpdateRequest to ES1 which pre-fetches the requested segments from the origin server and responds by sending CacheResponse where percentage is 100. Because the client is using ES1, the MMS does not have to send the NewBaseURL message to the client. When the second client sends the SegmentRequest listing 12 segments for R1-R4, the MMS sends a CacheStatusRequest with the same payload to ES1 and ES2. Now ES1 sends CacheReponse where percentage is 100. Hence, the MMS does not send CacheUpdateRequest to ES1 or a NewBaseURL message to the client. At n segment intervals, the clients send SegmentRequests listing the representations they might use for the n next segments. In Fig. 5 we can see a traffic peak from origin to ES1 when the pre-fetching occurs every two minutes. All the segments are available in ES1 when the clients request them and there is no cache miss. This minimizes the delay in transmission, and thus reduces any possible delay-induced oscillations in the bitrate adaptation.

IV. CONCLUSIONS

The paper presents a solution for optimizing CDN operation for adaptive HTTP video streaming services in terms of caching and network resource usage as well as response time. The paper also introduces a testbed implementation of the proposed advanced CDN solution. The testbed includes content- and service-aware network elements and utilizes SAND-like signaling. The paper includes preliminary experimental results obtained using the testbed, and the results attest the benefits of the proposed solution. The future work includes further development of the testbed implementation in order to support new use cases utilizing content-, service-, and network-aware decision-making in CDN and adaptive video streaming management. In addition, virtualization approaches for the CDN will be studied and developed in order to dynamically create and delete cache instances based on demand for further flexibility and optimization of the system.

ACKNOWLEDGMENT

The presented study was carried out in CELTIC-Plus NOTTS and H2B2VS projects and was partially funded by the Finnish Funding Agency of Technology and Innovation (Tekes). The authors would like to thank for the support.

REFERENCES

[1] Cisco, *Visual Networking Index: Forecast and Methodology, 2014-2019*, 2015.

[2] A. Vakali and G. Pallis, *Content Delivery Networks: Status and Trends*, IEEE Internet Computing, 7(6), pp. 68-74, 2003

[3] D.H. Lee, C. Dovrolis and A.C. Begen, *Caching in HTTP Adaptive Streaming: Friend or Foe?*, NOSSDAV'14, Singapore, 2014.

[4] ISO/IEC 23009-1:2014, *Information technology - Dynamic adaptive streaming over HTTP (DASH) Part 1: Media presentation description and segment formats*, 2014.

[5] ISO/IEC CD 23009-5:2015, *Information Technology Dynamic adaptive streaming over HTTP (DASH) - Part 5: Server and network assisted DASH (SAND)*, 2015.

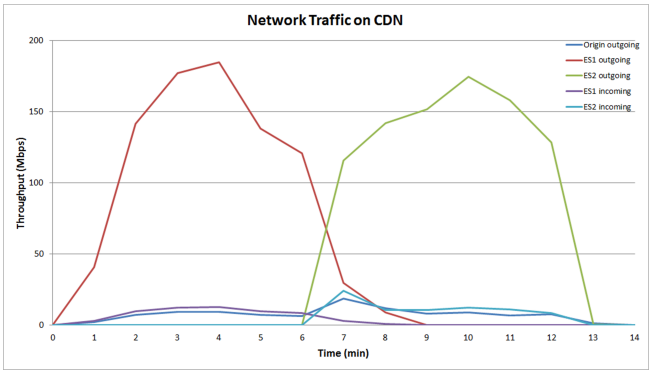

Fig. 4. Network traffic of 20 users on the CDN in the test case 2.

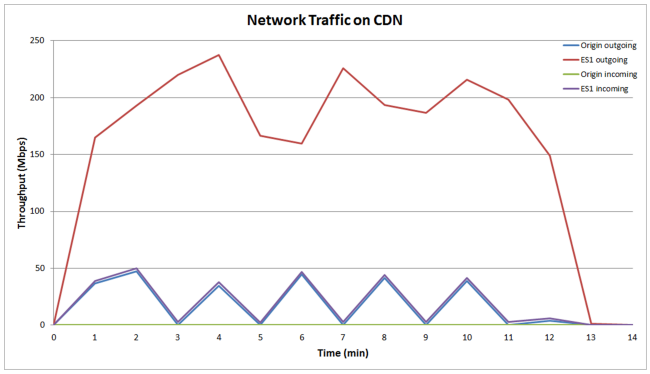

Fig. 5. Network traffic of 40 users on the CDN in the test case 3.