Adaptive Media Streaming over Emerging Protocols

Christian Timmerer, Christopher Mueller, and Stefan Lederer
bitmovin GmbH
Klagenfurt, Austria
{*firstname.lastname*}@bitmovin.net

Abstract - *The emerging MPEG standard Dynamic Adaptive Streaming over HTTP (MPEG-DASH) is designed for media delivery over the top of existing infrastructures and enables smooth multimedia streaming towards heterogeneous devices including both wired and wireless environments. The MPEG-DASH standard was designed to work with HTTP-URLs but mandates neither the actual version nor which underlying protocols to be used. This paper will provide a detailed introduction into emerging protocols (HTTP/2.0 and beyond) to be used in the context of adaptive media streaming, specifically DASH.*

INTRODUCTION

The Hypertext Transfer Protocol (HTTP) is currently one of most used protocols on the application layer as shown in [1]. In particular, real-time entertainment is one of the major drivers and currently accounting for more than 60% of the Internet traffic in North America's fixed access networks.

The adaptive delivery of multimedia content over HTTP is gaining more and more momentum, which resulted in the standardization of MPEG's Dynamic Adaptive Streaming over HTTP (DASH) [2]. The MPEG-DASH standard is designed for media delivery over the top of existing infrastructures and enables smooth multimedia streaming towards heterogeneous devices. In particular, it adopts the usage of HTTP-URLs to identify the segments available for the clients but neither mandates the actual version nor which underlying protocols to be used.

In this paper we describe the usage of MPEG-DASH within emerging protocols, namely HTTP/2.0 and Content-Centric Networking (CCN) based on [3][4].

DASH OVER HTTP/2.0

I. Introduction

This section describes HTTP/2.0 which is based on Google's SPDY protocol and at the time of writing of this paper available as Internet draft by the IETF [5].

The protocol is based on the Transmission Control Protocol (TCP) and maintains a single persistent connection for each session. During a session multiple streams can be opened between the client and the server in full-duplex mode. Typically, only one HTTP/2.0 connection between a server and a client exists until the client navigates to another server. The servers should leave connections open as long as possible until a given threshold timeout or when a client initiates a connection close.

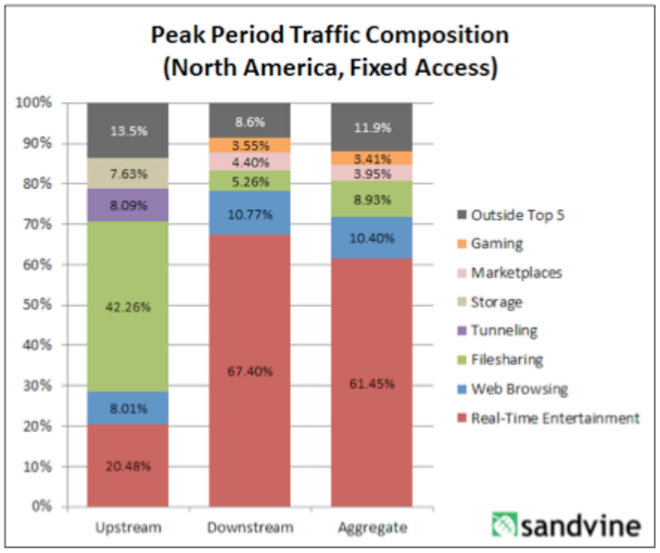

Figure 1. Peak Period Aggregation Traffic Composition - North America, Fixed Access [1].

The advantage of HTTP/2.0 is that it is fully compatible with HTTP/1.1 and can be integrated as a session layer between HTTP and TCP, hence, enabling incremental deployment. The HTTP request will be mapped into a HTTP/2.0 frame and vice versa for the HTTP response. Additionally, it is also possible to send multiple requests in parallel to support pipelining. Therefore, HTTP/2.0 offers an interface for HTTP which simplifies its integration for already existing HTTP applications. After this handover from HTTP/1.1 to HTTP/2.0 the whole communication will be handled on the HTTP/2.0 framing layer until a response arrives which will be passed to the HTTP/1.1 layer.

The HTTP/2.0 network communication is based on frames that are exchanged between the client and the server for a given TCP connection. The frame header is depicted in Figure 2.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| R |     Length (14)           |   Type (8)    |   Flags (8)   |
+-+-+-----------+---------------+-------------------------------+
|R|                 Stream Identifier (31)                      |
+-+-------------------------------------------------------------+
|                   Frame Payload (0...)                      ...
+---------------------------------------------------------------+
```

Figure 2. HTTP/2.0 Frame Header [5].

The fields indicated with *R* are reserved, with undefined semantics, must be set to zero, and ignored when receiving. The *Length (14-bit)* provides the length of the frame payload. The *Type (8-bit)* determines how the remainder of the frame header and the payload are interpreted. That is,

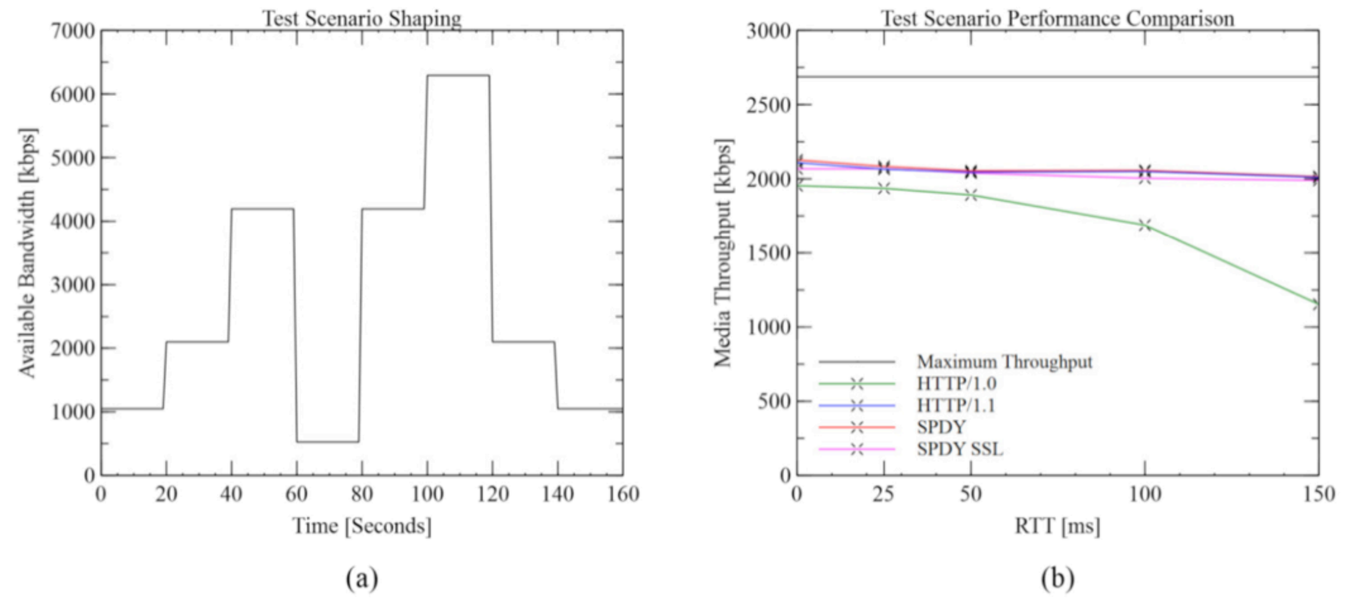

Figure 3. Average Media Throughput using SPDY given a predefined Bandwidth Trajectory for different RTTs [3].

also the *Flags (8-bit)* provide a set of frame-type specific Boolean flags. *The Stream Identifier (31-bit)* provides a mechanism to identify each stream within the persistent connection that may host multiple streams. Finally, the actual *Frame Payload* depends on the frame type.

Google further developed SPDY which is currently available as Draft 3.1 and still maintains two frame types for control and data frames but with similar functionality as within HTTP/2.0.

II. MPEG-DASH Client supporting HTTP/2.0

On the client side we have extended our open source MPEG-DASH VLC plugin with the most complete SPDY library, spdylay. The flexible architecture of the VLC plugin enabled a straightforward integration of SPDY and SSL-encrypted SPDY connections. Therefore, it is possible to objectively compare the performance of the MPEG-DASH client with different network protocols, i.e., HTTP 1.0, HTTP/1.1, SPDY, and SPDY with SSL encryption while maintaining the same behavior of the adaptation logic, buffer, etc.

Additionally, we have added a HTTP/2.0 branch to the MPEG-DASH reference access client libdash which now also supports SPDY and HTTP/2.0 respectively[1].

III. Experimental Results

In [3] we have performed various evaluations using the implementations outlined above in terms of overhead analysis and link utilization compared to existing HTTP versions. In this paper we focus on the client's media throughput given a predefined bandwidth trajectory within a test-bed comprising an HTTP/SPDY server, a DASH client, and intermediate network nodes to emulate the network conditions using a bandwidth shaper. For the details of the test-bed and evaluations in terms of overhead and link utilization, the interested reader is referred to [3].

Figure 3(a) shows the predefined bandwidth trajectory where the vertical axis describes the available bandwidth in kbps and the horizontal axis describes the time in seconds. Each experiment lasts exactly 160 seconds and the available bandwidth which is available during the experiment ranges from 1 Mbps to 6 Mbps. The content set provides 14 different media qualities ranging from 100 kbps to 4500 kbps which the client could individually choose at segment boundaries. All experiments have been performed with the same adaptation logic that is based on the buffer fill state and the measured throughput of the last segment. The buffer has been restricted for all experiments to 40 seconds. Each solution has been tested several times with different RTTs = 0, 25, 50, 100, and 150 ms and the average of each experiment is depicted in Figure 3(b). The vertical axis of Figure 3(b) shows the media throughput in kbps. It has been measured with the VLC MPEG-DASH plugin. The average maximum throughput has been calculated from the pre-defined bandwidth trajectory in Figure 3(a) which could be seen as the maximum achievable throughput. This line depicts the maximal achievable throughput without any overhead and optimal adaptation decisions, which is the upper bound for all transferring mechanisms in this laboratory setup.

Figure 3(b) shows that HTTP/1.1, SPDY, and SPDY with SSL encryption perform equally well and quite stable over all RTTs. As expected, HTTP 1.0 could not achieve the same media throughput especially for high RTTs due to the usage of one TCP connection per segment and slow start.

IV. Conclusions

HTTP/2.0 and SPDY operate on roughly the same performance as HTTP/1.1 with features such as persistent connections and request pipelining enabled. However,

[1] http://www.bitmovin.net/libdash.html (January 2014).

SPDY implicitly solves the Head-of-Line blocking problem of HTTP 1.0 and due to the lack of proper adoption of certain HTTP/1.1 features on caches it could definitely enhance the streaming performance of future networks. Interestingly, SPDY and SPDY with SSL encryption are very robust against increasing RTT because they are maintaining only one single TCP connection during the whole communication.

DASH OVER CONTENT-CENTRIC NETWORKING

I. Introduction

A variety of new Internet architectures have been proposed in the last decade and some of them seem to overcome the current limitations of today's Internet. One of these new Internet architectures is the Content Centric Network (CCN) approach [7], which moves the focus of traditional end-to-end connections to the content, rather than on addressing its location, i.e., devices in a network. CCN could eventually replace IP in the future, but it is also possible to deploy it on top of IP. In comparison to IP, where clients set up connections between each other to exchange content, CCN is directly requesting the content without any connection setup. This means that a client, which wants to consume content, simply sends an interest for this content into the network and the network responds with the corresponding content, wherever it may be located. Additionally, CCN is meant to provide security and trust as an integral part of the network.

Interestingly, CCN and DASH have several elements in common like, e.g., the client-initiated pull approach as well as the content being dealt with in pieces, and, thus, we have explored the possibility to integrate DASH with CCN.

In the CCN approach, there exist only two types of packets: *interest* and *data*. Interest packets are used for requesting the content whereas data packets are used for the actual data delivery. The maximum payload of a data packet is 4096 bytes and also referred to as a CCN chunk. Data packets are handled efficiently on the network nodes, e.g., to satisfy consolidated interest packets originating from multiple clients and, thus, providing implicit support for multicast and caching of data packets on CCN nodes within the delivery network.

II. Integration of DASH and CCN

In principle, there are two options to integrate DASH and CCN: (a) a proxy service acting as a broker between HTTP and CCN, and (b) the DASH client implementing a native CCN interface. The former transforms an HTTP request to a corresponding interest packet as well as a data packet to an HTTP response, including reliable transport as offered by TCP. The latter adopts a CCN naming scheme (CCN URIs) to denote segments in the Media Presentation Description (MPD). This requires an update to the MPD – including mitigating the requirement that only HTTP-URLs are allowed within the MPD – and the actual network component requesting the segments through CCN interest packets as well as handling the data packets.

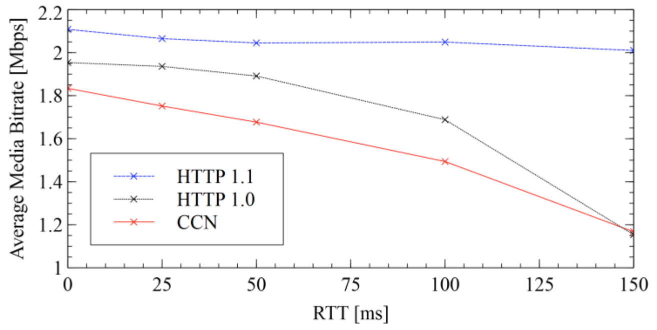

Figure 4. Average Media Throughput using CCN given a predefined Bandwidth Trajectory for different RTTs [4].

Initially, the DASH client retrieves the MPD containing the CCN URIs for the media segments. The naming scheme of the segments may reflect intrinsic features of CCN like versioning and segmentation support. Such segmentation support is already compulsory for multimedia streaming in CCN and, thus, can also be leveraged for DASH-based streaming over CCN. The CCN versioning can be adopted to signal different representations of the DASH-based content, which enables an implicit adaptation of the requested content to the client's bandwidth conditions. That is, the interest packet already provides the desired characteristics of a segment (such as bit rate, resolution, etc.) within the content name. Additionally, if bandwidth conditions of the corresponding interfaces or routing paths allow so, DASH media segments could be aggregated automatically by the CCN nodes, which reduces the amount of interest packets needed to request the content. However, such approaches need further research, specifically in terms of additional intelligence and processing power needed at the CCN nodes.

After requesting the MPD, the DASH client will start to request particular segments. Therefore, CCN interest packets are generated by the CCN access component and forwarded to the available interfaces. Within the CCN, these interest packets leverage the efficient interest aggregation for, e.g., popular content, as well as the implicit multicast support. Finally, interest packets are satisfied by the corresponding data packets containing the video segment data, which are stored on the origin server or any CCN node, respectively. With an increasing popularity of the content, it will be distributed across the network resulting in lower transmission delays and reduced bandwidth requirements for origin servers and content providers respectively.

III. Experimental Results

We have performed similar experiments as with HTTP/2.0 and detailed evaluation setup and results are available in [4]. The results for the average media throughput using the same predefined bandwidth trajectory as in the previous evaluation is depicted in Figure 4.

Considering that CCN is a new as well as experimental concept and the used CCNx implementation is a prototype and not integrated, e.g., into the system's kernel like TCP, the overall performance of DASH over CCN in terms of average media bitrate is relatively good. The difference of

117 kbps, or 6 %, to HTTP 1.0 and of 219 kbps, or 11 %, to HTTP/1.1 in the case of RTT = 0 ms is lower than expected, especially when considering the previously shown protocol overhead as well as the computational overhead at the CCN nodes, introduced by cache lookup, bloom filters, etc. Furthermore, the performance of DASH over CCN decreases monotonically in contrast to conventional DASH over HTTP 1.0, which finally leads to a 1 % better performance of CCN than HTTP 1.0 when the RTT increases to 150 ms. However, the CCN performance at this high network delay is still 735 kbps, or 39 % lower than HTTP/1.1 (i.e., using persistent connection and pipelining).

V. Conclusions

According to these results, the current CCN implementation has the potential to compete with DASH over HTTP 1.0, however, it definitely needs some work to come closer to the performance of HTTP/1.1. Therefore, the high overhead and the lower link utilization have to be addressed. In particular, the link utilization of DASH over CCN is influenced directly by the network delay. This influence can be reduced by improving the pipelining of CCN interest packets on the transport layer, as it is done by TCP. Furthermore, the performance can be increased by eliminating unnecessary interest packets at the end of the data transfer of a DASH segment.

The interest packets are requested in a pipelined manner, so non-existing data packets are requested while the last data packet containing the *FinalBlockID* field is received and processed by the client. These unnecessary interest packets can be easily avoided by sending the *FinalBlockID* field in an earlier data packet, e.g., the first one of the transfer, to notify the requesting node which is the last data packet. Of course, this also effects the bandwidth utilization in networks with higher delays, as instead of sending unnecessary interest packets, the client can already request data packets of the subsequent DASH segment and, therefore, reduce the time in which the link is unused. Additionally, research has to focus on a more efficient possibility for content encryption and signing which is currently mainly responsible for most of the header size and as a consequence also for the protocol overhead.

CONCLUSIONS

In this paper we have presented means for adaptive media streaming – specifically in the context of MPEG-DASH – over emerging protocols, i.e., HTTP/2.0 and CCN. The former is based on SPDY and is currently developed towards RFC within the IETF httpbis working group. Expected publication date of HTTP/2.0 is end of 2014. The latter is a hot research topic within the Future Internet research activity and pre-standardization efforts are conducted within the IRTF's Information-Centric Networking Research Group (ICNRG).

We conducted various evaluations based on a predefined test-bed and evaluation criteria showing the benefits of emerging protocols in the context of MPEG-DASH. The findings presented in this paper provide useful insights for current and future deployments of adaptive media streaming services over the top of existing infrastructures using HTTP and beyond.

REFERENCES

[1] Sandvine, "Global Internet Phenomena Report 2H 2013", *Sandvine Intelligent Broadband Networks*, 2013.

[2] Sodogar, I., "The MPEG-DASH Standard for Multimedia Streaming over the Internet", *IEEE Multimedia,* Vol. 18, No. 4, Oct.-Dec. 2011, pp. 62-67.

[3] Mueller, C., Lederer, S., Timmerer, C., Hellwagner, H., "Dynamic Adaptive Streaming over HTTP/2.0", *Proc. of International Conference on Multimedia and Expo (ICME) 2013*, San Jose, CA, USA, July 2013.

[4] Lederer, S., Mueller, C., Rainer, B., Timmerer, C., Hellwagner, H., "An Experimental Analysis of Dynamic Adaptive Streaming over Content Centric Networks", *Proc. of International Conference on Multimedia and Expo (ICME) 2013*, San Jose, CA, USA, July 2013.

[5] Belshe, M., et al., "Hypertext Transfer Protocol version 2.0", draft-ietf-httpbis-http2-09, Dec. 2013, http://tools.ietf.org/search/draft-ietf-httpbis-http2-09.

[6] Belshe, M., et al., "SPDY Protocol - Draft 3.1", http://www.chromium.org/spdy/spdy-protocol/spdy-protocol-draft3-1.

[7] Jacobson, V., et al., "Networking Named Content", *Proc. of the 5th Int. Conf. on Emerging Netw. Experiments and technologies (CoNEXT '09)*, ACM, New York, NY, USA, 2009.

AUTHOR INFORMATION

Christian Timmerer, bitmovin GmbH / Alpen-Adria-Universität Klagenfurt, Austria, Klagenfurt, christian.timmerer@bitmovin.net.

Christopher Mueller, bitmovin GmbH, Klagenfurt, Austria, christopher.mueller@bitmovin.net.

Stefan Lederer, bitmovin GmbH, Klagenfurt, Austria, stefan.lederer@bitmovin.net.