Paper

On Providing Cloud-awareness to Client's DASH Application by Using DASH over HTTP/2

Jordi Mongay Batalla[1], Piotr Krawiec[1], Daniel Negru[2], Joachim Bruneau-Queyreix[2], Eugen Borcoci[3], Andrzej Bęben[4], and Piotr Wiśniewski[4]

[1] *National Institute of Telecommunications, Warsaw, Poland*
[2] *CNRS-LaBRI, Bordeaux, France*
[3] *University Politehnica Bucharest, Bucharest, Romania*
[4] *Institute of Telecommunication, Warsaw University of Technology, Warsaw, Poland*

Abstract—Mobile Cloud Networks group together mobile users and clouds containing content servers. Hence, they are an ideal framework for media content delivery. Stream-switching adaptive video players cope well with some limitations of Mobile Cloud Networks as low bandwidth and bandwidth variability in access network. Nonetheless, other limitations, as cloud congestion, are difficult to be managed by the video players. This paper presents a system for discovering fault situations at the cloud (e.g., cloud congestion) and notifying to the video player, which will take appropriate actions for saving the quality of media transmission. In proposed implementation the video application is DASH-capable and adaptation action may be both stream rate adaptation and content server adaptation. The communication between client and server uses "bidirectional" communication feature of HTTP/2 thanks to the new deployed modules running DASH over HTTP/2 in both client's and server's applications.

Keywords—adaptive video streaming, DASH, HTTP/2.

1. Introduction

Mobile Cloud Networks (MCN) are a great opportunity for media delivery since they group together mobile users and media content servers (in cloud networks) under the same umbrella. However, the challenging issue still remains the quality of the delivery coming up to the user's expectations, without escalating the cost [1]. In fact, most of the recognized problems of MCN [2]–[5] directly affect the media streaming and should be solved for offering integrated solutions of media delivery. Specifically, the limitations of MCNs that mainly affect media delivery are:

- low bandwidth in the wireless access networks [2]. Even when 4G technology increased the bandwidth in mobile devices, the continuous raise of demand for mobile applications presents bandwidth limitations at users' disposal;
- excessive latency in wireless access networks [3], due to physical media delay caused by the low-quality connectivity with cell tower, and the latency of the protocol for access to the physical media;
- resource limitations of mobile devices [3], i.e., electric power, processing power and storage capacity. In the case of video applications, an added resource limitation in mobile devices is the poor display in comparison to laptops or TV;
- non-optimal management of access network resources [4], which influences negatively in the aforementioned points causing inappropriate access to the available bandwidth and increasing unnecessarily the latency in access networks;
- congestion of cloud networks [5], defined as the overload of any of the cloud resources (e.g., processing capacity, uplink bandwidth), which causes delay in the services damaging the delivery of media content and, as a consequence of this, impoverishing the quality of the media event.

Aforementioned limitations require extra-management mechanisms into the cloud directed to ensure appropriate user's satisfaction of the media event [6]. This paper centers on one of the mentioned limitations. It presents a management framework for discovering Cloud congestion situations and informing the user's video player about the predicted state of the cloud (avoiding to keep per-connection information in the cloud). The user's video player has then information for interpreting the cause of the reduction of bandwidth measured at the player. If the bandwidth reduction is caused by the cloud congestion, then the video player could adapt the video content server by switching the streaming to another server located in a different cloud. If the bandwidth reduction is caused by congestion in the user's mobile network, then the video player could adapt the streaming by reducing the media bitrate.
Even when the presented framework is independent of the video player (client application), the solution fits well in the case of stream-switching adaptive video players. Stream-switching adaptive players (e.g., Dynamic Adaptive Streaming over HTTP-DASH, Akamai HD Video Streaming – AHDVS, Adobe Dynamic Streaming and Apple HTTP Adaptive Live Streaming – HLS) request consecutive small portions of video (called chunks or segments), each one with the appropriate media rate (called representation rate).

The player decides both the representation rate and the content server from where the next segment will be downloaded, so it may adapt to the current state of the network and of the content server.

The paper is organized as follows. Section 2 gathers the state of the art of the proposed management framework and stream-switching adaptive protocols, whereas Section 3 provides some exemplary simulation results that prove the gain of considering the information about the state of the cloud into adaptation decisions. Section 4 presents the end-to-end framework considering both architecture and communication between involved entities and Section 5 shows implementation details of client's and server's applications deploying the presented framework. The proposed implementation has been optimized from the point of view of cloud management since authors avoid a separate channel for signaling between client's and server's applications. The communication uses the streaming channel instead, and it is based on DASH over HTTP/2, which makes bidirectional-like (initiated from the client or from the server) communication feasible.

An undoubted added value of this paper is the extension of *QTSamplePlayer* [7] for interworking with HTTP/2, which is available to the researchers in the web page: http://www.nit.eu/offer/research-projects-products/http2dash.

Results of the tests performed on the implemented software are presented in Section 6. At last, Section 7 summarizes the paper.

2. Background

Between all the limitations of MCN, stream-switching adaptive protocols cope pretty well with low bandwidth and bandwidth variability thanks to the adaptation mechanism that dynamically selects the media bitrate that fits better with the current download rate. Mobile networks with unmanaged and variable resources require from the client's application to constantly control the occupancy of its buffer [8] in order to avoid, from one side, re-buffering and, from the other side, suboptimal use of resources in the wireless link.

Even if stream-switching adaptive protocols are relatively new, the papers on this issue are countless. Until the publication of some open standards, the stream-switching adaptive protocols were, mainly, close solutions. Therefore, many of the first papers were based on inverse engineering, i.e., the analysis of existing solutions by testing them in different scenarios. A good recompilation of such analyses was made by Akhshabi *et al.* in [9], where the authors compared the adaptation mechanisms used by the most important service providers from the point of view of how aggressive/conservative they are in different scenarios. The authors of [10] compared the Smooth Streaming protocol which downloads video chunks periodically with traditional technique of continuous consecutive download. With the publication of the open MPEG standard Dynamic Adaptive Streaming over HTTP (DASH), the research centered on improving the adaptation algorithm in order to balance the stability of video playout and the stability of buffer occupancy. This trade raises due to the variability of download bandwidth. Since the instability of buffer occupancy is less harmful from the end user's point of view, the applications try, as the main goal, to trade off oscillations in video playout. In order to avoid such oscillations, Dobrian *et al.* [11] outlined the importance of the variability of the size of consecutive segments when estimating the download rate and Seo and Zimmermann [12] proposed to estimate the download rate from the rate measurements of a number of downloaded segments. The number of segments to be considered should be dependent of the state of the network. Other approaches for consolidating video playout consisted of monitoring the connections at the TCP level [8] or using not only measurements of download rate but also other measurements, such as packet losses, delay and TCP throughput [13], [14].

In MCNs the conditions, which the client's application should adapt to, are not controllable at the client's terminal. In other words, it is unlikely to differentiate the increase of the end-to-end delay from the decrease of bandwidth [11], so the congestion in the cloud is difficultly recognizable at the client's side. A more general vision about the situation of the server and network could improve the adaptation decisions. Some researchers have proposed to place the adaptation logic out of the end user's video player. For example Liu et al. proposed to deploy centralized video controllers with a global view of network and server conditions that may take decisions about adaptation for the clients [15]. Rejaie and Kangasharju proposed to introduce proxies for managing the quality of the media streaming at the network level [16]. At last, some papers showed several benefits of controlling the adaptation logic at the server side. For example, Sodagar [17] proposed to base adaptation decisions on the state of the sending buffer instead of the receiving one. Anyway, it seems that the adaptation logic should be located in the end user's video application, mainly by two reasons: the client application is in the best position to detect and respond to the dynamics on time and, on the other hand, there is a strong need for keeping minimal per-connection information in the cloud and servers [15].

In this paper a mechanism for informing the client about cloud congestion situations is proposed. With this information, the client may optimize the adaptation decisions by differentiating mobile access congestion from cloud congestion distinguishing two adaptation actions: media bitrate adaptation when there is mobile access congestion, and content source switching when there is cloud congestion. Moreover, the content server keeps no information about situation of the clients in presented solution.

3. Rationale

The simulation tests provided in this section aim to show that the adaptation decisions based only on user's side (video player) measurements are more error prone than in-cloud decisions assuming that further are supported by measurements at the network level (as it occurs in commercial cloud systems). Let us remark that the current simulation analysis does not aim to cover a wide range of cases and scenarios for offering exhaustive results about the necessity of considering the state of the cloud into adaptation decisions, but they only aim to confirm (in an example scenario) that more information (about the state of the cloud) is useful for taking better decisions about the media streaming.

With this scope, the downloading of segments and calculate the time of download of each segment (segment download time) in presence of background traffic is modeled. A posteriori (when the simulations are finished), the behavior of user- and cloud-based adaptation algorithms for the values of segment download rate obtained in the simulations will be compared.

The simulation scenario is shown in Fig. 1. The cloud uplink is modelled by a single server with infinite FIFO queue that serves (service time equal to 120 μs) the stream under test, which is composed of "segments" of 333 packets (10^5 of such segments), and the background traffic, which is Poisson traffic with a rate that varies for different tests: $R_{bg} = \{2,\ 3,\ 4,\ 5 \text{ and } 6\} \times 10^3$ packet/s. The packets of each segment of the stream under test arrive to the queue with an interarrival time equal to 12 μs (shaped packets arrival without considering TCP effects) and the first packet of each segment is sent only when the last packet of the previous segment finished its service in the queue.

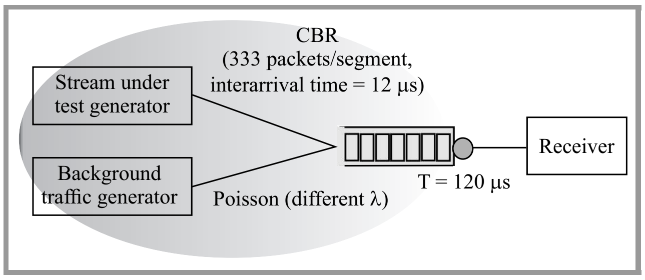
CBR
(333 packets/segment,
interarrival time = 12 µs)
Stream under test generator
Background traffic generator
Poisson (different λ)
T = 120 µs
Receiver

Fig. 1. Simulation model for rationale.

Note that if no background traffic were in the queue, then the complete segment would be downloaded in around 40 ms and the next segment could be sent immediately after that. Whereas, if there were 6×10^3 packet/s background traffic, then the segment would be downloaded in around 0.22 s, which means 7.5×10^3 packet/s of total traffic (stream under test and background) and server utilization $\rho \approx 0.9$.

The test is repeated for five values of background traffic rate indicated above. The mean rate of the stream under test (called *base rate*) equals $\{3.9,\ 3.3,\ 2.8,\ 2.2 \text{ and } 1.6\} \times 10^3$ packet/s, respectively for each aforementioned value of background traffic rate, which means queue utilization $\rho \approx \{0.7, 0.75, 0.8, 0.85 \text{ and } 0.9\}$, respectively. Let us remark that for each segment of the stream, the download rate is slightly different to the *base rate* due to the unpredictability of Poisson traffic.

When the simulations are finished, the download rate of each segment (10^5 segments in each one of the five tests), r^i, as the number of packets in the segment (333 packets) divided by the segment download time are calculated.

With the obtained values of segment download rate, the authors try to understand how the adaptation algorithm works in the case when it has no information about the state of the cloud (the server and the queue) and when it has this information. During adaptation decisions, three potential representations with rates: R_1, R_2 and R_3, where R_2 is the base rate minus 5% (of the base rate) are assumed. R_1 is the base rate minus 15% and R_3 is the base rate plus 5%, as indicated in Table 1. Note that R_2 is the reference representation, i.e., the representation that should be selected for all segments if the background traffic were Constant Bit Rate (CBR) instead of Poisson.

Table 1
Probability of erroneous decision

P	Representation (packets $\times 10^3$/s)	User-based P_{err}	Cloud-based P_{err}
0.70	$R_1 = 3.3$, $R_2 = 3.7$, $R_3 = 4.1$	9.7×10^{-3}	3.8×10^{-3}
0.75	$R_1 = 2.8$, $R_2 = 3.1$, $R_3 = 3.5$	1.1×10^{-2}	4.5×10^{-3}
0.80	$R_1 = 2.4$, $R_2 = 2.7$, $R_3 = 2.9$	1.2×10^{-2}	5.2×10^{-3}
0.85	$R_1 = 1.9$, $R_2 = 2.1$, $R_3 = 2.3$	1.4×10^{-2}	5.9×10^{-3}
0.90	$R_1 = 1.4$, $R_2 = 1.5$, $R_3 = 1.7$	1.5×10^{-2}	6.5×10^{-3}

For the adaptation algorithm which does not consider information about the state of the cloud (called user-based adaptation algorithm), authors assume an algorithm that adapts the representation bitrate of the next segment to the download rate of the last segment. This is, the representation selected for segment i, R_i, is calculated as:

$$R^i = \max_n \{R_n | R_n < r^{i-1}\}, \tag{1}$$

where r^{i-1} is the download rate of segment $i-1$.

The authors are conscious that the assumed adaptation algorithm is too simple (compared to commercial ones), but it is enough to compare user- and cloud-based adaptations. This adaptation algorithm is applied to the 10^5 ordered values of segment download rate r^i, obtained in the simulations. In this case, the representation R_1 (simulation results for total load in the server $\rho = 0.7$) will be selected for 472 segments (since 472 segments were downloaded with rate lower than R_2). The representation R_3 will be selected in 502 segments (since 502 segments were downloaded with rate higher than R_3). All the other segments will be requested with representation rate equal to R_2.

For analyzing the adaptation decisions, authors consider that the adaptation decision is erroneous when the selected representation for segment i, R^i, is higher than the download rate of segment i: r^i (resulted from the simulations). This is, the decision is incorrect if $R^i > r^i$. Let us remark that $R^i > r^i$ could cause image freezing in video players with short playback buffer. Note that R^i is a function of r^{i-1}, so the erroneous decision rate is closely related to the correlation of the segment download rate. The probability of erroneous decision for different values of ρ is presented in Table 1 (user-based column).

The second algorithm analyzed is the so-called cloud-based algorithm. It considers the information about the current state of the cloud and (based on historical data) the average conditions of the cloud. In the presented simulations, the average traffic is the same during all the simulations, so the average conditions (load, available bandwidth) of the cloud does not vary. Therefore, we assume that the cloud-based algorithm selects the same representation (R_2) for all the segments: $R^i = R_2$, $i = 1 \ldots 10^5$. Also in this case, we consider that taken decision for segment i was erroneous when $R^i > r^i$.

The probability of error (erroneous decisions divided by total decisions, i.e. 10^5) is presented in Table 1 for each of the five tests (different values of ρ), together with the values of the representation rates. As we can observe, the error for user-based decision is always much higher than in the case of cloud-based decisions since cloud-based decisions are based on the knowledge of the situation of the cloud bottleneck, unlike user-based decisions. For increasing values of ρ, the probability of erroneous decisions raises, which is explained by the higher variability of the state of the queue, which causes higher variability into the download time of the consecutive segments.

The authors are aware that the presented results are very dependent on the assumptions (especially on the assumed adaptation algorithms), but the aim of the simulation-based comparison was only to show that user-based decisions are less reliable since the user's video application does not have information about the bottleneck. A lector could find other algorithms that provide better results, but the conclusion would be the same. The cloud has information about the cloud's bottleneck that the user does not own, and such information may be useful for taking right adaptation decisions.

4. End-to-end Framework for Cloud-aware Adaptation

In order to provide awareness about the state of the cloud into adaptation decisions, authors propose that cloud system performs measurements at the cloud premises (generally, in the cloud access), which will be used for predicting the state of the cloud for the next few seconds (in proposed implementation the state of the cloud is predicted for the next 5 s). The information about potential restrictions of the cloud is then passed to the content server, which is responsible for sending it to the user's DASH application (client DASH application) by using the push function of HTTP/2. The DASH application, on its turn, will request the next segment by considering the state of the user's mobile access network (as measured by the DASH application) and the state of the cloud (as indicated by the information received from the content server). The decisions taken by the client application can be to perform the media bitrate adaptation or the content server adaptation (switching to another cloud for serving the request). Media bitrate adaptation is efficient if the congestion is in the wireless access since the unique possibility is to reduce bitrate for reducing congestion but, in the case of cloud congestion, better results by switching the content server while maintaining the previous media bitrate (saving the quality of the future streaming) can be obtained. Let us remark that both media bitrate and content server adaptation decisions should be in accordance with the original Media Presentation Description (MPD) file managed by the DASH application (different representations for media bitrate adaptation and different BaseURL tags for content server adaptation).

In order to obtain reliable information about cloud congestion, the system performs the next operations:

- The Monitoring Resource Mediator (MRM) (see Fig. 2) collects (in some time windows before a current instant of time) bandwidth information available at uplink of the cloud and processor load in the servers. Bandwidth information about aggregate traffic avoids potential problems of scalability during the measurements. This information is stored in the monitoring database at the Cloud Manager (CM). Let us remark that commercial clouds actually contain MRMs that monitor the state of the links.

- At time t, the Traffic Forecast within the CM (using the collected data read from the monitoring database) provides small-term bandwidth forecast for the next T seconds, i.e. $[t,\ t+T]$, and decides whether the cloud will experience in that time an over-load (congestion) that will be able to impoverish the quality of the media transmission. In that case, the CM informs the content server and the latter sends information to the client DASH application. The system finishes the above steps before time t, so that an over-load alert can be sent to the DASH application at time t for the period $[t,\ t+T]$, after which the above process is repeated for the next period $[t+T,\ t+2T]$. Let us remark that the time T is not synchronized with the segment duration or segment download duration but the two operations (DASH streaming and cloud congestion control) work independently.

Figure 2 shows the entities involved in providing cloud-awareness to the user's terminal.

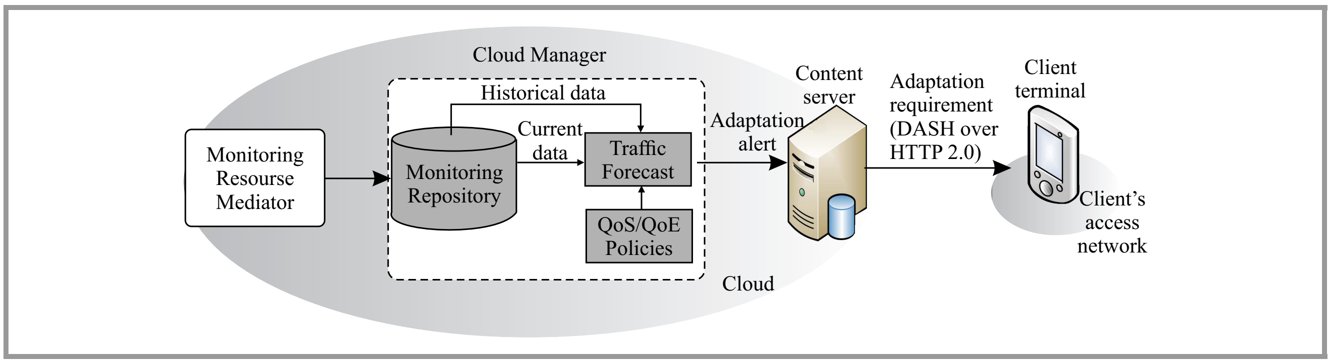

Fig. 2. Framework for cloud-aware adaptation.

The MRM is the monitoring tool that provides information about the traffic in selected points of the cloud (especially in the uplink of the cloud) and about the state of the processors in the servers. Many enterprize clouds, open-source clouds and CDNs provide a multimodal selection of possible monitoring metrics that span beyond the CPU utilization and Bulk Transport Capacity up to more detailed per service or user-defined metrics (e.g. [18]). In this case, the measurements used for predicting over-load of the cloud are the bandwidth of the aggregate (at cloud uplink) and the number of new requests arrived to the cloud during the last 30 s. The commercial Riverbed Stringray Traffic Manager tool [18] provides the requested measurements. Two different kinds of measurements are necessary for the traffic forecast algorithm: the current bandwidth measurements and the historical data (stored into the Monitoring Repository), as shown in Fig. 2.

The Traffic Forecast module has to efficiently predict the needed bandwidth capacity in the cloud based on observed fluctuations of cloud resources and to conclude which situations may lead to cloud congestion. For the implementation of the traffic forecast algorithm, a similar approach as in [19] is proposed but considering each http request (during one video session) as a separate video channel. The algorithm estimates the bandwidth required to the server (b_T) during the next instant of time, T, according to estimated values of active population (N_T) and target download rate which, on mean, each user will require (R), as indicated in Eq. (2).

$$b_T = R \times N_T \,. \tag{2}$$

Generally, the estimation of the population is based on the past measurements of population that are downloading content (active population) during a long period of time. These population time series are processed by using different mathematical techniques (e.g. Box-Jenkins) in order to eliminate periodicity and trends related to specific periods of time (e.g. daily periods) [19]. In this way, the data can be used independently of the moment when they were taken. The output of these operations can be characterized by autoregressive moving-average (ARMA) model. The characteristics of the past active population define the population during the next instant of time.

In presented implementation the active population that will download a content (N_T) to the server as the average of the measurements of last 30 s (which is the reference value of playback buffer for many video players) is calculated, and it is assumed that during this short period there is no periodicity trend that could have negative influence into the prediction. Note that the estimations presented are very sensitive to error since they use short-term measurements, but let us remember that the implemented system is a proof of concept. Its deployment in commercial networks should consider more sophisticated (by using long-term processed measurements) forecast algorithms.

The target download rate R is calculated as follows. Let us assume that the server has c different contents with the same popularity. Each content c contains i different representations with rate equal to R_{ic}. Then the average of the target download rate is:

$$R = \frac{1}{C} \times \sum_{c=1}^{C} \frac{\sum_{i=1}^{I_c} R_{ic}}{I_c} \,. \tag{3}$$

If the predicted bandwidth goes beyond a given threshold (90% of the uplink bandwidth of the server in author's implementation), then the traffic forecast algorithm predicts cloud congestion in the next time slot. In this case, the cloud manager contacts the content servers in order to inform about the situation. The interface implementation between CM and content server is based on JSON/RPC protocol. The content server contacts the users' terminals, which the server is actually serving in order to inform about the congestion situation. The server could decide to send such information only to a number of clients (e.g. one of five clients) in order to avoid avalanche situations (all the clients switch to another cloud). The communication between server and clients is based on the push functionality of HTTP/2. Details of such a communication are given in the Section 5.

At last, the client DASH application is responsible for taking the final decision about adaptation. Such a decision may include:

- media adaptation, i.e. switching media stream to lower representation without changing content server, as it is illustrated in Fig. 3,

- content server adaptation, i.e. downloading further segments from another content server (from the set of available servers specified in *MultiBaseURL* element into MPD) leaving representation unchanged.

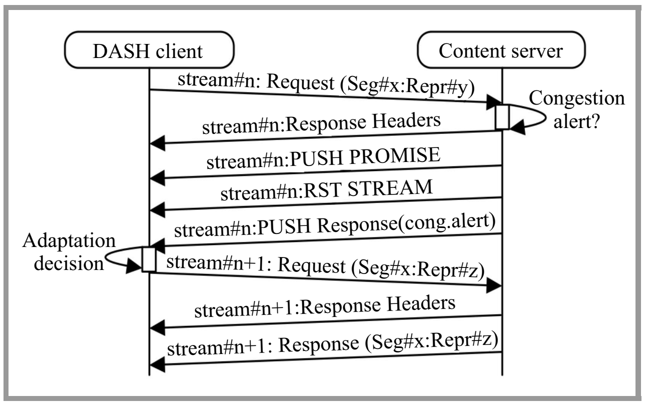
DASH client
Content server
stream#n: Request (Seg#x:Repr#y)
Congestion alert?
stream#n:Response Headers
stream#n:PUSH PROMISE
stream#n:RST STREAM
stream#n:PUSH Response(cong.alert)
Adaptation decision
stream#n+1: Request (Seg#x:Repr#z)
stream#n+1:Response Headers
stream#n+1: Response (Seg#x:Repr#z)

Fig. 3. Sequence diagram for push-based congestion alert.

For this, the DASH application should implement an enhanced adaptation algorithm that may adapt not only Media bitrate but also content server (e.g. to select another cloud for streaming the content) considering the information gathered by the same application and the information arrived from the content server. An example algorithm fulfilling such functionalities was presented in [20]. The selection of the server is, generally, a blind decision in the DASH application. It is responsibility of the service provider to inform the application about the characteristics of the new servers, but this is out of the scope of this paper. In case when DASH application switches the content server (content server adaptation), it should avoid to come back to previously used server, from which the client downloaded a previous segment, in order to avoid ping pong effect of endless switching between two overloaded clouds.
The scalability of the solution is ensured due to the fact that the unique interchanged information is about the cloud congestion, which is not specific per video session. cloud congestion can be predicted only by taking measurements of the aggregate traffic and marked flows, which saves most of the potential scalability issues in the monitoring tool. Moreover, the network measurements are taken in specific nodes, which are the bottlenecks of the cloud system (e.g. uplink). These points are well-known to the cloud provider and are constantly controlled by traffic manager tools.

5. Implementation Details of DASH over HTTP/2 Module

HTTP/2, which is still under development within the IETF HTTPbis Working Group, is a binary protocol that aims at better utilization of network capacity than previous versions while preserving compatibility with the transaction semantics of HTTP 1.1. HTTP/2 introduces a framing layer between HTTP and TCP used for multiplexing several HTTP requests into one TCP connection. HTTP/2 provides efficient header compression in order to reduce the protocol overhead [21] and also proposes server push mechanism, which allows a server to send a response without an explicit request from the client. In [22], the authors employ the server push to decrease media delivery latency in DASH live video streaming. In presented approach, authors apply the push feature to transfer information about cloud congestion from server to client without the need for client's request.
Figure 3 presents the communication between DASH client and content server assuming that the server received a congestion alert from the CM. The client DASH application downloads media segments in separate streams. The content server, after receiving a request for a segment, checks if during the time, which elapsed from the previous request, the CM signaled a congestion. If no, the server returns requested segment. If yes (what is depicted in Fig. 3), the server responses to the request with Headers frame and next sends Push_Promise frame to notify that the server intends to initiate new stream for "pushed" data. Then, the server sends RST_Stream header to reset current stream, followed by Push Response, which carries information about congestion. Canceling the current stream allows for faster reaction to congestion alert since the client does not need to wait with adaptation process until the whole required segment (Seg#x:Repr#y in Fig. 3) will be downloaded. Such process delays downloading of the current segment by Round Trip Delay required for transferring, one by one, Push_Promise, RST_Stream, Push Response frames and a new request for the segment. On the other hand, the client may ask for the new adapted segment (Seg#x:Repr#z in Fig. 3) just after receiving Push Response. Seg#x:Repr#z is the result of adaptation decisions after receiving cloud congestion alert (media bitrate adaptation or content server switching).
Let us remark that server push action can be executed only when server receives a segment request from the client (as the "supplementary response" to this request) since in HTTP client-server scheme an exchange of messages is initiated solely by the client.

5.1. DASH HTTP/2 Client Implementation

In the above-mentioned papers [21] and [22], the authors used implementations of SPDY protocol to develop HTTP/2-compilant DASH applications. Although HTTP/2 and SPDY have equivalent functionalities (Google's SPDYv2 protocol was chosen as the basis for HTTP/2), they are incompatible due to, for example, different header compression mechanisms (GZIP in SPDY, whereas HTTP/2 uses dedicated HPACK scheme). Therefore, authors have implemented own version of HTTP/2 DASH client using *nghttp2* library [23], which is compliant with IETF HTTP 2.0 Draft v13 [24].
For this purpose, the SPDY-based QTSamplePlayer, an open-source DASH application provided by Bitmovin [7] (which bases on *libdash3* library) has been extended, by implementing new class HTTP2Connection responsible for

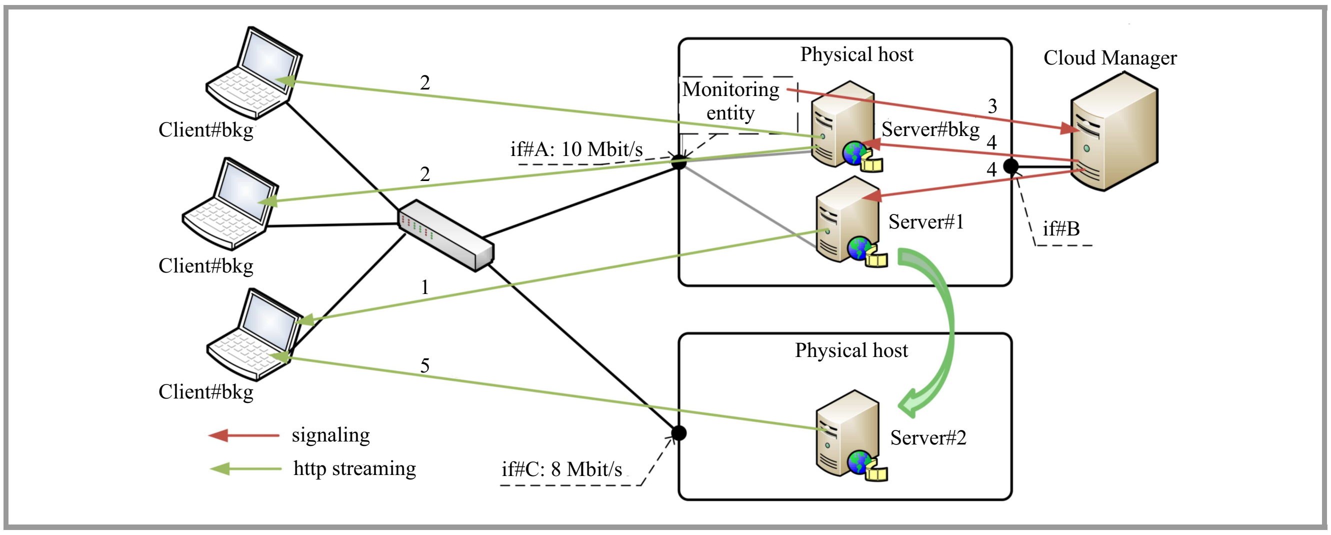

Fig. 4. Experimental setup for evaluation tests.

establishing and handling HTTP/2 connections. Moreover, we modified *DASHReceiver* module of the QTSample-Player to include *adaptationAlert()* method for interrupting the current segment downloading process whenever a server push occurs.

5.2. Content Server Implementation

The HTTP/2 content server was implemented based on *nghttpd* server implementation provided by *nghttp2* [23]. The authors deployed *AlertModule*, which contains JSON-RPC server for receiving congestion alerts from the CM. The *havePushData()* method of the *AlertModule* is called by main thread of content server whenever a new request for media segment arrives (which means opening a new stream). This method checks if a congestion alert from the CM exists and, in positive case, it records the label of congestion alert in order not to propagate the same alert in the future and returns data which should be pushed to the client. When the response of the *havePushData()* is positive, the server triggers push procedure and, at the same time, cancels the stream related with media segment request. Let us remark that additional functionality results in a very low overhead comparing to standard HTTP/2 server. This overhead is related with receiving a JSON message (of small size) from CM and performing one extra step in client's request handling flow to check if there is a message to be pushed.

Source code for both implementations (client and server), are available on web page http://www.nit.eu/offer/research-projects-products/http2dash.

6. Test Results

The experimental setup, presented in Fig. 4, includes two physical hosts with three virtual machines (labeled as server#bkg, server#1 and server#2) containing our afore-mentioned implementation of content streaming server. Each physical host emulates one separate cloud domain characterized by own IP prefix.

The servers in the first cloud are connected through a link that is constantly monitored (monitoring entity is located on physical host output interface if#A, see Fig. 4). The available bandwidth in the output interfaces if#A and if#C was restricted to 10 Mb/s and 8 Mb/s, respectively, by using the Linux Traffic Control system (*tc* command). The servers provide media content (Big Buck Bunny movie [25]) with 15 different representations, from 100 Kb/s up to 6 Mb/s, divided into segments of two seconds duration.

The client applications, client#1 and two client#bkg, run under Linux Ubuntu 14.04. The adaptation algorithm in the DASH client applications is based on mean download rate but it has been modified in order to switch the content server whenever congestion information arrived from the server (see [20]). Just after connecting with the server, the clients increased its HTTP/2 flow control window from default value equal to 64 KB up to 1048 KB using a Settings frame. In this way, streaming stop is avoid due to exhaustion of client's window space, and also we limit the number of Windows_Update frames generated by the clients, which indicate how many bytes the server is permitted to transmit.

The server#1 starts streaming the content to client#1 (arrow no. 1 in Fig. 4) with the highest representation. At second 60, both client#bgk start downloading the same content from server#bgk located at the same cloud domain as server#1 (arrows no. 2 in Fig. 4), so the uplink of the cloud becomes overloaded. The monitoring information of if#A arrives to the CM (arrow no. 3 in Fig. 4), which determines that there is bandwidth congestion since the occupancy of the interface is higher than 0.9 (simple prediction algorithm created for testing purposes). Then, the CM sends an alert about cloud congestion to the content servers: server#1 and server#bgk (arrow no. 4 in Fig. 4). The next request for media segment arrived to server#1 and server#bgk are used to perform server push to the clients. The DASH adaptation

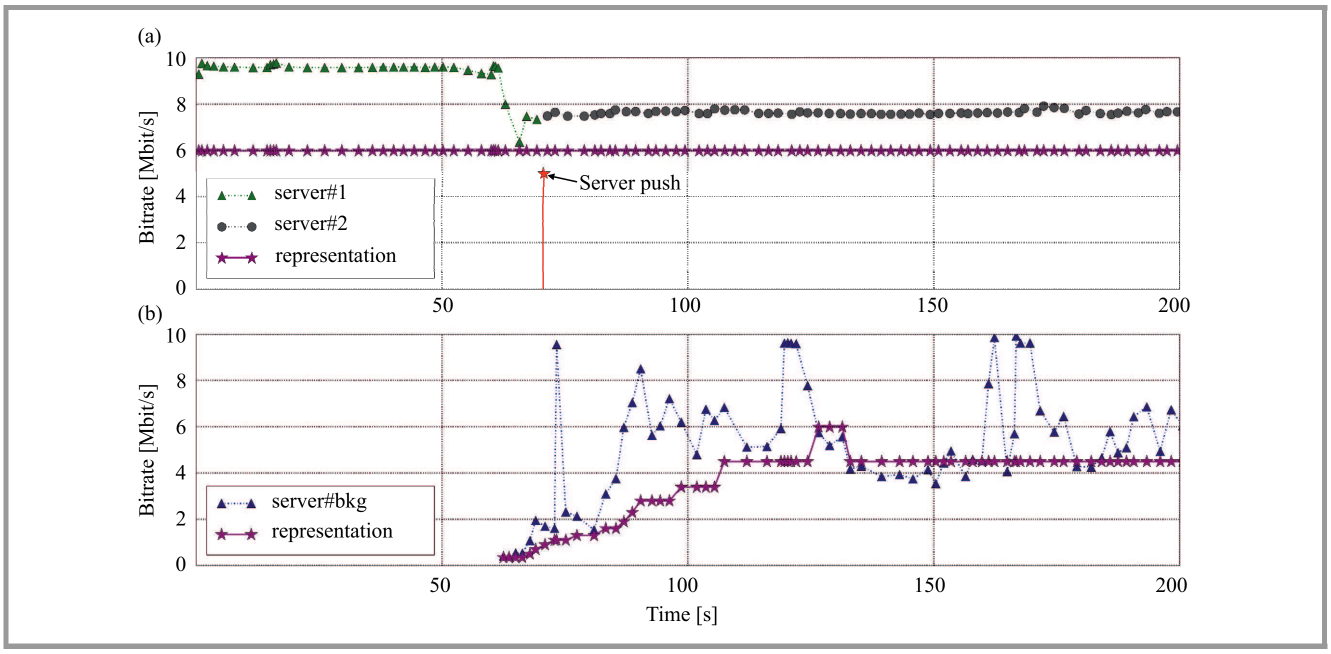

Fig. 5. Segment download rate with cloud-awareness.

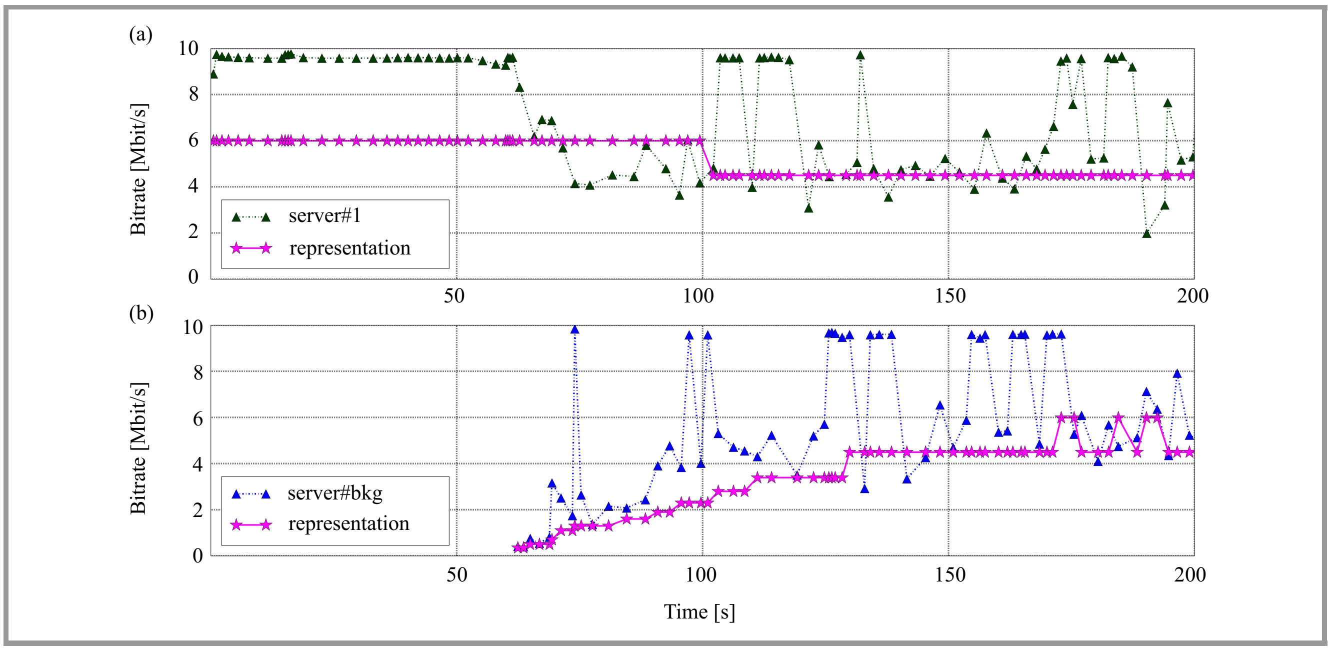

Fig. 6. Segment download rate without cloud-awareness.

mechanism at the client#1 terminal receives information about cloud congestion and performs content server adaptation by switching the streaming to server#2, as it is signaled by arrow no. 5 in Fig. 4. However, for demonstration purposes both client#bkg applications received MPDs without *MultiBaseURL* option, therefore they performed media adaptation only (without content server adaptation).

Figure 5a shows download rate of each segment received by client#1 from server#1 and server#2, as well as the representation rate selected by the client for each segment. The moment when server push occurred is indicated by a red vertical line in Fig. 5a (pointed also by an arrow). Figure 5b shows the segment download rate and selected representation for one of client#bkg.

As may be observed in the results presented in Fig. 5a, thanks to the cloud congestion notification, client#1 could switch the content server while maintaining the same (highest) representation during the whole downloading process. Moreover, thanks to prediction algorithm, the adaptation algorithm is able to react fast to the congestion situation (only 3 segments from the moment when server#bkg started streaming to client#bkg). Both client#bkg compete for the if#A bandwidth, so they download the content with lower representation (Fig. 5b).

Figure 6 shows the same scenario but, in this case, the cloud congestion information is not sent to the server, so DASH application of client#1 performs media bitrate adaptation with a delay of 14 segments from the moment when server#bkg started streaming (client#1 had to wait for collecting enough measurement data to discover bandwidth decrease). This delay may result in image freezing during video playout, if only the client buffer size is not enough to compensate degradation of downloading conditions.

By comparing Figs. 5 and 6, it may be concluded that cloud-awareness improves the performance of the system enhancing QoE due to the predictive feature that allows to fast reaction from the DASH application. Moreover, content server adaptation allows maintaining higher media bitrate by switching the transmission to another (non-overloaded) cloud.

7. Conclusions

The system presented in this paper allows for communication between Cloud Manager and video player in the end user's terminal by means of the content server. Such a communication is used for notifying situations of cloud congestion foreseen for close time. This way, the client video application may take proper decisions about adaptation taking into account both bandwidth limitations in the mobile access and congestion situations in the cloud. The results obtained by means of the system implemented on DASH-capable video player and DASH-capable server, present the applicability of the proposed system in situations of congestion in the cloud and compare the same situation when the DASH application does not own information about congestion. In the latter case, the application might not adapt bitrate in time, which would cause frozen image (in the case of strong degradation in the cloud). Moreover, making a distinction between cloud congestion and mobile access congestion allows for dual adaptation (media bitrate and content server), which may improve the quality of the media event experience.

Two issues will be addressed in planned future work: the use of bidirectional feature of HTTP/2 communication for sending dynamic MPD from the server to the client's DASH application and new dual adaptation algorithms in DASH application that integrate together rate measurements at the video player and information about congestion arrived from the Cloud Manager.

Acknowledgements

This work is supported by the European research project DISEDAN (http://wp2.tele.pw.edu.pl/disedan/) under the CHIST-ERA framework program.

References

[1] T. Jursonovics and S. Imre, "Quality-based charging solutions for wireless multimedia services", *Int. J. Netw. Manag.*, vol. 24, no. 5, pp. 357–401, 2014.

[2] M. Mehta, I. Ajmera, and R. Jondhale, "Mobile cloud computing", *Int. J. Elec. Commun. Engin. & Technol.*, vol. 4, no. 5, pp. 152–160, 2013.

[3] S. Qureshi *et al.*, "Mobile cloud computing as future for mobile applications – Implementation methods and challenging issues", in *Proc. IEEE Int. Conf. Cloud Comput. & Intell. Syst. CCIS 2011*, Beijing, China, 2011, doi: 10.1109/CCIS.2011.6045111.

[4] N. Fernando, W. L. Seng, and W. Rahayu, "Mobile cloud computing: A survey", *Future Gener. Comp. Sys.*, vol. 29, no. 1, pp. 84–106, 2013.

[5] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches", *Wirel. Commun. & Mob. Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013.

[6] J. Famaey and F. De Turck, "Federated management of the Future Internet: Status and challenges", *Int. J. Netw. Manag.*, vol. 22, no. 6, pp. 508–528, 2012.

[7] GitHub repository for bitmovin libdash library [Online]. Available: https://github.com/bitmovin/libdash/tree/http2 (last access: Aug. 2015).

[8] C. Dovrolis, M. Jain, and R. Prasad, "Measurement tools for the capacity and load of Internet paths" [Online]. Available: http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/bw-est/ (last access: Aug. 2015).

[9] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http", in *Proc. 2nd Ann. ACM Conf. Multimed. Syst. MMSys 2011*, San Jose, CA, USA, 2011, doi: 10.1145/1943552.1943574.

[10] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "What happens when http adaptive streaming players compete for bandwidth?, in *Proc. 22nd Int. Worksh. Netw. Oper. Sys. Supp. for Digit. Audio Video NOSSDAV'12*, Toronto, Ontario, Canada, 2012, doi: 10.1145/2229087.2229092.

[11] F. Dobrian *et al.*, "Understanding the impact of video quality on user engagement", in *Proc. ACM SIGCOMM 2011 Conf.*, Toronto, ON, Canada, 2011, doi: 10.1145/2043164.2018478.

[12] W. C. B. Seo and R. Zimmermann, "Efficient video uploading from mobile devices in support of http streaming", in *Proc. 3rd Ann. ACM Conf. Multimed. Syst. MMSys 2012*, Chapel Hill, NC, USA, 2012, doi: 10.1145/2155555.2155589.

[13] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction", *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, 2010, doi: 10.1109/TNET.2009.2037812.

[14] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer tcp throughput", *ACM SIGCOMM Comp. Commun. Rev.*, vol. 35, no. 4, pp. 145–156, 2005.

[15] X. Liu *et al.*, "A case for a coordinated internet video control plane", in *Proc. ACM SIGCOMM 2012 Conf.*, Helsinki, Finland, 2012, doi: 10.1145/2342356.2342431.

[16] R. Rejaie and J. Kangasharju, "Mocha: A quality adaptive multimedia proxy cache for internet streaming", in *Proc. 21st Int. Worksh. Netw. Operat. Syst. Support for Digit. Audio and Video NOSSDAV'11*, Vancouver, BC, Canada, 2011, doi: 10.1145/378344.378345.

[17] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet", *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[18] Riverbed Stingray Traffic Manager [Online]. Available: http://www.riverbed.com/

[19] D. Niu, Z. Liu, B. Li, and S. Zhao, "Demand forecast and performance prediction in peer-assisted on-demand streaming systems", in *Proc. 30th IEEE Int. Conf. Comp. Commun. IEEE INFOCOM'11*, Shanghai, China, 2011, doi: 10.1109/INFCOM.2011.5935196.

[20] J. Mongay Batalla and S. Janikowski, "In-segment content server adaptation for dual adaptation mechanism in DASH", in *Proc. 5th Int. Conf. Comput. Intell., Commun. Syst. Netw. IEEE CICSyN 2013*, Madrid, Spain, 2013, doi: 10.1109/CICSYN.2013.16.

[21] C. Mueller, S. Lederer, C. Timmerer, and H. Hellwagner, "Dynamic adaptive streaming over HTTP/2.0", in *Proc. IEEE Int. Conf. Multim. & Expo ICME 2013*, San Jose, CA, USA, 2013, doi: 10.1109/ICME.2013.6607498.

[22] W. Sheng and V. Swaminathan, "Low latency live video streaming over HTTP 2.0", in *Proc. 24st Int. Worksh. Netw. Operat. Syst. Support for Digit. Audio and Video NOSSDAV'14*, Singapore, 2014, doi: 10.1145/2578260.2578277.

[23] nghttp2 – HTTP/2 C Library. Project webpage [Online]. Available: http://nghttp2.org (last access: Aug. 2015).

[24] M. Belshe *et al.*, "Hypertext Transfer Protocol version 2", IETF HTTPbis Working Group Internet-Draft, 2014 [Online]. Available: http://tools.ietf.org/html/draft-ietf-httpbis-http2-13

[25] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP Dataset", in *Proc. 3rd Ann. ACM Conf. Multimed. Syst. MMSys 2012*, Chapel Hill, NC, USA, 2012, doi: 10.1145/2155555.2155570.

Jordi Mongay Batalla received his M.Sc. degree from Universitat Politecnica de Valencia (Spain) in 2000 and Ph.D. degree from Warsaw University of Technology (WUT) in 2009. His work experience includes jobs in Centro Nazionale di Astrofisica in Bologna, Italy as well as Telcordia Poland. Currently, he is with National Institute of Telecommunications as Head of Internet Architectures and Applications Department. He has also an Associate Professor position at WUT. His research interest focus mainly on quality of service in both IPv4 and IPv6 infrastructures, Future Internet architectures, as well as applications for Future Internet (Internet of Things, Smart Cities, IPTV).

E-mail: jordim@tele.pw.edu.pl
National Institute of Telecommunications
Szachowa st 1
04-894 Warsaw, Poland

Piotr Krawiec received his M.Sc. and Ph.D. degrees in Telecommunications from Warsaw University of Technology, in 2005 and 2011, respectively. Since 2012 he is an Assistant Professor at the Department of Internet Architectures and Applications, National Institute of Telecommunications, and Institute of Telecommunications, Warsaw University of Technology. His research areas include IP networks (fixed and wireless), Future Internet architectures and applications, prototyping and testbeds.

E-mail: P.Krawiec@itl.waw.pl
National Institute of Telecommunications
Szachowa st 1
04-894 Warsaw, Poland

Daniel Negru received his Ph.D. from the University of Versailles Saint Quentin en Yvelines in 2006 in the field of Broadcast and Internet convergence solutions at the network and service levels. In 2007, he became Associate Professor at ENSEIRB School of Engineers/University of Bordeaux, specializing in multimedia and networking. From 2010 to 2014, he has been coordinating the ICT FP7 ALICANTE IP project that tackles networking and multimedia research fields. He has participated to more than 10 collaborative research projects at the national or European level, published more than 50 papers, such IEEE Communication Magazine, IEEE Multimedia, Globecom, ISCC, FIA. In 2013, he received his Habilitation à Diriger des Recherches (HDR) from the University of Bordeaux.

E-mail: daniel.negru@labri.fr
CNRS-LaBRI, University of Bordeaux
351 cours de la Libération
33 405 Talence CEDEX, France

Joachim Bruneau-Queyreix received his M.Sc. in Telecommunications at ENSEIRB-MATMECA graduate school of engineering, Bordeaux, in 2014. Since 2014, he is pursing his Ph.D. at LaBRI/Bordeaux Computer Science Laboratory in the field of multi-criteria optimization for content delivery within the Future Media Internet. His research areas include video codecs, streaming protocols, Future Internet streaming systems and architectures as well as multimedia streaming application prototyping.

E-mail: jbruneau@labri.fr
CNRS-LaBRI, University of Bordeaux
351 cours de la Libération
33 405 Talence CEDEX, France

Eugen Borcoci, Ph.D, is full professor at University "Politehnica" of Bucharest (UPB), Electronics, Telecommunications and Information Technology Faculty. His expertise has been oriented to specific domains of telecommunications and computer networks architectures, technologies and services. Recently, his research interest and activities are on new architectural approaches: Future Internet, SDN/NFV, Content Aware/Centric Networking. He has published 5 books, 4 textbooks and over 130 scientific or technical papers and scientific reports. He has been UPB team leader in several European research projects. He is member of several International Conferences Committees and member of the Technical Sciences Academy of Romania.

E-mail: eugen.borcoci@elcom.pub.ro
University Politehnica Bucharest
Splaiul Indeependenei
Bucharest 5, 060042 Romania

Andrzej Bęben received his M.Sc. and Ph.D. degrees in Telecommunications from Warsaw University of Technology (WUT), Poland, in 1998 and 2001, respectively. Since 2001 he has been Assistant Professor at WUT, where he is a member of the Internet Architectures and Applications research group. His research areas cover Future Internet, IP networks, information centric networks, network virtualisation, traffic engineering, multi-criteria decision theory, simulation techniques, measurement methods, and testbeds.

E-mail: abeben@tele.pw.edu.pl
Institute of Telecommunication
Warsaw University of Technology
Nowowiejska st 15/19
00-665 Warsaw, Poland

Piotr Wiśniewski is a Ph.D. student at the Institute of Telecommunications at the Warsaw University of Technology, where he received his M.Sc. (2010) and B.Sc. (2009) degrees in Telecommunications. He is a specialist at the National Institute of Telecommunications and a Research and Teaching Assistant at the Warsaw University of Technology. His research interests include quality of service, Information Centric Networks, media streaming solutions, Future Internet architectures and applications.

E-mail: pwisniewski@tele.pw.edu.pl
Institute of Telecommunication
Warsaw University of Technology
Nowowiejska 15/19
00-665 Warsaw, Poland