DASCo: dynamic adaptive streaming over CoAP

Piotr Krawiec¹ · Maciej Sosnowski^{1,2} ·

Jordi Mongay Batalla^{1,2} ·

Constandinos X. Mavromoustakis³ · George Mastorakis⁴

Received: 3 February 2017 / Revised: 17 April 2017 / Accepted: 18 May 2017 /

Published online: 3 June 2017

© The Author(s) 2017. This article is an open access publication

Abstract This paper presents the Dynamic Adaptive Streaming over CoAP (DASCo), a solution for adaptive media streaming in the Internet of Things (IoT) environment. DASCo combines DASH (Dynamic Adaptive Streaming over HTTP), the widespread open standard for HTTP-compliant streaming, with Constrained Application Protocol (CoAP), the vendor-independent web transfer protocol designed for resource-constrained devices. The proposed solution uses DASH formats and mechanisms to make media segments available for consumers, and exploits CoAP to deliver media segments to consumers' applications. Consequently, the DASCo player offers native interoperability with IoT devices that are accessed via CoAP, thus it allows easy access to data collected by different sensors in order to enrich the multimedia services. In the paper we present an overview of constraints of default CoAP implementation with respect to media streaming, and propose guidelines for development of adaptive streaming service over CoAP. Moreover, we discuss the features of CoAP that can be investigated when designing an efficient adaptive algorithm for DASCo. Presented experimental results show poor performance of DASCo when default values of CoAP transmission parameters have been used. However, adjusting the parameters according to the network conditions considerably improves DASCo efficiency. Furthermore, in bad network conditions the enhanced DASCo is characterized by a more stable download rate compared to DASH. This feature is important in the context of dynamic media adaptation, since it allows an adaptation algorithm to better fit media bit rate with download conditions.

Keywords Adaptive streaming · CoAP · Internet of things · Multimedia IoT · Dash

✉ Piotr Krawiec
p.krawiec@itl.waw.pl

¹ National Institute of Telecommunications, Szachowa Str. 1, 04-894 Warsaw, Poland

² Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland

³ University of Nicosia, 46 Makedonitissas Avenue, P.O.Box 24005, 1700 Nicosia, Cyprus

⁴ Technological Educational Institute of Crete, Estavromenos, Heraklion, 71500 Crete, Greece

1 Introduction

Streaming of multimedia content is one of the primary Internet services nowadays, since it accounts for majority of traffic transferred through the global network. What is more, an increasing number of Internet users, jointly with their augmented interest in video services, will cause video streaming contribution in global IP traffic to grow constantly until 2020 [11]. To handle demands related to such massive traffic volume, HTTP-based adaptive streaming solutions, such as Apple's HLS (*HTTP Live Streaming*), Adobe's HDS (*HTTP Dynamic Streaming*), Microsoft's ISS *Smooth Streaming*, and DASH (*Dynamic Adaptive Streaming over HTTP*) standardized by ISO/IEC [19], are commonly used. One of the factors that determine the leading position of HTTP-based adaptive streaming in multimedia services is that it can be deployed using ordinary web servers and existing, well-proven infrastructure of CDNs (*Content Delivery Networks*). Moreover, HTTP relies on TCP as a transport protocol, and congestion control provided by TCP helps to adjust the quality of ongoing media sessions to current network conditions.

Internet of Things (IoT) is considered as a further milestone of the ongoing digital revolution. According to the IoT idea, data from the physical world, gathered by a vast number of different sensors and devices connected, in some form, to the Internet, will be readily accessible for applications. In this way, the web services, which are essential building blocks of the current Web, can be elevated to a completely new level, by introducing the knowledge about the surrounding environment to web applications.

Miniaturized, resource-constrained devices that communicate through lossy, low bit rate wireless networks, form the foundation of IoT. Due to rigorous limitations in power, memory capacity and computational capabilities, new protocols had to be developed to allow interaction of IoT devices with the Internet. An example is 6LoWPAN [18, 31] that enables transferring IPv6 packets through IEEE 802.15.4 wireless networks. At the application level, as a substitute of heavy text-based HTTP, which is a backbone of the Web, the Constrained Application Protocol (CoAP) [38] has been proposed. It is expected that CoAP will play a similar role in IoT domain to that of HTTP in the traditional Internet, enabling realization of access to IoT objects and their services in the form of web services.

The increasing prevalence of low priced camera devices in various forms, from surveillance cameras, body worn cameras, in-vehicle video recorders, to cameras in smartphones, brings the expectations that in the near future the distribution of traffic load within IoT will follow a similar trend as the one observed on the traditional Internet, with predominant role of multimedia streaming. A small form factor stand-alone cameras that are available nowadays are equipped, besides visual sensor and optic, with modules responsible for hardware-based data encoding and wireless transmission [10]. The new generation of video codecs provides higher effectiveness of video compression, also in wireless environment [33], resulting in a satisfying quality of multimedia content jointly with low bit rates usually required in IoT scenarios. Moreover, the new codecs can be merged with different encryption schemes that offer a good protection of multimedia streaming while preserving the bit rate and minimizing the computational requirements [26].

The growing demand for multimedia services in IoT outlines two research directions. One of them is focused on adaptation of IoT technologies to the requirements imposed by multimedia streaming, for example by modifying MAC layer ([35, 44]) or transport layer ([16]) of IEEE 802.15.4 networks for proper handling of the traffic generated by delay-sensitive streaming applications. On the other hand, research community investigates

approaches that consider the utilizing existing solutions for multimedia services within IoT domain. Examples are 802.11n and 802.11ah specifications that adjust popular WiFi technology to low power IoT devices by introducing a set of energy saving mechanisms. In [4] the authors propose a framework that is based on 802.11n standard and enables energy efficient multimedia communication over IoT.

In this paper we present DASHCo: *Dynamic Adaptive Streaming over CoAP*, an adaptive streaming platform for IoT. Our solution is compliant with DASH specification from the point of view of metadata formats and basic principles of action, but exploits CoAP instead of HTTP as a delivery protocol. In this way, DASHCo relies on application layer mechanisms for adaptive streaming that have been successfully deployed on the Internet and, at the same time, it provides native interaction between streaming applications and CoAP-enabled objects from IoT domain.

The rest of the paper is organized as follows. In the next section we will briefly discuss the suitability of adaptive streaming in IoT. Then, our proposition of CoAP-based adaptive streaming framework will be presented. In Section 4 we will show experimental results for performance evaluation of proposed framework, comparing them with traditional DASH streaming. Finally, the paper will be summarized with our major conclusions and the outline of further works.

2 Adaptive streaming in IoT scenario

IoT deployment is based mainly on wireless technologies as primary connectivity platforms. A characteristic feature of a wireless environment is large variability of network conditions even in neighboring locations. Interferences, fading and other factors affect the wireless channel, causing the quality of transmission links to vary significantly in space and time [34]. Consequently, two adjacent terminals can experience wholly different packet losses and delays when they communicate with the same remote server. On the other hand, users may use different devices to play multimedia content: from high-end tablets to low-end smartphones and wearables such as smart glasses or smart watches. This is an area, where adaptive streaming is of particular importance due to its ability to deliver media content with the quality adequate to current network conditions and terminal capabilities.

Among the adaptive mechanisms existing on the market, DASH attracts a particular interest since its specification has been released as an open standard. Vast majority of multimedia streaming companies (Netflix, Comcast, Akamai, just to mention a few) are engaged in the DASH Industry Forum (DASH-IF) [13], an association that focuses on supporting the implementation of DASH specification into real business activities. Same as its corporation-controlled counterparts (HLS, HDS, Smooth Streaming), DASH is based on splitting multimedia content into many pieces, called segments, that contain a small portion of a video (usually a few seconds). Each segment is encoded with different media bit rates, called representations, which are characterized by different media quality and bandwidth requirements. Information about the location of the segments, their available representations and bandwidth requirements, is provided through a manifest file called Media Presentation Description (MPD). On the basis of the data derived from an MPD file, user's application continuously downloads successive media segments, selecting a representation with the highest bit rate for each segment which is, however, not higher than the available bandwidth on the downloading path. The download of both, the MPD file and the video segments, is realized in the form of typical HTTP GET requests.

Adaptive streaming applies when multiple users want to stream the same multimedia content but they experience different network conditions or they use devices that have distinct technical parameters such as display resolution or decoding capabilities. Good representatives of such scenarios in IoT domain are various monitoring web applications (security/surveillance monitoring of a building or a factory, elderly monitoring at a nursing home etc.), since due to an occurrence of a specific event many persons may need to keep their eyes on the same object, i.e. they want to stream visual documentation of the given object. A related use case is depicted in Fig. 1, where a set of video acquisition devices, such as IP cameras, monitor a building area. Captured video signal is encoded with default bit rate for a given device and uploaded on a streaming server, which is responsible for storing the video and sharing it with interested persons. In order to make media content available in adaptive streaming service, the streaming server divides received video content into segments, transcodes them to obtain a set of representations and creates an associated MPD file. When insecure event occurs, a control center may trigger the video captured by a selected camera to be displayed on the handheld devices carried by security officers. Then, client applications at the handheld devices send a request to the streaming server to obtain the MPD file for indicated media content, and start streaming requested content, adjusting quality of the streamed media to the available download rate. Moreover, during a streaming session, media representation can be switched when network conditions change, for example due to a movement of the officer.

Another example is remote healthcare system deployed in a hospital that allows a nurse to display patient's diagnostic information, such as electrocardiograph's screen, on her tablet. When the nurse notices something disturbing, she contacts a doctor who visits the other room of the hospital (at the edge of wireless network). The doctor also streams diagnostic information on his tablet, but with the quality adequate to his network conditions. We can also mention a Smart Museum use case presented by MPEG 3DG subgroup [27], which concentrates on activities related to Media Internet of Things and wearable devices. In this scenario, a museum visit is enriched with augmented information such as a video clip explaining a history of the object being viewed. Multimedia content associated with an exhibit is streamed from a server in an adaptive way, according to capabilities of the device used by a visitor: a smartphone, a tablet or a smart glasses.

The smartness of multimedia IoT applications results from their interaction with the surrounding environment. To enrich security officers' knowledge about investigated object

Fig. 1 Use case: surveillance monitoring

and its surroundings, their terminals can interact with different IoT devices to gather additional information, for example whether a door is locked/open or about the last entry to a room recorded by a motion detector, according to multisensory smart surveillance concept [21]. The doctor examining patient's electrocardiogram can obtain other medical parameters of the patient by connecting with patient's temperature, pressure and pulse sensors. A hand gesture recognition system deployed in Smart Museum can send to visitor's terminal recognized commands related to play control (start, pause, stop, next clip), while the visitor can use her/his multimedia application to interact with observed object (for example, visitors can control miniaturized models exhibited in the museum).

We assume that the communication between the multimedia application and IoT objects is provided through CoAP, an application layer protocol intentionally designed to operate in a resource-constrained IoT environment. CoAP is characterized by a small overhead, since it uses binary header and stateless UDP protocol in the transport layer. At the same time, it complies with HTTP syntax by implementing widely used GET, PUT, POST and DELETE methods, as well as HTTP's response codes. In this way, the development of CoAP-based applications can benefit of existing knowledge and well-known, commonly used web technologies, because IoT objects and services offered by them can be exposed in the form of web services, fully compatible with REST principles [37]. Consequently, CoAP is anticipated as one of the major web transfer protocols in IoT in the near future [41].

3 CoAP-based adaptive streaming

DASH has aroused a lot of interest on the traditional Web-based video market. Netflix and Youtube chose DASH as their main streaming technology [3, 24]. 3GPP, in the Release 10 specification [1], adopted DASH as the solution for streaming media over LTE Advanced mobile networks. It is therefore no surprise that researchers have started to think about incorporating DASH approach also in the IoT domain.

In [25] the authors propose video streaming framework dedicated for IoT that is built on the top of Content Centric Networking (CCN) [20] architecture. They use DASH mechanisms to create a set of video segments with different representations. Then, segment requests generated by DASH client are translated to CCN interests, whereas responses from HTTP server are translated to CCN content objects. Although it is considered that CCN architecture, as well as other name-based approaches such as the ones presented in [29], can provide many benefits to IoT [25], the main drawback of the proposed solution is that CCN follows a clean slate design concept and operates on the new network stack, thus it is incompatible with the existing network infrastructure and its deployment requires building the network from scratch [5]. In turn, [36] presents DASH-based video platform for IoT, called WVSNP-DASH, that provides a specific naming syntax for video segments. In this way, essential metadata required for video playout is embedded in the name of every segment and a client can play the segment without the need for downloading a manifest file. However, the authors of [36] do not investigate a communication overhead introduced by WVSNP-DASH, which can be significant, especially when video content is divided into a set of small segments and URL of each segment is enlarged by video metadata. In addition, WVSNP-DASH proposes inserting only a slight set of parameters into a segment's name: information about segment's index and representation, video container format, the highest available representation and the total number of segments for given media content. In turn, MPD file defined by DASH specification contains much

more information that are crucial for video playout, such as data related to video resolution, DRM (Digital Right Management) or timeline. Consequently, presented WVSNP-DASH, in fact, is not fully compliant with DASH data model.

In this paper, we propose Dynamic Adaptive Streaming over CoAP – DASCo. To the best of our knowledge, DASCo is the first solution that integrates DASH with the CoAP protocol designed for IoT. DASCo is compatible with the DASH semantic but exploits CoAP as a platform for accessing video segments. It works on a usual CoAP server, which means that its implementation can be based on existing CoAP infrastructure. On the other hand, compatibility with DASH specification means that DASCo inherits main DASH's advantages, such as: support for different streaming services (On-demand, Live and Time-shift), support for various DRM solutions, capability to work with a number of different codecs and video container formats.

CoAP has been designed to run on resource-constrained devices. Therefore, unlike HTTP, it operates on UDP protocol, which provides lower communication overhead (the header length is 8 bytes only, no three-way handshake procedure since it is a connectionless protocol). In general, we assume that streaming services will be requested by handheld devices as smartphones or tablets which in fact have enough resources to deal with HTTP/TCP stack implementation. However, these devices can also take advantage of using the lightweight protocol stack defined for constrained IoT objects, because data transfer can be performed with lower costs in terms of energy and bandwidth consumption.

Another benefit of using CoAP in streaming application is a native interoperability between the application and CoAP-compliant IoT domain. The application can directly communicate with IoT objects to get universal and fast access to observations performed by sensors or actuators inputs. CoAP offers an *observe* option [17] that enables implementation of publish/subscribe interaction model. Using it, sensors can push information to the streaming application whenever the current state of the monitored quantity changes. As a result, video streaming service can be easily enhanced with the knowledge of physical environment and interaction with it. The all-CoAP model simplifies a structure of such enhanced streaming application and reduces its development costs, since all communication is performed using the same platform (at least at the application level). In addition, the speed and reliability of the application may increase, because there is no need to use additional devices responsible for protocol translations at the application layer (e.g. HTTP-CoAP proxies). Considering oncoming IEEE 802.11ah (a.k.a. Wi-Fi HaLow) standard that enables low power connectivity [2], the protocol stack matching between user handheld device with DASCo player and IoT objects spans from the application layer to the data link layer (in case of 802.15.4 sensors, their interoperability with user device is usually provided through layer 2 gateways located at the edge of the wireless sensor network).

3.1 CoAP constraints for media streaming

The central premise of CoAP was to provide RESTful access, but at the same time with a low communication overhead, to scalar data acquired by resource-constrained sensors or to send short commands to actuators. The CoAP header is fixed with a length of 4 bytes only, and it can be extended with additional Options headers. As a result, CoAP's average transaction size in bytes, in typical IoT scenario, is almost 10 times smaller compared to HTTP [12]. Thus, several approaches have been published in the recent literature that consider CoAP as a lightweight transfer protocol for scenarios other than interaction with IoT devices. For example, A. Eriksson et al. in [14] propose a solution that uses CoAP to implement point-

to-multipoint communication for Information-Centric Networking (ICN) services such as content distribution or live video streaming. The authors of [39] exploited CoAP to handle large amounts of data during the deterministic data transfer in Industrial Ethernet environment. Nevertheless, streaming of multimedia content shows distinct characteristics compared to the transfer of scalar data generated by typical sensors, since it involves the transmission of much larger data sets and, particularly in the case of live streaming service, requires delay-sensitive communication.

Standard CoAP model includes two abstract layers. The upper layer implements REST semantic with Request/Response communication scheme realized by means of GET, PUT, POST and DELETE methods. The lower layer controls the transfer of CoAP messages through the underlying UDP protocol. This messaging layer supports two modes: reliable and unreliable. Reliable data transfer is based on retransmissions and timeout mechanisms and is carried out by confirmable (CON) messages. Each CON message must be acknowledged with an ACK message before the timeout expires. Otherwise, the message is retransmitted. On the other hand, non-confirmable (NON) messages are used for unreliable communication as their sender does not expect ACK as confirmation. All CoAP messages are identified by Message ID (MID), which is used to detect duplicates, and also to bind ACK with its CON message during reliable communication.

To limit the communication overhead resulting from packet fragmentation at the IP layer, the CoAP specification provides default upper bound for a message payload size that equals 1024 bytes. Taking into account that size of video segment usually exceeds this value, media streaming through CoAP can be performed in a block-wise fashion specified in [7]. The block-wise option introduces an intermediate abstract layer, as it is depicted in Fig. 2, which is responsible for segmenting and resequencing operations.

The block-wise layer divides a payload into a set of blocks. The client sends a block request as a CON message and the server replies with ACK that carries a piggybacked response with the requested block. Each single block transfer is handled separately by the messaging layer and CoAP specification [38], by default, allows only one pending interaction between the client and the given server at a certain time. Consequently, there is a set of individual operations of block downloading that are performed in a Stop-and-Wait manner at the message layer (see Fig. 3) between the two events at the request/response layer that are related to the generation of a request and the receipt of a complete response. This Stop-and-Wait data transfer is very ineffective for a streaming service due to the following reasons:

- A single block request results in a transmission of precisely one block. In networks that are characterized by a long Round Trip Time (RTT), the client wastes most of the time on waiting until the request arrives to the server and the server's response goes back to the client. In [9] an improvement has been proposed that allows the server to send several blocks as a response to one block request. However, this solution is not compatible with the current CoAP specification because it requires modifications in the messaging layer.
- One block carries a limited amount of data, since a block-wise option header defines the maximum block size as 1024 bytes only [7].
- In the reliable transfer mode CoAP implements a simple congestion control mechanism with retransmission timeout and exponential back-off between retransmissions. Default values of retransmission parameters defined by CoAP specification are very conservative. The value for an initial timeout is randomly chosen between 2 and 3 s and it is doubled on every successive timeout expiration. These values result from the characteristics of

Fig. 2 CoAP abstract layering for block-wise transmission

wireless sensor networks, where a response can be delayed due to entering an endpoint into a sleep mode for sake of energy saving. Nevertheless, the default CoAP congestion control may lead to significant delays in media streaming process when packet losses occur. Consequently, delivery of a whole media segment, which consists of many blocks, may consume too much time resulting in undesired freezing events during video payout.

The continuity of multimedia transmission can be provided by using NON messages in a block-wise transfer. In this case, the client can send a number of block requests without waiting for receiving a response for the previous request. However, the non-confirmable transfer mode may result in too high and uncontrolled packet losses, particularly in a congested wireless environment. Furthermore, the advantage of using TCP in DASH streaming is not only a strict lossless transmission, since multimedia application can tolerate the loss of single packets [23]. Another benefit of the congestion control mechanism provided by TCP is its ability to prevent congestion in multimedia transmission while simultaneously keeping the ongoing multimedia sessions. With a non-confirmable transmission, the CoAP client may generate a number of

Fig. 3 Block-wise transfer sequence chart

requests that exceeds network and/or server capabilities, thereby introducing a distortion in multimedia streaming service.

All the above mentioned obstacles should be carefully considered during the design of adaptive streaming in a CoAP-compliant environment.

3.2 Dynamic adaptive streaming over CoAP – DASCo

The proposed DASCo streaming framework provides communication principles between DASCo client application (i.e. media player) and a standard CoAP server. DASCo maintains compliance with DASH specification, thus it uses similar metadata formats and methods, but relies on a distinct delivery protocol. Information about the delivery protocol is included in the MPD file, which provides instructions to the client application on how to construct segment's URL (e.g., by concatenation of *<BaseURL>* and *<SegmentURL>* elements from MPD).

CoAP follows the REST architecture principles and uses URI (Uniform Resource Identifier) to name resources hosted by a server. For DASCo purposes two kinds of resources can be registered on the CoAP server that are bound to a streaming service. The first one is *mpd* and the associated URI: *coap://streaming_server_host/mpd/* is a list of all MPD files hosted by streaming server. The second resource is *media_content*. When service operator wants to provide new content for users, for example a video recorded by a camera marked as *camera1*, he/she registers new resource *camera1video* as a child resource of *media_content*, using POST method. Next, streaming server performs segmentation of the uploaded media content and transcodes segments to different representations. Afterwards, the server creates an MPD file for given content and registers the file as a child resource of *mpd*.

The specification [37] provides resource discovery mechanism with resource description for CoAP environment, which can be exploited by a client application to query a server for its list of hosted MPD files. A user who wants to stream the new media content sends a request to the server to obtain the related MPD file, for example:

```
GET coap://streaming_server_host/mpd/cameralvideo
```

Next, based on metadata available in the obtained MPD, user's application consecutively requests media segments:

```
GET coap://streaming_server_host/media_content/cameralvideo?repID=1&no=1
```

where query parameter *repID* indicates requested segment's representation, and *no* refers to the sequence number of the segment. If segmentation process is implemented by means of a byte range option, the range attribute should be sent as a query parameter instead of segment's sequence number:

```
GET .../cameralvideo?repID=1&mediaRange=863-13826
```

DASCo streaming server can easily make the same content available outside the CoAP domain, via HTTP service for standard DASH clients. This can be done through the deployment of two manifest files: DASCo MPD and DASH MPD, in parallel, both having almost the same structure. The only difference is in values of elements that identify delivery protocol in

segment's URL, such as $\langle BaseURL \rangle$. On-the-fly conversion between DASCo and DASH MPDs may also be applied when a request for DASH MPD arrives, since it requires nothing but a simple operation of swapping URL prefix from “coap” to “http” (alternatively, values of $\langle SegmentURL \rangle$ and $\langle SegmentTemplate \rangle$ elements may need to be changed if we want to specify segment's location in DASH as a plain path to the segment file rather than URI to REST resource, as it is in CoAP). Deployment of DASH streaming together with DASCo is even easier when only relative URLs are used. In this case the MPD file does not contain any information about host's URL where segments are stored, thus the MPD does not specify a delivery protocol. The absolute segment's URL, jointly with a protocol part, is then constructed taking into account the service context, i.e. location of video segments is solved based on the location of the associated MPD file. As a result, the same MPD can be used by the streaming server as a response to the request received via CoAP (DASCo) or HTTP (DASH).

It is worth noting that placing a streaming server in the same network domain with video sources and DASCo clients, as it is presented in Fig. 1, results in low response times what is, in particular, crucial for live streaming service (or rather the “quasi-live”, since we allow a small delay related to transcoding and segmentation processes). Moreover, the server can gather statistics about the network and then use them during segmentation and transcoding processes in order to better adapt the created segments to network conditions. Such cooperation of codec and streaming protocol allows increasing the overall efficiency of adaptive streaming system [28].

As it was discussed in subsection 3.1, default CoAP retransmission timeout of 2 s is relatively conservative and may lead to inefficiency in streaming service. To cope with this issue, monitoring and self-tuning mechanisms should be incorporated into DASCo client application to estimate the timeout value based on RTT measurements. Such approach is in line with CoAP specification [38], which allows application to dynamically adjust values of different transmission parameters. An example of the timeout tuning mechanism is presented in [8].

Moreover, higher performance of streaming service can be achieved by increasing the maximum message payload size over the default 1024 bytes. For example, WiFi devices usually use MTU of 1500 bytes, which allows allocating, without unnecessary packet fragmentation, 1492 bytes for the CoAP message (8 bytes is consumed by UDP header) and 1364 bytes for the message payload (we assume that 128 bytes is reserved for the CoAP header and potential CoAP options). Although the specification permits the increase of the message payload size, unfortunately it does not apply to block-wise transmission. The reason lies in Block Option structure that is used during block-wise transfer. The *SZX* (“size exponential”) field of this structure, which determines the size of the transferred block, has a constant length and 1024 is the maximum possible value that can be assigned to it. Therefore, an increase of bandwidth utilization through increasing message payload can be achieved by using standard CoAP messages only, without block-wise transfer mode. Such solution involves a significant growth of DASCo applications' complexity, because they need implementation of additional mechanisms responsible for fragmentation of the video segment into smaller pieces that match the message payload size (at the server side), their assembly by the client, detection of duplicate messages etc.

Designing an efficient adaptation algorithm for DASCo is out of scope of this paper. Many propositions that try to solve the problem: how to select the highest bitrate representation of a segment, whose bitrate is not higher than the download rate, have been published in the last few years. In general, the proposed adaptation algorithms for DASH can be classified into

three basic classes: rate-based, buffer-based and time-based (a comprehensive overview of the main features of each class is presented in [6, 30]). Choosing the most appropriate algorithm for CoAP environment is an interesting research challenge. Time-based adaptation algorithms, such as presented in [6, 43], which operate on download times of media segments are worth considering. Their important feature is awareness of varying segments sizes, which is relevant in congested wireless environment with limited download rates, specific to IoT.

Taking into account Stop-and-Wait data transfer when block-wise option is used, an adaptation algorithm designed for DASCo may include a feature that allows downloading several segments and/or blocks in parallel. It can be reached by increasing the value of NSTART parameter. According to the CoAP standard [38], the default value for NSTART is 1, thus only a single ongoing transaction between the client and the given server is allowed. Enhanced DASCo adaptation algorithm may decide not only about the representation of the downloaded segment, but also control the number of simultaneously downloaded segments by changing NSTART value, depending on the monitored network conditions.

Furthermore, block-wise transmission gives a DASCo client the possibility to precisely control the segment download process, since the application can be aware of the download time for each block of a given segment. The client can know the total size of the requested segment (for example by using option Size2 defined in block-wise specification [7]) and how many blocks of the segment have been already downloaded. Based on this information, DASCo adaptation algorithm can evaluate the progress of the download process and eventually decides to stop downloading the current segment, and then requests a new segment with lower representation (i.e. smaller size) to avoid video playout stall. Such functionality is hard to implement in TCP-based DASH, since the entity that controls TCP connection is the operating system. On the other hand, tearing down a TCP connection at the application level is inefficient, because the application is not aware if request for the subsequent portion of data has already been sent by the TCP client or not (if yes, then the server response with requested data is useless) and next, a costly establishment of new connection (TCP three-way handshake, slow start phase) needs to be performed.

The CoAP specification also allows an application to change the value of MAX_RETRANSMIT parameter, which defines the maximum number of retransmissions. With this feature, DASCo adaptation algorithm can control number of retransmission attempts, since multimedia content can tolerate some losses, but often requires delay-sensitive communication, particularly in case of live streaming.

4 Experimental results

We have performed experimental studies to prove the ability of the proposed DASCo to efficiently stream video segments. Tests were performed within the PL-LAB2020 experimental infrastructure [32]. In order to run tests in parallel, a few physical servers were used, and on each server three virtual machines (VMs) were established with bridged network interfaces to create topology as presented in Fig. 4.

Edge nodes run client and server applications, while the intermediating one emulates wireless network environment by the usage of the Traffic Control (*tc* command) tool available on Linux. Particularly, bandwidth limitation is emulated by the Token Bucket Filter (*tbf* option in *tc*), when latencies and losses are emulated by the NetEM (*netem* option in *tc*). FIFO queueing discipline prevents packet reordering. Parameters of the emulated network are the

Fig. 4 Testing topology

same in both directions. Client VM operates on Ubuntu Desktop 16.04, while two other VMs run on Ubuntu Server 14.04.

Applications that emulate the client and the server for two versions of Dynamic Adaptive Streaming (over HTTP and over CoAP – DASH/DASCo) were implemented in Python. DASH implementation exploited standard Python libraries, while DASCo was developed on the basis of the CoAPthon [40]. The client collected statistics related to downloaded segments and Wireshark [42] was used to get detailed information about individual packets.

Client requested consecutive “Big Buck Bunny” [22] media segments with the 2-s segment length. Two representations of the movie, in resolution 240p (bit rate 150 kbps) and 360p (bit rate 250 kbps) were available on the server, corresponding to typical handheld device streaming scenario. Nevertheless, within each scenario the client downloaded segments of one representation, since no adaptation was performed. Each test lasted one hour, during which the client downloaded the movie in the loop. The tests were repeated from 12 to 24 times.

Two scenarios were considered. The first one was adequate to the situation when WiFi network provided relatively good network conditions, with network delay equal to 2 ms with 2 ms of delay variation, packet loss probability equal to 0.1% and available bandwidth equal to 5 Mbps. In this scenario the client requested for the higher representation (250 kbps). When DASCo was considered, two values of retransmission timeout, defined in CoAP specification as ACK_TIMEOUT parameter, were used: the default one equal to 2 s, and enhanced one that matched measured RTT, i.e. 0.05 s. Results for this scenario are depicted in Fig. 5, which presents histograms of segment download times (SDT) distribution (with 95% confidence intervals). The second scenario was performed assuming network conditions corresponding to congested WiFi environment, with network delay equal to 2 ms with 20 ms of delay variation, packet loss probability equal to 2% and network bandwidth equal to 512 Kbps. The client downloaded media content with lower representation (150 Kbps). In DASCo case, two values of ACK_TIMEOUT, the default 2 s and the enhanced 0.1 s, were investigated. Results for this scenario are depicted in Fig. 6.

As expected, in good network conditions DASH provided better performance (lower SDT) compared to DASCo, thanks to its advanced congestion and flow control algorithms (Fig. 5a). Since the bandwidth was relatively high, TCP window at DASH server reached large sizes, so the server could transmit many packets simultaneously. On the other hand, CoAP, because of its Stop-and-Wait data transfer manner, could not exploit available bandwidth as well as TCP did. When a packet loss occurred, the download time of the whole media segment was enlarged by at least the ACK_TIMEOUT value. In case of the default value of retransmission timeout (Fig. 5b), SDT for about 10% of segments exceeded 2 s, thus it had a greater value than the segment playout duration. Adjusting the value of

Fig. 5 Histograms of the segment download time in good network conditions (media representation: 250 kbps)

retransmission timeout to the network conditions significantly improved DASHCo performance (Fig. 5c), since 99% of segments were downloaded by the client with the delay not greater than 1.2 s. This result is comparable to DASH (download time for 99% segments was below 1.4 s), however mean value of SDT for DASH streaming is 0.4 s, while for DASHCo it equals 0.8 s.

In the bad network conditions scenario, DASHCo system with default CoAP transmission parameters ($\text{ACK_TIMEOUT} = 2$ s), as in the previous tests, indicated a very poor performance. Taking into account high packet loss ratio and the fact that in order to deliver a media segment in block-wise mode 60–70 packets must be transferred (blocks and corresponding GET requests), it is very likely that more than one loss may occur during the segment download. Therefore, the histogram of SDT distribution is multimodal (Fig. 6b) with modes correlated to the number of retransmissions performed per segment, and 73% of video segments were downloaded within time longer than segment playout duration, which can be unacceptable for adaptive streaming service.

The situation changes when we consider tuning the CoAP retransmission timeout according to network conditions. When the value of ACK_TIMEOUT parameter decreased to 0.1 s, the histogram became unimodal and efficiency of DASHCo (Fig. 6c) was even better than DASH (Fig. 6a). The reason lies in poor performance of TCP congestion control mechanism in lossy wireless environment, since TCP treats channel losses as congestions, which considerably decreases DASH throughput. Moreover, the measured maximum size of a CoAP block was equal to 1076 bytes (including headers) while TCP segments can be 140% of that size (1514

Fig. 6 Histograms of the segment download time in bad network conditions (media representation: 150 kbps)

bytes), hence during DASH retransmissions more data need to be re-sent compared to DASHCo. As a result, in DASH streaming 90% of segments were downloaded in time no longer than 2 s, whereas in DASHCo this ratio reached 99.9%. In addition, the shape of SDT distribution for DASH is wider compared to the DASHCo histogram, which means that DASH streaming is less stable. This result is also illustrated in Fig. 7, which presents the time plot of a download rate for media segments. One can observe that download rates in case of DASHCo are characterized by lower fluctuations than rates achieved within DASH. From the perspective of dynamic media adaptation, it is a promising result in favour of CoAP.

We also investigated the total number of exchanged packets and total amount of exchanged bytes during a single streaming of a 10-min long “Big Buck Bunny” movie. The measurements were performed on the server side, therefore the results include all packets sent by server and received by server (without lost packets that had been sent by a client).

It has already been shown in many papers (e.g. [15]) that in the standard IoT scenario, when CoAP is used to communicate with sensors and carries scalar sensors data in the order of few tens of octets, it outperforms HTTP in terms of throughput efficiency. The reason stems from the larger header overhead of HTTP and TCP than CoAP and UDP, and also the additional packets required to establish the TCP connection. In contrast, the results presented in Table 1 show that in streaming scenario amount of bytes required to transfer the movie from the server to the client is almost the same for both solutions. This is due to the fact that the advantage of CoAP, related to smaller header size and lack of packets needed for connection establishment/

Fig. 7 Download rate in bad network conditions (media representation: 150 kbps)

termination, is eliminated by lower payload size in the CoAP block messages, therefore more packets must be sent to transfer a media segment via DASHCo than via DASH.

5 Conclusions and future work

In order to provide the best possible quality of a streaming service, the media stream must adapt to the current network conditions. One of the most popular standard for this purpose is DASH, which exploits HTTP protocol to download media segments from a server.

In this work we propose a framework that integrates the idea of DASH with Constrained Application Protocol (CoAP), which is considered as a common, vendor-independent protocol for interaction between IoT devices. We call it DASHCo – Dynamic Adaptive Streaming over CoAP. Our solution is based on DASH metadata formats, but uses CoAP as a delivery platform. In this way, the DASHCo player can natively interact with different IoT devices and collect additional, contextual data that enrich the multimedia service. The usage of a common protocol within one application makes the application development process easier.

As a proof of concept, experimental results are provided. Client exploits DASH/DASHCo to stream a video content from a server, while different network conditions are emulated. Results of the experiments indicate that DASHCo, which operates on the top of default CoAP implementation, has poor efficiency in media streaming, due to its conservative congestion control

Table 1 Throughput measured in bad network conditions (DASHCo with ACK_TIMEOUT = 0.1 s)

	Number of packets	Transferred bytes [KB]
DASH	16,800 (± 70)	11,160 (± 25)
DASHCo	19,935 (± 15)	11,055 (± 25)

mechanism. However, by tuning the CoAP retransmission timeout according to the current network conditions, the performance of DASCo significantly improves and measured results are close to the reference DASH system.

Furthermore, in bad network conditions the use of DASCo with enhanced CoAP transmission parameters is even more efficient, in terms of segment download time and download rate fluctuations, than standard DASH. This feature is relevant in IoT scenarios, where large number of devices communicate through a wireless network, which influences network characteristics.

Our further work will focus on designing an efficient adaptation algorithm dedicated for DASCo, which will benefit from the features of UDP-based streaming through CoAP. Contrary to HTTP, the CoAP specification allows an application to tune some transmission parameters, such as retransmission timeout and maximum number of retransmissions, and this capability can be utilized by the DASCo adaptation algorithm to improve media streaming efficiency.

Acknowledgements This work was partially founded by the Narodowe Centrum Badań i Rozwoju in Poland, under grant Celtic-Plus Monalis.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. 3GPP Technical Specification, “Transparent end-to-end packet-switched streaming service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH)”, 3GPP TS 26.247 V10.9.0 (2014–12), Release 10, http://www.3gpp.org/ftp/Specs/2016-12/Rel-10/26_series/. Accessed 19 December 2016
2. Adame T, Bel A, Bellalta B, Barcelo J, Oliver M (2014) IEEE 802.11AH: the WiFi approach for M2M communications. IEEE Wireless Communications Magazine
3. Adhikari V, Guo Y, Hao F, Varvello M, Hilt V, Steiner M and Zhang Z (2012) Unreeling Netflix: Understanding and improving multi CDN movie delivery. Proc. IEEE INFOCOM, pp. 1620–1628.
4. Afzal B, Alvi SA and Shah GA (2016) Adaptive duty cycling based multi-hop PSMP for internet of multimedia things. 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, pp. 895–900
5. Beben A et al (2012) Content Aware Network Based on Virtual Infrastructure, in: Proceedings of 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing / Hochin Teruhisa, Lee Roger (eds.), IEEE Computer Society
6. Beben A, Wisniewski P, Mongay Batalla J, Krawiec P (2016) ABMA+ : lightweight and efficient algorithm for HTTP adaptive streaming. In the Proc. of the 7th international Conference on Multimedia Systems ACM MMSys2016, Klagenfurt, Austria
7. Bormann C, Shelby Z (2016) Block-Wise Transfers in the Constrained Application Protocol (CoAP), Internet Engineering Task Force (IETF), RFC 7959
8. Bormann C, Betzler A, Gomez C, Demirkol I (2016) CoAP Simple Congestion Control/Advanced. Internet Engineering Task Force (IETF), Internet-Draft, October 20
9. Choi G, Kim D, Yeom I (2016) Efficient streaming over CoAP. 2016 International Conference on Information Networking (ICOIN), Kota Kinabalu, pp. 476–478
10. Chotard L, Lagorce X, Posch C (2016) Ultra-low bandwidth video streaming using a neuromorphic, scene-driven image sensor. 2016 I.E. international symposium on circuits and Systems (ISCAS), Montreal, QC, pp. 455–455
11. Cisco White Paper. “Cisco Visual Networking Index: Forecast and Methodology, 2015–2020”, June 2016. <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni-hyperconnectivity-wp.pdf>. Accessed 19 Dec 2016
12. Colitti W, Steenhaut K, De Caro N (2011) Integrating Wireless Sensor Networks with the Web. Extending the Internet to Low power and Lossy Networks Workshop (IP+SN 2011), Chicago, IL
13. DASH Industry Forum. Webpage: <http://dashif.org> Accessed 19 Dec 2016

14. Eriksson E A, Ohlman B, Persson KÅ, Malik AM, Ihlar M, Sunde L (2015) Scalable point-to-multipoint communication for cloud networking using information-centric networking. 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, pp. 654–662
15. Gao W et al. (2016) Assessing Performance of Constrained Application Protocol (CoAP) in MANET Using Emulation. Proceedings of the International Conference on Research in Adaptive and Convergent Systems RACS '16, Odense, Denmark, pp. 103–108
16. Gowri S R et al. (2015) Dawn: Dependable Networking Framework for Multimedia-enabled Internet-of-Things. Proceedings of the 13th International Conference on Advances in Mobile Computing and Multimedia MoMM 2015, Brussels, Belgium, pp 211–215
17. Hartke K (2015) Observing Resources in the Constrained Application Protocol (CoAP), Internet Engineering Task Force (IETF), RFC 7641
18. Hui J, (ed.), Thubert P (2011) Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. Internet Engineering Task Force (IETF), RFC 6282
19. ISO/IEC 23009-1 (2014) Information technology - Dynamic adaptive streaming over HTTP (DASH) - Media presentation description and segment formats
20. Jacobson V et al (2009) Networking named content. In Proc. of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '09). New York: ACM, pp. 1–12
21. Kokkonis G, Psannis KE, Roumeliotis M, Schonfeld D (2017) Real-time wireless multisensory smart surveillance with 3D-HEVC streams for internet-of-things (IoT). J Supercomput 73(3): 1044–1062
22. Lederer S, Müller C, Timmerer C (2012) Dynamic Adaptive Streaming over HTTP Dataset. In Proceedings of the ACM Multimedia Systems Conference 2012, Chapel Hill, North Carolina, February 22–24
23. Lin C-H, Ke C-H, Shieh C-K and Chilamkurti NK (2006) The Packet Loss Effect on MPEG Video Transmission in Wireless Networks. 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06), Vienna, pp. 565–572
24. Live Streaming API. YouTube on-line guides and tutorials for developers. <https://developers.google.com/youtube/v3/live/guides/encoding-with-dash>. Accessed 19 December 2016
25. Martinez-Julia P, García ET, Murillo JO, Skarmeta AF (2013) Evaluating Video Streaming in Network Architectures for the Internet of Things. 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Taichung, pp. 411–415
26. Memos VA, Psannis KE (2016) Encryption algorithm for efficient transmission of HEVC media. J Real-Time Image Proc 12(2):473–482
27. Mitrea M, Kim S-K, Chun S, Use cases for Internet of Media Things and Wearables, MPEG 3DG Subgroup, ISO/IEC JTC, October 2016, <http://mpeg.chiariglione.org/standards/exploration/internet-media-things-and-wearables>. Accessed 10 April 2017
28. Mongay Batalla J (2015) Advanced Multimedia Service Provisioning based on efficient interoperability of adaptive streaming protocol and High Efficient Video Coding. Springer Journal of Real-Time Image Processing
29. Mongay Batalla J, Krawiec P (2014) Conception of ID layer performance at the network level for internet of things. Springer J Pers Ubiquit Comput 18(2):465–480
30. Mongay Batalla J, Krawiec P, Beben A, Wisniewski P, Chydzinski A (2016) Adaptive video streaming: rate and buffer on the track of minimum Rebuffering. IEEE J Sel Areas Commun 34(8):2154–2167
31. Montenegro G, Kushalnagar N, Hui J, Culler D (2007) Transmission of IPv6 Packets over IEEE 802.15.4 Networks. Internet Engineering Task Force (IETF), RFC 4944
32. PL-LAB2020 experimental infrastructure. Website: <http://www.pllab.pl/>. Accessed 19 Dec 2016
33. Psannis KE (2016) HEVC in wireless environments. J Real-Time Image Proc 12(2):509–516
34. Psannis KE, Ishibashi Y (2009) Efficient error resilient algorithm for H.264/AVC: mobility Management in Wireless Video Streaming. Springer Telecommun Syst J 41(2):65–76
35. Rikli NE, Alabdulkarim M (2014) Cross-layer-based adaptive video transport over low bit-rate Multihop WSNs. Can J Electr Comput Eng 37(4):182–191
36. Seema A, Schwoebel L, Shah T, Morgan J, Reisslein M (2015) WWSNP-DASH: name-based segmented video streaming. IEEE Trans Broadcast 61(3):346–355
37. Shelby Z (2012) Constrained RESTful environments (CoRE) link format. Internet Engineering Task Force (IETF), RFC 6690
38. Shelby, Z., Hartke, K., and Bormann C (2014) The Constrained Application Protocol (CoAP). Internet Engineering Task Force (IETF), RFC 7252
39. Skodzik J et al (2015) CoHaRT: a P2P-based deterministic transmission of large data amounts using CoAP, 2015 I.E. International Conference on Industrial Technology (ICIT), Seville, pp. 1851–1856
40. Tanganelli G, Vallati C, Mingozzi E (2015) CoAPthon: Easy Development of CoAP-based IoT Applications with Python. IEEE World Forum on Internet of Things (WF-IoT 2015), Milan, pp. 63–68

41. Thoma M, Braun T, Magerkurth C, Antonescu AF (2014) Managing things and services with semantics: A survey. 2014 I.E. Network Operations and Management Symposium (NOMS), Krakow, pp. 1–5
42. Wireshark: network protocol analyzer: <http://www.wireshark.org>. Accessed 19 Dec 2016
43. Wisniewski P, Beben A, Mongay Batalla J, Krawiec P (2015) On delimiting video rebuffering for stream switching adaptive applications. IEEE Int. Conference on Communications ICC, London
44. Xu J, Andreopoulos Y, Xiao Y, van der Schaar M (2014) Non-stationary resource allocation policies for delay-constrained video streaming: application to video over internet-of-things-enabled networks. IEEE J Sel Areas Commun 32(4):782–794

Piotr Krawiec (P.Krawiec@itl.waw.pl) received a Ph.D. degree in telecommunications from Warsaw University of Technology (WUT), Poland in 2011. He is an assistant professor at the Internet Technologies and Applications Department at the National Institute of Telecommunications, Poland, and concurrently holds a research position at WUT. His research interests lie in the areas of multimedia delivery, streaming protocols, wireless communications and the Internet of Things.

Maciej Sosnowski (m.sosnowski@tele.pw.edu.pl) received M.Sc. degree in telecommunications from Warsaw University of Technology in 2012, currently he is a Ph.D student. He is a member of the Internet Technologies and Applications Department at the National Institute of Telecommunications, Poland, and concurrently holds a research position at WUT. His research interests lie in the areas of queuing theory, virtualization techniques and the Internet of Things.

Jordi Mongay Batalla (J.Mongay@itl.waw.pl) PhD, is the Head of Internet Technologies and Applications Department at the National Institute of Telecommunications. He is also with Warsaw University of Technology where he is Assistant Professor and provides research in Internet protocols and applications, especially multimedia delivery, Internet of Things and Cloud/Edge computing. He has written more than 100 research papers in known international journals and conferences and is part of the editorial board of several journals.

Constandinos X. Mavromoustakis (mavromoustakis.c@unic.ac.cy) is currently a Professor at the Department of Computer Science at the University of Nicosia, Cyprus where he is leading the Mobile Systems Lab. He is a vice-chair of IEEE/R8 regional Cyprus section since Jan. 2016, and since May 2009 he serves as the Chair of C16 Computer Society Chapter of the Cyprus IEEE section. He has a dense research work outcome (more than 220 papers) in Distributed Mobile Systems and spatio-temporal scheduling, consisting of numerous refereed publications including several books, whereas he has served as track Chair and co-Chair of various IEEE International Conferences.

George Mastorakis (mastorakis@gmail.com) received his B.Eng. degree from the University of Manchester, his M.Sc. from University College London, and his Ph.D. from the University of the Aegean. He is Associate Professor at the Technological Educational Institute of Crete and Research Associate at the Centre for Technological Research of Crete. His research interests include cognitive radio networks, network traffic analysis, and radio resource management.