Multimed Tools Appl (2018) 77:15829–15849
DOI 10.1007/s11042-017-5157-0

Toward a full peer to peer MPEG-DASH compliant streaming system

Achraf Gazdar[1] · **Lamia Alkwai**[2]

Received: 18 October 2016 / Revised: 2 July 2017 / Accepted: 28 August 2017 /
Published online: 7 September 2017

Abstract MPEG-DASH standard has been proposed to standardize the proprietary solutions proposed to stream multi-qualities encoded video/audio over HTTP. Although MPEG-DASH relies on the client/server communication model, in this paper we show that it could be also used in a full P2P streaming system after making the necessary changes in the MPEG-DASH standard as well as in the main modules of the P2P streaming system. The main changes are made in the Media Presentation Description (MPD) file to support the P2P network bootstrapping mechanism (peers selection algorithm) as well as in the pieces selection algorithm to make the peer quality adaptation enabled as imposed by the MPEG-DASH. We choose the famous BitTorrent protocol as an example to implement the required changes. Extensive simulations have been conducted under the OMNeT++ simulator. The obtained results in terms of the average missed segments and the average waiting time on playback are reasonable with regards to 1) the BitTorrent known issue in the streaming context, to 2) the hard conditions we considered in our simulation set-up and 3) compared to some representative related works. This results show that MPEG-DASH could be easily supported by P2P streaming systems.

Keywords Adaptive streaming · BitTorrent · MPEG-DASH · Peer-to-peer

✉ Achraf Gazdar
agazdar@ksu.edu.sa

Lamia Alkwai
lalkwai@kacst.edu.sa

[1] Software Engineering Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

[2] National Center for Computation Technology and Applied Mathematics, King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia

1 Introduction

As the streaming of multimedia over the Internet has became more and more popular in the last decade, many streaming protocols and standards have been proposed. In this context, Real-Time Streaming Protocol (RTP) was proposed as a standard to stream media over IP networks. RTP allowed for the delivery of audio and video packets in managed IP networks with low overhead [12]. Usually it works combined with Real Time Control Protocol (RTCP) to manage the streaming session between the server and the clients. Unfortunately, RTP suffered from several disadvantages such as its inability to exploit the features of the existing Internet infrastructure, which was designed for file exchange, as well as the blocking of RTP packet by firewalls. Furthermore, heterogeneous terminals capabilities (resolution, processing powers, codecs. etc.) are very difficult to handle using RTP. Therefore, an alternative concept of streaming was proposed, which relies on Hypertext Transfer Protocol (HTTP) instead of RTP [11]. HTTP streaming has several benefits including the full exploitation of existing Internet infrastructure, such as the existing proxy caches, and Content Distribution Networks (CDNs). Moreover HTTP is firewall friendly and makes the client taking the responsibility to manage the streaming session, causing no overhead on the server.

Despite all the appealing advantages, HTTP streaming is not able to handle varying bitrates for heterogeneous clients, which is the case for most nowadays networks where the devices connected to the network are diversified. As a consequence, industry consortia addressed this problem by proposing adaptive streaming solutions. The Third Generation Partnership Project (3GPP) [1] proposed the Adaptive HTTP Streaming (AHS) [35].

The basic idea in AHS is to divide the media file into segments. Each segment is encoded in different bitrates. These segments are available on a web server and can be downloaded by clients using a classical HTTP requesting schema as shown in Fig. 1.

The client is responsible of requesting bitrates which are best suitable for its conditions, such as the current available bandwidth and the screen resolution. Therefore, the adaptation is done at the client side rather than at the server side. The client can dynamically switch between different bitrates as its conditions change as well.

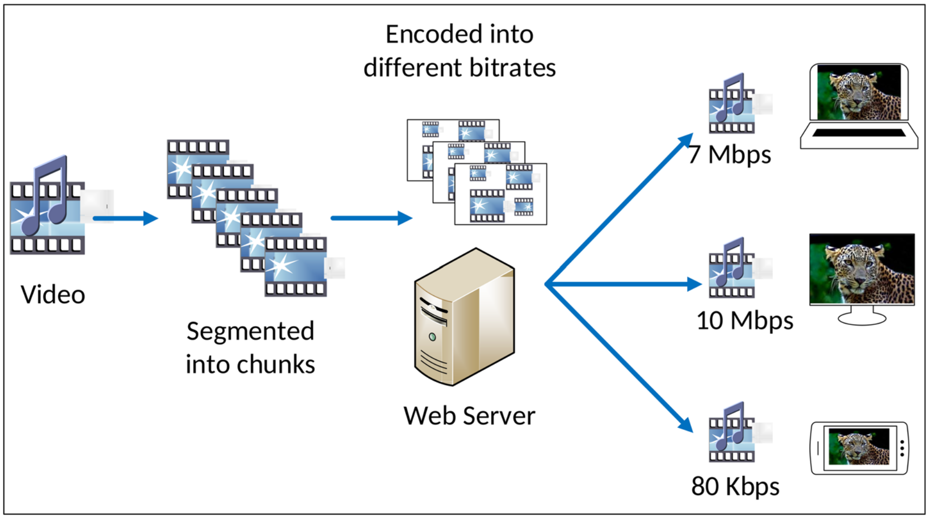

Fig. 1 Adaptive HTTP streaming (AHS)

To allow clients to know important streaming information such as the segments' start and end times, the available bitrates and the Uniform Resource Locator (URL) for each segment, a manifest file called Media Presentation Description (MPD) file was proposed. It is an Extensible Markup Language (XML) based meta-data file. Upon its arrival to the CDN, the client starts by downloading the MPD file to prepare the preferred segments downloading schedule for it which may vary at runtime.

The advantages of adaptive HTTP Streaming has led to the development of proprietary streaming platforms from different companies such as Apple's HTTP Live Streaming [30], Microsoft's Smooth Streaming [25], and Adobe's HTTP Dynamic Streaming [4]. Each of these proprietary streaming platforms implements a different manifest file and supports different segment formats, therefore, a client must be aware of -and able to interpret- the specific protocol which the platform implements for it to stream content from its servers. For this reason, the Moving Pictures Expert Group (MPEG), with the participation of the Third Generation Partnership Project (3GPP), developed a standard to allow interoperability between proprietary protocols and to combine their features, which will make it possible for any standard-based client to stream from any standard-based server. This standard is called Dynamic Adaptive Streaming over HTTP (DASH), also known as MPEG-DASH [36], and was published as ISO/IEC 23009-1:2012 in April, 2012. The standard's scope specifies:

1. The MPD data model, which describes the available content and representations, and other characteristics of the media.
2. The formats of the segments, which are the existing MPEG formats and any other compatible format.

A high level representation of an MPEG-DASH client-server architecture is shown in Fig. 2.

The scope of the MPEG-DASH specification does not include the means of delivering and distributing the MPD, the client's functionalities for fetching the content, the segment selection heuristics, and the decoding of the media content.

Furthermore, MPEG-DASH deals only with client-server based architectures which immediately raises the question (Q1) on the possibility to use the same MPD based signaling mechanisms in a full P2P (Video-on-Demand) VoD streaming context.

Actually, most P2P streaming systems offer a single selection for streaming parameters (such as bitrate and resolution), which are often based on average peer resources. However, given the high heterogeneity of peers in today's networks ranging from mobile phones to high-definition televisions, their streaming demands and preferences differ significantly according to their characteristics such as their bandwidth , decoding, and display capabilities. Therefore, distributing content encoded in a single resolution and bitrate does not guarantee the same level of quality for all peers.

This raises a second question (Q2) on how to adapt the P2P neighbors/media pieces (segments) selection algorithms to implement the adaptive media content downloading.

In this paper, we show the feasibility of a quality-adaptive P2P streaming system which adheres to the MPEG-DASH communication standard with regards to what have been done in related works to answer Q1 and Q2. We have chosen BitTorrent [17, 23] as an example for our feasibility study as it is one of the most commonly used P2P protocol with a well detailed technical documentation. Extensive simulations show reasonable performances of the system in terms of the average missed pieces and the waiting time on playback with regards to the BitTorrent limits on which our system is based and to the new media adaptation constraints imposed by the MPEG-DASH standard and compared to some representative works from the literature [21] and [22].

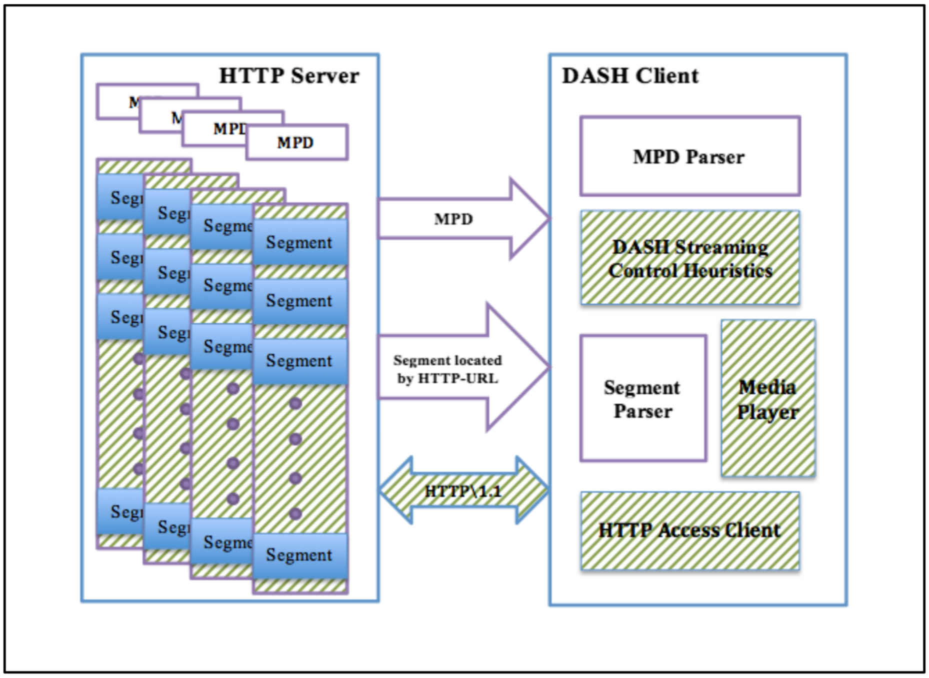

Fig. 2 High-level MPEG-DASH client-server architecture

The rest of the paper is organized as follows. In the next section we present the literature related works. Following, we present in the third section the design of the main changes to be done in BitTorrent to be an MPEG-DASH compliant protocol. Next, we introduce in the fourth section the implementation details, the simulation setup and the main performance evaluation results. Finally, we conclude with our main findings.

2 Related works

Many P2P streaming applications and systems have been successfully implemented, deployed, and used by customers, such as Coolstreaming [8], Anysee [5], and Joost [16]. Therefore, the majority of the literature in the field of P2P streaming came after the success of P2P streaming applications, and focuses on discussing the related challenges and opportunities, evaluating the performance of different systems and approaches [21] [22], and some suggesting enhancements and improvements in existing systems. The work in [31] is an example, which discusses the challenges and opportunities related to P2P video streaming. They studied the issue of which type of P2P networks is best for providing high quality video streaming, tree-based vs. mesh based, structured P2P network vs. unstructured, or maybe a hybrid of more than one type. Also, the ideas to ensure efficient data dissemination via different data replication strategies were investigated. Furthermore, the existing incentive mechanisms to encourage peer contribution in P2P networks were explored. The issue of peer heterogeneity was addressed by suggesting scalable coding video standards, and some other solutions such as deployment of a multi-tree structure.

The author in [21], evaluated the performances of the new Coolstreaming protocols. The main results show that the continuity index of the playback can reach 95% for an average upload link equal to 475 kbps and for a video streaming bitrate equal to 400 kbps.

In [22], the authors investigated a set of peers selection and pieces selection algorithms in a P2P streaming network. They found that the Random Useful Packet Forwarding (RUPF) outperforms the rest of the algorithms when combined with different peers selection algorithms. Some results in this work as well as in [21] will be used to compare the performance of our proposed system especially in terms of the average missed segments.

In [24], authors have evaluated the performance of video streaming over P2P networks. A simulation was conducted to analyze the performance of network level parameters which are (packet loss, transmission delay, and achieved throughput), and an application level parameter which is Peak Signal to Noise Ratio (PSNR). This work revealed some interesting results and conclusions, such as that packet loss is mostly caused by network congestion, while low video quality is more related to low bandwidth.

The work in [10] have focused on how to provide fault tolerant VoD in P2P networks. They have realized the differences in live and VoD streaming requirements and have proposed a P2P architecture for VoD which fulfills its special needs. The authors have also proposed a novel caching scheme to ensure a fast and localized failure recovery by introducing the concept of generation. A generation is a group of peers who arrive at close time intervals, therefore, they have the same beginning blocks in their caches and can substitute each other in case one member of a generation (which serves other peers from another generation) leaves. They have also focused on allowing quick peer join, which is done by minimizing the number of contacts a peer has to do in the joining phase and keeping the control information between peers as minimal as possible, or even on-demand in some cases.

Most P2P streaming systems offer single selection for streaming parameters, which are often based on average peer resources. Only some works [2, 3, 32] have focused on heteregonuous P2P streaming systems in terms of multi-qualities media encoding.

In [2], the authors have proposed a P2P streaming system based on scalable video coding (SVC). The proposed system is a mesh-based P2P streaming system, which allows both VoD and live-streaming. An SVC stream is divided into chunks which contain layers to improve quality in three dimensions: spatial, temporal and signal-to noise ratio (SNR). Quality Adaptation is performed in two steps: Initial Quality Adaptation (IQA), and Progressive Quality Adaptation (PQA). IQA is done at the beginning of a streaming session, to choose the appropriate quality level, based on static parameters (such as screen resolutions and processing power). PQA is performed periodically to adapt the quality selection to network conditions changes (such as number of peers and achieved throughput).The work in [3] presents an evaluation of the mechanism proposed in [2]. For the evaluation, two metrics related to the perceived video quality were defined, which are: session quality (which reflects the user's watching experience in terms of start-up delay, number of stalling events, and the average stalling duration), and the SVC quality (in terms of the received layers by a peer in relation to its static resources, and the number of layer changes during the video playback). The results showed that a quality adaptive VoD system provides better streaming sessions also, better SVC quality was achieved.

The authors in [32] have proposed PALS (P2P Adaptive Layered Streaming), which is a receiver-driven quality adaptive streaming approach. Using PALS, a receiver peer controls the delivery of a layered stream from multiple sender peers. This was performed in four steps: first a peer selects a number of sender peers which can provide a maximum throughput. Second, it decides the quality level based on monitoring the throughput periodically. Third, it determines the distribution of throughput among sender peers. Finally, it distributes the available bandwidth among the required layers from each sender peer. By adopting this approach, the receiver can control the stream quality, but has no control over the incoming throughput. Although preliminary results where presented in this paper, it forms a starting point for many streaming quality adaptation control approaches in P2P streaming system, in which the quality adaptation is performed at the receiver.

Some other works have dealt with the p2p video streaming adaptation using HTTP and caching (Hybrid P2P/CDN) techniques namely [33] and [34]. They propose to spread caches all-over the P2P networks to make the video segments as near as possible to the requesting peers. Peers can play the role of caches as well to avoid deploying more resources in the network.

The works presented until now demonstrate the utilization of the characteristics of adaptive video encoding standards and the HTTP based video streaming adaptation to achieve better streaming performance for heterogeneous nodes in a P2P network which partially answer Q2 (raised in the introduction) but not in the context of MPEG-DASH we deal with in this paper.

Works in the literature and the industry related to MPEG-DASH can be categorized into three directions:

1. Exploring different MPEG-DASH Applications, such as medical video streaming based on MPEG-DASH.
2. Using specific video coding standards in DASH streaming solutions, such as combining SVC with DASH.
3. Utilizing DASH in other Network Architectures, such as CDNs and CCNs.

In this paper, we only focus on the work which applies the MPEG-DASH standard into network architectures other then the one applied in the standard specification, which is the client-server architecture. Some of the architectures that were exploited, include: Content Centric Networks (CCN), Cloud environments, and peer-assisted client-server based networks.

In [9], the authors propose an Information Centric Network (ICN) P2P application for live streaming of video in the MPEG-DASH streaming format. The motivation is to exploit the useful functionalities offered by ICNs such as: routing by name, multicast delivery, and in-network caching to allow peers to improve playback quality. This improvement is achieved by having the prefetcher in the peer to concurrently download segments from the cellular (connection to the Internet) and the proximity (connection to full mesh one hop network) interfaces. Although the results show that peers where able to play a video at a coding rate close to the sum of their cellular capacities, testing the application on five peers only does not guarantee to achieve the same results as the system scales.

The work in [20] combines CCN and MPEG-DASH, and was motivated by having several characteristics in common, such as, the content is dealt with in pieces, and the pull of content initiated by the client. First, a demonstration of how to integrate MPEG-DASH into CCN was presented. This was done by having the DASH client to implement a native CCN interface. After that, overhead introduced by CCN header and protocol requirements was evaluated. The results show that the overhead introduced by CCN protocol is high in comparison to HTTP, which is a main drawback of the proposed technique. Also a comparison of the performance of delivering DASH-based content over HTTP and CCN was conducted. The comparison showed that the achieved performance is comparable with HTTP 1.0, however HTTP 1.1 outperforms it significantly.

The authors in [19] have extended their implementation in [26] to enable a peer-assisted streaming architecture instead of client-server, while keeping conformance to the

MPEG-DASH standard. Therefore, only DASH-compliant communication exists between the servers and the peers, and among the peers. In order to do this, some enhancements were presented to the MPD file, the functionalities of the client, and the HTTP web server operations, as summarized in Table 1.

A simulation of the proposed architecture was implemented, including 35 peers and an HTTP server, and based on the HTTPTools framework [15], and the VLC plug-in described in [26]. The results showed that, using the peer assistance, the server bandwidth usage was reduced by about 15%. Also, the P2P traffic contributed to more than 50% of the downloaded segments of a client after a specific time period.

Although this work pioneered the idea of adopting P2P streaming to the MPEG-DASH standard, it still relies on the client-server architecture, and does not support a full P2P model. Only P2P streaming is allowed if certain conditions were met, otherwise the content would be streamed from the web server. Also the selection of a neighboring peer to stream content from -in the case where there are several neighbors having the required segment - is done randomly, and thus better decisions may be possible by applying a different selection strategy, resulting in better load balancing.

The most relevant work to us is MyMedia [18]. The authors contribution can be summarized into 1) a semantic based video searching module and 2) an MPEG-DASH based P2P live streaming module. Video searching is out of the scope of our paper however we are interested in contribution 2). The proposed solution proposes to modify the MPD file as done by [19] to add the eventual base url for each video segment and as well as a new peer context section to describe the neighbor capabilities in terms of the maximum supported downloading peers and the currently connected peers. Based on this new section the peer selects the best suited neighbors to download segments from. While this mechanism can work perfectly in a live context, it can induct a considerable overhead in VoD scenarios. In fact, the number of segments to be encoded in the MPD file in a live context is very small compared to a VoD scenario where a very great number of segments should be added to the MPD file especially when the media is encoded in many representations (qualities). This mechanism makes almost impossible to the peer to download the MPD files from its neighboring to perform the peer/segment selection algorithm. What we propose instead in our current work, is to download once the MPD file from the tracker, contact the peers encoded in the base URL and build the bitmap data structure to perform the segments selection algorithm. The bitmap

Table 1 Changes in components for allowing peer-assisted DASH [26]

MPD	The MPD was modified by adding base-URLs of the peers which have already downloaded a segment to the corresponding representation. Also, MPD is updated to incorporate peers that are online only, and for a specified time interval to keep the size of the MPD manageable
Peer	Three modifications were made to clients: First, a peer's incoming connections were restricted to 4 connections maximum at any given time, to guarantee a minimum upload bandwidth. Second, is the modification regarding the download and stream switching logic, such as a peer only allowed to stream from another peer if its buffer fill level is higher then 50%, to ensure smooth playback. The third was to make available large local caches, and using commercial systems to maintain even bigger caches, to increase the P2P traffic and decrease the server load
HTTP Server	A central segment tracker was implemented to run on the HTTP server to track each peer that has requested a segment, in addition to its IP address, and a request timestamp. This tracker assisted the MPD generator to update the MPD

is continuously kept up to date through the HAVE mechanism (to send the id of the recently downloaded segment to all the neighbors) instead of downloading MPD files for each new refresh period. More details on our proposed system will be given in the next section.

Table 2 summarizes the main relevant related works to us in the literature. As it is shown in Table 2, no work has dealt with the adaptation in a full P2P system (and not a peer-assisted solutions) within the MPEG-DASH context except MyMedia [18] but in a live streaming context. However, we discussed in the previous paragraph its inefficiency in a VoD oriented P2P system which we consider in our current paper.

3 Toward a compliant MPEG-DASH P2P streaming system

As we have announced in the introduction, we selected BitTorrent as base for our feasibility study of a full MPEG-DASH compliant P2P streaming system. The new component we designed and added as a first contribution is the DASH based signalizing mechanism instead of the torrent based mechanism. In other words, upon its arrival a peer starts by downloading an MPD from a public web server instead of downloading a BitTorrent file. More details about the MPD file based starting process will be given in the next subsection.

Our second contribution deals with the implementation of the HTTP media downloading capability in the OMNET++ BitTorrent module [17] to be compliant with the MPEG-DASH standard as well as the adaptation capability through a quality driven segments selection algorithm. This latter is a new constraint imposed by the MPEG-DASH standard to any P2P streaming protocol.

3.1 MPD based starting process

As portrayed in Fig. 3, the only change we have made in the MPD file to keep it as closest as possible to the standard structure is the adding of the tracker attribute in the BaseURL tag. It is a boolean attribute allowing the peer to know whether it should proceed with the MPD file as a normal MPEG-DASH CDN meta-file or as a P2P meta-file. When the value is equal to *yes* then the P2P mode is activated.

The rest of the initialization process is kept as done in the BitTorrent specification. The new arrival peer sends an announce request to the Trackers and gets back its randomly

Table 2 Related work in terms of supported architecture, MPEG-DASH compliance, and video streaming type

Reference	Architecture	MPEG-DASH compliance	Video streaming
[5, 8, 16]	P2P	No	Live streaming
[10]	P2P	No	VoD
[2]	P2P	No	VoD and Live streaming
[32]	P2P	No	Live streaming
[9]	ICN P2P	Yes	Live streaming
[20]	CCN	Yes	VoD
[19]	Peer-assisted	Yes	VoD
[18]	P2P	Yes	Live streaming
Our system	**P2P**	**Yes**	**VoD**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- MPD file Generated with GPAC version 0.5.1-DEV-rev4229 -->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.5S" type="static"
  mediaPresentationDuration="PT1H26M" profiles="urn:mpeg:dash:profile:isoff-main:2011">
 <BaseURL>./</BaseURL>
 <BaseURL tracker="yes">http://tracker.com</BaseURL>
 <BaseURL>tracker="no">download.tsi.telecom-paristech.fr/gpac/dataset/dash/mmsys13/</BaseURL>
 <BaseURL>...</BaseURL>
 <Period duration="PT1H26M">
  <AdaptationSet segmentAlignment="true" group="1">
   <Representation id="1" mimeType="video/mp4" codecs="avc1.42c00d" width="320" height="240"
                   frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="101355">
    <SegmentList timescale="1000" duration="2001">
    <Initialization sourceURL="redbull_240p_100kbps_2sec_segmentinit.mp4"/>
    <SegmentURL media="redbull_240p_100kbps_2sec_segment1.m4s"/>
    <SegmentURL media="redbull_240p_100kbps_2sec_segment2.m4s"/>
    <SegmentURL media="redbull_240p_100kbps_2sec_segment3.m4s"/>
    <SegmentURL media="redbull_240p_100kbps_2sec_segment4.m4s"/>
    <SegmentURL media="redbull_240p_100kbps_2sec_segment5.m4s"/>
    <SegmentURL media="redbull_240p_100kbps_2sec_segment6.m4s"/>
    <SegmentURL media="redbull_240p_100kbps_2sec_segment7.m4s"/>
    <SegmentURL media="redbull_240p_100kbps_2sec_segment8.m4s"/>
        ...
        </SegmentList>
   </Representation>
  </AdaptationSet>
 </Period>
</MPD>
```

Fig. 3 New MPD file structure

generated neighbors list. A normal handshake process starts between the peers and the downloading/playback process starts. The downloading process is done following a segments selection algorithm driven by the current suited quality of the peer. More details on the quality driven segments selection algorithm will be given in the next subsection.

During the download/playback session, a peer periodically sends a new announce request to the trackers to update his neighbors list. The tracker selects the peers' list to be sent back to the peer as done by the BitTorrent protocol.

3.2 Quality-driven segments selection algorithm

Each peer gets configured by its maximum supported quality (the maximum downloading rate it can manage). When it parses the MPD file, it allocates a downloading buffer large enough to accommodate the media segments of the maximum supported qualities as well as the segments of lower qualities. For example, assuming that a movie is encoded in 3 different qualities requiring 1 Mbps, 512 Kbps, 256 Kbps playback rates (CBR playback assumed) respectively where 5 segments are present in each representation (quality). Let's assume also, that the maximum supported quality by a newcomer peer p_1 is 1Mbps. In this case the peer should allocate a buffer able to store 15 media segments. The downloading of segments with lower qualities allows a smooth adaptation of the peer whenever it experiences difficulties to download higher media quality. It permits the peer to be able to interact with heterogeneous neighbors in terms of segments qualities available in their buffers as well. Figure 4 portrays an example of buffers' structures of a peer with the highest quality 1 Mbps, a peer with the medium quality 512 Kbps and a peer with the lowest quality 256 Kbps after elapsing some time in the system. Peer p_1 could have a buffer structure similar to the buffer structure illustrated in the top of Fig. 4.

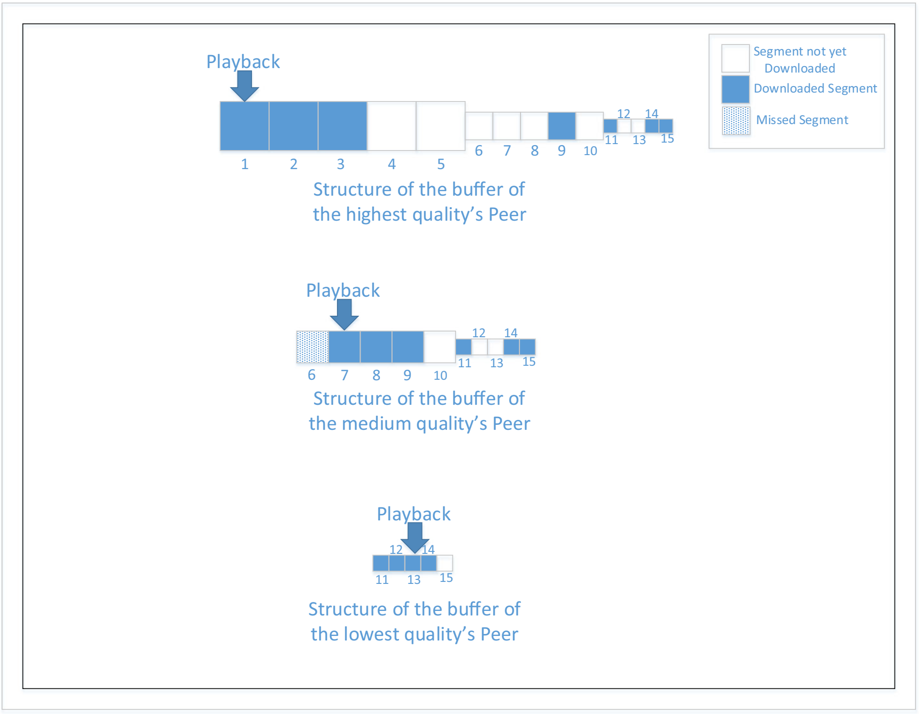
Segment not yet Downloaded
Downloaded Segment
Missed Segment
Playback
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Structure of the buffer of the highest quality's Peer
Playback
6 7 8 9 10 11 12 13 14 15
Structure of the buffer of the medium quality's Peer
Playback
11 12 13 14 15
Structure of the buffer of the lowest quality's Peer

Fig. 4 Examples of peers' buffers structures

The playback of the media starts after a buffering period denoted b. During the buffering and the playback process, the peer runs a FIFO segment selection algorithm in the last selected quality as shown in line #7 of Algorithm 1. The selected quality is initially set to the maximum supported quality. When the difference between the latest downloaded segment in the last selected quality and the currently playing segment exceeds a threshold denoted f the peer switches the segments selection algorithm into RAREST-First (line #5 in algorithm 1). This reminds the Bitos [37] pieces' selection algorithm but here we consider many qualities instead of only one quality as considered in [37]. Note that the last downloaded segment considered in the segments selection algorithm is the last segment in the longest sequence formed by the consecutive segments starting by the last played segments. As shown in line #3 of Algorithm 1, the *ToDownloadSet* is calculated each time the *SeekNextSegmentToDownload* is called in order to inform the segments selection algorithms (FIFO and RAREST-First) which segments are to be downloaded for playback. As an example we can consider the peer with the highest quality in Fig. 4. Let's assume that $f = 2$. As portrayed in the figure, the difference between the last downloaded segment in the highest quality segment #3 and the last played segment #0 (because the peer is still playing the first segment) is equal to 3 (greater than f) which allows the peer to download the rest of the segments using the RAREST-First algorithm. This explains the presence of segments #9, #11, #14 and #15 from the lower qualities in the peer's buffer.

Algorithm 1 Segments selection algorithm

```
1:  procedure SEEKNEXTSEGMENTTODOWNLOAD(
    $lastQuality, lastPlayedSegmentIndex, lastDownloadedSegmentIndex, f$)
2:      $segmentToDownload \leftarrow -1$
3:      $ToDownloadSet \leftarrow getListOfSegmentsToDowload(localBitMap,$
    $lastPlayedSegmentIndex)$
4:      if $(lastDownloadedSegmentIndex - lastPlayedSegmentIndex) > f$ then
5:          $segmentToDownload \leftarrow$
    $RarestSegmentSelection(lastDownloadedSegmentIndex,$
    $remoteBitmaps, ToDowloadSet)$
6:      else
7:          $segmentToDownload \leftarrow$
    $FIFOSegmentSelection(lastDownloadedSegmentIndex,$
    $remoteBitmaps, ToDownloadSet)$
8:      end if
9:      return $segmentToDowload$
10: end procedure
```

As detailed in Algorithm 2, each time the playback process misses one segment in the last selected quality (denoted *lastQuality* in Algorithm 2) it performs a small waiting period of the segment to be available in any supported qualities (the maximum supported qualities

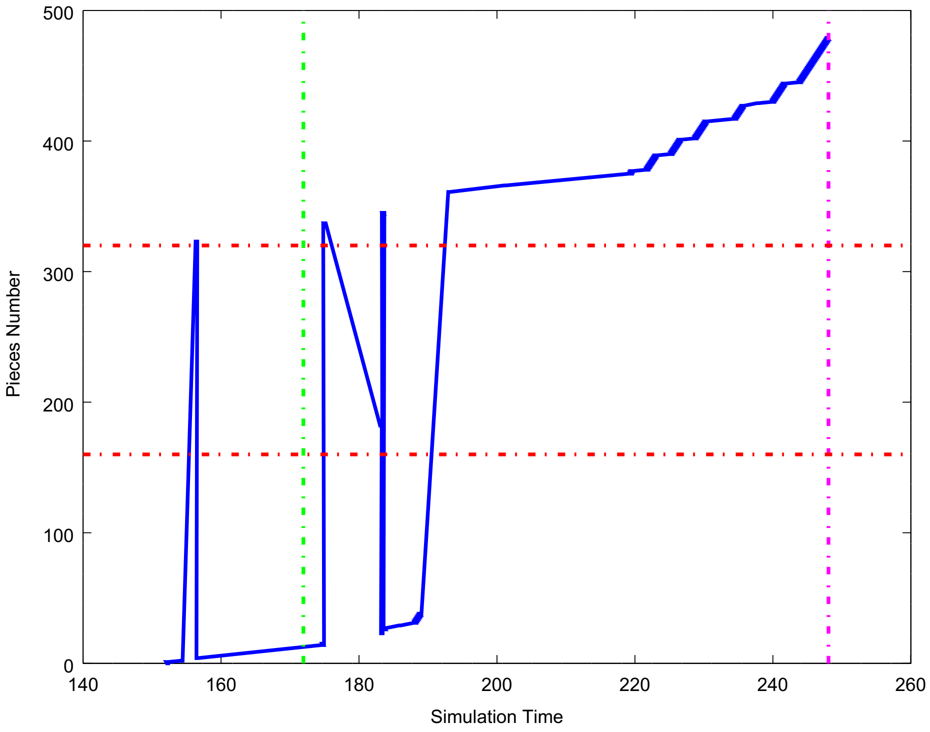

Fig. 5 An example of the behavior of a peer running the quality driven adaptation algorithm

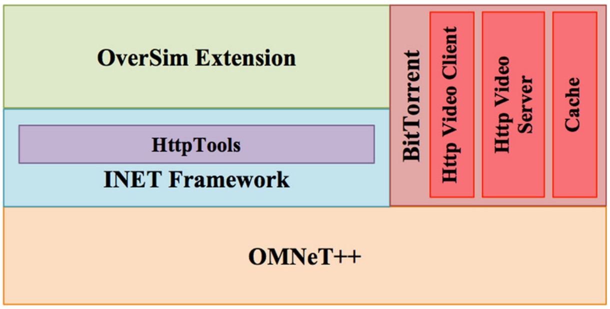

Fig. 6 New components added to the bittorrent peer (in red color)

and the lower qualities as well). This waiting period is denoted np (line #24 in Algorithm 2). Note that when a required segment is not found in the last selected quality, the player tries to get the same segment but in a lower quality in the buffer (from line #9 to line #12 in Algorithm 2). After np seconds, if the required segment is still not available the playback process skips to the next segment in the last selected qualities (line #34 in Algorithm 2). A successive missed segments counter denoted $missed$ is incremented each time a segment is missed during the playback (line #28 in Algorithm 2). If the number of the successive missed segments exceeds a threshold denoted m_thresh ($missed > m\_thresh$) the peer downgrades the last selected quality to the next lower quality (from line #29 to line #31 in Algorithm 2). A second counter denoted $succ$ is used to count the successive downloaded segments in a given quality. If $succ$ exceeds a threshold denoted s_thresh, the peer upgrades its last selected quality to the next higher supported quality (from line #17 to line #19 in Algorithm 2).

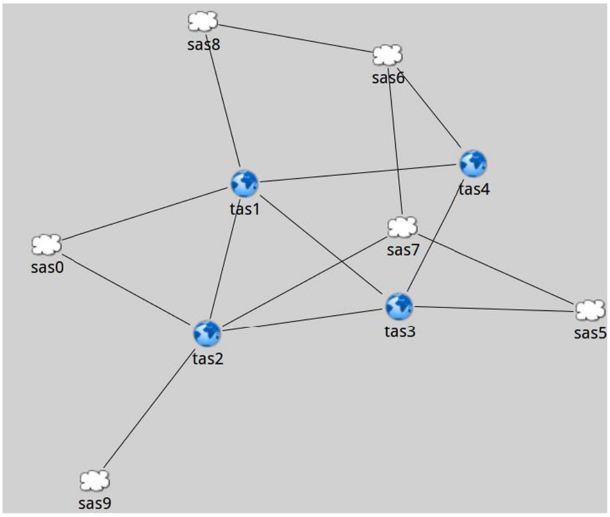

Fig. 7 Topology used in our simulations

Algorithm 2 Playback adaptation algorithm

```
1: procedure PlayTheNextSegment($segmentIndex, lastQuality, np, missed, m\_thresh, succ, s\_thresh, playTentative$)
2:     $qualityIndex \leftarrow lastQuality$
3:     $nextSegmentToPlay \leftarrow qualityIndex * videoSizePerQuality + segmentIndex - 1$
4:     $found \leftarrow false$
5:     while $qualityIndex \leq lowestQualityIndex$ do
6:         if $isInBuffer(nextSegmentToPlay)$ then
7:             $found \leftarrow true$
8:             $break$
9:         else
10:            $qualityIndex++$
11:            $nextSegmentToPlay \leftarrow qualityIndex * videoSizePerQuality + segmentIndex - 1$
12:        end if
13:    end while
14:    if $found$ then
15:        $missed \leftarrow 0$
16:        $succ++$
17:        if $succ > s\_thresh$ then
18:            $UpgradeQualityIndex(lastQuality)$
19:        end if
20:        $PlaySegment(nextSegmentToPlay)$
21:        $playTentative \leftarrow 1$
22:    else
23:        if $playTentative \leq 1$ then                ▷ Try to play after np seconds
24:            $scheduleTask(PlayTheNextSegment, np)$
25:            $playTentative++$
26:        else
27:            $succ \leftarrow 0$
28:            $missed++$
29:            if $missed > m\_thresh$ then
30:                $DowngradeQualityIndex(lastQuality)$
31:            end if
32:        end if
33:    end if
34:    $segmentIndex++$                  ▷ get ready to play the next segment
35: end procedure
```

Figure 5 illustrates the behavior of a peer set to the highest quality bitrate (1Mbps) running the quality driven adaptation algorithm. As we can see in this figure, the peer switches many times from one quality to another while downloading the media segments. Finally it decides to finish the playback of almost the half of the movie in the lowest quality. The set-up simulation considered in this example is summarized in Table 4. Note here that the highest quality is represented by the lower horizontal band in the figure, the medium quality is represented by the horizontal band in the middle while the lowest quality is represented by the higher horizontal band. The bands are bounded by the red lines.

The peers use the HTTP GET method to ask for segments from their neighbors. Let us consider the example of the first paragraph in the current subsection. The segment number should belong to {1, 2, ..., 15}. When a peer receives an HTTP GET request for segment number 12, it immediately determines that it should send back the segment number 2 in the lowest quality as $12 = 5 * 2 + 2$. Here, we suppose that the highest quality (1Mbps) corresponds to the lowest index 0 while the lowest one corresponds to the greatest index 2.

4 Performance evaluation of the MPEG-DASH compliant P2P system

To evaluate the performances of our system, we have introduced changes on the OMNET++ [28] implementation of the BitTorrent protocol [17] which can be summarized as follows:

- Porting the BitTorrent implementation to version 4.x of OMNET++ simulator
- Add support for multiple seeders
- Add support for HTTP media segments retrieval to be DASH-compliant
- Add support of HTTP caches in the P2P network to be compliant with the real world PROXY/CACHE based networks

The implementation is spread among the INET [27] and the OVERSIM frameworks [6]. We made the whole implementation available in Github [14, 29].

4.1 OMNET++ simulator implementation details

In addition to the implementations of the adaptation algorithm and the segments selection algorithm presented in the previous section, we have added support for HTTP segments media retrieval to be compliant with the MPEG-DASH standard as portrayed by Fig. 6. An HTTP server has been added to the BitTorrent node using the HttpTools API provided with OMNET++ [15]. The HTTP server has full access to the local bitmap of the peer (on which it runs) as well as the media buffer to know whether a “404 NOT FOUND” HTTP response or a “200 OK” HTTP message with the media segment enclosed should be sent back to the requesting peer.

We implemented an HTTP client as well in each BitTorrent peer to enable requesting media segments using the HTTP protocol.

A new kind of node (Hostmodule) has also been implemented and added to the simulated network which is the cache host. It allows the HTTP segments caching in the P2P network. The caching replacement policy uses the LRU algorithm (Least Recently Used) but it can be easily replaced by any other policy in the simulation model. The implementation of this new node is based on the HttpTools API [15] too. The cache nodes should be connected to the access routers. To be able to communicate with the cache node, a cache sub-module has also been implemented in each BitTorrent peer as illustrated by Fig. 6.

4.2 Simulation setup

Three main simulation configurations have been investigated to evaluate the performances of our new MPEG-DASH compliant P2P system in terms of the segments missing rate which affects the playback smoothness and the waiting time experienced during the playback. In the first configuration, we vary the number of peers in the network. In the second configuration, we vary the playback startup delay. In the third one, we vary the size of the caches deployed in the caches node all over the network. In order to run realistic simulations, we used the ReaSE topology generator [7] which allows to generate an Internet-like topology with background traffic. The topology that we used is portrayed in Fig. 7. It contains 6 *Stub Autonomous System* (SAS) interconnected with 4 *Transit Autonomous System* (TAS). The TAS and SAS are composed of core routers, gateway routers and edge routers. The background traffic we used involves several types of traffic servers including mailing servers, streaming servers, naming servers and web servers. The average number of terminals connected to each edge router is 120 with an average number of 10 servers. Each terminal randomly opens connections with the servers asking for services. Other types of interactive traffic could happen between terminals as well. We considered the same medium background traffic profiles used in [7] in our simulated network. Those traffic profiles are summarized in Table 3.

In this paper, we suppose a stable DSL based network without any peer's churn. Table 4 summarizes the simulation parameters set-up used in all the simulation configurations. It is worthy noting that 30 runs are done for each value of the upcoming studied parameters (the number of peers, the playback start-up delay, the cache size).

4.3 Simulation results

In Fig. 8a and b, we plot the average number of the missed segments and the average waiting time during the playback as a function of the number of peers in the network.

Caches are randomly deployed in the network with maximum capacity equal to the 25% of the media file size. The average cache concentration (the number of peers behind one common cache) is equal to 2 peers per cache. This choice is made in order to attenuate the caching effect in the whole network. The start-up delay is fixed to 20 seconds.

It is clear that the peers playing the highest quality are paying the highest cost in terms of missed segments and waiting time. However the overall, network is showing a reasonable performance where the average missing segments do not exceed 18% of the total segment number (160) compared to [21] where the missed segments percentage varies from 60% (continuity index equal to 0.4) to 10% for its best configuration with a startup delay exceeding 20 seconds and a single quality video with a bitrate equal to 400 kbps and an average upload capacity of peers equal to 475 kbps.

Table 3 Background traffic profiles

Parameters/Profile	Interactive	Web	Mail	Streaming	UDP Misc	Ping
Request Length (KB)	100	200	4000	250	300	64
Requests Per Session	70	10	10	1	10	3
Reply Length (KB)	400	1000	100	250	300	64
Reply Per Request	1	30	1	1000	2	4
Time Between Requests (sec)	0.5	2	0.1	1	1	1
Time To Respond (sec)	0.1	0.5	0.1	0.5	0.5	0.2
Time Between Sessions (sec)	1	3	3	5	3	1
Ratio in the Network (%)	3.2	36	2.8	3.9	9.7	4.3
Ratio in WAN (%)	11	73	5	27	4	50

Table 4 Simulation parameters setup

Attribute	Value
Media file size	20 MB encoded into 3 qualities
Considered Qualities' Rates	1 Mbps, 512 Kbps, 256 Kbps
Number of Trackers	1
Piece (Segment) size	128 KB
Number of Blocks per Piece	1
Total Number of Segments per Quality	160
Peers' Access Technology	DSL (average 4 Mbps)
Peer's Arrival	Poisson (0.002777 peers/sec)
Number of seeders	10% of the total peers number
Next Piece Waiting Duration if Not Found	2 seconds
FIFO to RAREST-First switching window f	6 pieces

Our quality driven algorithm introduces an average overhead of almost 12% compared to the random piece selection policy (RP) investigated in [22] and about 14% to the best piece selection policy named Random Useful Packet Forwarding (RUPF) presented in the same work considering an average peer upload rate of 1 Mbps and a single quality video streaming of rate varying from 480 kbps to almost 1 Mbps.

The overall waiting time on playback for the whole network increases when the number of peers on the network increases. This means that the peer takes more time in looking for pieces when the number of peer increases especially the peers playing the media with the highest quality. Nevertheless, we should think again on redesigning the segments lookup and exchanging to enhance the system performances in this regard especially for the peer asking for the highest quality even if the whole system is tested in hard conditions with an average of 4Mbps DSL access technology as well as a very reduced peers' arrival rate.

It is worth noting that our quality driven algorithm tends to a stable network status starting from a peers' number equal to 60 as portrayed in Fig. 8a and b.

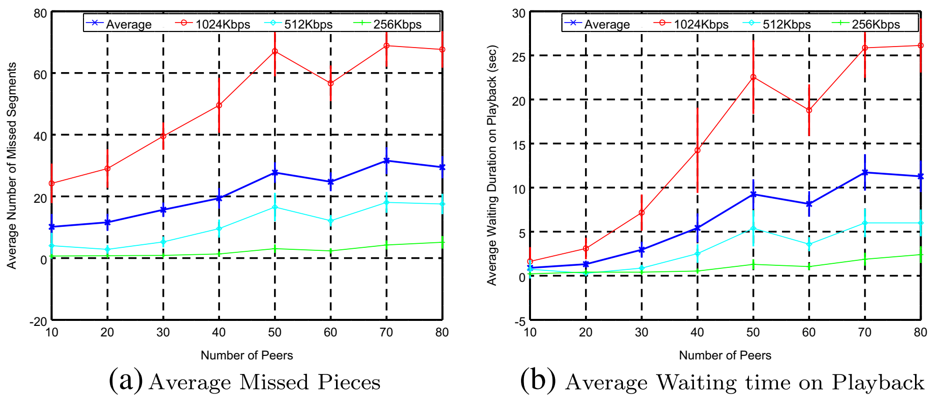

Fig. 8 Performances when varying the number of peers

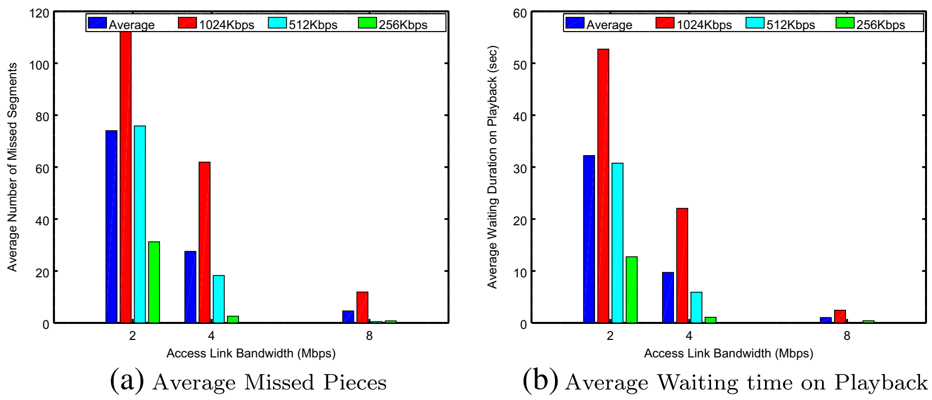

Fig. 9 Performances when varying the access link bandwidth

Besides, when we increased the average bandwidth of the access link to 8Mbps, all the peers experienced a smooth playback with almost no missed pieces and no waiting time on playback. We plot these results in Fig. 9.

Figure 10a and b plot the average number of missed segments and the average waiting time on playback as a functions of the startup delay respectively. 60 peers are considered in the network with the same caches deployment policy as in Fig. 8.

Again the peers playing the media with the highest quality are paying the highest cost in terms of the missed segments and the waiting time during the playback. However, the plots have a decreasing trends when the startup delay increases which show that the system performances have gradually improved as a function of the startup delay. Nevertheless, more improvement could be done through adding special nodes (CDN-infrastructure or a special peer) in the network with special capabilities to enhance the overall system performance as done in [13], μTorrent, eMule, Kazaa and Tribler P2P systems.

Figure 11 plots the average number of missed pieces and the average waiting time on playback as a function of the cache size for a cache concentration equal to 6. These figures

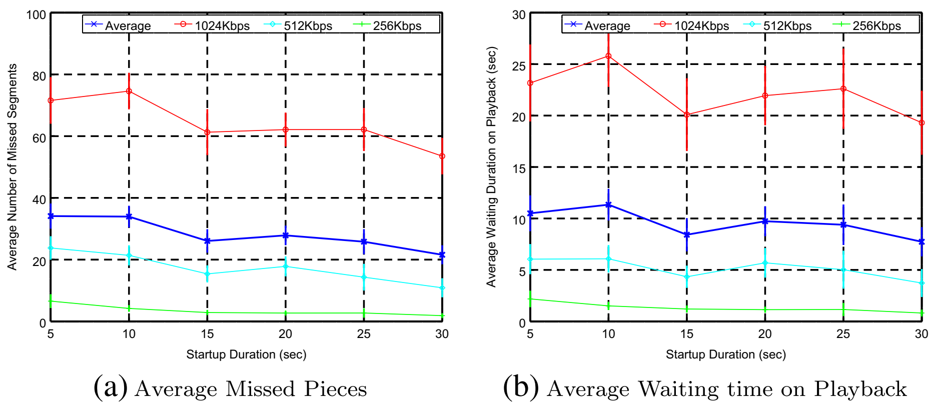

Fig. 10 Performances when varying the startup delay

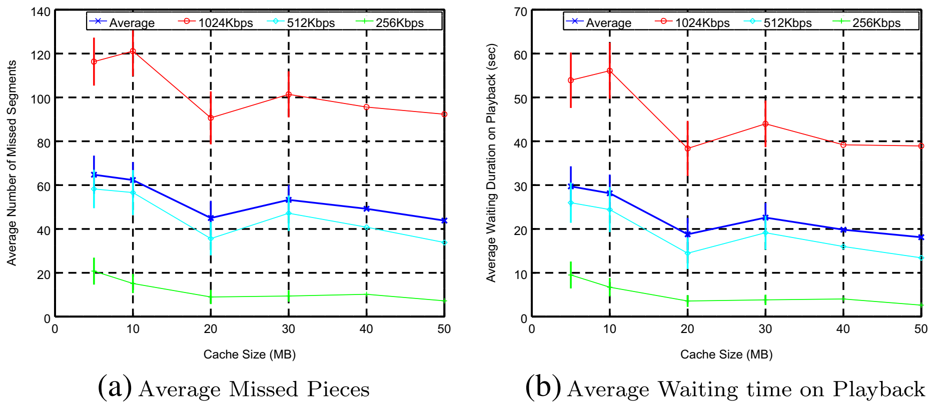

Fig. 11 Performances when varying the cache size

portray a decreasing trend of both plots which show that the overall system performances in terms of the average number of missed pieces and the average waiting time on playback get improved when the cache size increases.

From an ISP (Internet Service Provider) friendly point of view, the cache deployment allows to reduce the traffic in the considered network and allows a natural physical clustering of the peers interested in the same media content. This result is illustrated by Fig. 12.

In Fig. 12a, we plot the cache hits and misses when varying the cache size while we vary the cache peers' concentration in Fig. 12b. 60 peers are considered in the simulations scenario and the rest of the parameters are the same as presented in Table 4. As we can see in these figures, increasing the cache size and the cache concentration allows to minimize the ISP network traffic. Indeed, in both figures the number of hits increases when the cache size and the cache concentration increase. As shown in Fig. 12a, the cache size value which maximizes the number of hits is equal to 40 MB which is the closest simulations setup to the theoretical optimal value equal to 35 MB = 20 MB + 10 MB + 5 MB where 20 MB is the file size having the highest encoding quality 1 Mbps, 10 MB is the file size in the medium

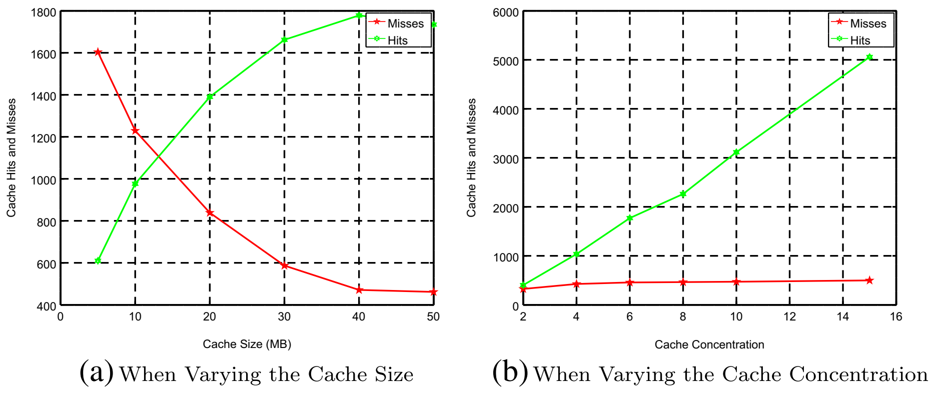

Fig. 12 Average cache hits and misses

encoding quality 512 Kbps and 5 Mb corresponds to the file size in the lowest encoding quality 256 Kbps.

5 Conclusion

The MPEG-DASH standard has been initially proposed for adaptive live and VoD streaming CDN context only. Most of the related works dealt with MPEG-DASH in a peer assisted live streaming context or in CCN networks.

In this paper, we investigated the feasibility of a full MPEG-DASH compliant P2P streaming system. We have showed that with a small modification of the MPEG-DASH standard and especially the MPD file we can build a full P2P streaming system compliant with the standard. In this work, we have integrated the MPEG-DASH standard within a BitTorrent based system but it can also be added into any existent P2P system without any problem. We have modified the segments'(pieces) selection algorithm to make it quality driven based and to allow automatic adaptation of the peer.

We have implemented and evaluated our P2P system in the Omnet++ simulator. The simulation results show that the overall system performances of our proposed system in terms of the average missed segments and the waiting time on playback are reasonable with regards to 1) the BitTorrent known issue in the streaming context [37], to 2) the hard conditions we considered in our simulation set-up and 3) compared to some representative works namely [21] and [22] .

As a future work, we are planning to:

1. revise the peers' selection algorithm and the segments (pieces) selection algorithm to enhance the performances of our system especially for the peers demanding the highest streaming quality,
2. study other parameters of the system such as the segment size variation, FIFO to RAREST-FIRST switching window and the Next Piece Waiting Duration,
3. investigate the system performances in a peers churn context.

Acknowledgements This work was supported by the Research Center of College of Computer and Information Sciences, King Saud University. The authors are grateful for this support.

References

1. 3gpp (2015) Online http://www.3gpp.org/, accessed January 6
2. Abboud O, Pussep K, Kovacevic A, Steinmetz R (2009) Quality adaptive peer-to-peer streaming using scalable video coding. In: Wired-wireless multimedia networks and services management. Springer, pp 41–54
3. Abboud O, Zinner T, Pussep K, Al-Sabea S, Steinmetz R (2011) On the impact of quality adaptation in svc-based p2p video-on-demand systems. In: Proceedings of the second annual ACM conference on multimedia systems. ACM, pp 223–232
4. Adobe (2015) Online http://www.adobe.com/products/hds-dynamic-streaming.html, accessed January 6
5. Anysee (2015) Online http://www.anyseedirect.eu, accessed March 4
6. Baumgart I, Heep B, Krause S (2007) Oversim: a flexible overlay network simulation framework. In: IEEE global internet symposium, 2007. IEEE, pp 79–84
7. Baumgart I, Gamer T, Hübsch C, Mayer CP (2011) Realistic underlays for overlay simulation. In: 4th international ICST conference on simulation tools and techniques, SIMUTools '11, Barcelona, Spain, March 22–24, 2011, pp 402–405. https://doi.org/10.4108/icst.simutools.2011.245526
8. Coolstreaming (2015) Online http://www.coolstreaming.us, accessed March 4

9. Detti A, Ricci B, Blefari-Melazzi N (2013) Peer-to-peer live adaptive video streaming for information centric cellular networks. In: IEEE 24th international symposium on personal indoor and mobile radio communications (PIMRC), 2013. IEEE, pp 3583–3588
10. Do TT, Hua KA, Tantaoui MA (2004) P2vod: providing fault tolerant video-on-demand streaming in peer-to-peer environment. In: IEEE International conference on communications, 2004, IEEE, vol 3, pp 1467–1472
11. Fecheyr-Lippens A (2010) A review of http live streaming. Internet Citation, pp 1–37
12. Frederick R, Jacobson V, Design P (2003) RTP: a transport protocol for real-time applications. IETF RFC3550 http://fumblog.um.ac.ir/gallery/187/RFC203550.pdf
13. Hammami C, Jemili I, Gazdar A, Belghith A, Mosbah M (2015) HLPSP: a hybrid live P2P streaming protocol. TIIS 9(3):1035–1056. https://doi.org/10.3837/tiis.2015.03.011
14. INET-Authors (2016) Dash bittorrent implementation in inet. Online, https://github.com/hyperonex/INET-OverSim-20101019
15. Jónsson KV (2009) Httptools: a toolkit for simulation of web hosts in omnet++. In: Proceedings of the 2nd international conference on simulation tools and techniques, ICST, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, p 70
16. Joost (2015) Online http://www.joost.com/pause/, accessed March 4
17. Katsaros K, Kemerlis VP, Stais C, Xylomenos G (2009) A bittorrent module for the omnet++ simulator. In: IEEE international symposium on modeling, analysis & simulation of computer and telecommunication systems, 2009. MASCOTS'09. IEEE, pp 1–10
18. Klusch M, Kapahnke P, Cao X, Rainer B, Timmerer C, Mangold S (2014) Mymedia: mobile semantic peer-to-peer video search and live streaming. In: Proceedings of the 11th international conference on mobile and ubiquitous systems: computing, networking and services, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, MOBIQUITOUS '14, pp 277–286. https://doi.org/10.4108/icst.mobiquitous.2014.258026
19. Lederer S, Muller C, Timmerer C (2012) Towards peer-assisted dynamic adaptive streaming over http. In: 19th international packet video workshop (PV), vol 2012, pp 161–166. https://doi.org/10.1109/PV.2012.6229730
20. Lederer S, Mueller C, Rainer B, Timmerer C, Hellwagner H (2013) An experimental analysis of dynamic adaptive streaming over http in content centric networks. In: IEEE international conference on multimedia and expo (ICME), 2013. IEEE, pp 1–6
21. Li B, Xie S, Qu Y, Keung GY, Lin C, Liu J, Zhang X (2008) Inside the new coolstreaming: principles, measurements and performance implications. In: INFOCOM 2008. The 27th conference on computer communications. IEEE. https://doi.org/10.1109/INFOCOM.2008.157
22. Liang C, Guo Y, Liu Y (2009) Investigating the scheduling sensitivity of p2p video streaming: an experimental study. Trans Multi 11(3):348–360. https://doi.org/10.1109/TMM.2009.2012909
23. Mason M (2014) Coming soon: smartphone streaming, powered by bittorrent live. BitTorrent Blog http://blog.bittorrent.com/2014/02/14/coming-soon-smartphone-streaming-powered-by-bittorrent-live/
24. Memon SA, Hassan SR, Memon NA (2014) Evaluation of video streaming performance over peer-to-peer network. In: International conference on collaboration technologies and systems (CTS), 2014. IEEE, pp 413–420
25. Microsoft (2015) http://www.iis.net/downloads/microsoft/smooth-streaming, accessed January 6
26. Müller C, Timmerer C (2011) A vlc media player plugin enabling dynamic adaptive streaming over http. In: Proceedings of the 19th ACM international conference on Multimedia. ACM, pp 723–726
27. OpenSim (2008) Inet framework for omnet++. INET Home Page, online https://inet.omnetpp.org/
28. OpenSim (2015) OMNeT++ Home Page, online https://www.omnetpp.org, accessed March 20
29. OVERSIM-Authors (2016) Dash bittorrent implementation in oversim. Online, https://github.com/hyperonex/OverSim-20101103
30. Pantos R (2015) HTTP live streaming. http://tools.ietf.org/html/draft-pantos-http-live-streaming-12, accessed January 6
31. Ramzan N, Park H, Izquierdo E (2012) Video streaming over p2p networks: challenges and opportunities. Signal Process Image Commun 27(5):401–411
32. Rejaie R, Ortega A (2003) Pals: peer-to-peer adaptive layered streaming. In: Proceedings of the 13th international workshop on network and operating systems support for digital audio and video. ACM, pp 153–161
33. Roverso R, El-Ansary S, Haridi S (2012) Peer2view: a peer-to-peer http-live streaming platform. In: IEEE 12th international conference on peer-to-peer computing (P2P), 2012, pp 65–66. https://doi.org/10.1109/P2P.2012.6335813

34. Roverso R, El-Ansary S, Haridi S (2012) Smoothcache: Http-live streaming goes peer-to-peer. In: Bestak R, Kencl L, Li L, Widmer J, Yin H (eds) NETWORKING 2012, lecture notes in computer science, vol 7290. Springer, Berlin, pp 29–43. https://doi.org/10.1007/9783642300547_3
35. S4 (2015) 3GPP specification: 26.234. Online http://www.3gpp.org/DynaReport/26234.html, accessed January 6
36. Stockhammer T (2011) Dynamic adaptive streaming over http–: standards and design principles. In: Proceedings of the second annual ACM conference on multimedia systems. ACM, pp 133–144
37. Vlavianos A, Iliofotou M, Faloutsos M (2006) Bitos: enhancing bittorrent for supporting streaming applications. In: INFOCOM 2006. 25th IEEE international conference on computer communications, Proceedings. IEEE, pp 1–6

Achraf Gazdar is currently an Assistant Professor at College of Computer and Information Systems at King Saud University. He received his PhD (2007) and MSc (2002) diplomas both in computer sciences from the National School of Computer Sciences (École Nationale des Sciences de l'Informatique), University of Manouba in Tunisia and His Bachelor degree in computer sciences (2000) from the Higher School of Management (Institut Supérieur de Gestion de Tunis), University of Tunis in Tunisia. His research focuses on Video on Demand (VoD) systems design and architectures, video streaming systems, multimedia P2P networks and protocols.

Lamia Alkwai is a researcher at King Abdulaziz City for Science and Technology in the National Center of Computer Technology and Applied Mathematics since 2010. She earned her MSc. degree in software engineering from King Saud University where she continues to seek her Phd in computer science. Her research interests falls into computer networks, internet of things, and network security.