DASH-based Peer-to-Peer Video Streaming in Cellular Networks

Ala'a Al-Habashna[1], Stenio Fernandes[2,1], Gabriel Wainer[1]
[1]Dept. of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada
[2]Center of Informatics, Federal University of Pernambuco, Recife, Brazil
alaaalhabashna@sce.carleton.ca, sflf@cin.ufpe.br, gwainer@sce.carleton.ca

Abstract — **Cellular networks now have increased demands for video streaming applications, which makes it challenging providing a high Quality of Experience (QoE). We propose improving this QoE by using Device-to-Device (D2D) communications in Base-Station (BS) assisted Peer-to-Peer (P2P) video streaming. Our architecture also employs Dynamic Adaptive Streaming over HTTP (DASH), an adaptive bit rate video streaming technique. We provide a detailed description of the proposed architecture. We used the Discrete EVent System Specification (DEVS) formalism to build a model for the proposed architecture in an LTE-A network, and use the model to study the performance achieved by the proposed architecture in terms of many video streaming QoE metrics. We also used the model to simulate a conventional DASH video streaming over a cellular network. Simulation results show that our architecture achieves significant improvement in video streaming QoE.**

Keywords—DASH; P2P streaming; D2D communication; DEVS

I. Introduction

The improvements in mobile devices have led to a significant growth in video traffic over cellular networks [1]. Now, cellular networks users tend to watch longer videos with higher resolution in advanced smartphones. In addition, we now have many live and on-demand video streaming platforms such as YouTube and Hulu. As per [1], 75% of the world's mobile data traffic will be video by 2020.

This increasing demand for video streaming presents a serious challenge for cellular networks service providers. Providing a high Quality of Experience (QoE) video streaming over cellular networks is challenging for many reasons. First, the scarcity of the radio spectrum in cellular networks makes it hard to provide the necessary high data rates for users to enjoy video streaming with high QoE. Furthermore, the variability of bandwidth of cellular links makes it harder; fluctuations in data rates could cause frequent video stalling, which degrades the QoE significantly. As such, we need new techniques to improve video streaming in cellular networks.

In this work, we propose and evaluate a new architecture that improves the QoE of video streaming in cellular networks. The proposed architecture employs the following techniques,

- The Cached and Segmented Video Download (CSVD), an algorithm we proposed in [2]-[3]. This algorithms focuses on Base-Station (BS) assisted Peer-To-Peer (P2P) video transmission in cellular networks. CSVD itself makes use of Device-to-Device (D2D) communication, as well as video caching in the User Equipment (UE). Here, CSVD is adapted in the context of video streaming as opposed to the original video transmission discussed in [2]-[3]. We evaluate the proposed architecture in terms of video streaming QoE metrics.
- Dynamic Adaptive Streaming over HTTP (DASH); an adaptive bitrate video streaming technique that provides advantages to video streaming such as efficient bandwidth utilization and improved streaming quality. This is used to allow support for DASH-based applications.

The proposed architecture is called DABAST: **DA**SH-based **BS**-**A**ssisted P2P video **ST**reaming in cellular networks. This is the first work that investigates P2P video streaming on cellular networks with DASH. This architecture brings many benefits to video streaming over cellular networks. We will provide further description of the DABAST and discuss each of the techniques above in more detail in Section 3. Furthermore, we use the Discrete EVent System Specification (DEVS) formalism to build a model for the proposed architecture in an LTE-A network. Simulations based on this model are used to evaluate the performance of DABAST in terms of video streaming QoE metrics. Simulation results show that the proposed architecture achieves significant improvements in terms of QoE when compared to conventional video streaming over a cellular network, i.e., DASH streaming without D2D/P2P streaming. Here we summarize our main contributions:

- A novel architecture, namely DABAST, which improves the QoE of video streaming in cellular networks.
- A DEVS model for DABAST in an LTE-A network
- An implementation of a DEVS model of DABAST using the CD++ toolkit to study its performance

II. Background and Related Work

In video streaming, a user can start playing a video segment before the whole video is downloaded. Due to the continuously increasing popularity of streaming applications and their high bandwidth requirements, video streaming has received much interest for over two decades [4]. Initial research on video streaming focused on developing new streaming protocols for client-server video streaming on wired networks. As the popularity and number of users of video streaming continued to increase, P2P video streaming became popular. A wide varied research has been conducted in the last decade on the design of new P2P streaming protocols over wired networks [5].

P2P communication in wireless networks has been studied in the context of wireless ad hoc networks [6], but it has not been considered in cellular networks until the recent emergence of D2D communications [7]-[10]. The introduction of D2D communications in the LTE-A standard has opened the door for P2P communications between UEs in cellular networks [11]. D2D communications provides a direct link between two UEs without going through the BS or the core network. Using D2D communications in cellular networks provides good means for improving performance. As such, it has been investigated in the last few years [7]-[10]. Some of the work focuses on finding incentive mechanisms to motivate involvement of UEs in D2D communication as the success of D2D communication depends on the participation of users to share their content [12]-[13].

To use P2P streaming, users were required to install dedicated applications. Furthermore, streaming traffic could be blocked by firewalls. These facts motivated the concept of streaming over the web. HTTP video streaming is the most popular form of video streaming nowadays due to the convenience of using HTTP [4], which eliminates the need to install and use a dedicated streaming application and helps to get the streaming traffic past firewalls. However, it was still challenging to stream video in wireless and mobile devices due to the high bandwidth variability of the wireless links. DASH came as bid to deal with this issue [15], as it allows changing the quality of video streaming to adapt to network conditions.

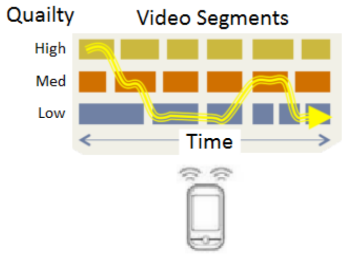

Fig. 1. DASH operation

As we can see in Fig. 1, DASH provides two features that helped improving video streaming. First, it breaks down the video into small, easy to download segments (for example 5-seconds chunks). Second, each segment is encoded at multiple bit rates, providing multiple quality levels for each segment, which allows adaptive streaming. Clients will choose between various bit rates to adapt to the network conditions. This helps improving the bandwidth utilization and reducing the interruptions of the video playback, which results in a higher QoE.

The adaptation strategy is one of the most important parts of DASH on the client side. It determines how the client selects the streaming quality to adapt to the varying network conditions. These strategies usually try to balance between two factors. They try to maximize the video quality by selecting the highest video rate the network can support, and at the same time minimize rebuffering.

A. Related work

There has been some work on P2P video streaming in cellular networks. In [16], a system, called MicroCast, was designed and evaluated using a testbed. MicroCast is used by a group of smart phone users who trust each other, are interested in watching the same video at the same time, and who are within proximity of each other. Users use their cellular connection to download segments of the video, and use their WiFi connections to share among them the downloaded content to improve the user experience. While this could result in some improvement for a group of users, the scope of the system is limited. Furthermore, users usually do not use their cellular connection for downloading video segments when WiFi is available.

A protocol for P2P video streaming on mobile phones, called RapidStream, was proposed in [17]. It is similar to many of the P2P streaming protocols on wired networks that involve the dissemination of buffer maps and video chucks between peers. While such protocols work well in wired networks, they involve too much signaling and transmission (dissemination of buffer maps) to be appropriate for UEs that has limited power, processing, and transmission resources (especially on a large scale). In [18], multi-source video streaming was proposed where mobile users can connect through WiFi direct to other users to get some of the video content. Such system requires the device to perform device discovery to find neighbors, and service discovery to find services offered by neighboring device. These requirements along with the signaling needed to exchange content consume a significant amount of resources.

In [19], the authors proposed a D2D communication system where multiple helpers collaborate to send a video segment to the requesting UE. The video, which is assumed to be in scalable video coding standard, is encoded by applying multiple description coding by each helper, and each helper sends a different description to the requesting UE. The authors analytically studied the problem of optimizing the number of transmitted descriptions to the requesting UE to maximize the video quality and efficiently consume the helpers' energy. However, the work only considers the energy consumed by the helpers to send the segments without considering the processing power and energy needed to encode the video segments. Encoding the video segments is a big favor to ask for, considering the limited energy and processing power of UEs. Furthermore, the optimization problem assumes that the BS knows the energy level of helpers and D2D channel characteristics between each pairs.

None of the research studies above on P2P video streaming in cellular networks considers how the video segments are actually cached. When evaluating the performance, they consider that requested segments are cached. Furthermore, they consider small-scale networks, i.e., up to 10 UEs including the helpers. We will show that using clustering and BS assistance, the potential of collaborative D2D communication between UEs is significant. In our previous research, we proposed the CSVD algorithm to improve video transmission [2]-[3]. CSVD is based on the architecture in [20], which employs video caching and D2D communication. In [2]-[3], CSVD is deployed in an LTE-A network with large number of UEs to evaluate how files are cached and exploited later for P2P video transmission. Simulation results showed that significant improvements could be achieved in terms of cell's aggregate rate and average data rate. Further details will be provided in the next section; as it is a main component in the DABAST.

Here, we extend CSVD by proposing and evaluating DABAST; a novel architecture that improves the QoE of video streaming in cellular networks. DABAST employs BS-assisted P2P video transmission between UEs (CSVD) and DASH. To the best of our knowledge, this is the first work that combines and investigates both P2P video streaming on cellular networks with DASH. Both DASH and P2P communication between

UEs over D2D are very beneficial for video streaming on cellular networks. Combining both techniques in one system is a necessary step that brings many benefits and achieves performance gains to video streaming on cellular networks. However, this system will need much study. Issues to investigate include finding the best DASH adaptation strategies suiting the architecture and strategies to select where video segments are sent from. Such architecture is worth investigating due to the improvements it brings to video streaming in cellular networks. We used the DEVS formalism [21] to build a model for the DABAST architecture, and used that model to test and evaluate the performance of the DABAST using simulations.

DEVS provides a formal framework for modeling generic dynamic systems. It has a hierarchical, modular, and component-oriented structure and formal specifications for defining structure and behavior of a discrete event model. A DEVS model is composed of structural (Coupled) and behavioral (Atomic) components, in which the coupled component maintains the hierarchical structure of the system, while each atomic component represents a behavior of a part of the system. The atomic component uses I/O ports and a finite state timed automaton representing the behavior of the model. A model is in state *s* for a specified time *ta(s)*, after which it produces an output *y* and changes its state based on the internal transition function. If it receives an input *x* before *ta(s)*, it invokes its external transition function, which can change the model's state.

The CD++ toolkit [22] was used to implement our model of DABAST. CD++ is an open-source simulation software written in C++ that implements the DEVS abstract simulation technique. The simulation engine tool of CD++ is built as a class hierarchy [22]. C++ is used to develop the atomic components of the model. These components can be incorporated into the class hierarchy. Passive classes can be also used to model components of the system. Coupled models can be created using a language built in the simulation engine. Modeling the DABAST using DEVS will be discussed in Section 4.

III. The DABAST Architecture

The DABAST architecture focuses on providing video streaming services to the users with better QoE. Fig. 2 shows the DABAST architecture.

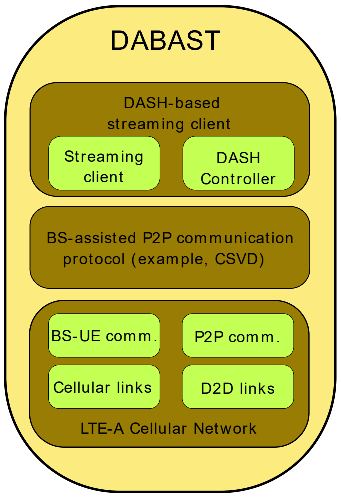

Fig. 2. The DABAST architecture

At the bottom, we have the LTE-A network that involves the communication between the BS and UEs over cellular links, and the communication between UEs over D2D links where the UEs exchange data directly without going through the BS. A BS-assisted P2P communication protocol is implemented on top of that (here, as an example, we use CSVD, which uses both cellular and D2D communication). DASH-based Video streaming takes place on top of these layers, as the transmission of video segments is implemented as per the communication protocol at the layer below. The DASH-based streaming has two parts: a streaming client and a DASH controller where the adaptation algorithm is implemented.

Following, we provide detailed description of the components in the top two layers. For further details on modeling the LTE-A network, the reader is referred to [3].

A. *The CSVD algorithm*

We used the CSVD algorithm [2]-[3] as the BS-assisted P2P communication protocol in DABAST. CSVD focuses on BS-assisted P2P transmission of video files using D2D communications.

In CSVD, the BS divides the coverage area into non-overlapping subareas, each of which is a cluster. The BS assigns UEs to clusters based on their locations, and it selects the UEs in the central area of each cluster as Storage Members (SMs) of that cluster. SMs are UEs that are used as helpers in the cluster. Only the UEs in the middle of each cluster are selected as SMs, in order to prevent inter-cluster interference when the SMs transmit to other UEs in the same cluster using D2D links. After clustering, when a UE requests a video file from the BS, it will process the request and respond as follows:

- Send With Assistance (SWA): if the file (or parts of it) is available in any of the SMs, the BS will ask the SMs to send the pieces to the requesting UE over D2D links.
- Send To a SM (STSM): if the requested file is not available in the distributed cache (or more copies need to be cached in the cluster) and the requesting UE is a SM, the BS will send the file to that UE over a cellular link, and it will ask the UE to cache the file. This case allows the SMs to cache video files. These files will be available for UEs in the cluster when requested later.
- Send To a UE (STUE): otherwise, the BS will send the file directly to the requesting UE over a cellular link.

As simulation results showed in [3], CSVD achieves significant improvements in terms of both Cell's aggregate data rate and average data rate.

B. *Streaming client*

CSVD is used to exploit both cellular links and D2D links for BS-assisted P2P video streaming between UEs in cellular networks. Here we provide an overview of the main video streaming concepts and measured metrics. Video streaming can be seen as a combination of download and concurrent playing, i.e., playout starts before the download is complete [23]. Playout usually starts after receiving a certain "sufficient" number of video segments. The received segments are buffered in a video/application buffer. The application that plays the video is usually referred to as the client. The client receives the pieces from the video buffer and the number of pieces available for playout is called the playout buffer length. Bad network conditions (insufficient bandwidth, delay, etc.) may cause the playout buffer to get empty as the video bit rate is higher than the video streaming rate, which causes video playout interruptions. These interruptions are referred to as video stalling or rebuffering. When stalling occurs, playout stops until sufficient data is buffered again.

QoE is used to measure the quality of video streaming; it is a measure the customer's streaming experience. There are many factors that are used to measure the quality of experience, here we present the most important ones [23].

- Video stalling (rebuffering): the stopping of video playback as the playout buffer gets empty. Increasing video stalling decreases the QoE. Many studies [23] have showed that video stalling has the biggest impact on QoE, and thus, should be avoided as much as possible.
- Video continuity index: a measure of the extent by which rebuffering pauses are avoided [24]. The continuity index is measured as follows,

$$\eta_c = 1 - \frac{\Delta T_{rb}}{\Delta T}, \qquad (1)$$

where ΔT_{rb} is the total time the client remains paused due to rebuffering events and ΔT is the duration of the experiment (playing time and rebuffering time).
- Initial (startup) delay: the delay from the request to stream the video until the playback starts. A certain number of video segments should be received before decoding and playback starts.
- Video bit rate: it is a measurement of the amount of data in one second of the video. Video bit rate is determined by many quality factors of the video such as video frame rate, resolution, and quantization parameters. As the video bit rate increases, the video quality increases, which increases the QoE.

C. *DASH controller*

As discussed in Section 2, DASH is an adaptive video streaming technology employed to help improving the bandwidth utilization and reducing the interruptions of the video playback, which results in a higher QoE.

Much work has been done on the adaptation strategies of DASH [25]. These can be classified into three categories; Adaptive Bit Rate selection (ABR) algorithms, buffer-based algorithms, and hybrid algorithms. In the ABR algorithms, the video client selects the video bit rate by monitoring network conditions and estimating the available network capacity. The problem with these algorithms is that in environments with highly variable bit rate, accurate estimation of future capacity could be challenging. Buffer-based algorithms were inspired by the fact that the occupancy of the playback buffer is the primary state variable we are trying to manage. Hence, video bit rate can be selected based only on the length of the playout buffer. Hybrid approaches try to employ both the estimation of network capacity and the length of playback buffer.

Here, we propose using the buffer-based approach proposed in [25]. In our architecture, the UE could receive a video segment from the BS or from any SM in the cluster. As such, it would be difficult to estimate the bit rate at which the next segment will be received. The use of hybrid approaches will be considered in future work. The adaptation algorithm we used is a piecewise function, *f(B)*, that uses the length of the playout buffer to determine the video bit rate as shown in Fig. 3.

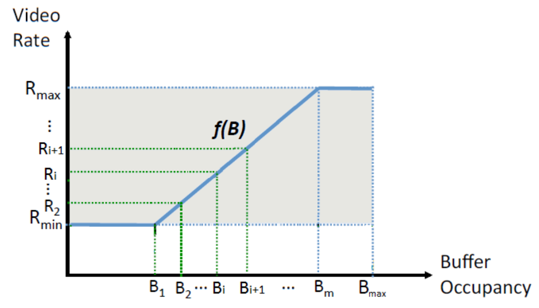

Fig. 3. Buffer-based adaptation algorithm

At the beginning when the buffer is empty, the video bit rate will be set to the minimum level, R_{min}. When the playout buffer length reaches a certain value (B_1 in Fig.3), the client will ask for next higher video bit rate (R_2 in Fig.3). The algorithm follows a simple rule, stay at rate R_{i+1} as long as the playout buffer length is between B_i and B_{i+1}. We refer to the component of the video client that runs the adaptation algorithm as the DASH controller.

D. *The DABAST algorithms*

When a streaming client starts a video stream, the UE will send a request to the BS to download video segments. The communication between the BS and the UEs in terms of signaling and transmission of video segments takes place according to a BS-assisted P2P communication protocol (here, we use the CSVD). The BS will send the pieces directly to the requesting UE over cellular links if the segments are not available in the distributed cache of the cluster. Otherwise, the BS will ask SMs to send the segments to the requesting UE. The BS keeps sending pieces with a certain video bit rate to a UE unless the UE requests changing the video bit rate. The BS sends a video segment from the distributed cache (when found) even if the segment found in the distributed cache does not match the video bit rate requested by the UE.

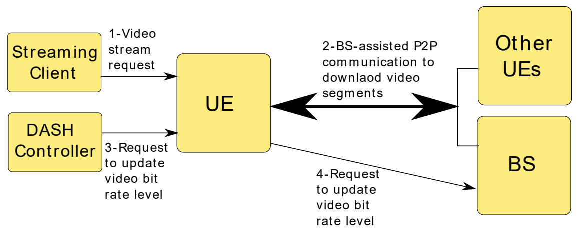
Streaming Client
1-Video stream request
DASH Controller
3-Request to update video bit rate level
UE
2-BS-assisted P2P communication to downlaod video segments
4-Request to update video bit rate level
Other UEs
BS

Fig. 4. DABAST operation

The DASH controller in each UE will adjust the video rate according to the length of the playout buffer. When the video bit rate needs to be changed, the UE sends a request to the BS to change it. This reduces the signaling and latency between the BS and the UE when compared to current implementations where the UE sends a request for each video segment.

IV. Modeling the DABAST with DEVS

As discussed earlier, we used DEVS to build a model of DABAST. Fig. 5 shows the coupled DEVS model of the top level architecture. We can see we have defined a *Cell* coupled model that contains a *BS*, a *Transmission Medium*, and many *UE* coupled models. The Cell coupled model also contains a *Cell Manager* atomic model.

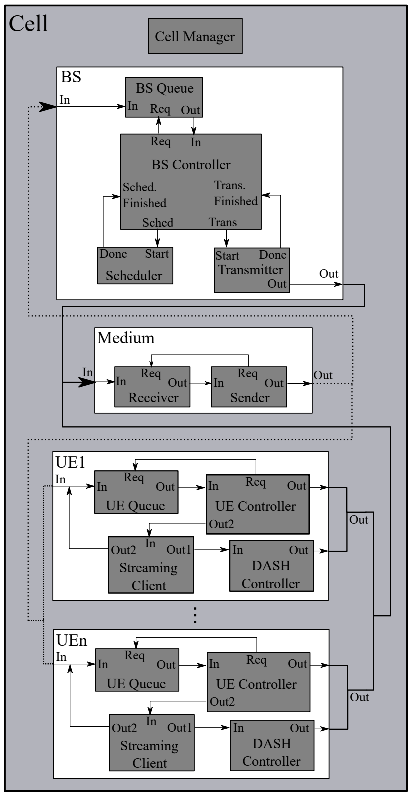

Fig. 5. Coupled DEVS model of DABAST

The *BS* coupled includes four atomic models: *BS Queue*, *BS Controller*, *Scheduler*, and *Transmitter*. The messages received messages are buffered at the *BS Queue*. The *BS Controller* is where the BS part of the CSVD algorithm is implemented [3]. The *Scheduler* schedules the messages to be transmitted in the next Transmission Time Interval (TTI), which is 1 ms. Every TTI, the BS Controller also asks the *Transmitter* to send messages that were scheduled for transmission during this TTI.

The *UE* coupled models contain four atomic models: *UE Queue*, *UE Controller, Streaming Client, and DASH controller*. Messages received are buffered at the *UE Queue*. The *UE Controller* is where the UE part of the CSVD algorithm is implemented.

The DASH-based streaming client is implemented in the *Streaming Client* and *DASH controller* atomic models. The streaming client manages the video buffer. It adds video segments received to the video buffer and removes video segments that were played from the buffer. The video is implemented as a sliding window. Video segments that were already played will be removed from the video buffer and the buffer slides to cover the next segments in the stream. The *DASH controller* implements the adaptation algorithm. It monitors the video playout buffer, and updates the video bit rate accordingly. When the video bit rate is to be updated, a request is sent to the BS with the new video bit rate.

The *Medium* model simulates the transmission medium and the *Cell Manager* atomic model initializes and sets the parameters of the cellular DLs and uplinks (ULs) between the BS and the UEs, as well as the D2D links between the UEs. For further details on the communication models used for simulation of the LTE-A cellular links and D2D links, the reader is referred to [3]. In addition to the atomic models above, many other passive classes where developed to model other components of the system such as classes to model the cellular and D2D links, download sessions the BS has with UEs, etc.

We used the CD++ toolkit to implement our model. A sample code snippet is shown in Fig. 6, which includes parts of the implementation of the *BS controller* atomic model (class BS).

```
BS::BS(const string &name) : Atomic(name),
  in(addInputPort("in")), req(addOutputPort("req")),
  sched(addInputPort( "sched" )),
  schedFinished(addOutputPort( "schedFinished" )),
  transmit(addInputPort( "trans" )),
  transFinished(addOutputPort( "transFinished" )){
  ...    }

Model &BS::externalFunction(ExternalMessage &msg){
  receivedMsg = msg.valueO();
  int msgType = receivedMsg->getMsgType();
  const Time msgTime = msg.time();
  if (state == CHECK_QUEUE){
    switch(msgType){
     case MSG_DOWNLOAD:
      dlMsg = (DownloadMsg*)receivedMsg;
      state = RCV_MSG;
      keepHoldInTime = receiveDownloadReq
                          (dlMsg, msgTime);
      this->getSessionPtr(receivedMsg->getSrcID())
      ->state = RCV_DOWNLINK_REQ; break;
     ...         }
}

Model &BS::internalFunction(InternalMessage &msg ){
  switch(state){
    case INITIAL:
      state = CHECK_QUEUE;
      this->bs->holdInActive(Time::Zero); break;
    case CHECK_QUEUE:
      this->bs->passivateBS(); break;
    case RCV_MSG:
      state = CHECK_QUEUE;
      bs->holdInActive(keepHoldInTime); break;
      ...
    }
}

Model &BS::outputFunction(InternalMessage &msg){
  switch(state){
    CHECK_QUEUE:  //request the next message
      bs->sendReq(msg.time(), 1, NULL); break;
    RCV_MSG:
    if(receivedMsg != NULL) //delete msg from queue
      bs->sendReq(msg.time(), 2, NULL);
  break;
  ...
  }
}
```

Fig. 6. Code snippet from BS *Controller* atomic model in CD++.

The model has five states; *Inital, Check_Queue, Rcv_Msg, Schedule, and Send.* At the beginning of each iteration, the model is in the *Initial* state as can be seen in the internal function. Then it goes to the "*Check Queue*" state, during which, the BS sends requests (as shown in the output function) to the queue asking for the messages from UEs. The queue will send the next message in line. When the model receives a message (See the external function), it goes to *Rcv_Msg* state. During this state, the BS processes the message and then sends another request to the Queue. The external function shows, as an example, the receipt of *MSG_DOWNLOAD*, which is the message sent by a UE to download video segments (See [3]). When no more messages are available, the queue sends *EMPTY_QUEUE* message to the *BS controller*. When the *BS controller* receives *EMPTY_QUEUE* message, it goes to the *Schedule* State. In this state, the BS sends a message to the *Scheduler* to schedule the messages to be sent during the next TTL. When the scheduler finishes scheduling, it sends a *SCHED_FINISHED* message to the *BS Controller* to indicate that scheduling is finished. When The *BS Controller* receives this message, it goes to the *Send* state. In this state, the model sends a message to the *Transmitter* model to start transmission. When the Transmission is done, the *Transmitter* model sends a *TRANS_FINISHED* message to the BS controller to indicate that transmission is finished. Upon receipt of this message, the BS goes back to the *Check_Queue* state.

V. SIMULATING DABAST

We executed system-level simulations to evaluate the performance of DABAST in terms of the QoE metrics presented in Section III. Table 1 shows the simulation setup.

TABLE I. SIMULATION SETUP

Parameter	Value
Cellular Channel BW (MHz)	10
Cell Range (m)	500
Number of clusters	4
BS antenna gain (dB)	12
BS transmission power (dBm)	43
UE antenna gain (dB)	0
UE transmission power (dBm)	21
Noise spectral density (dBm)	-174
Antenna height (m)	15
Transmission model	UTRA-FDD
Carrier frequency	900MHz
File requests	20
Area configuration	Urban
Number of files	500
D2D Channel BW (MHz)	50
D2D Carrier frequency	24 GHz
D2D transmitter TX Power (dBm)	23
D2D Large-scale fading std deviation (dB)	4.3
D2D Receiver noise figure (dB)	9
D2D TX/RX Height from Ground (m)	1.5
Segment length (second)	10
Number of buffered segments to start playout	4
Video bit rate levels (kbps)	384, 768, 2000, 4000
Videos length (second)	441

We compared the performance of DABAST with a conventional DASH system, i.e., without P2P/D2D communications. The simulations consider a single LTE-A cell with 500 UEs. The urban macro propagation model [26] was used for cellular links with a DL operating carrier frequency of 900 MHz, and a transmission bandwidth of 10 MHz. A millimeter-wave channel model at 24 GHz is used for D2D transmission [27].

In each iteration of the simulation, the UEs are randomly distributed throughout the cell using a uniformly distributed distance from the BS. Clustering takes place in the beginning in case of DABAST where the cell is divided into 4 clusters. The UEs then start requesting video streams. During each iteration of the simulation, each UE will request two video streams. A UE requests a video stream, and after finishing the playout, it will request a second video. Before generating each request, a UE waits for a random period according to a Poisson distribution with mean of 10 seconds. The popularity of videos is generated according to a Zipf distribution to simulate the variable popularity of the videos, as it has been established this is a good model for this purpose [28]. Using this distribution, some videos are requested more often than others. The length of the videos is 441 seconds, which is the mean length of a YouTube video [29]. We used a video buffer of 240 seconds, as in [25]. Four video bit rate levels where used as shown in Table 2. These are adapted from the H.264/AVC video coding standard [30]. The Zipf exponent is 1.5, and the number of video requests made by each UE in a simulation run is 2.

We measured the number of rebufferings, video continuity index, initial delay, and video bit rate levels of the received video segments. Table 2 shows the mean values for these measurements, along with the Margin of Error (MoE) for a 95% confidence interval. The first three measurements are based on 30,000 values (30 runs x 500 UEs x 2 requests). For the video bit rate level, the measurements are based on 1,350,000 values (45 video segments received per request).

TABLE II. SIMULATION RESULTS

	Conventional DASH		**DABAST**	
	Mean	MoE	Mean	MoE
Rebufferings	3.4201	0.0083	1.6410	0.0189
Cont. index	0.7323	0.0007	0.8693	0.0015
Initial delay (sec)	56.415	0.2507	24.647	0.2217
Video bitrate (kbps)	396.34	0.1935	442.25	0.4333

As we can see in Table 2, there is a clear improvement achieved by DABAST over conventional DASH in terms of all the metrics above. The average number of rebufferings with DABAST is less than half of that for conventional DASH. As such, there is a significant increase in the continuity index with DABAST. The average initial delay with DABAST is also less than half of that for conventional DASH, which is a significant improvement. The average initial delay for conventional DASH is relatively high because in this scenario, there are 500 UEs in the cell requesting video streams, and sharing fixed cellular frequency resources (10 MHz). The average video bit rate achieved with DABAST is also higher than that of conventional DASH. These significant improvements are achieved by DABAST because video segments are delivered to UEs much faster than in the case of conventional DASH. Video segments are delivered faster in the case of DABAST because the CSVD algorithm is employed, where video segments are sent to many UEs from both the BS (over cellular links) and SMs (over D2D links) as opposed to only from the BS. This reduces the initial delay to receive the first 4 segments needed to start playing, and consequently, it reduces the initial delay. This also reduces the events of video buffer stalling, and consequently reduces the number of rebufferings. There is only a small improvement achieved by DABAST in terms of average video bitrate. This is due to two reasons. First, with both conventional DASH and DABAST, the DASH controller will resort to choosing a lower video bitrate level to increase the video playout buffer length and reduce the number of rebufferings. As such, the improvement in the number of rebufferings is usually achieved on the expense of video bit rate. Second, as mentioned in the Section III, in this implementation, the BS will send a video segment from the distributed cache (when found) even if the segment found in the distributed cache does not match the video bit rate requested by the UE. This is to increase the utilization of the available D2D channel, and to speed up the transmission of video segments as sending from the distributed cache is faster. As such, sometimes the BS sends video segments with lower video bit rate than the requested.

Fig. 6 shows an histogram of the number of rebufferings for both the conventional DASH and DABAST. As we can see, over 96% of the streaming requests have 3 or 4 rebufferings in the case of the conventional DASH. With DABAST, on the other hand, about half of the streaming requests have 0 rebufferings, and slightly less than half of the requests have 3 or 4 rebufferings. These streams with 0 rebufferings are streams with video segments satisfied from both the BS and distributed cache. This demonstrates the importance of using the CSVD in DABAST, which improves the transmission of video segments to the UEs.

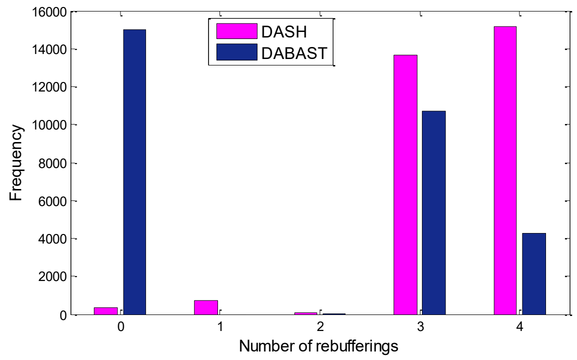

Fig. 7. Histogram of the number of rebufferings for conventional DASH and DABAST.

Fig. 7 shows a histogram of the continuity index for both conventional DASH and DABAST. The continuity index results match these in Fig. 6 for the number of rebufferings.

Half of the requests with DABAST have a continuity index of 1, which corresponds to zero rebufferings. Less than half of the video streams have a continuity index less than 0.77 with DABAST (corresponding to 3 and 4 rebufferings). However, in the case of conventional DASH, over 96% of the video streams have a continuity index less than 0.77.

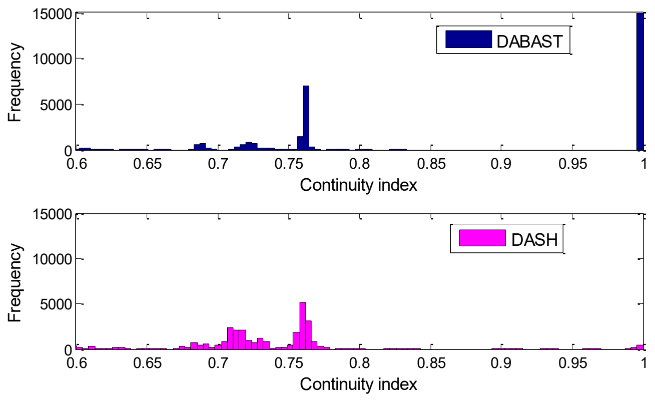

Fig. 8. Histogram of the continuity-index for conventional DASH and DABAST.

Fig. 8 shows the Empirical Cumulative Distribution Function (ECDF) for the initial delay of both conventional DASH and DABAST.

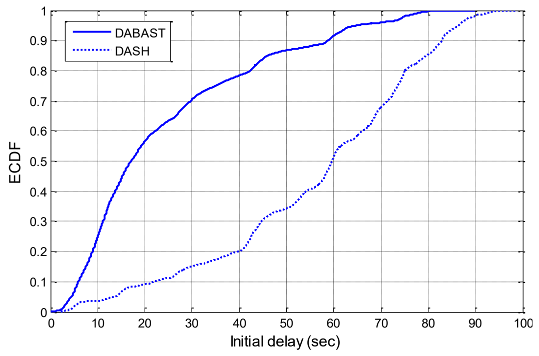

Fig. 9. ECDF of the initial delay for conventional DASH and DABAST.

We can see that the ECDF of DABAST is always higher than that of conventional DASH. For example, the probability of having a stream with initial delay of 20 seconds or less is 0.56 with DABAST, and only 0.09 with conventional DASH. Fig. 8 also shows that 50% of the streams have initial delay of 17.08 seconds or less with DABAST while 50% of the streams have 59.60 seconds or less with conventional DASH. As previously mentioned, with conventional DASH, all the UEs in the cell share a fixed frequency resources (10 MHz cellular channel), while with DABAST, D2D communication exploits the millimeter wave channel for P2P communication in addition to cellular resources. The transmission of video segments from the distributed cache of the cluster speeds up the delivery of video segments and significantly reduces the initial delay.

TABLE III. COUNT OF THE VIDEO BITRATE LEVELS

	Count	
Video bitrate (Kbps)	Conventional DASH	DABAST
384	1325268	1224024
768	18918	105072
2000	5814	18648
4000	0	2256

Table 3 shows the count for the received video segments with each video bitrate level, for both conventional DASH and DABAST. The results show that with DABAST, fewer video segments with 384 kbps were received and more video segments with higher levels (768, 2000, and 4000 kbps) were received. This explains why the average video bitrate (Table 2) for DABAST is higher than that for conventional DASH.

With DABAST, the video segments are delivered faster to the requesting UEs as explained above. As such, clients will have more video segments in the playout buffer, i.e., higher playout buffer length. Consequently, the requested video bit rate will be higher in the case of DABAST.

The results presented in this section show that DABAST provide improvements over conventional DASH in terms of all measured metrics, which significantly improves the QoE of video streaming in cellular networks.

VI. Conclusion

The increasing demands for video streaming poses a big challenge for cellular networks service providers. Providing a high Quality of Experience (QoE) video streaming over cellular networks is difficult due to the limited frequency resources and variable network conditions. As such, new techniques that help improving the QoE of video streaming in cellular networks are needed. In our previous work, we proposed an algorithm for improving video transmission in cellular networks, namely the Cached and Segmented Video Download (CSVD). CSVD employs Device-to-Device (D2D) communication for Base-Station (BS) assisted Peer-to-Peer (P2P) video transmission in cellular network. Here, we extend our work by developing a novel architecture for Dynamic Adaptive Streaming over HTTP (DASH) Based BS-Assisted video STreaming in cellular networks (DABAST).

DABAST is used to improve the QoE of video streaming in cellular networks. In the first implementation of DABAST, we use the CSVD in the context of BS-assisted P2P video streaming. We use the Discrete EVent System Specification (DEVS) formalism to build a model for the proposed architecture in an LTE-A network, and use the model to study the performance achieved by the proposed architecture in terms of many video streaming QoE metrics. We also use the model to simulate a conventional DASH-based video streaming over a cellular network, i.e., without D2D/P2P. Simulation results show that DABAST achieves significant improvements in terms of many video streaming QoE metrics. In future work, we will further investigate the DABAST architecture at a finer level by studying various implementations of the different components. This will include implementing and evaluating different P2P communication algorithms between the UEs as well as different DASH controllers.

References

[1] Cisco. "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update." Internet: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. Feb. 2016 [Feb. 28 2016].

[2] A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman. "Improving Wireless Video Transmission in Cellular Networks using D2D Communication." Canada. Provisional patent P47111. May 2015.

[3] A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman, "Cached and Segmented Video Download for Wireless Video Transmission," *in proc. ANSS*, 2016.

[4] B. Li et al., "Two decades of internet video streaming: a retrospective view, " *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, no. 1, pp. 1-20, 2013.

[5] I. Ullah et al., "A Survey and Synthesis of User Behavior Measurements in P2P Streaming Systems," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 3, pp. 734-749, 2011.

[6] J Zhao, et al., "Cooperative Caching in Wireless P2P Networks: Design, Implementation, and Evaluation," *IEEE Trans. on parallel and distributed systems*, vol. 21, no. 2, pp.229-241, 2010.

[7] A. Asadi, Q. Wang, and V. Mancuso, "A Survey On Device-To-Device Communication In Cellular Networks," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 4, pp. 1801 - 1819, 2014.

[8] B. Kaufman and B. Aazhang, "Cellular networks with an overlaid device to device network," *in Proc. of Asilomar Conference on Signals, Systems and Computers*, 2008, pp. 1537–1541.

[9] K. Doppler et al., "Device-to-device communication as an underlay to LTE-advanced networks," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 42–49, 2009.

[10] K. Doppler et al., "Device-to-device communications: functional prospects for LTE-Advanced networks," *in Proc. of IEEE ICC Workshops*, 2009, pp. 1–6.

[11] S. Parkvall and D. Astely, "The Evolution of LTE Towards IMT-Advanced," *Journal of Communications*, vol. 4, No. 3, pp. 146-154, 2009.

[12] Y. Zhang, L. Song, W. Saad, Z. Dawy, and Z. Han, "Contract-Based Incentive Mechanisms for Device-to-Device Communications in Cellular Networks," *IEEE Journal on Selected Areas in Communications*, pp. 1-12, 2015.

[13] L. Gao, J. Huang, Y. Chen, and B. Shou, "Contract-based cooperative spectrum sharing," *IEEE Transactions on Mobile Computing*, vol. 13, no. 1, pp. 174–187, 2014.

[14] L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand, "Motivating smartphone collaboration in data acquisition and distributed computing," *IEEE Transactions on Mobile Computing*, vol. 13, no. 10, pp. 2320–2333, 2014.

[15] T. Stockhammer "Dynamic adaptive streaming over HTTP: standards and design principles," *in Proc. ACM MMSys'11*, 2011, pp. 133-144.

[16] L. Keller et al., "MicroCast: cooperative video streaming on smartphones," *in proc. MobiSys*, 2012, pp. 57-70.

[17] P. Eittenberger, M. Herbst, U. Krieger, "RapidStream: P2P Streaming on Android," *in proc. IEEE International Packet Video Workshop*, 2012, pp. 125-130.

[18] V. Siris and D. Dimopoulos, "Multi-Source Mobile Video Streaming with Proactive Caching and D2D Communication ," *in IEEE WoWMoM*, 2015, pp. 1-6.

[19] T. Duong et al. "Energy-aware rate and description allocation optimized video streaming for mobile D2D communications," *in Proc. IEEE ICC*, 2015, pp. 6791 - 6796.

[20] N. Golrezaei, P. Mansourifard, A. F. Molisch, and A. G. Dimakis, "Base-Station Assisted Device-To-Device Communications For High-Throughput Wireless Video Networks," *IEEE Transactions on Wireless Communications,* vol. 13, no. 7, pp. 3665-3676, 2014.

[21] B. Zeigler, H. Praehofer, and T. Kim, Theory of modeling and simulation. San Diego: Academic Press, 2000.

[22] G. Wainer, Discrete-event modeling and simulation: a practitioner's approach. Boca Raton: CRC/Taylor & Francis Group, 2009.

[23] M. Seufert et al., "A servey on quality of experience of HTTP adaptive streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469-492, 2015.

[24] L. Cicco and S. Mascolo, "An Adaptive Video Streaming Control System: Modeling, Validation, and Performance Evaluation," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 526-539,2014.

[25] T. Huang et al., "A buffer-based approach to rate adaptation: evidence from a large video streaming service," *in Proc. ACM SIGCOM'14*, 2014, pp. 187-198.

[26] 3rd Generation Partnership Project, "Technical Report 36.942, V12.0.0," 2014.

[27] A. Al-Hourani, S. Chandrasekharan, and S. Kandeepan, "Path loss study for millimeter wave device-to-device communications in urban environment," *in Proc. ICC*, 2014, pp. 102-107.

[28] M. Cha et al., "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proc. ACM SIGCOMM,* 2007, pp. 1–14.

[29] S. Ahsan et al., Characterizing Internet Video for Large-scale Active Measurements, Submitted to the *Networking and Internet Architecture*, arXiv preprint arXiv:1408.5777v1, 2014.

[30] "Infrastructure of audiovisual services – Coding of moving video," ITU-T, Recommendation H.264, 2012.