Cached and Segmented Video Download for Wireless Video Transmission

Ala'a Al-Habashna Gabriel Wainer
Dept. of Systems and Computer Engineering
Carleton University
Ottawa, ON, Canada
{alaaalhabashna; gwainer}@sce.carleton.ca

Gary Boudreau Ronald Casselman
Ericsson Canada
3500 Carling Avenue, K2H 8E9
Ottawa, ON, Canada
{gary.boudreau; ronald.casselman}@ericsson.com

Abstract—**The continuously increasing demand for higher data rates is a challenge for cellular networks service providers. In recent years, wireless video has been one of the main drivers of wireless data traffic, and this kind of traffic is going to continue growing. The LTE-Advanced standard (LTE-A) introduced new methods to cope with the increasing users' demands, but performance is still limited. Combining existing algorithms with Device-to-Device (D2D) communications is a promising solution to deal with these issues. In D2D, transmission between users' devices takes place over direct links, without going through the Base-Stations (BS). Here, we introduce an algorithm for improving the throughput of video transmission in cellular networks called the Cached and Segmented Video Download (CSVD). The algorithm splits video files into pieces that are cached in selected user equipment (UEs) in the cellular network, and employs D2D communication between UEs to exchange files' pieces. We have built different models and simulated the CSVD algorithm using the Discrete Event System Specification (DEVS) formalism. The models are used to study the performance improvement achieved by the CSVD in terms of cell's aggregate data rate as well as the average data rate per user. Simulation results show that the CSVD achieves significant improvement over the conventional transmission approaches.**

I. Introduction

In recent years, the advance in cellular networks and mobile devices has led to major improvements in the services provided to cellular networks users. This has caused the rate of adoption of mobile devices to grow exponentially [1]. Nowadays, many web users use smart phones as their primary way to access the web. Due to the nature of these newly provided services, and to the increased number of users, the demand for higher data rates has increased exponentially. Providing such high data rates for users has become one of the main challenges for cellular service providers.

Wireless video has been one of the main drivers of wireless (cellular) data traffic, and its importance is going to grow. As the storage and the screen size of smart devices such as tablets and smart phones continues to increase, users will watch longer videos with higher resolution. This will significantly increase the amount of data to be transmitted. Recent estimates show that approximately three-fourths of the world's mobile data traffic will be video by 2019. Mobile video will increase 13-fold between 2014 and 2019, accounting for 72 percent of total mobile data traffic [2].

The scarcity of the radio spectrum is a major reason for the inability to provide higher data rates. As most of the licensed frequency bands are allocated, it is difficult to provide sufficient resources to the users and their increased bandwidth. As such, there is a need for new techniques to increase the data rates by utilizing the radio spectrum more efficiently or by improving the transmission link capacity.

The LTE-Advanced (LTE-A) standard was introduced by the 3rd Generation Partnership Project (3GPP) to cope with these increasing user demands and to meet the requirements of International Mobile Telecommunications-Advanced [3]. Although LTE-A has higher performance than 2G and 3G networks, its performance is still limited, and the increase in video transmission rates will make it worse.

Here, we propose a method for improving the throughput of video transmission, as this could help with the increasing wireless data traffic. One way to do this was proposed in [4], in which the authors proposed a new communication architecture that exploits Device-to-Device (D2D) communications for improving the throughput of video transmission.

D2D communication provides direct communication between two User Equipment (UEs) without going through the Base-Stations (BS) or the core network. Users nowadays use high data rate services, which could significantly benefit from direct communication between users. In recent years, we have seen various efforts combining current standards with D2D communications [5-8] in order to improve performance.

In particular, architecture proposed in [4] exploits the fact that video files have a high degree of content reuse: usually, a few popular files are requested by a large number of users (such as viral YouTube videos, reports from recent sport events, etc.). The basic idea is to cache popular video contents in the UE devices. If cached video files are requested, they will be sent to the requesting devices over D2D links.

The original architecture is limited: it assumes that the files are pre-cached at random in the nodes, and complete files are cached and exchanged between the network nodes. Furthermore, they did not define a messaging protocol between the UEs and BS to exchange such files. Instead, a simple model was used to study the the performance of the architecture analytically.

Here, we propose a variation of the original architecture, called the Cached and Segmented Video Download (CSVD) algorithm [9]. Our objective is improving the throughput of video transmission in cellular networks. CSVD is based on the architecture described above, but instead of caching complete files, the files are split into pieces and cached in multiple nodes. We assume that no files are cached in the beginning, and that files are stored upon request. The algorithm defines how the files are cached and exchanged among the UEs. The algorithm also considers the messaging/signaling protocol between the BS and the UEs. Only the selected UEs in each cluster are used for caching to reduce inter-cluster interference. A complete detailed protocol has been defined, including a variety of messages necessary for this communication, and a complete definition of the protocol is described.

We built a suite of models using the Discrete Event System Specification (DEVS) formalism [20] and the CD++ DEVS toolkit to model an LTE-A cellular network. The hierarchical and modular nature of DEVS was useful for modeling and simulating this network, which was built using different submodels, in which each one implements a different component of the wireless network. Each of the submodels could be tested and verified separately, and later integrated into the whole model, and reused. This made it easy to design, implement, and test. System level simulations were performed to evaluate the performance of the CSVD algorithm with different parameters. Simulation results show that CSVD achieves a significant improvement over the conventional transmission approach where D2D communication is not employed. Furthermore, simulation results show the performance of CSVD under different parameters (number of users, files' popularity, etc.).

The rest of this paper is organized as follows, In Section 2, we provide an overview of related work. In Section 3, we present the CSVD algorithm. In Section 4, we describe the modeling of the LTE-A cellular network in DEVS. The simulation scenarios and results are presented in Section 5.

II. Background

Recently, various research efforts focused on improving the data rates of cellular networks, and on how to meet the increasing performance demands. A great deal of this research has been conducted at the physical layer, focusing on the transmission link capacity [11]. However, such improvement is limited, as the rates achieved by modern networks are close to the theoretical limits. Another approach has been to decrease the size of the cells in order to provide higher spectral efficiency [12]. This is a costly approach, which requires increasing the number of cells in the network.

As we discussed in the Introduction, improving the transmission of video could improve the overall performance of the network. In [13], the authors proposed caching popular video files at the BSs or at the mobile switches, reducing the traffic on the backhaul network and eliminating the need to fetch the file from the Internet. Nevertheless, this caching does not improve the transmission between UEs and BS over radio links.

In [14], the authors proposed creating simultaneous coded-multicasting opportunities. The contents are placed in order to allow the central server to satisfy the requests of several users with different demands with a single multicast stream. The multicast streams are generated by coding across the different files requested by the users. Each user exploits the content stored in the local cache memory to enable decoding of its requested file from these data streams. In [15], another approach was proposed for caching video files. The idea is to cache contents expected to be requested by a user on their device. This can be applied by using a virtual home-box layer for distributing efficient and scalable future internet video on demand streaming services. In these two approaches, cached contents can be only used locally by the caching devices, and cannot be exploited by others in the network.

In recent years there has been a great deal of research on using D2D communication in cellular networks, as it provides a good means for improving performance [5-8], and different authors started to use this method combined with caching. The *mobile content delivery network* is a proposed technology in which devices that are designated as caching servers are used to provide nearby users with cached contents on demand`, and content delivery could take place over D2D links [16]. While this technology could help improve the data rates in cellular networks, it is costly as these designated devices need to be placed throughout the network, configured, and maintained. Instead, the architecture, proposed in [4] can improve the throughput of video transmission and overcome the problem of rapidly increasing wireless video traffic. In current cellular networks, when a UE wants to download a video file, it sends the request to its BS. The BS will get the file and send it to the requesting UE through a cellular downlink (DL). In the approach presented in [4], the cell is divided into "clusters" of nodes. Each cluster contains a group of nodes that can exchange information with each other by using D2D links. The nodes in each cluster can save video files. When a video file is requested by a UE, the BS will check to see if the file is stored in the virtual storage of that cluster. If the requested file is found, it will be transmitted from the UE that has the file to the requesting UE over a D2D link. The network model in [4] is oversimplified. It also is assumed that video files are already saved in the UEs and random storage of the video files is considered. How the files are cached is not considered. Furthermore, it is assumed that UEs should save and forward complete files over D2D links.

In [17], the authors provide analytical results on the throughput scaling laws of wireless networks with caching and asynchronous content reuse, and they compare the method above with coded multicasting. Analytical results showed that the proposed approach achieves the same throughput scaling law of the infrastructure-based coded multicasting scheme. In [18] the authors studied coded multicasting analytically in order to allow nodes in an "infrastructure-less" network to exchange pre-cached files. This means that no BS is considered in the network. Files are assumed to be pre-cached. They do not discuss how the files are placed in the nodes, and the signaling/messaging between the UEs and the BS is not studied.

D2D communication highly depends on the participation of users in sharing contents. As such, it is important to find different approaches to motivate such user involvement. There has been much research on incentive mechanisms to motivate user involvement in D2D communication [19-21].

As mentioned in the introduction, we used the architecture proposed in [4] to define the CSVD algorithm for improving the throughput of video transmission in cellular networks. We studied the performance of the algorithm by building a complex model and running varied simulations of the new protocol. We used the DEVS formalism [22] to build our models and to test and evaluate the CSVD algorithm. DEVS has been proposed as a sound formal framework for modeling generic dynamic systems and includes 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 [22]. An input to the atomic component via an input port triggers a state transition (referred to as "external transition"), and in contrast the state transition (referred to as "internal transition") at the end of the time-delay of each state leads to an output generation through an output port.

We used the CD++ toolkit [23] to implement our LTE-A network DEVS model. 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 [23]. Using CD++, atomic models are developed using C++ programming language and can be incorporated into the class hierarchy. In addition to the atomic models, passive classes can be also used. Coupled models can be created using a language built in the simulation engine.

Modeling the LTE-A network using DEVS will be discussed in Section 4. In the following Section, we provide a detailed description of the CSVD algorithm and how it operates.

III. The CSVD Algorithm

CSVD focuses on scenarios where there is limited area with high density of users, for instance:

- Sport events in which users want to download instant replays from this event, or videos of other events taking place at the same time.
- Live concerts with detailed video feeds of the arena.
- Massive religious events (i.e., a Pope's Mass in St. Peter's Basilica in The Vatican).
- Large political events (i.e., election results or inauguration speeches).
- University convocations.

Let us consider one cell in a cellular network, in which the BS is in the middle of the coverage area, as seen in Fig. 1.

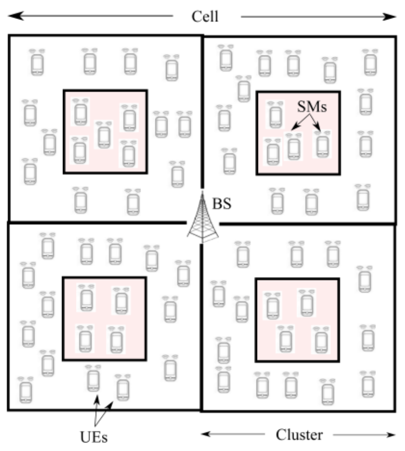

Fig. 1. Cell after clustering.

In this research, we only considered cases where UEs are stationary. Furthermore, we did not consider UE power failure. At the beginning, the BS starts by dividing the cell into clusters, as follows:

1) The BS logically divides the coverage area into non-overlapping subareas. Each one of these will be a cluster.

2) The BS sends a broadcast Clustering message telling the UEs that a cluster formation is about to start.

3) The UEs reply with a Clustering Response message indicating their location.

4) The BS assign 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, as in Fig. 1. We only choose UEs *in the middle* of each cluster as SMs, in order to prevent inter-cluster interference when the SMs transmit to other UEs in the same cluster using D2D links.

After completing the clustering phase, the transmission phase can begin. In this phase, the UEs send requests to download video files to the BS. When the BS receives a download request, it processes the request, and responds as follows.

- Send With Assistance (SWA): if the file (or parts of it) is available in any of the SMs, the BS will ask them 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.

In the following sections, we discuss the different cases discussed above in detail.

A. Send With Assistance case

Fig. 2 shows the steps of the download process of a file for the SWA case.

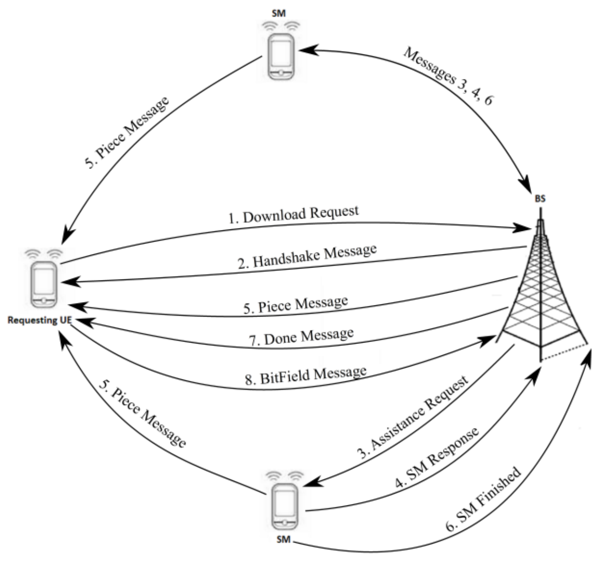

Fig. 2. CSVD algorithm.

As we can see, the BS, a requesting UE, and SMs are in the same cluster as the requesting UE. In this case, the download process will be as follows:

1) The UE sends a *Download Request* message to the BS.

2) As this file has already been sent before to an SM to cache it, the BS has a MetaInfo file that describes the parameters for the download session of this video file. The MetaInfo file contains the following fields.

TABLE I. METAINFO FILE

Field	Description
File Size	The file size in bytes
Number of Pieces	The number of pieces
Piece size	The piece size in bytes
Last piece size	Last piece size in bytes
File name	A string representation of the file
Info	A dictionary that describe the file

The fields in the MetaInfo file represent the parameters for this download session. This will be sent to the requesting UE to set the parameters of this download (Number of pieces, file name, etc.). The first field is the size of the file to be sent in bytes. As the piece size is fixed, the number of pieces can be found. The size of the last piece is variable, and it is indicated in the fourth field. The File name is a string representation of the file. This is generated by the BS, and serves as a unique identifier of the file for this algorithm. The BS then sends a handshake message to the requesting UE. The handshake message contains the MetaInfo file above.

3) The BS will check its database to find out which of the SMs have the pieces of the cached file. Then, the BS will send an *Assistance Request* message to these SMs asking for their assistance to send pieces to the requesting UE. The *Assistance Request* message has a field indicating the number and indexes of the pieces that the SM needs to send to the requesting UE.

4) The SMs will send a *Response* message. The SMs will indicate whether they are available to assist with this download session or not. The Response message also contains a field that indicates the maximum number of outstanding assists the BS should send, i.e., the maximum number of assists this SM can handle at a time. The SM finds this number based on the current processes/transmissions it is handling. The SM can send this message to BS during the assistance session if it wants to change the maximum number of concurrent Assists.

5) The BS and the SMs starts sending the pieces to the requesting UE. Each time the BS wants an SM to forward new piece(s) of the file, it will send that SM an *Assistance Request* message indicating the piece(s) to forward. Each piece message has an index that identifies the piece.

6) When an SM finishes sending pieces, it will send an *SM_Finished* message to the BS, acknowledging the transmission of the piece(s).

7) When the BS receives *SM_Finished* for the pieces from the SMs participating, and when it finishes sending its pieces, it will send a *Done* message to the requesting UE.

8) When the requesting UE receives a *Done* message, it will send a *BitField* message indicating the pieces it has received.

Using D2D links to transmit cached pieces of the files can save a considerable amount of cellular resource blocks (RBs) that can be used to serve other users, which will increase the cell throughput and the video transmission rate.

As part of the CSVD algorithm, the BS should keep track of the followings in its database:

- A list of the current clusters
- For each cluster, the BS should keep a list of the members and SMs of each cluster
- A list of cached files/pieces, and the caching SMs of each piece

B. Send To a SM case

In this case, the file transfer starts by the SM sending a Download Request message to download a file. After receiving the request, the BS will start a session with this UE. If this is the first time a SM request this file, the BS creates a MetaInfo file that contains information about this transfer. The MetaInfo file is the same as in table 1. The BS also creates a handshake message and sends it to the requesting SM. The BS then starts sending pieces of the file directly to the SM over cellular link. The Save bit in the Piece message is always set to indicate that the SM should cache the received piece. The SM keeps a BitField to keep track of the received pieces. After sending all the pieces, the BS will send a *Done* message. When the SM receives all the pieces and the *Done* message, it will send a message containing the BitField to the BS to indicate the end of the file transfer.

C. Send To a UE case

In this case, the requesting UE is not an SM, and the file is not available in the cluster. There are two differences between this case and STSM. First, the BS always splits the file into pieces and creates a new MetaInfo file as the file was not requested by a SM before. Second, the Save bit is always zero in the Piece message so that the piece will not be cached.

D. The SVD algorithm

We call our implementation of the conventional download process the Segmented Video Download (SVD), as video files are sent in pieces. In SVD, we do not use file caching or D2D communications. Instead, the files will be sent as in STUE.

IV. MODELING OF THE MOBILE NETWORK

As discussed earlier, we built a DEVS model of the proposed architecture and executed numerous simulation scenarios, some of which will be discussed in these sections.

Fig. 3 shows the structure of a coupled DEVS model used for a single cell network. This model was used to evaluate the performance of the CSVD and SVD. As can be seen, at the top level, we have the Cell coupled model, which contains the BS, Transmission Medium, and many UE coupled models. We can have an unlimited number of UEs. The Cell coupled model also contains the Cell Manager atomic model.

The BS coupled model is used to simulate the BS in the cell. It contains two atomic models, BS Queue and BS controller. The BS controller is where the BS part of the algorithm is implemented (for example, steps 2, 3, 5, and 7 in Fig. 2). The *BS Queue* atomic model is used to buffer messages sent to the *BS Controller*. The BS Queue atomic model buffers messages sent for the BS controller. It checks the destination address of each message to see if the message is intended for the corresponding controller. If it is, it will buffer this message to be sent to the controller. Otherwise, the message will be discarded. The UE coupled models are used to model the UEs in the cell. Each one of these models contains two atomic models: *UE Queue*, and *UE Controller*. The UE part of the algorithm (for example, steps 1, 4, 5, 6, and 8 in Fig. 2) is implemented in the

UE controller atomic model. As with the BS Queue, the UE Queue buffers messages that are sent to the corresponding controller.

Aside from the atomic model components above, many other passive classes have been added. These include classes to model the cellular DLs and uplinks (ULs), D2D links, download sessions the BS has with UEs, cell clusters, etc.

The *Cluster* passive class is used to model the division of the cell into clusters. An object of the Cluster class has *x* and *y* coordinates that specify the area of the cluster. Each object of the Cluster class has a list of pointers called *Members*. Items in this list point to the UE Controller objects that are members of this cluster. Furthermore, each object of the cluster class has a list of pointers called *SMs*. Items in this list point to the UE Controller objects that are SMs of this cluster.

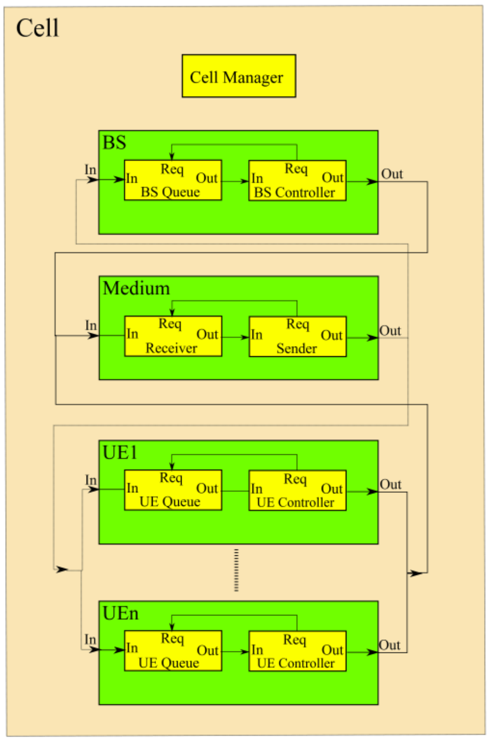

Fig. 3. Coupled DEVS model of the cellular network.

The transmission *Medium* model represents the interaction between the UEs, BSs, and pairs of UEs in the model. The Medium model receives all the messages sent and broadcasts them to all the other nodes and the BS. As mentioned above, the Queue of the BS and the UEs will use the destination address to recognize their messages.

The cellular DLs and ULs between the BS and the UEs, as well as the D2D links between the UEs are initialized by the Cell Manager at the beginning of the simulation. We consider path loss and shadowing. For cellular links, the Macro cell propagation model for urban area is employed [24]. The propagation model (*L*) is given by,

$$L = 40*\left(1-4*10^{-3}*Dhb\right)*\log_{10}\left(d\right) - 18\log_{10}\left(Dhb\right) + 21\log_{10}\left(f\right) + 80dB \tag{1}$$

where *d* is the base station-UE separation in kilometers, *f* is the carrier frequency in MHz, and *Dhb* is the base station antenna height in meters, measured from the average rooftop level. The path loss (*PL*) is then can be calculated as

$$PL = L + LogF \tag{2}$$

where *LogF* is a log-normally distributed shadowing with standard deviation of 10dB. The received signal is then can be calculated as

$$P_{RX} = P_{TX} - MAX(PL - G_{TX} - G_{RX}, MCL) \tag{3}$$

where P_{RX} is the received signal power, P_{TX} is the transmitted signal power, G_{TX} is the transmitter antenna gain, and G_{RX} is the receiver antenna gain, and *MCL* is the minimum coupling loss.

Considering Additive White Gaussian Noise (AWGN), the link data rate, *R*, can then be calculated as

$$R = B * \log_2(1 + \frac{P_{RX}}{N_0 B}), \tag{4}$$

where N_0 is the noise variance and *B* is the transmission bandwidth.

For D2D transmission, we used a Millimeter wave channel model at 24 GHz defined in [25]. The path loss for D2D links, PL_{D2D}, can be calculated as

$$PL_{D2D} = 60.05 + 1.95 * 10\log_{10}(d) + LogF_{D2D}, \tag{5}$$

where $LogF_{D2D}$ is log-normally distributed shadowing with standard deviation of 4.3 dB. The data rate is calculated as in equation (4).

The BS Controller and the UE Controller models were built using a similar approach. For a detail description of these models and their functionalities, the reader is referred to the Appendix.

V. Simulation Scenarios and Results

We used system level simulations to evaluate the potential of CSVD. The DL cell's aggregate data rate and average data rate per user for CSVD were compared to those of the SVD algorithm. For SVD, the aggregate rate is the rate of the data transmitted from the BS to all the UEs. For CSVD, the aggregate rate is the total rate of data transmitted from the BS to all UEs, and the data transmitted from SMs to other UEs over D2D links. The average data rate per user is the average rate at which a single user receives data. The simulation scenarios presented here consider a single cell with a variable number of UEs (ranging from 100 to 550). The urban macro propagation model [24] 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 [25]. Table 2 shows the simulation setup.

In each iteration of the simulation, the UEs are randomly distributed throughout the cell using a uniformly distributed distance from the BS. The cell is divided into 4 clusters and UEs will be assigned to clusters based on their location as shown in Fig. 1. UEs in each cluster will also be marked as a SM or non-SM. UEs in the central area of each cluster will be marked as SMs. The central area of each cluster forms 1/4 of the total area of the cluster. Hence, roughly, one fourth of the UEs in each cluster will be SMs. The received signal was calculated according to the corresponding model for the DLs, ULs, and D2D links (refer to Section 4 for these models).

The UEs then start sending download requests to the BS. The UEs generate requests to download video files from a list. The popularity of such files is generated according to Zipf distribution to simulate a variable popularity of files, as it has been established that this is a good model for video files popularity [26]. Using this distribution, some files are requested more often than others are. The Zipf exponent, β, controls the relative popularity of the files. The size of the video files will be generated according to a logNormal distribution as in [27].

TABLE II. 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
File size range	1-100 Megabytes
Area configuration	Urban
Piece size (KB)	512
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

Each UE sends one request at a time, and after downloading the whole file, the UE will attempt to generate another request. Before generating each request, a UE waits for a random period according to a Poisson distribution with mean of 10 seconds. Each UE will send 2 download requests in total during each iteration (i.e., each UE will download 2 files). At the end of each iteration, we calculate the cell's aggregate data rate and the mean of the average data rate per user for all users. At the end of the simulations, we calculate the mean of the cell's aggregate data rates and the mean of the average data rates from all the iterations. The number of iterations in the presented simulations is 30. In addition to the mean, we show the margin of error for the sample mean for each value with a confidence interval of 95%. Unless stated otherwise, the number of UEs is 500, the Zipf exponent is 1.5, and the number of requests made by each UE is 2. We chose 500 as the default number of UEs because at 500 UEs we can better see the effect of changing the number of requests or Zipf exponent values. We use two requests as a reasonable number for file requests per UE, but we also investigate different values. We also study the impact of changing the Zipf exponent.

In addition to comparing CSVD and SVD, we also use our protocol to study the improvement achieved by our algorithm when compared to the architecture in [4]. In [4], one copy of a file is kept in each cluster in one UE and the whole file is sent from that UE to the requesting UE. In our algorithm, we keep multiple copies of each piece of a file (up to a maximum number) in multiple SMs. This will make pieces available in multiple SMs, which allows further parallelism when sending pieces, and allows more load balancing between SMs. We call this parameter, the number of cached copies of each piece.

Fig. 4 shows the cell's aggregate data rate versus the number of UEs in the network for SVD and CSVD with 1, 3, 5, and 7 cached copies, respectively. The number of cached copies indicates the maximum number of copies of each piece cached in each cluster (each copy is cached in a different SM). A 5 cached copies, for instance, means that 5 copies of a file will be saved in each cluster. This means, with 4 clusters, 20 copies will be cached in the cell.

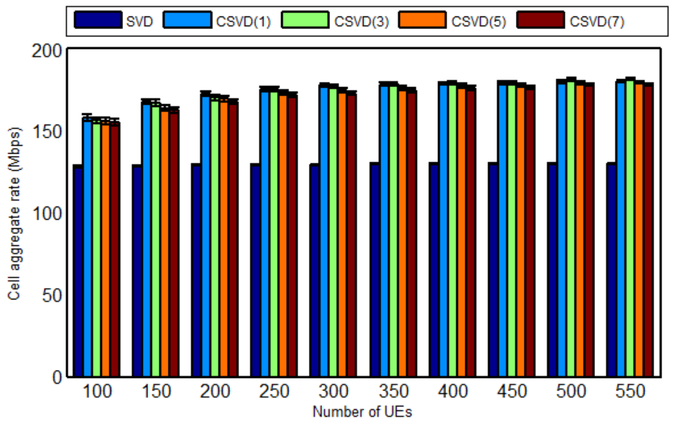

Fig. 4. Cell's aggregate data rate vs. number of UEs. 2 requests, β = 1.5.

Fig. 4 shows the improvement of CSVD over SVD. This improvement is caused by having more resources, i.e., the millimeter wave channel used for D2D communication to send video files between UEs, as opposed to only cellular frequency resources. As such, more data will be transmitted in the cell and aggregate data rate will increase. The maximum aggregate rate achieved using the SVD is around 130 Mbps, while with the CSVD, we can achieve an aggregate rate of 180 Mbps. This means that a significant improvement of 50 Mbps on the cell's aggregate data rate can be achieved.

As we can see, there is no significant effect for the number of UEs on the aggregate data rate for SVD. There is a small increase on the aggregate data rate of SVD when the number of UEs increases from 100 up to 300 (from 128 Mbps to 130 Mbps) Beyond 300 UEs, there is no effect for increasing the number of UEs. In SVD, each cell has fixed cellular resources (for example 10 MHz cellular channel) and as the number of UEs increases, the cellular frequency resources used to transmit data to the UEs will increase until they are fully utilized. As such, we can say from Fig. 4 that with SVD, at 100 UEs, the cell is overloaded and cellular channel is fully utilized. This explains the small effect of increasing the number of UEs byond 100 on the aggregate rate with SVD.

For CSVD, the figure shows that there is an increase in the aggregate data rate with the number of UEs in the network. This is caused by having a millimeter wave channel with high bandwidth (50 MHz) used for D2D communications. As the number of UEs increases, there will be more requests, and more SMs. As such, more files will be cached and later sent from the distributed cache. Hence, the D2D channel will be further utilized and the aggregate data rate will increase. The aggregate data rate increases from 159 Mbps to 179 Mbps when the number of UEs increases from 100 to 350. However, the increase in the data rate slows down after 350 UEs. This is due to many reasons. First, as the number of UEs increases, a 50 UEs increment will have less effect. Second, the algorithm is also still limited by the degree of file reusability. Only few files are popular and the rest of the requested files are not. Furthermore, each UE requests only 2 files in total. Later in this section, we will show the effect of increasing file reuse.

One can see that there is no significant effect for the number of cached copies for the CSVD algorithm on the aggregate data rate, because increasing the number of cached copies will not cause much increase in the utilization of D2D channels. Because for high number of UEs, the SMs in each cluster will be busy sending pieces to the requesting UEs, even if only 1 copy of each file is cached in the cluster.

Fig. 5 shows the average data rate per user versus the number of UEs in the network for SVD and CSVD with 1, 3, 5, and 7 cached copies, respectively. The average data rate decreases with the number of Ues because the available resources (frequency resources, BS processing power) is divided over higher number of UEs. As can be seen, a significant improvement is achieved by the CSVD over the SVD. At 100 UEs, the average data rate per user is around 2 Mbps for the SVD, while the average data rate per user is around 5.6 Mbps for the CSVD with a number of cached copies of 3 or more. This is because with the CSVD, the files will be transmitted to the requesting UE in parallel from multiple sources and from the cluster cache, which speeds up the transmission process. The impact of the CSVD increases with increasing the number of users. For instance, at 550 UEs, the average data rate per user for the SVD is 0.33 Mbps, while for the CSVD and 3 cached copies it is 2.9 Mbps, which is about 9 folds improvement. As mentioned before, this is because increasing the UEs also increases the available SMs and requested and cached files. Thus, more data will be transmitted from the local cache over D2D links rather than being sent from the BS over cellulr links. As such, increasing the number of UEs will cause less decrease in the average data rate per user than in the SVD, where it decreases drastically due to dividing fixed cellular resources over higher number of users.

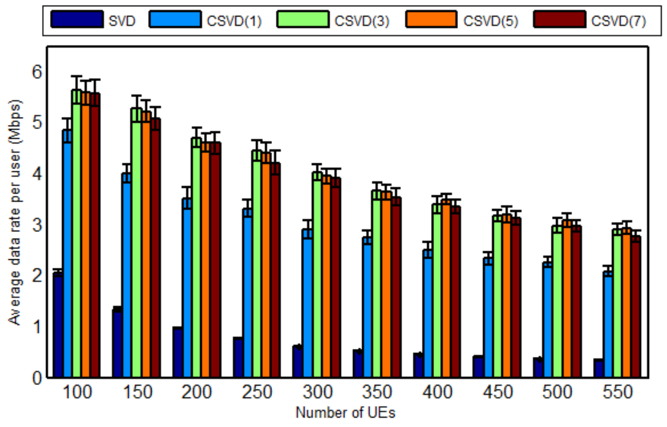

Fig. 5. Average data rate per user versus the number of UEs. 2 requests per user, and β = 1.5.

Furthermore, it can be seen from Fig. 5 that significant improvement is achieved with the CSVD when the number of cached copies is increased from 1 to 3. This improvement is caused by having the popular files available in more SMs, which allows further parallelism when sending pieces, and allows more load balancing between SMs, which speeds up the transmission of video files to the requesting UEs.

After 3 cached copies, there is no significant effect for the number of cached copies in the cluster, which means that 3 copies in the cluster are enough, especially for low number of UEs. This is explained in the following. The BS sends a piece to a SM directly (STSM) if the number of copies in the cluster is less than the intended number of cached copies. Otherwise, the piece will be sent to the SM from the distributed cache. Hence, increasing the number of cached copies will increase the average data rate up to a certain point. After some point increasing the number of cached copies beyond a certain value might cause a slight reduction in the average data rate, as too many copies will need to be sent to SMs from the BS over cellular links even when enough copies are already cached in the cluster. This explains why the average data rate for CSVD with 7 cached copies is slightly less than that of CSVD with 3 and 5 cached copies. This also depends on the number of UEs in the cluster. If the number of UEs in the cluster is large, there will be too many requests and hence it is worth it to cache a file in many SMs. However, if the number of UEs is small, caching files in too many SMs might cause a reduction in the average data rate as explained above. This explains why the average data rates for CSVD with 3 cached copies is slightly higher than that with 5 cached copies when the number of UEs is less than 350, and the other way for number of UEs greater than 350.

The Zipf exponent β controls the relative popularity of files. Higher values of β lead to higher content reuse. This means that higher β means that the popularity of the first files in the list will increase, and they will be requested more often. Fig. 6 shows the average data rate per user versus the Zipf distribution exponent for CSVD. As expected, the average data rate increases by increasing the Zipf distribution. This is because when the popularity of some files increases, they will be cached and requested by more UEs. This will increase content reuse and speeds up the transmission process as more files will be delivered from the local cache rather than from the BS over cellular links. However, the increase in the download rate will eventually stop, as the algorithm is limited by the available cellular and D2D frequency resources.

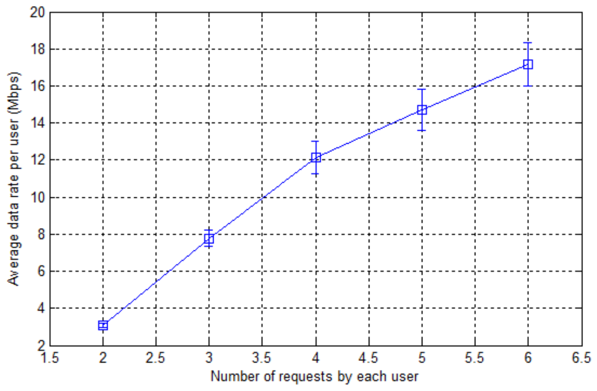

Fig. 6. Average data rate per user versus Zipf exponent for CSVD. 500 UEs and 5 cached copies.

Fig. 7 shows the average rate versus the number of files requested by each UE during each iteration of the simulations for CSVD. As can be seen, the average download rate increases by increasing the number of requests by users. This is because as the number of file requests increases, more files will accumulate in the cluster cache, and consequently, the cached files will be further reused by the later requests. As such, more requests will be satisfied from the cluster cache over D2D links, which results in higher average data rate per user. Fig. 7 shows that the increase in the data rate eventually starts to slow down. This is because the algorithm is still limited by the available D2D subchannels and the available cellular RBs (cellular resources are used to send Assistance Request messages to SMs).

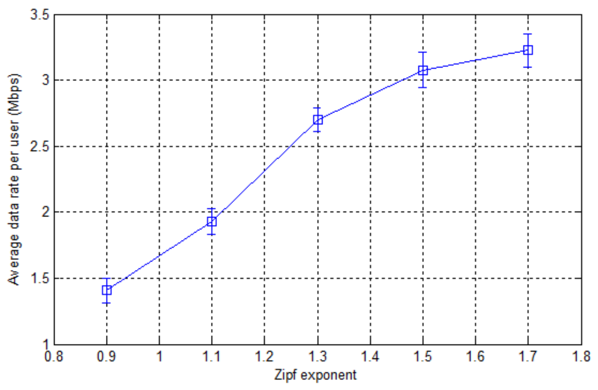

Fig. 7. Average data rate per user versus Number of requests per user for CSVD. 500 UEs, 5 cached copies, and $\beta = 1.5$.

VI. Conclusion

Wireless video has been one of the main drivers of wireless data traffic recently. This continuously increasing video traffic causes a major challenge to the already overburdened cellular networks. Here, we proposed the Cached and Segmented Video Download (CSVD) algorithm to improve the throughput of video transmission in cellular networks. The algorithm splits video files into pieces that are cached in selected user equipment (UEs) in the network, and employs the Device-to-Device (D2D) communication between UEs in cellular networks to exchange files' pieces. We use the Discrete Event System Specification (DEVS) formalism to model an LTE-A cellular network. The model is used to study the performance improvement achieved by the algorithm in terms of cell's aggregate video transmission rate as well as average data rate per user. Simulation results show that CSVD achieves very significant improvement over the conventional transmission approach where D2D communication is not employed. User mobility and power constraints will be considered in future work.

References

[1] ICT Data and Statistics Division, Telecommunication Development Bureau, ITU. "ICT facts and figures." Internet: http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2014-e.pdf. Apr. 2014 [Apr. 20 2015].

[2] 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/white_paper_c11-520862.html. 2015 [May 5 2015].

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] B. Cohen. "The BitTorrent protocol specification," Internet: http://bittorrent.org/beps/bep_0003.html. Oct 2013 [May 5 2014].

[11] R. Clarke, "Expanding mobile wireless capacity: The challenges presented by technology and economics," *Telecommunications Policy*, vol. 38, no. 8-9, pp. 693-708, 2014.

[12] V. Chandrasekhar, J. Andrews, and A. Gatherer, "Femtocell networks: a survey," *IEEE Communications Magazine*, vol. 46, no. 9, pp. 59–67, 2008.

[13] H. Ahlehagh and S. Dey, "Hierarchical video caching in wireless cloud: Approaches and algorithms," in *Proc. IEEE ICC*, 2012, pp. 7082–7087.

[14] M. A. Maddah-Ali and U. Niesen, "Decentralized caching attains order-optimal memory-rate tradeoff," Accepted in *IEEE/ACM* Transactions on Networking, arXiv preprint arXiv:1301.5848, 2014.

[15] S. A. Chellouche et al., "Home-box-assisted content delivery network for Internet video-on-demand services," in *Proc. IEEE ISCC*, 2012, pp. 544–550.

[16] H. J. Kang et al., "Mobile caching policies for device-to-device (D2D) content delivery networking," in *INFOCOM WKSHPS*, 2014, pp. 299 - 304.

[17] M. Ji, G. Caire, and A. F. Molisch, "Wireless Device-to-Device Caching Networks: Basic Principles and System Performance," Submitted to *Networking and Internet Architecture*, arXiv preprint arXiv:1305.5216v2, 2014.

[18] M. Ji, G. Caire, and A. F. Molisch, "Fundamental Limits of Caching in Wireless D2D Networks," Submitted to *IEEE Transactions on Information Theory*, arXiv preprint arXiv:1405.5336v1, 2014.

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 3rd Generation Partnership Project, "Technical Report 36.942, V12.0.0," 2014.

[25] 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.

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

[27] 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.

APPENDIX: FUNCTIONALITY OF THE BS AND UE CONTROLLERS

Note: This appendix can be deleted from the final version of the paper to comply with the paper size limitations. The purpose of this appendix is to provide further details on the implementation of the BS and UE Controllers to the reviewers.

A. BS Functionality

The BS manages all the data transmission in the cell, the sessions with all the UEs, and allocates the radio frequency resources to the UEs. Messages sent to the BS from UEs are buffered in the BS queue. Every Transmit Time Interval (TTI) which is 1 ms, the BS will check its queue for messages from UEs. The BS will process received messages and update the state of the corresponding download session. For instance, if the received message is a new download request, the BS will create a new Session object to that node. If it is a *BitField* message, the BS will delete the corresponding session, as the file transfer is complete. If the received message is *SM_Finished* message, the BS will update the statistics of that session (number of transmitted pieces, transmitted bits, etc.)

At the beginning of the simulation, the BS controller is at the *Initial* state. Then it goes to the "*Check Queue*" state, during which, the BS sends requests to the queue asking if there are any messages from UEs. The queue will send the next message in line. When BS receives a message, it goes to "*Receive message*" state. During this state, the BS processes the message and then sends another request to the Queue. 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 will go to the "*Schedule*" State.

Scheduling is the most complex functionality of the BS model. During this state, the BS will go through the list of sessions. Depending on the state of the session, the BS will take action,

- If the session is new, the BS will prepare the MetaInfo and *Handshake* message.
- If the BS is awaiting a message from the corresponding UE, the BS will skip to the next session.
- If a piece message should be transmitted for that session, the BS will prepare an *Assistance Request* message if the piece is cached in the cluster, or prepare a *Piece* message if the piece is not cached in the cluster.
- If the session has a message to transmit (*Handshake* message, *Done* message, *Assistance Request*, or a *Piece* message) the BS will call the message schedule function to allocate cellular RBs enough to transmit the whole message. If there are not enough RBs to transmit this message, the BS will allocate the remaining RBs in the next TTI.

When the message scheduling function is called, it will calculate the needed RBs to transmit this message. The function then allocates RBs from this TTI to transmit the current message. If RBs in this RB are not enough, the model will allocate all the available RBs to this session (the needed RBs to transmit this message will be assigned from the next TTI), and go to the *Send* state, to send messages that are already scheduled for this TTI. A message is only sent when enough RBs are allocated. In the case in which the number of remaining RBs is greater than the number of needed RBs, the model will update the number of remaining RBs in this TTI, update its statistics, and push the current message to the list of messages that will be sent during the next *Send* state, and then go to the next session.

After allocating all the RBs in TTI or scheduling all the messages that should be transmitted, the model will go to the *Send* state. At this stage, the model will go through the messages in the "Messages to send" list, and send them. The BS will send the messages that are scheduled to be sent in the first TTI slot, and then send the remaining messages.

The BS also divides the D2D channel into subchannels. Every time the BS sends an *Assistance Request* message to a SM, it will assign a subchannel to this SM to use it to send the piece. When the BS receives *SM_Finished* message, it will get the subchannel back, and can use it with another assistance request.

B. UE Functionality

Each UE starts at the *Idle* state as shown in Fig. 8. Each time the UE at the *Idle* state, it will generate a request with a probability of 0.5 in 5 seconds, or wait and try again after 5 seconds. After generating a request, a UE will wait for a *Handshake* message. After receiving the *Handshake* message, the UE will set the parameters for this session (file size, number of pieces, piece size, etc.). The UE will then wait for pieces of the file. After receiving all the pieces, and a *Done* message, the UE will send a *Bitfield* message. When the file download is complete, the UE goes back to *Idle* state, and tries to generate another download request as described above.

When a UE receives a *Piece* message while it is in *Await Piece* state, it updates the statistics for this session, and goes back to *Await Piece* state. If the UE is an SM and the *Save* bit in the *Piece* message is set, the UE will save the *Piece* message. The Save bit is always set for *Piece* messages in STSM case (As described in section 3).

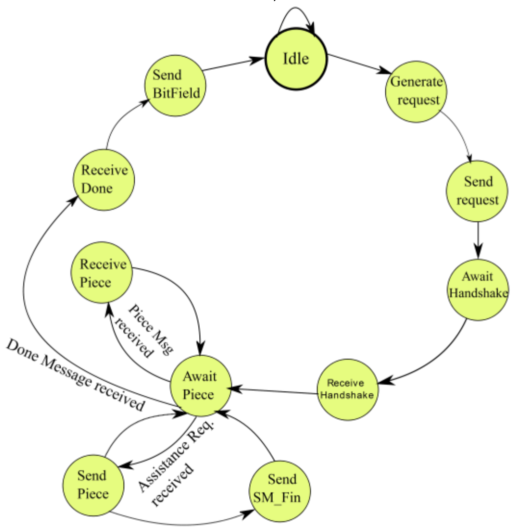

Fig. 8. UE Controller's state diagram.

When an SM UE is in the *Await Piece* state, and it receives an *Assistance Request* message, it will go to *Send Piece* state where it sends a *Piece* message to the requesting UE. Then it could go to back to *Await Piece* state, or could go to Send *SM_Finished* message to acknowledge the transmission of Piece(s), before going back to *Await Piece* state.