Distributed Cached and Segmented Video Download for Video Transmission in Cellular Networks

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*—Wireless video accounted for more than half of the total data traffic in cellular networks in 2015, and this is expected to further increase in the upcoming years. Device-to-Device (D2D) communication, introduced by the LTE-Advanced (LTE-A) standard, allows direct communication between devices in cellular networks. Here, we introduce the DIStributed, Cached, and Segmented video download (DISCS) algorithm for improving the throughput of video transmission in cellular networks based on D2D communications. The algorithm splits video files into pieces, which are distributed over selected user equipment (UEs) in the cellular network, to cache and forward the pieces using D2D communication. We used the Discrete EVent System Specification (DEVS) formalism to build an LTE-A network model and used the model to study the performance of DISCS. Simulation results show that DISCS achieves significant performance improvements in terms of the cell's aggregate data rate as well as the average data rate per user.**

Keywords—LTE-A; Cellular networks; D2D communication; DEVS

I. INTRODUCTION

The demands for higher data rates in cellular networks have been continuously increasing due to the improvements on mobile devices, the services provided, and the increasing number of users [1]. As most of the licensed frequency bands are now allocated, it is difficult to provide sufficient resources to these demands. Furthermore, the data rates provided by radio links have been approaching their theoretical capacity. As such, it has become a main challenge for cellular networks operators to provide such high data rates for users.

Due to the improvements on smart devices, wireless and mobile users recently tend to watch longer videos with higher resolution on their phones and tablets. Consequently, in 2015 55 percent of the total mobile data traffic was due to wireless video traffic [2]. Furthermore, it is predicted that three-fourths of the world's mobile data traffic will be video by 2020 [2]. This increase in video traffic will increase the data traffic and further complicate things for cellular networks operators. Hence, new techniques are needed to help serving the video traffic, which is becoming the majority of data traffic.

Device-to-Device (D2D) communication is an innovative feature that was introduced by the LTE-Advanced (LTE-A) standard [3]. D2D enables direct communication between nearby user equipments (UEs) without routing the traffic through the Base Stations (BS) and the network infrastructure. Exploiting D2D transmission brings many benefits to communication in cellular networks. Capacity gains can be achieved by sharing spectrum resources between cellular and D2D users. Furthermore, data rate gains can be achieved due to proximity and potentially favorable propagation conditions. There have been various efforts to combine current standards with D2D communications in order to achieve such gains [4-7].

In [8-9], we proposed the Cached and Segmented Video Download (CSVD) algorithm for improving the throughput of video transmission in cellular networks, and showed that this method helps with the increasing data traffic. Our algorithm is based on the architecture proposed in [10], which exploits D2D communications for BS-assisted peer-to-peer (P2P) video transmission in cellular networks. The main 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 from the caching UEs over D2D links.

Here, we present an improvement to such algorithm, namely, the DIStributed, Cached, and Segmented video download (DISCS). In DISCS, a video file is divided into pieces. The pieces of a video file are distributed over multiple Storage Members (SMs) to be cached and forwarded to the requesting UE. This provides further parallelism when transmitting video files and further load balancing among SMs, which speeds up the transmission process. Furthermore, in DISCS, the SMs will be required to receive and forward pieces when asked for assistance (as opposed to just forwarding pieces they already have) which helps accumulating video files faster in the distributed cache.

We built a suite of models using the Discrete EVent System Specification (DEVS) formalism [11-12] and the CD++ DEVS toolkit to model an LTE-A cellular network that employs DISCS. System level simulations were performed to evaluate the performance of the DISCS algorithm with different parameters and under different simulation scenarios. Simulation results show that DISCS achieves significant improvements over the CSVD algorithm in terms of both the cell's aggregate data rate as well as the average data rate per user.

The rest of this paper is organized as follows, In Section 2, we provide an overview of the related work. In Section 3, we present the DISCS algorithm. In Section 4, we describe the modeling of the LTE-A cellular network in DEVS. Simulation scenarios and results are presented in Section 5. The conclusions are stated in Section 6.

II. Background

The high demand for video content in cellular networks has increased the data rate requirements [2]. One solution that has been implemented to increase the frequency efficiency and provide higher data rates is decreasing the size of cells [13]. However, there are some challenges for having small-sized cells. First, it becomes more difficult to handle mobility and control interference as the size of the cells decreases. Second, it becomes more costly to implement a high-capacity wired backbone as the number of cells increases. As such, innovative solutions are required to increase the provided data rates and help serving the video traffic that would take up a major portion of the overall traffic.

Caching popular video files at the BSs or mobile switches has been employed to help improve the transmission of video traffic in cellular networks [14]. When a popular file is cached at the BS, it will be available when requested by the UEs, which eliminates the need for requesting the video from the web and reduces the amount of traffic on the backhaul network. However, this solution does not reduce the amount of traffic between the BSs and UEs over cellular frequency links.

Other researchers proposed that the users should cache on their devices the contents that are expected to be requested in the future. For instance, the authors in [15] proposed an adaptive popularity-based video caching strategy. The strategy enables strong collaboration between users and service environments for ensuring better quality of services to end-users. Video contents are dynamically cached in home-boxes following users' demands, allowing their delivery from optimal places. Although such approach helps reducing the amount of video traffic, cached contents can be only used locally by the caching devices, and cannot be exploited by others in the network.

In [16], the authors proposed an approach where the content is cached on central servers close to the users. Furthermore, a central server can use simultaneous coded-multicasting to satisfy the requests of several users with different demands with a single multicast stream. However, this scheme relies on a carefully designed placement phase in order to create coded-multicasting opportunities. Since the content placement is performed before the actual user demands are known, it has to be designed carefully such that these coded-multicasting opportunities are available simultaneously for all possible requests.

The approaches above represent client-server models, which do not scale well for video content distribution in cellular networks with high number of users and limited resources. As such, there has been a need for P2P communication models. In particular, D2D communication presents a promising solution as it allows direct communication between UEs [4]. D2D communication depends on the participation of users for 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 such user involvement in D2D communication [17-19].

Combining D2D communication and video caching has been proposed recently. The mobile content delivery network has been introduced recently as a distributed system where 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 [20]. While this technology could help improving the data rates in cellular networks, it is costly as these designated devices need to be placed throughout the network, configured, and maintained.

There has been some work on the use of D2D for P2P video communication in cellular networks [21-23]. All the previous works adapt algorithms that are similar to P2P streaming protocols on wired networks that involve the dissemination of buffer maps and video pieces between peers. While such protocols are suitable for communication on wired networks, they involve too much signaling and transmission (such as dissemination of buffer maps) to be appropriate for UEs with limited power and resources. Furthermore, the previous work considers small-scale networks (up to 10 UEs). The number of UEs of an LTE-A cell in urban areas is usually higher. Here, we show that using clustering and BS assistance, the potential of collaborative D2D communication between UEs is significant.

The architecture, proposed in [10], employs D2D communication to improve the throughput of video transmission and overcome the problem of rapidly increasing wireless video traffic. In this architecture, the cell is divided into "clusters". Each cluster contains a group of nodes that can exchange information with each other 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 [10] is oversimplified and the original architecture is limited; it assumes that the files are pre-cached in the nodes. As such, the architecture needs a complex and carefully designed placement phase. Since the content placement is performed before the actual user demands are known, it has to be designed carefully so that users make use of the cached content. The architecture also assumes that 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 video files. Instead, a simple model was used to study the performance of the architecture analytically.

In [8-9], we proposed the CSVD algorithm that is based on the architecture described above, but instead of caching complete files, the files are split into pieces and multiple copies of a file can be cached at multiple SMs. 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. Only 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.

Here we extend our work in [8-9], and propose the DISCS algorithm. In DISCS, the pieces of a requested video file are distributed over multiple SMs to be cached and forwarded to the requesting UE. This provides further parallelism when transmitting video files and further load balancing among SMs, which speeds up the transmission process. Furthermore, in DISCS, the SMs will be required to receive and forward pieces when asked for assistance (as opposed to just forwarding pieces they already have), which helps accumulating files faster in the distributed cache. The DISCS algorithm will be described in details later in Section 3.

We used the DEVS formalism [11] to build a model for an LTE-A network that employs DISCS. DEVS provides a sound formal framework for modeling generic dynamic systems. DEVS 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.

We used the CD++ toolkit [12] 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. We use the developed model to study the performance of the DISCS algorithm and compare it to the CSVD algorithm by running various simulations. In the following Section, we provide a detailed description of DISCS, and show how it operates.

III. The DISCS Algorithm

Both the CSVD and DISCS are designed for scenarios where there is a high density of users in the cell, such as,

- 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.

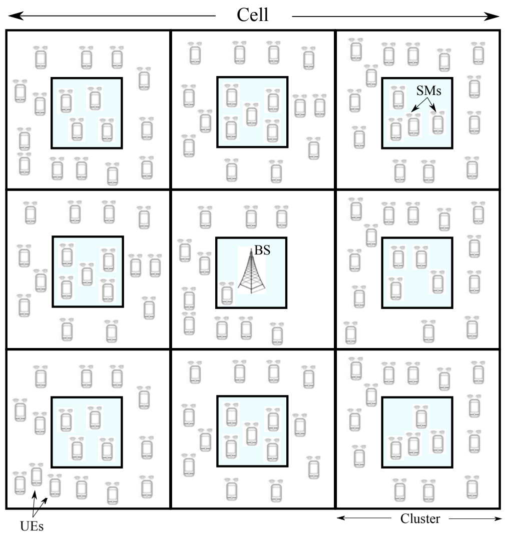

Fig. 1. Cell divided into 9 clusters

As in the CSVD, the BS initially divides the cell into non-overlapping subareas, each one of which will be a cluster. The BS then assigns UEs to clusters based on their locations [9]. The UEs that are in the central area of each cluster will be selected as SMs of that cluster. Only the UEs in the central area are chosen as SMs in order to prevent inter-cluster interference when the SMs transmit to other UEs in the same cluster using D2D links. Fig. 1 shows a cell that is divided into 9 clusters.

After dividing the cell into clusters, the transmission phase starts. The UEs send their requests to download video files to the BS. The BS processes a download request, and responds differently depending on the case. We consider four different cases for DISCS:

- Send With Assistance (SWA): if the file (or a part of it) is available in any of the SMs of the cluster, the BS will ask these SMs to send the pieces they have 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), and the requesting UE is a SM, the BS will send the file to that SM over a cellular link, and ask the SM to cache the video file. These files will be available for UEs in the cluster later.
- Distribute to SMs (DTSMs): this new case proposed by DISCS is as follows. If a requested video is popular and it is not available in the distributed cache of the cluster, the BS will distribute the pieces among the SMs. The BS asks the SMs to cache the pieces (as the file is popular), and asks them to forward the received pieces to the requesting UE.
- 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 described above in detail.

A. *Send With Assistance*

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. Table 1 shows the fields of the *MetaInfo* file.

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. The BS then sends a Handshake message to the requesting UE. The Handshake message contains the *MetaInfo* file.

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 should 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.

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 that piece.

6) When an SM finishes sending piece(s), 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 to the BS indicating the pieces it has received.

B. *Send To a SM case*

In this case, the file transfer starts by the SM sending a *Download Request* message to download a video file. After receiving the request, the BS will start a session with this UE. If this is the first time an SM requests 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 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 the pieces and the *Done* message, it will send a message containing the *BitField* to the BS.

C. *Distribute to SMs case*

As mentioned above, this is the main contribution of the DISCS algorithm. In this case, a popular file (for instance, one requested *n* times) is requested by a non-SM UE. The BS distributes the video file pieces over SMs and asks them to cache the pieces and forward them to the requesting UE (as the file is popular, and distributing it will be beneficial for the cluster). The download process is as follows:

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

2) The BS creates a *MetaInfo* file that describes the parameters for the download session of this video file (as in table 1). The BS then sends a *Handshake* message (containing the *MetaInfo* file) to the requesting UE.

3) The BS then sends *Assistance Request* messages to the SMs of the cluster asking their help to send the pieces to the requesting UE. There is a field in the message that is set to indicate that this is a "receive and forward" request, i.e., the SM is needed to receive the piece, cache it, and forward it to the requesting UE.

4) The SMs will send a *Response* message to indicate their availability for assistance, and to indicate the maximum number of outstanding assists the BS can send.

5) The BS then starts distributing the pieces to the SMs. Each piece message has an index that identifies that piece.

6) When an SM receives a piece, it will cache it, and send it to the requesting UE over D2D link.

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

8) When the BS receives *SM_Finished* for all the pieces from the SMs participating, it will send a *Done* message to the requesting UE.

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

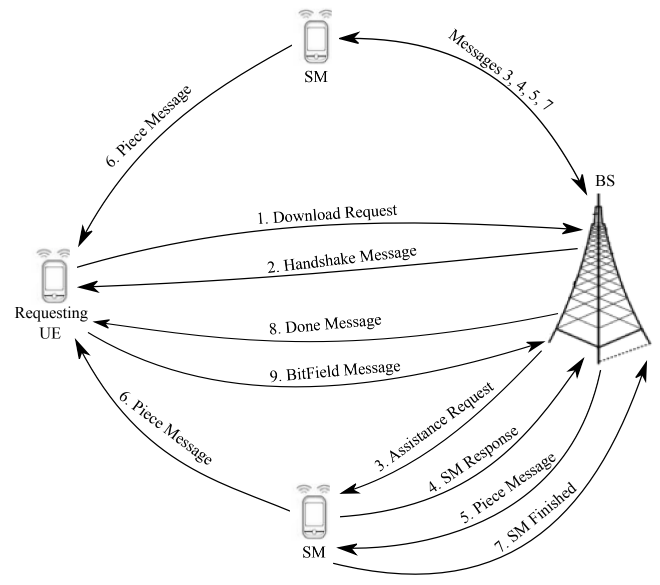

Fig. 2. DTSMs case.

This case further helps accumulating popular video files in the distributed cache of the cluster. It also allows for more parallelism and load balancing among SMs when sending video files from the distributed cache of the cluster. This should increase the utilization of the D2D channel and speeds up the transmission, and consequently increase the average data rate.

In addition to the data the BS needs to keep in its database for the CSVD (list of the clusters, list of the members and SMs of each cluster, and list of cached files/pieces), the BS keeps track of the number of times the files were requested recently.

D. *Send To a UE case*

In this case, the requesting UE is not an SM, and the file is not available in the cluster. Hence, the BS will transmit the file directly to the requesting UE over cellular links. This case is similar to STSM (case B). However, in this case, the Save bit is always zero in the *Piece* message so that the piece will not be cached.

E. 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 always sent as in STUE (case D above).

IV. Modeling the LTE-A Network with DEVS

Fig. 3 shows a DEVS coupled model definition of the LTE-A network we want to simulate. At the top level, we have a *Cell* coupled model, which contains the *BS*, *Transmission Medium*, and many *UE* coupled models. It also contains a *Cell Manager* atomic model.

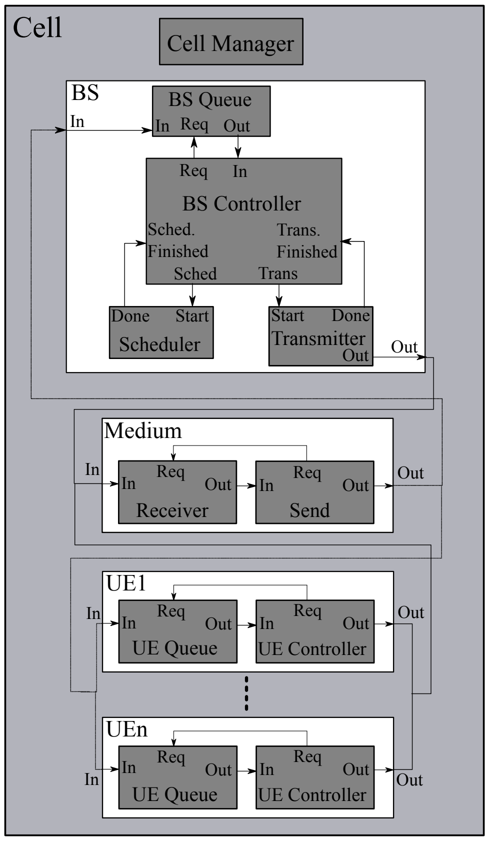

Fig. 3. DEVS model of the cellular network.

The BS coupled model is in charge of modelling the BS in the cell. It has four atomic models; *BS Queue, BS controller, Scheduler, and Transmitter*. Messages are buffered at the *BS Queue* atomic model, which also checks the destination address of a received message. If it matches that of the BS, the message will be buffered, otherwise it will be discarded. The *BS Controller* processes received messages and it implements the algorithms above (for example, steps 2, 3, 5, and 8 in Fig. 2). Every Transmission Time Interval (TTI), which is 1 ms, the BS processes received messages and asks the Scheduler to schedule the messages to be sent in the next TTI. Every TTI, the BS Controller also asks the Transmitter to send messages that were scheduled for transmission during this TTI.

The UEs in the cell are modeled with the *UE* coupled models. A *UE* coupled model contains two atomic models; *UE Queue*, and *UE Controller*. Received messages are buffered at the *UE Queue*. The *UE Controller* is where the UE part of the algorithm is implemented.

The *Medium* model receives a message sent from the BS or any UE and broadcasts it to the other receivers (BS/UEs) in the cell. As mentioned above, the queue of the BS and the UEs will use the destination address to recognize their messages.

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. Path loss and shadowing is considered here. The urban macro propagation model [24] was used for cellular links and the D2D channel model at 24 GHz, defined in [25], was used for D2D communication.

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 DLs and ULs, D2D links, download sessions the BS has with UEs, cell clusters, exchanged message, etc.

V. Simulation Scenarios and Results

We implemented our DEVS model of the LTE-A network using the CD++ toolkit. System level simulations were performed to evaluate the performance of DISCS and compare it to the CSVD and SVD in terms of DL cell's aggregate data rate and average data rate per user.

A. Simulation scenarios

In the Simulations, we consider a single LTE-A cell. 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. According to [24], the propagation model (*L*) is given by,

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

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

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

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

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

where P_n is the noise power and *B* is the transmission bandwidth.

For D2D transmission, we used the D2D channel model at 24 GHz, defined in [25]. According to that model, the path loss for D2D links, PL_{D2D}, can be calculated as,

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

where d is the transmitter-received distance in meters, and $LogF_{D2D}$ is a log-normally distributed shadowing with standard deviation of 4.3 dB. The data rate is calculated considering AWGN. Table 2 shows the simulation parameters we used.

TABLE II. SIMULATION SETUP

Parameter	Value
Cellular Channel BW (MHz)	10
Cell Range (m)	500
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 (MHz)	900
File size range (MB)	1-100
Area configuration	Urban
Piece size (KB)	512
Number of files	500
D2D Channel BW (MHz)	60
D2D Carrier frequency (GHz)	24
D2D transmitter TX Power (dBm)	23
D2D Large-scale fading std deviation (dB)	4.3
UE receiver noise figure (dB)	9
D2D TX/RX Height from Ground (m)	1.5

In the beginning of each iteration of the simulations, the UEs are uniformly distributed throughout the cell. The cell is divided into 9 clusters. According to their location in the cell, UEs are assigned to clusters as shown in Fig. 1. Furthermore, the UEs in the central area of each cluster are 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. Each iteration in the simulations is divided to two phases; a transient phase, followed by a steady state phase. At the beginning of the transient phase, there are no files cached in the clusters. As UEs download videos during the transient phase, video segments will accumulate in the distributed cache of each cluster. Each UE performs two requests during the transient phase. At the beginning of the steady phase, there will be many pieces in the distributed caches of the clusters that were cached during the transient phase. We present results for both phases. During each phase, each UE sends 2 download requests in total (i.e., each UE downloads 2 video files). A UE sends one request at a time, and after downloading the whole video file, it generates another request. Before each request, a UE waits for a random period using a Poisson distribution with mean of 10 seconds. At the end of each phase, we calculate the cell's aggregate data rate and the mean of the average data rate per user. The mean of the cell's aggregate data rate and the mean of the average data rate from all the iterations are calculated at the end of the simulations. The results show the mean values based on 40 simulation runs along with the margin of error for 95% confidence interval.

The UEs generate requests to download video files from a list. The popularity of videos is generated according to a 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]. Unless stated otherwise, the number of UEs is 500, the Zipf exponent is 1.5, and the number of requests made by a UE during each phase is 2.

B. Simulation results

Fig. 4 shows the Cell's aggregate data rate versus the number of UEs in the cell, for the SVD, CSVD, and DISCS algorithms, respectively, in the steady state phase. Up to one copy of each piece of a file is cached in a cluster in the case of CSVD and DISCS. As Fig. 4 shows, CSVD and DISCS provide significant improvement over the SVD. The maximum aggregate rate achieved using the SVD is around 130 Mbps, while with the CSVD and DISCS, aggregate rates of 490 Mbps and 690 Mbps can be achieved, respectively, at 700 UEs. This significant improvement on the aggregate data rate is due to having more resources, i.e., the D2D channel with large bandwidth (60 MHz) available in each cluster, and used for D2D communication.

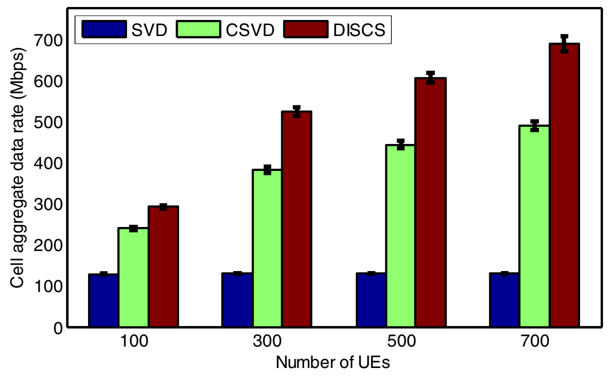

Fig. 4. Cell's aggregate data rate vs. number of UEs (steady state phase). 2 requests per user and β = 1.5.

Furthermore, DISCS achieves significant improvement over CSVD. This is because in CSVD, when a video file is cached in a cluster, it is always cached in one SM, while in DISCS, a cached file is distributed over many SMs in the cluster in the case of DTSMs. As such, when video files are transmitted from the distributed cache, multiple SMs will be sending pieces in parallel to the requesting UE in the case of DISCS. As such, the D2D channel will be further utilized and the aggregate data rate will increase.

Fig. 4 also shows that with CSVD and DISCS, the aggregate data rate increases with increasing the number of UEs in the network. Increasing the number of UEs increases the number of requests for video files and the number of SMs in each cluster. This increases the number of cached files in a cluster and the number of requests that would be satisfied from the cluster cache. Hence, the D2D channel will be further utilized and the aggregate data rate will increase. With SVD, the aggregate data rate does not increase with the number of UEs in the cell. In SVD, each cell has fixed cellular resources (a 10 MHz channel is used here) and as the number of UEs increases, the utilization of the cellular channel will increase, until it is fully utilized. As such, we can say from Figure 4 that with SVD, at 100 UEs, the cell is overloaded and the cellular channel is fully utilized.

Fig. 5 shows the average data rate per user versus the number of UEs in the network for SVD, CSVD and DISCS, respectively (steady state phase). Up to one copy of each piece of a file is cached in a cluster in the case of CSVD and DISCS. As Fig. 5 shows, CSVD and DISCS provide important performance gains due to the transmission of video segments from the BS and SMs (distributed cache), as opposed to only transmitting video files from one source (the BS). This speeds up the transmission process and increases the average data rate. In the SVD, the average data rate decreases faster with increasing the number of UEs. For instance, the average data rate decreases from about 2 to 0.63 Mbps when the number of UEs increases from 100 to 300 UEs. This is because the fixed available frequency resources are divided over higher number of UEs. The improvement achieved by the CSVD and DISCS over the SVD increases when the number of UEs increases. This is because increasing the UEs also increases the available SMs and requested and cached files. Thus, more data will be transmitted from the cluster caches over D2D links rather than being sent from the BS over cellular links. As such, increasing the number of UEs will cause less decrease in the average data rate per user than in the SVD.

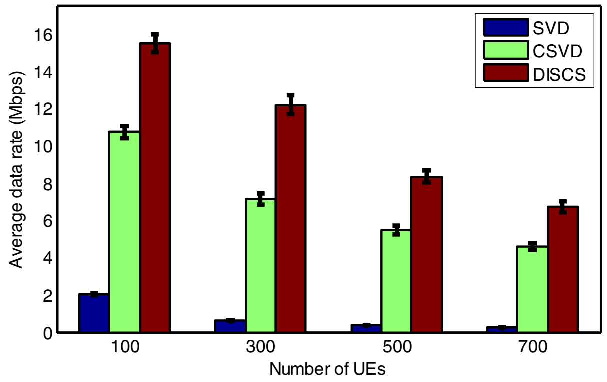

Fig. 5. Average data rate per user versus the number of UEs (steady state phase). 2 requests per user and β = 1.5.

Fig. 5 also shows that DISCS achieves significant improvement over CSVD. This is because in the case of DISCS, many files will be sent in parallel from multiple SMs (as opposed to one SM). This causes further parallelism in sending video files and better load balancing between SMs, which speeds up the transmission of video files and increases the average data rate.

As mentioned in the previous section, the simulations were divided into two phases. The first phase is the transient phase that starts with no video files saved in the distributed caches of the clusters, and the pieces of the video files accumulate in the distributed caches during this phase as requested by UEs. In the beginning of the steady phase, there will by many files in the clusters that were cached during the transient phase. Fig. 6 and 7 show the aggregate data rates and average data rates, respectively, for the transient phase versus the number of UEs.

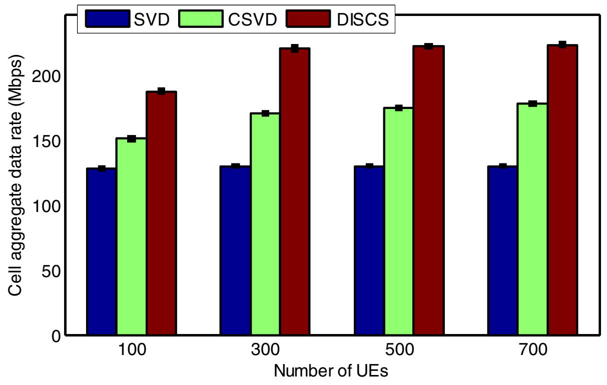

Fig. 6. Cell's aggregate data rate vs. number of UEs (transient phase). 2 requests per user and β = 1.5.

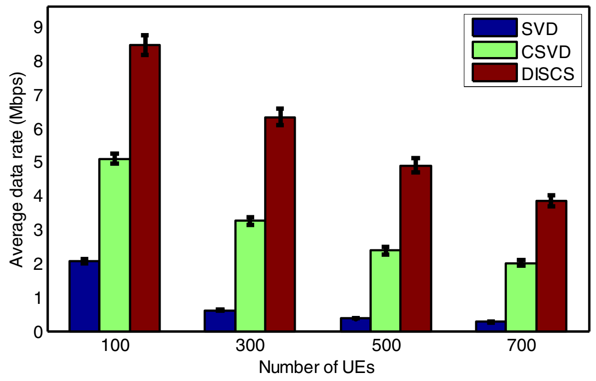

Fig. 7. Average data rate per user versus the number of UEs (transient phase). 2 requests per user and β = 1.5.

As expected, more improvement is achieved by CSVD and DISCS in the steady state phase. This is because in the steady state phase, there are more cached files in the clusters. Hence, more video files will be sent from the distributed cache, which increases the D2D channel utilization and speeds up the video transmission. However, good improvements are still achieved by both algorithms over SVD in the transient phase.

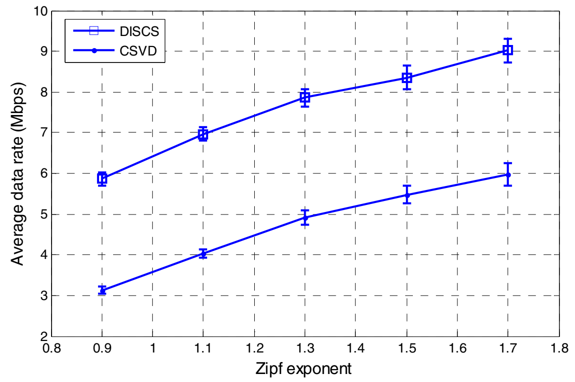

Fig. 8. Average data rate per user versus Zipf exponent (steady state phase). 500 UEs, 2 requests per user.

The Zipf distribution has one parameter, namely the Zipf exponent. This exponent controls the relative popularity of files. When the Zipf exponent increases, the content reuse increases. This is because higher β means that the popularity of the first files in the list will increase, and they will be requested more often. The impact of the Zipf distribution exponent on the performance of the CSVD and DISCS is shown in Fig. 8. As can be seen, the average data rate increases by increasing the Zipf exponent. As the number of popular files increases, the content reuse increases because more files will be cached and delivered later from the distributed cache rather than from the BS. This speeds up the transmission process and increases the average data rate. The increase in the average data rate will eventually slow down. This is because in our scenario, each UE requests only two video files. Increasing the number of requests made by each UE further increases content reuse and improves the average data rate. This is because cached files will be further used by the later requests.

VI. Conclusion

Wireless video accounted for more than half of the total data traffic in cellular networks in 2015, and this is expected to further increase in the upcoming years. This made it challenging for cellular networks operators who are already struggling to cope up with the increasing demands for higher data rates. As such, new techniques are needed to help serving the video traffic that is becoming the majority of the data traffic. In our previous work, we proposed the Cached and Segmented Video Download (CSVD) algorithm to improve the throughput of video transmission in cellular networks. The algorithm caches video segments in selected user equipment (UEs) in the network, and employs Device-to-Device (D2D) communication between UEs in cellular networks to exchange video segments.

In this work, we propose an improved algorithm, namely, the DIStributed, Cached, and Segmented video download (DISCS). In DISCS, the pieces of a video file are distributed over multiple Storage Members (SMs) to be forwarded to the requesting UE. This provides further parallelism when transmitting video files and further load balancing among SMs, which speeds up the transmission process. We use the Discrete Event System Specification (DEVS) formalism to model an LTE-A cellular network that implements DISCS. The model is used to study the performance improvement achieved by DISCS over CSVD in terms of cell's aggregate video transmission rate as well as average data rate per user. Simulation results show that DISCS achieves significant improvements over the CSVD.

References

[1] ICT Data and Statistics Division, Telecommunication Development Bureau, ITU. "ICT facts and figures." Feb. 2016 [Online]. Available: https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf.

[2] Cisco. "Cisco visual networking index: global mobile data traffic forecast update." Feb. 2016 [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

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

[4] 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, Nov. 2014.

[5] 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.

[6] 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, Dec. 2009.

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

[8] 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.

[9] A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman, "Cached and segmented video download for wireless video transmission," *in proc. ANSS*, 2016, pp. 1-8.

[10] 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, Jul. 2014.

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

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

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

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

[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] M. A. Maddah-Ali and U. Niesen, "Decentralized caching attains orderoptimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029-1040, Aug. 2015.

[17] Y. Zhang, L. Song, W. Saad, Z. Dawy, and Z. Han, "Contract-based incentive mechanisms for device-to-device communications in cellular networks," *IEEE JSAC,* vol. 33, no. 10, pp. 1-12, Oct. 2015.

[18] L. Gao, J. Huang, Y. Chen, and B. Shou, "Cooperative spectrum sharing: a contract-based approach," *IEEE Transactions on Mobile Computing*, vol. 13, no. 1, pp. 174–187, Jan. 2014.

[19] 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, Oct. 2014.

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

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

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

[23] V. Siris and D. Dimopoulos, "Multi-source mobile video streaming with proactive caching and D2D communication ," *in IEEE WoWMoM*, 2015, pp. 1-6.

[24] 3GPP TR36.942, "Evolved universal terrestrial radio access; RF system scenarios," Dec. 2015.

[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 measuremen*t, 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, Aug. 2014.