Joint Content-Mobility Priority Modeling for Cached Content Selection in D2D Networks

Vinicius Silva, Vinícius Mota, Daniel Macedo, Marcelo Dias de Amorim

Joint Content-Mobility Priority Modeling for Cached Content Selection in D2D Networks*

Vinicius F. Silva Vinícius F. S. Mota Daniel F. Macedo
Marcelo Dias de Amorim

Abstract

One key component for efficient opportunistic device-to-device (D2D) deployment is cache management. It determines which content to store opportunistic D2D communications. Existing solutions focus on the nature of content or mobility attributes, but most of them neglect their joint influence. Moreover, most solutions rely on a preloading phase, filling caches with content that the respective users may not consume, but that may be of interest to other nodes, and increasing traffic overhead in the core network. Further, a popular file may be a lousy candidate for opportunistic D2D because contact opportunities may not provide enough transfer capacity. To solve this issue, we propose a model that computes priority values based on both content and mobility attributes. Our approach considers only files that users have consumed, therefore eliminating a preloading phase. Using real-world and synthetic mobility traces, we compare our solution with Least Recently Stored replacement, as well as a state-of-the-art approach that also considers content and mobility attributes. Results show an increase in the global cache hit rate of almost 80% in scenarios that offer many files, and of around 420% in scenarios with a few users. The priority model generates 90% lower overhead in terms of the control bytes. We also apply our solution in a chunk-based adaptive video streaming application. We observe that our solution leads to higher video delivery ratios when compared to the baselines.

1 Introduction

In the context of opportunistic device-to-device (D2D) communications, it is crucial to adopt a cache management scheme that reacts well to the dynamics of the network, and that raises the probability that nodes both *find* and *retrieve* content in the neighborhood. Several works in the literature propose interesting methods for cache management but seldom consider content and mobility attributes at the same time [3, 7, 17]. As a matter of fact, it is not enough to determine the most popular files to cache if contacts among nodes do not have enough time to transfer these files when nodes meet. Moreover, most

*This work received funds from the following Brazilian government agencies: *Coordination for the Improvement of Higher Education Personnel* (CAPES), *National Council for Scientific and Technological Development* (CNPq), *Foundation for Research of the State of Espírito Santo* (Fapes), and *Foundation for Research of the State of Minas Gerais* (Fapemig).

solutions also rely on a preloading phase where a device's cache receives content that the device may not consume, but its neighbors might. Such a strategy results in traffic overhead in the core network [3, 7, 17].

In this paper, we investigate several tradeoffs between file popularity, file size, and contact time, so devices do not waste transfer opportunities. We propose a model that computes priority values to files that a user has consumed, giving higher values to those of small size and high popularity, since their probability of successful transmission through opportunistic D2D communications is higher, in comparison with large, unpopular files. The novelty of our proposal is the union of content and mobility attributes to define priority levels on a per-file basis.

Figure 1 illustrates the three steps of our prioritization scheme. The first step occurs when the device has files *A* and *B* in the cache, and downloads file *C* that it cannot store due to lack of space. In this case, the device proceeds to the second step, where it runs the model to assign priorities to all files, including *C*. In this example, file *A* receives the lowest priority, since it is the least popular of all three, despite its median size in comparison to the other two files. On the other side, file *C* receives a higher priority in comparison with file *A* because it has higher popularity, despite being bigger than file *A*. File *B* receives the highest priority since it is the smallest of all three and has the highest popularity.

After calculating the priorities, the device then goes to the final step, where it evaluates each file from the most to the least priority. In our example, the device ranks files as B-C-A; therefore, it stores B and C in the cache, and removes A due to the lack of space.

In practice, our priority model combines two independent evaluations. The first one focuses on mobility in terms of contact time, to define a file size limit that a device should transmit successfully through opportunistic D2D communications. On the one hand, files that are larger than such a limit receive smaller scores. On the other hand, the second evaluation focuses on the popularity of a file and its respective size, giving higher scores to popular and small files.

The performance of the proposed model is evaluated through simulations using real-world and synthetic mobility traces, as well as real metadata collected from a video platform to generate file requests. We compare the proposed model with a classical technique that replaces Least Recently Stored files when a device cache is full, as well as a state-of-the-art approach that uses content and mobility attributes to select the most popular content in the cache. The proposed priority model increases the global cache hit rate in almost 80% for scenarios with a large number of files, and by up to 420% for scenarios with few users. Compared to the state-of-the-art solution, the priority model reduces by 80% the overhead of control messages and by 90% the overhead of control bytes. We also evaluate the impact of each component of the model on the hit rate, and observe that each variant of the model offers different improvements depending on the characteristics of the scenario. Finally, we apply the priority model in a chunk-based adaptive video streaming application, once again demonstrating the superiority of the model in comparison with the baselines.

The rest of this paper is organized as follows. Section 2 presents the network model as well as the role of each entity of the network. The computation of the priority model is presented in Section 3. Section 4 describes the simulation setup, while the

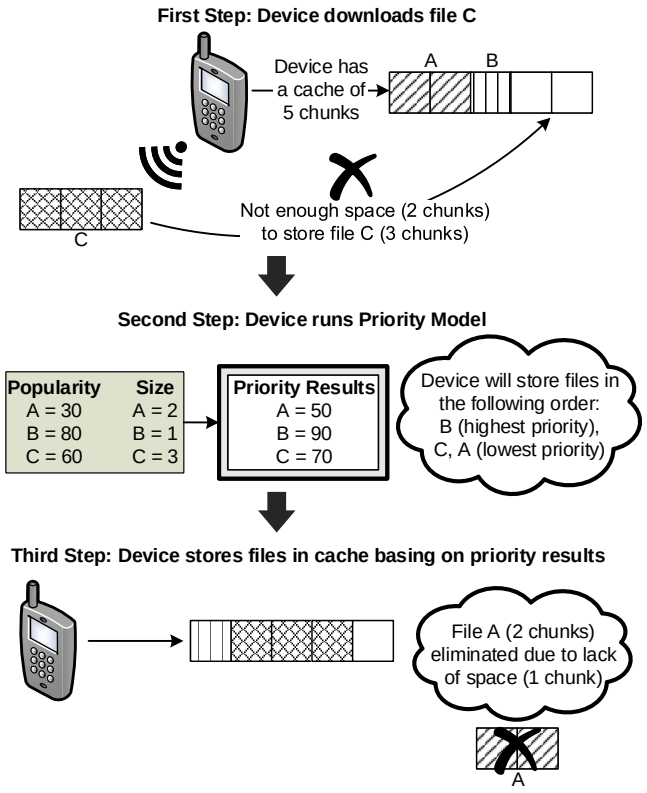

Figure 1: Use case of the proposed priority model. For the sake of illustration, we consider the priority and popularity of a file in a scale between 0 and 100, and the file size in terms of chunks with fixed length

the simulation results are presented in Section 5. We postpone the related work to Section 6, so that the reader gathers enough material to better understand the originality of our work. Finally, Section 7 concludes the paper and proposes ideas for future work.

2 Network Model

We model the network as a set $D = \{d_1, d_2, ..., d_{|D|}\}$ of mobile devices, and we denote d_i as the i-th device in D. Each device has a cache of fixed size of $C_{\max}$ bytes. We consider that a content server (CS) has the file set $F = \{f_1, f_2, ..., f_{|F|}\}$ out of which nodes request files. Every file is an indivisible data unit of varied size, which does not change over time. We denote f_j as the j-th file in the F set, with a size s_j ranging between 1 and $C_{\max}$ bytes. Figure 2 depicts the network model.

Both the CS and the devices monitor the file requests in the network in discrete monitoring time intervals. Each device also monitors the D2D contact duration with its

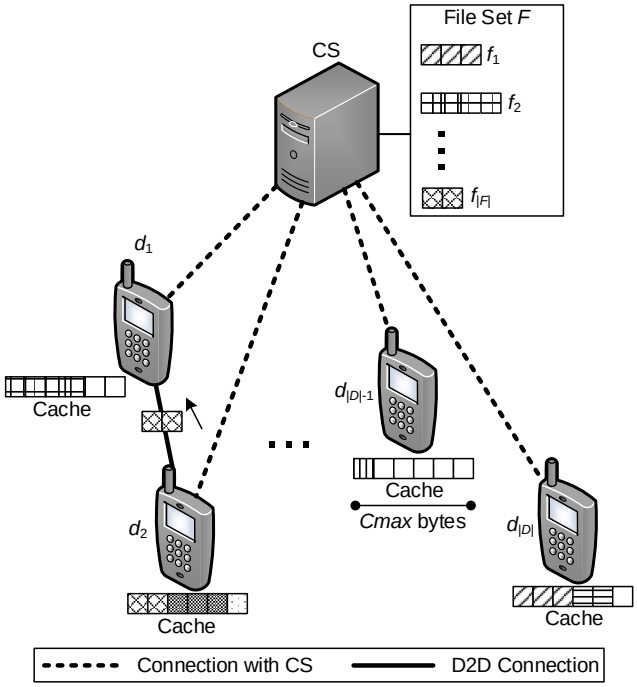

Figure 2: Network model

neighbors. For each request of file f_j, the requesting device first searches for it in the neighborhood, during some predefined time interval. If the device finds f_j at any of its neighbors, it triggers opportunistic D2D sharing. If the device cannot find f_j after the predefined interval, it forwards the request to the CS.

3 Building the Priority Model

The priority for a file f_j at the cache of device d_i, here denoted ζ_{f_j,d_i}, is a real number with an undefined maximum value. We compute it by multiplying two main terms, which we call *Transmission Potential* (TP) and *Dissemination Potential* (DP), each one focusing on a different aspect:

$$\zeta_{f_j,d_i} = \text{TP} \times \text{DP}. \tag{1}$$

The first term, TP, focuses on the mobility aspect, by relating the size s_j of file f_j to a size threshold λ_{d_i}, which relies on d_i's contact pattern. This threshold is the maximum file size that d_i would be able to transmit successfully through opportunistic D2D communications. Therefore, files that are larger than λ_{d_i} receive lower values following an exponential factor. We show how we obtain λ_{d_i} in detail in Section 3.1. For the time being, we compute the transmission potential as:

$$\mathrm{TP} = \frac{\lambda_{d_i}}{s_j}. \tag{2}$$

The second term, DP, on the other hand, focuses on the popularity aspect. We denote the popularity of file f_j for device d_i as p_{f_j,d_i} (we derive and explain this variable, step by step in Section 3.2). As stated before, devices are less likely to share large, unpopular files through opportunistic D2D communications, then we should give higher priorities to files that are short and popular at the same time. To make this possible, we weight the file popularity with its respective size:

$$\mathrm{DP} = \frac{p_{f_j,d_i}}{\log_{C_{\max}} s_j}. \tag{3}$$

Note that for the largest files ($s_j = C_{\max}$ and $\log_{C_{\max}} s_j = 1$), the resulting value only depends on the popularity p_{f_j,d_i}.

In the following, we present a detailed description on how to derive λ_{d_i} and p_{f_j,d_i}.

3.1 Deriving a Content Size Threshold for each Device

The value of a size threshold at device d_i (λ_{d_i}) relies on two main attributes: the exponentially weighted moving average (EWMA) contact time between d_i and its neighbors, which we denote as $\overline{t_{d_i}}$, and the device's achievable throughput through opportunistic D2D communications, which we denote as ε. By multiplying these two values, we define the heaviest file that d_i would be able to transmit successfully to its neighbors:

$$\lambda_{d_i} = \overline{t_{d_i}} \times \varepsilon. \tag{4}$$

Device d_i updates $\overline{t_{d_i}}$ every time it disconnects from a neighbor. The new value of $\overline{t_{d_i}}$ is a EWMA, weighted with α, that depends on its previous value and on the duration of the latest contact t_{last,d_i}:

$$\overline{t_{d_i}} = \alpha \times t_{\mathrm{last},d_i} + (1-\alpha) \times \overline{t_{d_i}}. \tag{5}$$

The longer the EWMA contact time ($\overline{t_{d_i}}$), the larger the file that this device should be able to share successfully with its neighbors through opportunistic D2D communications.

3.2 Deriving the Popularity of a File in a Device's Communication Range

The popularity of file f_j at device d_i, denoted as p_{f_j,d_i}, is a real value that relies on the number of requests processed in the whole network, as well as those that d_i and its neighbors processed through D2D communications. We compute the EWMA global popularity $\overline{p_{f_j}}$ for file f_j in discrete monitoring time intervals. We define it as the percentage of devices among D that requested f_j in the latest monitoring interval, denoted as p_{last,f_j}, weighted with its previous value (using parameter β):

$$\overline{p_{f_j}} = \beta \times p_{\text{last},f_j} + (1-\beta) \times \overline{p_{f_j}}. \quad (6)$$

It is important to note that the global popularity computed in Equation 6 does not consider the geographic distribution of file requests. Although we find essential to express a global monitoring that detects file request trends that will raise the file's priority, it is also important to consider the device's point of view in an isolated form, i.e., how popular each file is in the device's communication range. In this way, we avoid computing high priorities to files that are not so popular in one or more devices, at the same time not denying the trends in the whole network.

To take that into account, we weight the global popularity with a local metric, which is the EWMA number of requests that device d_i and its neighbors processed through D2D for file f_j; we denote this value as $\overline{r_{d_i,f_j}}$. We update $\overline{r_{d_i,f_j}}$ in fixed monitoring intervals, weighting by γ the latest number of requests computed for f_j (we denote this value as r_{last,d_i,f_j}), and $(1-\gamma)$ the current EWMA number of requests (Equation 7):

$$\overline{r_{d_i,f_j}} = \gamma \times r_{\text{last},d_i,f_j} + (1-\gamma) \times \overline{r_{d_i,f_j}}. \quad (7)$$

By combining the two popularity metrics from Equations 6 and 7, we obtain the popularity for a file f_j in device d_i:

$$p_{f_j,d_i} = \overline{p_{f_j}} \times \overline{r_{d_i,f_j}}. \quad (8)$$

4 Simulation Setup

This section describes the simulation setup. We used the *ONE* opportunistic networking simulator for all experiments [5].

The following subsections explain how we define and generate file requests, as well as mobility patterns during the simulations, through different content and mobility datasets (Section 4.1). Next, we list and discuss our simulation parameters (Section 4.2).

4.1 Content and Mobility Datasets

We use metadata from YouTube[1] to obtain realistic content characteristics for our simulations. The raw dataset contains a daily recording from up to 200 videos classified by YouTube servers as "trending", for a set of countries that include the USA, Great Britain, Germany, Canada, France, Russia, Mexico, South Korea, Japan, and India, between November 2017 and June 2018. For each day, the dataset presents each trending video with the number of requests received by YouTube's servers during that day. For our experiments, we considered one day of measurements from French YouTube's dataset, in November 2017. We consider the trending video set as the catalog of files that users may request. We use the number of requests of each video to define how likely devices request it during the simulations.

[1] https://www.kaggle.com/datasnaek/youtube-new

Considering more realistic scenarios, there is also the impact of video streaming protocols, such as the Dynamic Adaptive Streaming over HTTP (DASH) [15], which divides the video in chunks with variable video bitrate. In these scenarios, it is important to note that our priority model provides a joint adjustment on the video resolution, since a real-time selection of video chunks would be done in each device cache, keeping chunks that each device will be able to share in its respective D2D neighborhood.

The raw dataset described above does not provide the real size (in bytes) of each video or its chunks. Therefore, for simplicity, in most part of our experiments we consider each video consisting of only one chunk, as well as the dropping of a complete video/file from cache when replacing it for a new file. We associate the size of each file to a random value according to a Weibull distribution, based on previous studies of Pang *et al.* [10]. Then, we calculate the percentage of views for each video and apply them in the simulator in the form of a probability distribution. Once we generate a request, we assign it to a device according to a uniform distribution.

Although we consider a "video download only" behavior in all experiments, our solution applies to other types of content. It is important to note that we adopt a generic definition of a file in our network model (see Section 2 for more details). Therefore, for other types of content, we expect similar performance results.

For the mobility part, we applied two contact traces collected from real devices. The first one is *Infocom06* [12], which comprises 98 users (78 mobile + 20 static) carrying devices around office, conference, and city environments. The second trace is *Reality Mining* [2], which gathers, among several types of events, proximity information from 97 subjects at MIT.

Although contact traces represent a real scenario observed by real devices, these traces have two drawbacks: i) lack of details on the users' mobility profile, an information that would help to understand the impact of user mobility on the performance of our solution; and ii) the impossibility of analyzing the scalability of our solution, since contact traces have fixed number of users.

To overcome the aforementioned drawbacks, we also evaluate our solution using a synthetic mobility model called *Shortest Path Map Based Movement*. In this model, users start in a random uniform position in the map. During the simulation, they choose a destination in a map and take the shortest path to it, with random uniform velocities as well as random uniform wait times between two movements.

4.2 Simulation Parameters

Table 1 lists the parameters that we use in our experiments. We explain some of them below:

Simulation Time. We set up the time of 86,400 seconds (one day), since the portion that we consider in the YouTube Trending dataset comprises data for a whole day (see Section 4.1 for more details).

Number of Files. Considering that the YouTube Trending dataset has 200 files for each day of measurements, for numbers of files inferior to 200, we consider only the top

Table 1: Simulation parameters

Parameter	Experiments varying the Number of Files	Experiments varying the Number of Users
	General Parameters	
Number of Runs	10, with confidence interval of 95% according to Student's t distribution	
Simulation Time	86,400 seconds	
Number of Files	10,20,50,100,150,200 98 (*Infocom06*)	200
Number of Users	97 (*Reality Mining*) 100 (Synthetic)	10,20,50,100,200 (Synthetic)
Transmission Speed	5 MB/s	
Device Cache Capacity	193,277,600 bytes	
Request Generation Interval	1,000-85,400 seconds	
File Request Frequency	One every 8.44 seconds	
D2D File Search Delay	50 seconds	
File Consumption Time	35.4 seconds	
	Synthetic Mobility Model Specific Parameters	
Map Dimensions	4,500 x 3,400 meters (Helsinki City Map)	
Device Movement Speed	0.5-1.5 meters/second (random uniform values)	
Wait Time between Two Movements	0-120 seconds (random uniform values)	
Communication Range	15 meters	
	Priority Model Specific Parameters	
Popularity Monitoring Interval	10,000 seconds	
EWMA Weights (α-β-γ)	0.1	

first lines, and calculate the percentage of views only considering this line subset (see Section 4.1 for more details).

Transmission Speed and Communication Range. We chose Wi-Fi Direct as the transmission technology, considering its simplicity to use and the support on almost all mobile devices nowadays. We retrieved both the transmission speed and the communication range from recent studies made by Reis *et al.*, where the authors evaluated the performance of D2D with real mobile devices using Wi-Fi Direct, in different scenarios [11]. The authors measured, among many other performance metrics, the average throughput for the transmission of files of different sizes. We chose the highest dis-

tance considered by the authors as the devices' communication range (15 meters), and the average throughput for the largest file as the devices' transmission speed (5 MB/s). In that way, our experiments consider a realistic scenario with limited contacts and transmission performance. Finally, it is important to highlight that the communication range of 15 meters is only valid for the synthetic mobility model, since the contact traces already consider the variable range of real devices that collected the data. It is also important to note that the transmission speed does not vary in our experiments, due to limitations inherent to our simulation scenario. Therefore, the transmission capacity of a node with its neighbors depends on the duration of D2D contact. Thus, the longer the contact time, the higher the transmission capacity.

Device Cache Capacity. To ensure that all files fit in the device cache, we define the cache capacity as the size of the largest file retrieved from the YouTube Trending dataset, which in our case was 184.32 MB (193,277,600 bytes).

File Request Frequency. In a first moment, we consider a scenario where each user downloads one copy of all available content. However, during our experiments we vary the number of files and users, comparing the global hit rate between different parameter combinations. Aiming to do fair comparisons, we set the same file request frequency for all simulations, considering a scenario with 100 users and 100 different files, which would generate 10,000 file requests. We do not consider the first and last 1,000 seconds of the simulation time for warmup purposes either, and we distribute the requests in the remaining time. Therefore, to generate 10,000 file requests, we must generate one request every 8.44 seconds.

File Consumption Time. Since all our experiments consider only video content, we chose the file consumption time basing on Krishnan and Sitaraman's study, where they evaluate the behavior of video users, basing on traces that include 23 million views from 6.7 million viewers. In their results, they state that the user has a median play time of 35.4 seconds, before abandoning the video [6]. This is a behavior that the authors call *video surfing*, where the viewer watches several videos before deciding which ones he will watch completely. For convention purposes, we assume this viewer behavior for all experiments.

5 Simulation Results

This section presents the simulation results. The experiments are organized in four parts: **(i)** Tuning of important parameters for the priority model (Section 5.1); **(ii)** Comparison of the priority model with a classical technique that considers neither content nor mobility attributes, and with a state-of-the-art approach that considers them (Section 5.2). In this part, we evaluate the global cache hit rate by varying the *D2D File Search Delay* (Section 5.2.1), as well as the number of files and users (Section 5.2.2). We finish this part evaluating the overhead by also varying the number of files and users (Section 5.2.3); **(iii)** Investigation of the impact of each component of the priority

model on the hit rate, to demonstrate the importance of a model with all the attributes that we described in Section 3 (Section 5.3); **(iv)** Evaluation of the priority model with a chunk-based adaptive video streaming application (Section 5.4). We compare it with the baselines by observing the global chunk ratio and the chunk quality rate, when we vary the number of video files and users.

The performance metrics considered in this section are:

- *Global cache hit rate*. The percentage of files that devices have requested *and* received through opportunistic D2D communications or found in their cache (files that devices have downloaded before);
- *Number of files consumed per minute* by all devices, through each option (D2D, Server and Cache);
- *Control message overhead*. The ratio between the number of control messages sent in the network and the number of files consumed by all devices;
- *Control bytes overhead*. The ratio between the number of bytes of control messages sent by all devices in the network, and the total of bytes transmitted (control + files).
- *Global chunk ratio*. Similar to the *Global cache hit rate*, described above. It is the percentage of chunks that devices have requested *and* received through opportunistic D2D communications or found in their cache (chunks that devices have downloaded before).
- *Chunk quality rate*. The percentage of chunks of each quality consumed by all devices through opportunistic D2D communications and the cache, independently of which video each chunk belongs to.

5.1 Parameter Tuning

In this part of the experiments, we evaluate the following simulation parameters: the popularity monitoring interval and the EWMA Weights. In these experiments, we consider a scenario with 100 users and 100 different files, and apply the synthetic mobility model.

5.1.1 Popularity Monitoring Interval

The popularity monitoring interval indicates how frequently the global and local popularity metrics, in terms of the EWMAs (see Section 3.2 for more details), for each file, receive an update cycle.

We evaluated monitoring intervals between 5,000 seconds (around 1.39 hours) and 50,000 seconds (around 13.89 hours) with steps of 5,000 seconds, taking into account the total simulation time of 86,400 seconds (24 hours). According to results that we present in Figure 3, the hit rate reaches a peak with a monitoring interval of 2.78 hours, and has a gradual decrease for higher values. Basing on this behavior, we chose 2.78 hours as the monitoring interval for all experiments.

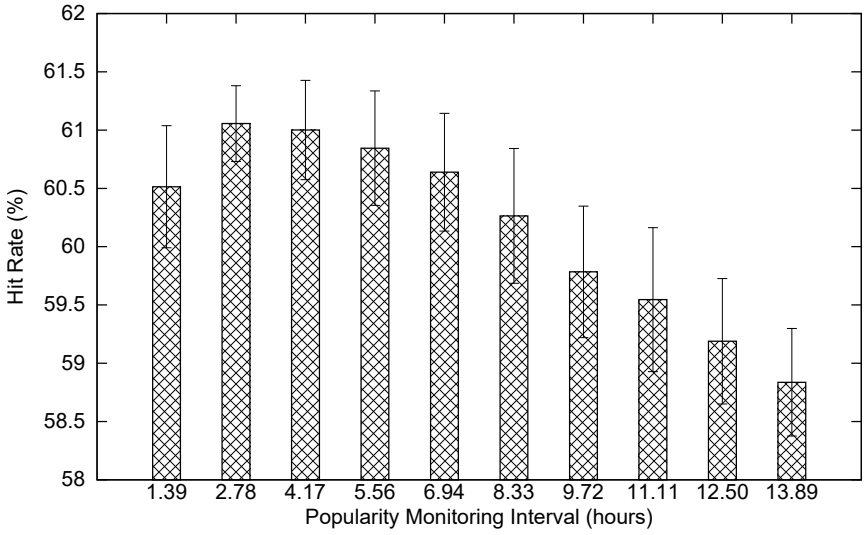

Figure 3: Varying the popularity monitoring interval

Even though the highest hit rate obtained (for a monitoring interval of 2.78 hours) was 3.77% superior to the smallest one (for a monitoring interval of 13.89 hours), we need to consider that our simulations are limited to one day, as well as fixed content and mobility datasets. Moreover, the results presented in Figure 3 show that, for each scenario, the popularity monitoring interval has a respective optimal value, therefore demanding a real-time adjustment of this attribute. We leave such an adjustment as future work.

5.1.2 EWMA Weights

The EWMA weights (α, β and γ) define how relevant the last average values are, in comparison with the last observed values. As described in Section 3, we employ EWMAs to monitor the contact time between devices, as well as the global and local popularity of each file. In our experiments, for convenience and simplification purposes, we consider the same value for α, β, and γ.

We evaluated values between 0.1 and 0.9 with steps of 0.1. As shown in Figure 4, the cache hit rate has a gradual increase with the decrease of the weights. For our experiments, we chose the value that got the highest global cache hit rate, which was 0.1 in our case.

5.2 Comparison with the Baselines

In this part, we evaluate the priority model in comparison with two different baselines. The first baseline is a greedy, classical technique that we call LRS (*Least Recently Stored*). It works as follows: when the device downloads a new file, it tries to

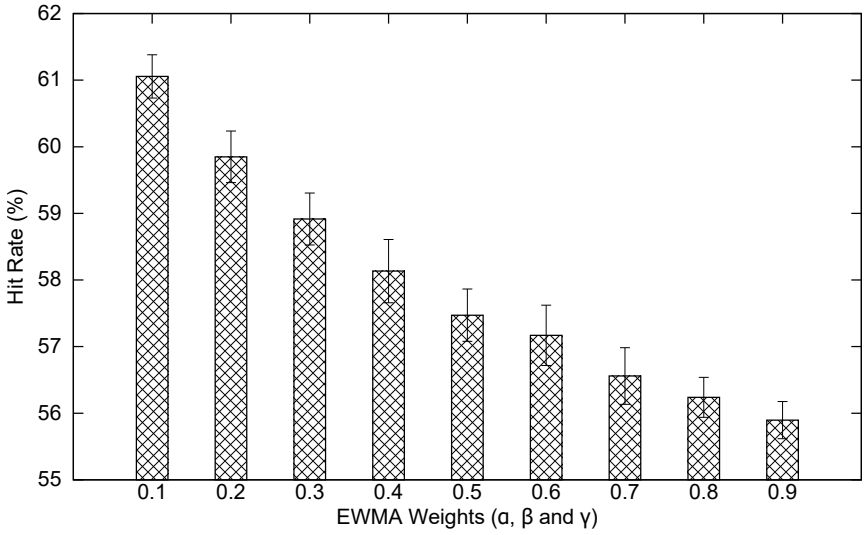

Figure 4: Varying weights for the EWMAs

store such a file in its cache. If the available space is not enough, the device removes the file with the oldest storage time. The user repeats this step until the cleared space is enough for the new file. Unlike our solution, LRS considers neither the nature of content, in terms of size and popularity, nor the nature of contacts.

The second baseline strategy is a state-of-the-art approach, called SACC (*Social-Aware Cooperative Caching*) [18]. SACC employs a cache replacement mechanism that computes in real-time a popularity metric for each file. This metric considers social variables such as the social tie strength, trustworthiness, and encounter probability, between users that provide a file, and users without the file. SACC replaces files in the cache based on the popularity. Least popular files are dropped when the cache is overloaded. In our experiments, we consider that all devices always have the same file interests and then belong to the same community. Therefore, we define a single community with all the devices [18].

We evaluate our priority model, LRS, and SACC in terms of the global cache hit rate, as well as the overhead under different scenarios.

5.2.1 Global Cache Hit Rate: Varying the D2D File Search Delay

The D2D file search delay indicates how long the device searches for the desired file in the D2D neighborhood before redirecting the file request to the content server. For the file search delay, we refer again to the study of Krishnan and Sitaraman, who state that users take up to 50 seconds to give up downloading a video [6]. Basing on such a limit, we evaluated the file search delay by measuring the global cache hit rate, for search delay values between 0.5 and 60 seconds, with steps of 0.5 seconds, by applying *Infocom06* and *Reality Mining* contact traces, as well as the synthetic mobility model

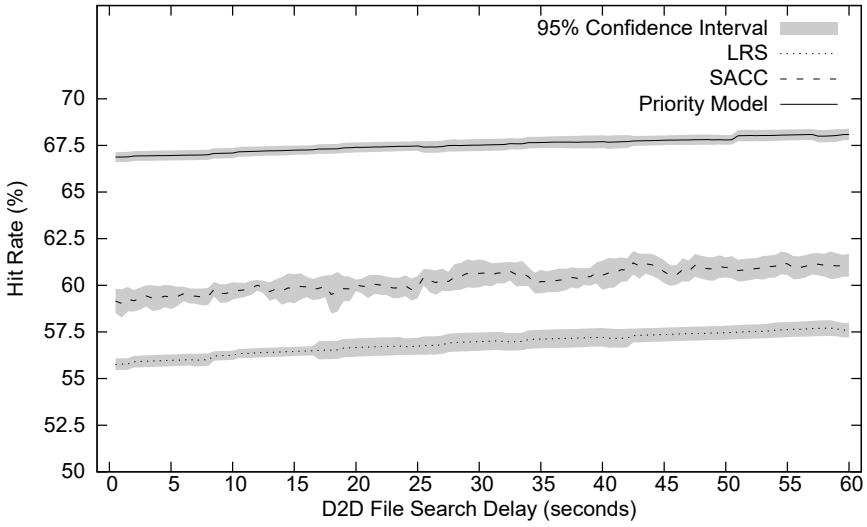

Figure 5: Varying the D2D file search delay: contact trace *Infocom06*

with 100 users, in a scenario with 100 different files.

According to results presented in Figures 5 to 7, users with shorter and longer tolerances to search delays have similar performance, with a difference on the hit rate of less than 1%. Nevertheless, we also observe higher hit rates with longer search delays, for all scenarios and solutions. This is an expected behavior, since more patient users tend to find the desired file in the D2D neighborhood with more frequency. Another interesting point to observe is the fact that, for all scenarios, the priority model outperforms the baselines, besides presenting a more predictable behavior with shorter confidence intervals.

Since we intend to observe the optimal performance of our solution, and to make fair and more detailed comparisons with the baselines between different scenarios, for the following experiments in this section we normalize users with a *patient* profile, with a search delay of 50 seconds.

5.2.2 Global Cache Hit Rate: Varying the Number of Files and Users

Figures 8 to 10 show the global cache hit rate when we vary the number of files, with the application of each contact trace as well as the synthetic mobility model.

For all figures and scenarios above, for our solution, LRS and SACC, we observe a gradual decrease in the hit rate when the number of files in the network grows. This is an expected behavior, since a high number of different files reduces the probability of finding a specific file in the device's cache or the D2D neighborhood. In these experiments, the average number of neighbors was 1.63, 0.13 and 0.07, for contact traces *Infocom06*, *Reality Mining* and the synthetic mobility model, respectively.

Still regarding the decrease in the global cache hit rate, when the number of files

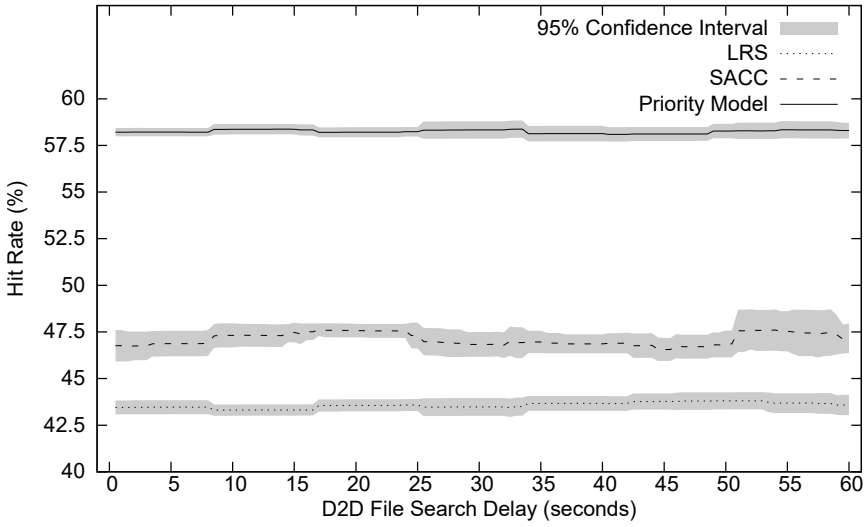

Figure 6: Varying the D2D file search delay: contact trace *Reality Mining*

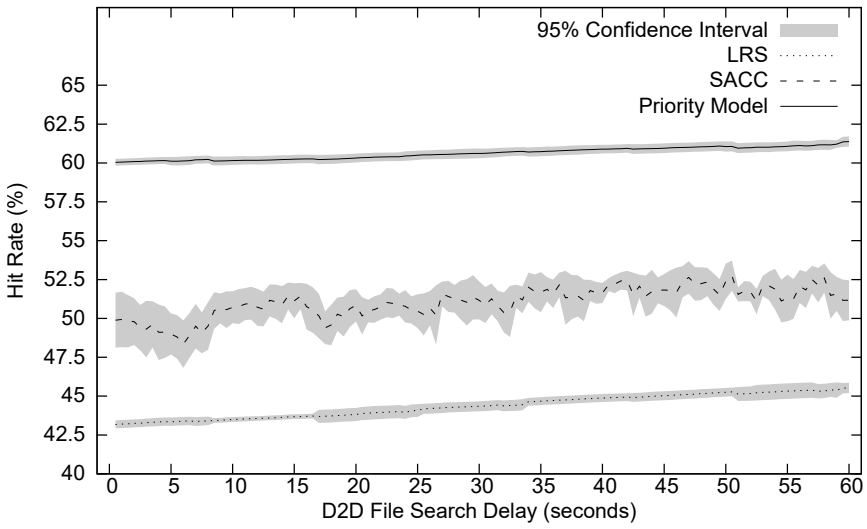

Figure 7: Varying the D2D file search delay: synthetic mobility model

varies from 150 to 200, the decrease in the hit rate is smaller than when varying from 100 to 150 for LRS, SACC, and the priority model. This suggests that scenarios with more files tend to cause smaller hit rates in the network, and the performance for all solutions would stabilize, not offering more significant improvements in the hit rate.

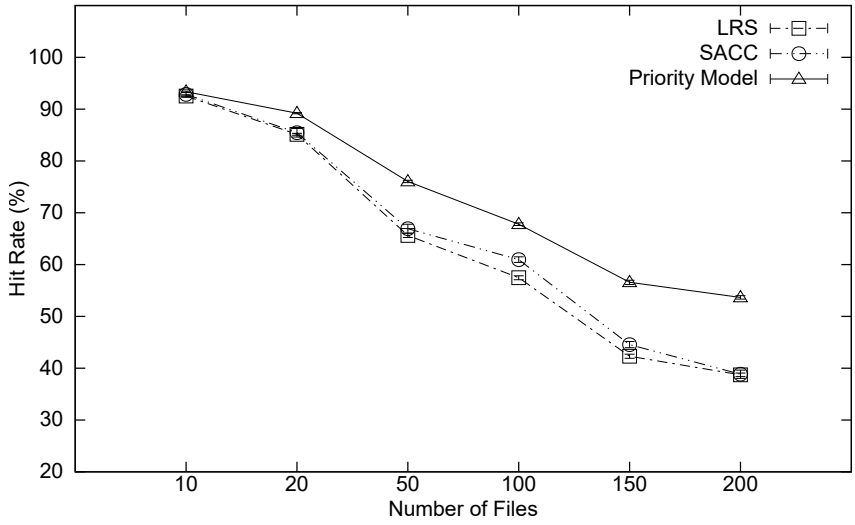

Figure 8: Varying the number of files: contact trace *Infocom06*

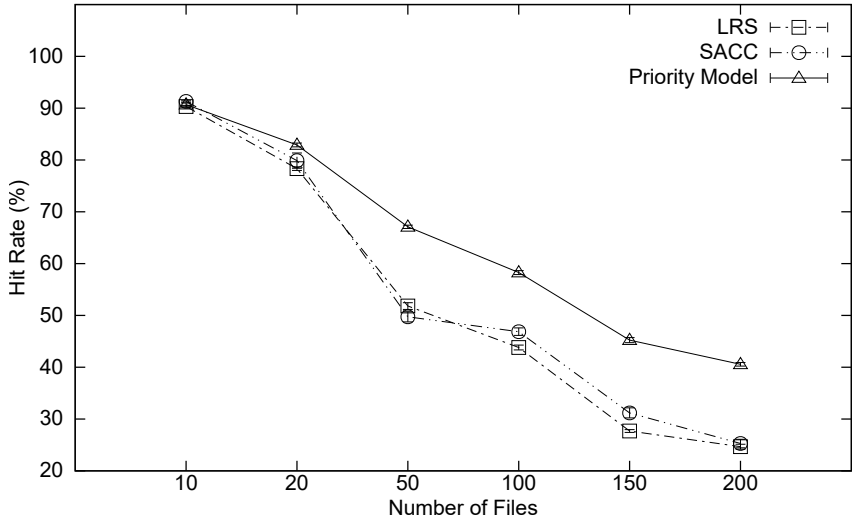

Figure 9: Varying the number of files: contact trace *Reality Mining*

We can also observe the highest differences between our solution and LRS in all scenarios with 200 files. In these cases, our solution outperforms LRS by approximately 38.6%, 64.48% and 76.56%, when applying contact traces *Infocom06* and *Reality Mining* and the synthetic mobility model, respectively. By comparing our solution

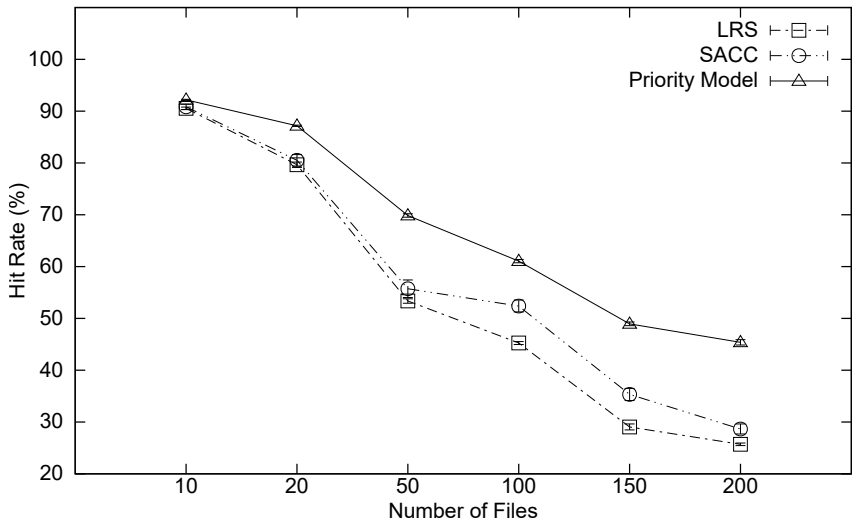

Figure 10: Varying the number of files: synthetic mobility model

with SACC, there is an improvement of 38.2%, 60.46% and 58.33% for the same respective scenarios. The superiority of our solution, in comparison to LRS and SACC, indicates the importance of computing priority values for file selection in scenarios with a large number of files.

For scenarios with 10 files, all solutions provide a similar hit rate. In these scenarios, in the beginning, all devices download most of the files through D2D or from the server; later, they download almost everything from the cache. Therefore, scenarios with low numbers of available files provide hit rates of above 90%.

Figure 11 shows the number of downloaded files in intervals of 30 minutes, for each download option (through D2D, from the server and from the cache). These values reflect one single run of 24 hours for our solution, in a scenario with 100 users, and applying the synthetic mobility model. In Figure 11, we can observe that after 5 hours of running time, all devices start to consume most of the files directly from their cache. For the sake of simplicity, we do not present results for LRS and SACC, since they presented the same file download behavior.

As described in Section 4, we also evaluate the effects of varying the number of users on the global cache hit rate. To do such an evaluation, we keep fixed the number of files in 200 and apply just the synthetic mobility model, due to the impossibility of doing the same with the application of contact traces. In these experiments, the average number of neighbors was 0.006, 0.015, 0.034, 0.07, and 0.139 for 10, 20, 50, 100 and 200 users, respectively.

Figure 12 shows the global cache hit rate. We observe a gradual decrease over the hit rate for the priority model when the number of users grows in the network. The same behavior could also be observed for the SACC mechanism, although with less intensity and not so predictable as our solution, due to the high variation observed

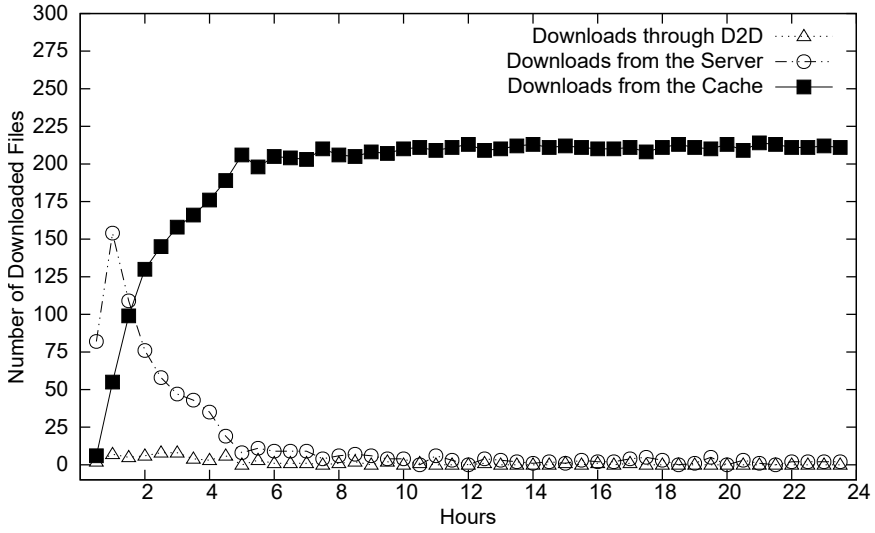

Figure 11: Number of downloaded files through D2D, from the server and the cache during one run for the proposed solution

through the confidence interval. This is also an expected behavior, since users have a limited communication range and are also limited to move in the streets of the map specified in our experiments, a fact that helps to restrain the average number of neighbors in the network. It is also worth noting that each device can communicate through D2D to only one device at a time. A limited number of neighbors reduces the number of potential D2D contacts and transmissions in the network, which in turn causes a "dispute" for the use of opportunistic D2D communications. Such a phenomenon reduces the probability of a device finding an available D2D neighbor, with the desired file. As a consequence, the effects of selecting files in the cache with the priority model and the SACC mechanism also reduce for scenarios with more users, since part of them are forced to request the desired file to the server.

Reversely, we observe that LRS had a slight increase over the global cache hit rate with the increase over the number of users. However, we can also observe that the hit rate starts to stabilize between scenarios with 50 and 200 users, which indicates an optimal performance and a subsequent decrease for scenarios with even more users, thus following the same behavior already observed for the priority model and the SACC mechanism.

Another interesting point is how our solution outperforms LRS and SACC for scenarios with 10 users. For that scenario, the hit rate of our solution outperformed LRS by 423.63% and SACC by 40.41%, demonstrating the importance of having such a cache selection technique when we have more isolated users. This was our case, since the synthetic mobility model leads to a normal distribution of users in the map, therefore reducing the occurrence of long-time D2D links.

We can also note that the superiority of our solution decreases as the number of

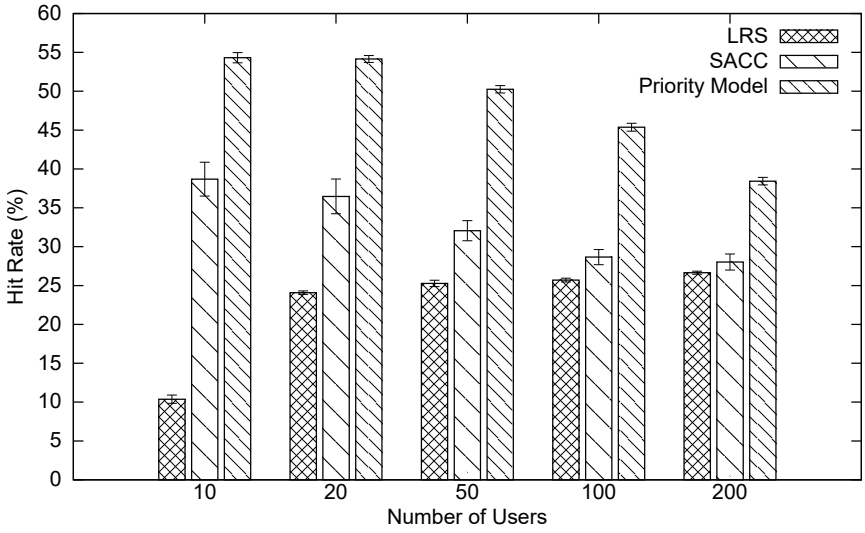

Figure 12: Varying the number of users: synthetic mobility model

users increases, attaining 44.22% over LRS and 37.1% over SACC for the scenario with 200 users. We can explain this phenomenon by the fact that each device selects content basing on their local observations; therefore, we expect differences in the cache of different devices. If we consider realistic scenarios with many users, where a device tends to have more neighbors, and a fixed number of files, the effects of selecting files in the cache is limited, since there is a higher probability of a device finding what it needs in its neighborhood.

We cannot make the same statement above for realistic scenarios where the number of users and the number of files vary at the same time. In this section, we also demonstrated the efficacy of our solution in scenarios with more files. Therefore, it is important to evaluate the tradeoff between this parameter and the number of users in the network, to determine in which cases our model provides better performance. A proper triggering of our model would also help reduce the control overhead in the network, as well as the energy consumption of each device, since the model would cease to run sometimes with no avail. Defining what we call here "operating regions" (when and how we should apply our model) is left as future work.

5.2.3 Overhead: Varying the Number of Files and Users

This section discusses the control message and the control bytes overhead of the priority model, in comparison with our baselines. We describe below how each solution behaves in terms of the control messages sent in the network.

LRS relies on the storage timestamp of each file that a device consumed. Therefore, there are no control messages for cache replacement. As a result, the overhead of LRS comprises only of file requests. Meanwhile, SACC requires to exchange control

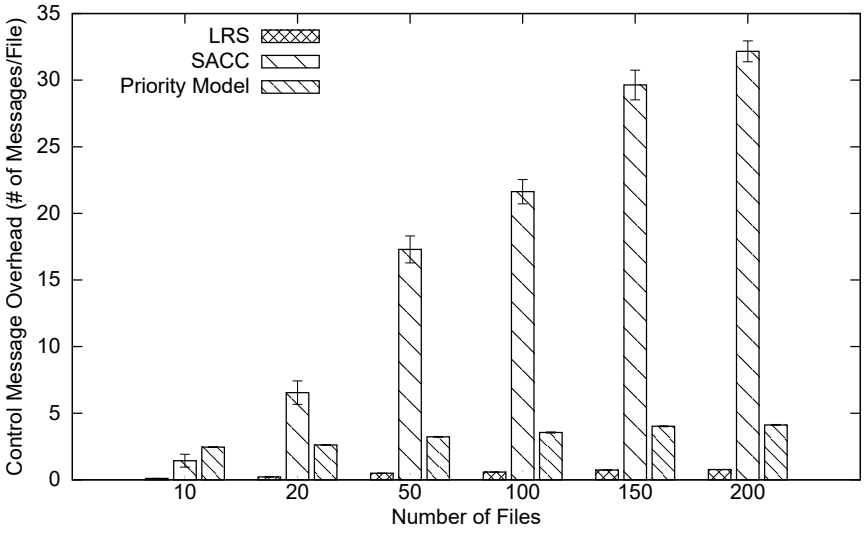

Figure 13: Control message overhead: Varying the number of files

messages with all devices and the server, to discover which device has each file to compute its popularity. Furthermore, SACC requires control messages to update the social metrics among devices. Similarly, our priority model exchanges control messages to monitor global and local file popularity, as well as to monitor contact time among users.

Figure 13 shows the control message overhead behavior when we vary the number of files, for scenarios with 100 users and applying the synthetic mobility model. For the sake of simplification, we omitted the results for *Infocom06* and *Reality Mining* contact traces since they provide similar behavior.

As we can see in Figure 13, LRS provides the lowest control message overhead between all solutions, computing an average of less than one control message per consumed file. Even with the absence of significant overhead levels, LRS shows poor performance in scenarios with a few users. In terms of the global cache hit rate, our solution outperforms LRS by at least 38.6% in all scenarios (see Section 5.2.2).

In the case of the SACC mechanism, the control message overhead has a linear growth, when we increase the number of files in the network. This is because a large variety of files reduces the chance of a device finding the desired file in the D2D neighborhood or its cache. We observed this behavior in Section 5.2.2, with a decrease in the global hit rate. Finding fewer files in the cache implies that new files arrive in a device with more frequency, overloading the cache more often and, therefore, demanding more cache replacement operations. As stated above, for each cache replacement procedure under the SACC mechanism, the monitoring server must exchange a broad set of control messages to update a set of social metrics between devices and to collect information about the files stored in each cache.

Unlike SACC, the priority model exhibits a nearly constant behavior when the number of files grows. This happens for two reasons. The first one is the constant moni-

toring costs, since, for each file download through D2D or from the server, the number of control message exchanges is the same, to update global and local popularity, as described in the beginning of this section. The same happens when a cache replacement occurs, where a device updates the global popularity of its files with just one control message exchange with the CS. The overhead due to contact time monitoring is constant, in the average measurements for each scenario presented in Figure 13, since the number of users and their mobility pattern is the same.

The second reason for an almost constant overhead, for the priority model, refers to the number of file requests generated in the network. It is worth noting that the file request frequency was the same for all experiments, as described in Section 4. If we consider an hypothetical scenario where the number of requests is higher, the overhead of our solution would present a linear grow, since devices always would send a fixed number of control messages in the D2D neighborhood and to the server, for each file consumption, to update their popularity monitoring metrics.

Although an increase in the number of files causes a decrease in the hit rate, as also observed in Section 5.2.2, the use of the priority model allows optimizing the set of files stored in the cache, reducing the number of cache replacements and also helping to stabilize the control message overhead of our solution.

Besides being 58.33% superior to SACC in terms of the global cache hit rate in the scenario with 200 files, as demonstrated in Section 5.2.2, the priority model provides a control message overhead 87.19% inferior to the value obtained with SACC, for the same scenario. If we consider the scenario of a dense network with many devices, with many contact events between them, many replacement operations tend to happen, and the popularity of each file needs to be updated frequently. As a consequence, many control message exchanges between devices and the monitoring server also happens in SACC. Moreover, many of the control messages tend to be big, due to the extended contact and file-sharing history of many devices, after some operating time.

To understand the impact of the number of users on the overhead, we vary the number of users, fixing 200 files, and applying the synthetic mobility model. Figure 14 shows the results.

As we can see, LRS presents a constant control message overhead. This is expected since there are no control message exchanges to replace files in the cache. Although the low and constant overhead, it is important to remember that LRS attains a hit rate of around 10%, for scenarios with 10 users, as demonstrated in Section 5.2.2. With the priority model, and with the cost of less than 2 control messages per downloaded file, such a hit rate is around four times bigger, as also demonstrated in Section 5.2.2. For SACC, although being almost three times bigger in comparison to LRS, it has the cost of around 4 control messages per downloaded file, therefore doubling the control message overhead in comparison to the priority model, which provided a higher hit rate as mentioned above.

Considering scenarios with more than 20 users, we can observe a linear behavior for both SACC and the priority model. In the case of SACC, such growth is due to the constant increase in the monitoring costs, which demands the collection of monitoring data from all devices during the cache replacement procedure. Therefore, with the growth in the number of users, more control messages need to be sent between them and the monitoring server.

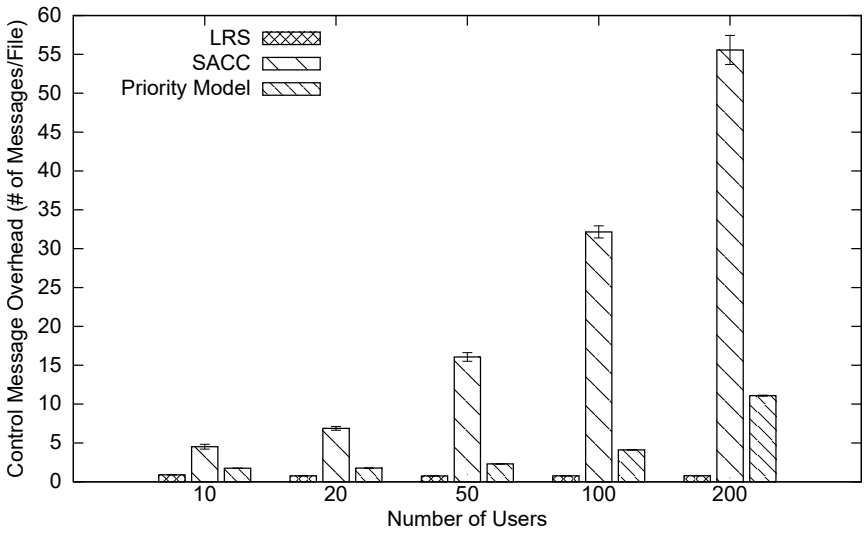

Figure 14: Control message overhead: Varying the number of users

In the case of the priority model, the growth in the number of contacts between devices explains the growth in the control message overhead. The priority model demands the exchange of control messages to monitor the contact time. Although the growth in the number of contacts in the network, the control message overhead of priority model was around 80% inferior in comparison with SACC, for the scenario with 200 users, at the same time providing superiority in the hit rate of over 37% over LRS and SACC, as demonstrated in results from Section 5.2.2.

Figure 15 shows the control bytes overhead when we vary the number of files. As we can observe, LRS has constant control byte overhead. Indeed, LRS control byte overhead comprises only of file requests, as stated before.

By comparing the priority model with SACC, the control bytes overhead was almost 96% smaller. This is due to the smaller monitoring costs for the priority model, besides being constant in the most part, as described before.

Another interesting point to observe is that all solutions maintained almost the same control bytes overhead for all scenarios. We explain this behavior through the decrease in the global cache hit rate, with the increase in the number of files for all solutions, a behavior that we demonstrated in Section 5.2.2. Such a decrease causes a growth in the number of bytes sent between devices and the server, in terms of file requests, monitoring information, or files, a fact that helps to stabilize the control bytes overhead between all scenarios. It is also important noting that content replacement procedures are running in real-time, which aim to reduce the number of file exchanges between devices and the server. Such a reduction may not be effective, as demonstrated in Section 5.2.2 with the reduction in the effects of selecting content in the cache, when the number of users grows. Moreover, there is the cost of real-time monitoring, a fact that increases the number of control bytes sent in the network. Both cases described above

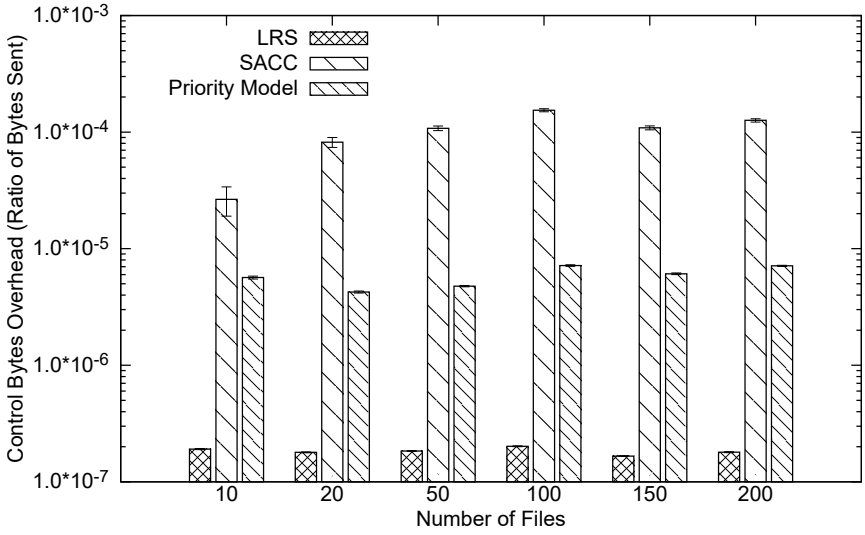

Figure 15: Control bytes overhead: Varying the number of files

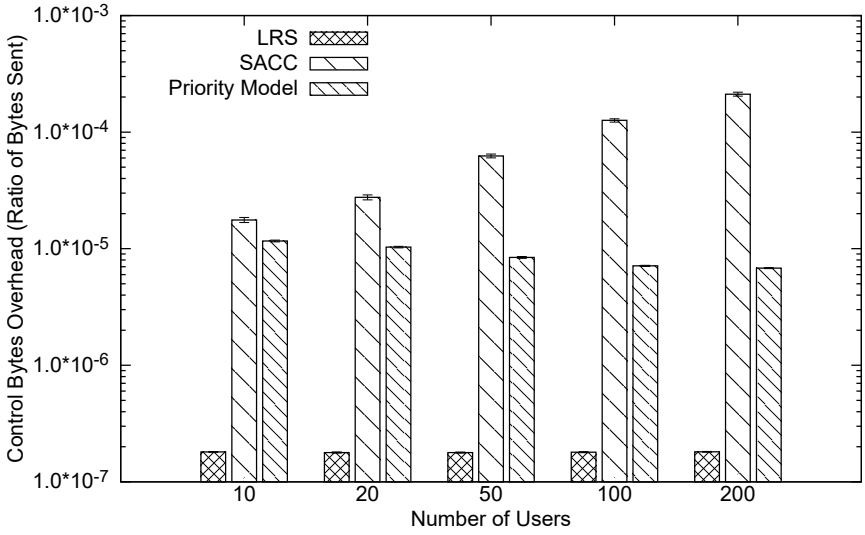

Figure 16: Control bytes overhead: Varying the number of users

jointly provide a stabilization effect in the control bytes overhead for all scenarios.

Figure 16 shows the control bytes overhead, with the increase in the number of users. As we can observe, LRS presents a constant behavior for the control bytes overhead.

We can also observe an expected linear growth in the control bytes overhead for SACC. It means that a growth in the number of users means that more connections tend to happen between users, a fact that increases the monitoring costs of SACC, demanding the transmission of a more substantial amount of social monitoring data to the centralized server.

In comparison with SACC, the priority model was able to provide higher differences in the control bytes overhead, with the increase in the number of users. For the scenario with 10 users, the control bytes overhead was 34.04% smaller, while for the scenario with 200 users the overhead was 96.78% smaller. In a similar way as observed for Figure 14, it is important to add that for the scenario with 10 users, besides providing a smaller control bytes overhead, the priority model surpassed LRS and SACC by 423.63% and 40.41%, respectively, in terms of the global cache hit rate.

Another interesting behavior that we can observe in Figure 16 is how the control bytes overhead decreases for the priority model, with the increase in the number of users. Between scenarios with 10 and 200 users, the decrease was of 41.47%. We can explain such behavior by the almost constant growth in the amount of monitoring data exchanged between devices and the server, with the increase in the number of users in the network. This phenomenon allies to the faster growth in the number of downloads from the server, for scenarios with more users, as demonstrated in Section 5.2.2, with the decrease in the global cache hit rate. A higher number of downloads from the server means that fewer control messages are exchanged between devices through D2D, therefore reducing the control bytes overhead.

5.3 Influence of each Component of the Proposed Model

In this part of the experiments, we evaluate the pertinence of our model by observing the achievable cache hit rate with variants of it. In practice, we re-run all experiments from Section 5.2.2, this time focusing on the proposed model.

We vary the priority model in five different ways, as follows:

1. **Complete model**, as explained in Section 3 and formalized in Equation 1.
2. **Dissemination Potential (DP) only (Equation 3):** Eliminates the Transmission Potential (TP) term (Equation 2), which relies on a file size threshold, as described in Section 3. It is worth recalling that TP computes the threshold basing on the EWMA contact time. Therefore, we remove the mobility aspect by removing this term and leave just the evaluation that focuses on the popularity and size of a file (DP).
3. **Transmission Potential (TP) only (Equation 2):** Eliminates the Dissemination Potential (DP) term (Equation 3), which evaluates the popularity aligned with the size of a file, as described in Section 3. Without the DP term, the device considers just the mobility aspect, and each file receives a priority basing on its size in comparison to the size threshold.
4. **File popularity only (Equation 8):** This model just considers the popularity variable derived in Section 3.2, therefore considering neither the mobility nor the size factors. The higher the popularity, the higher the priority of a file.

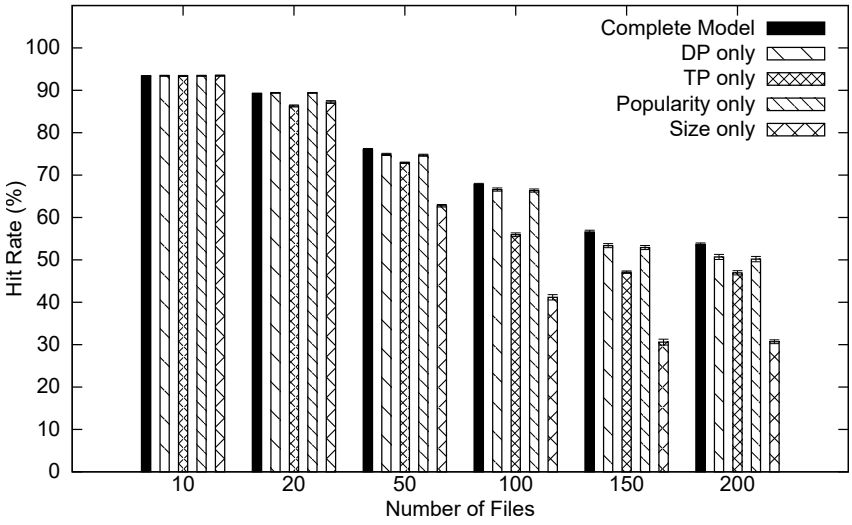

Figure 17: Varying the model and the number of files: contact trace *Infocom06*

5. **File size only (s_j):** We consider this one as the simplest of all variants of our model, giving higher priorities to smaller files, therefore preserving more files in the cache.

Figures 17 to 19 show the global hit rate for each model described above, for each number of files between 10 and 200, with the application of each contact trace and the synthetic mobility model. When applying traces *Infocom06* and the synthetic mobility model (Figures 17 and 19, respectively), we can see that the complete model outperforms the other ones for numbers of files higher than 50, which confirms our observations in Section 5.2.2, where we state the importance of the complete model in scenarios with a large number of files. At the same time, we can observe a similar performance for all models in scenarios with less than 50 files for all cases. This is another behavior that we observed in results in Section 5.2.2, where a small number of files raises the probability of a device finding everything it wants in its neighborhood or its cache, therefore reducing the effects of a cache selection procedure.

Still considering the *Infocom06* contact trace and the synthetic mobility model, when comparing the hit rate from models *DP only* and *TP only*, model *TP only* leads to a lower hit rate, mainly for numbers of files smaller than 150. Both terms rely on the file size, which takes us to the conclusion that the popularity metric (part of the DP term) gives higher positive effects, in comparison with the mobility aspect (part of the TP term). Such a phenomenon occurs since most variations in the priority values come from the nature of the files, as well as the fact that the complete model prioritizes files that are small and popular at the same time. Moreover, both TP and DP terms rely on trends of EWMAs (as seen in Section 3), while trends for mobility mostly bases on a large number of contact events along the time. This is not the case of popularity, which

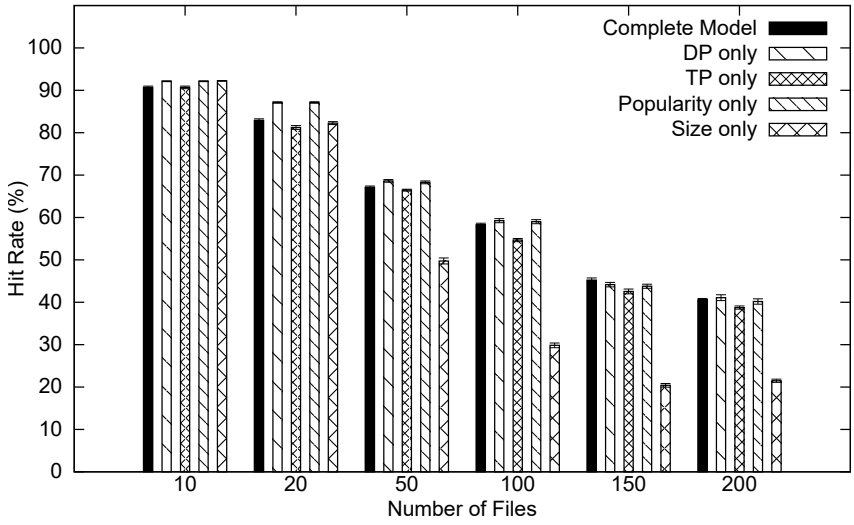

Figure 18: Varying the model and the number of files: contact trace *Reality Mining*

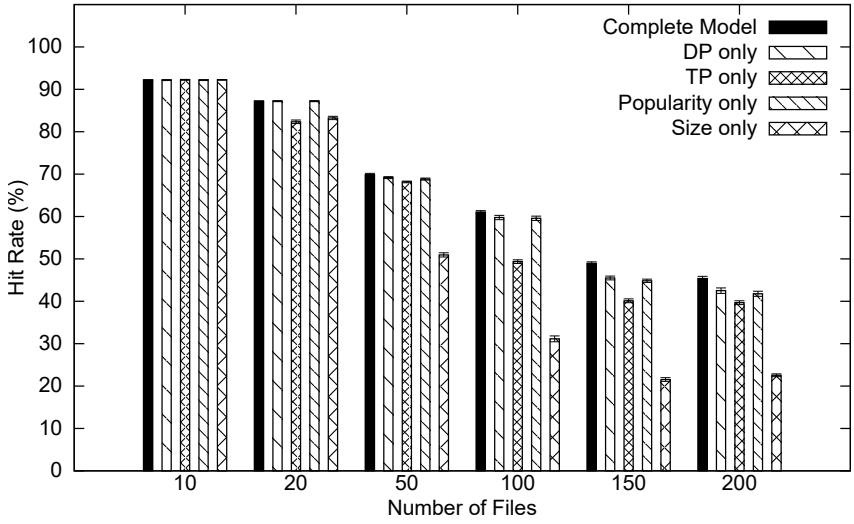

Figure 19: Varying the model and the number of files: synthetic mobility model

bases on a limited content set that contributes to a higher trend accuracy, therefore having more impact on the priority.

We reinforce the importance of file popularity with model *Popularity only*, which obtained a cache hit rate similar to *DP only* for scenarios with many files, but slightly

lower (a difference of less than 1% in all cases), thus showing the need to keep the file size attribute along with popularity at the DP term.

It is also important to note that computing priority, just relying on the TP term, does not prevent the device from storing a file with size above the threshold (see Section 3.1 for more details), if its other files fit in a similar situation. Therefore, the risk of not being able to transmit files through opportunistic D2D communications is higher if we do not consider other aspects (i.e., the DP term) when computing the priority. We demonstrate this phenomenon by observing model *TP only* in comparison with the complete model, being this last one superior in almost all scenarios.

Considering just the model *Size only*, we got the lowest cache hit rates for most scenarios, in comparison with the other four models. The use of this model raises the probability of storing a higher number of files in the device's cache. This operation helps to solve, at least partially, the problem of storing only files that the device can transmit, during their opportunistic D2D links. However, the fact that the device does not consider the nature of each file prevents it from storing some files that their neighbors may request in the future. Moreover, the model's low hit rate also shows the need for a fair evaluation of file sizes from another point of view, an issue that we solve with model *TP only*, that defines a size threshold to compare with the size of each file. The resulting model outperforms model *Size Only* by up to 108.48% at the hit rate, a peak value that we obtained with the *Reality Mining* contact trace. Finally, Figures 17 to 19 demonstrate the critical importance of applying the file size attribute at both TP and DP terms, considering that model *Size only* attained cache hit rates superior to 40% of those the other models obtained.

In the opposite way of *Infocom06* contact trace and the synthetic mobility model, with the application of *Reality Mining* contact trace (Figure 18), we obtained higher cache hit rates with variants of the complete model, being this one superior only in the scenario with 150 files. This complements our observations on the results from Section 5.2.2, where we expect some scenarios where the complete model does not make a positive impact on the cache hit rate. The results presented in Figure 18 show that we can still improve the hit rate of these scenarios with variants of the proposed model. As mentioned in Section 5.2.2, the definition of a triggering module to select the most suitable model, for each scenario, is left as future work.

Although all observations above, it is worth recalling that we did not vary the number of users in these scenarios. Changing this number implies that the network density is higher, as the target area is limited in the simulations. A higher network density impacts directly in the way our model works, as we can see in Figure 20, which presents the global cache hit rate for the complete model and its variants, by varying the number of users in the network.

As we can see in Figure 20, the complete solution outperformed all the other models, in accordance to results from Section 5.2.2. We observe the highest positive effects for scenarios with 10, 20, and 50 files, which made the complete model 23.29%, 21.76% and 15.74% superior to the second-best option, respectively.

Another interesting point to observe in Figure 20 is the fact that model *TP only* was superior to models *DP only* and *Popularity only*, for numbers of users inferior to 50, therefore showing the higher importance of the mobility aspect in these scenarios. We can explain this behavior by considering that a small number of users reduces the num-

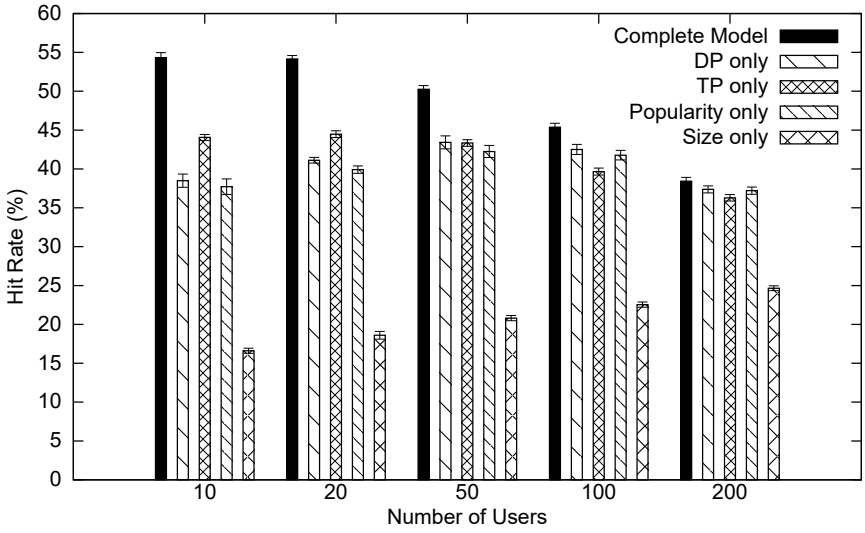

Figure 20: Varying the model and the number of users: synthetic mobility model

ber of contacts between them. In those cases, their transmission capacity through opportunistic D2D communications has more importance, and small files receive higher priorities than big ones. This does not mean that the file size is enough to obtain a proper cache selection, as we can see for model *Size only*, which attained the lowest cache hit rate in comparison with the *TP only* model. Even if we combine mobility attributes to the file size (the case of model *TP only*), we still attain lower cache hit rates in comparison with the complete model, a fact that demonstrates the importance of combining the content and mobility aspects in a single model.

Considering model *Size only*, we can see an approximation of its cache hit rate to the other four models, for the scenario with 200 users. We explain this behavior by remembering that our simulation environment considers a limited area for movements, as mentioned before. Therefore, when we vary the number of users, we also vary the network density. If we consider a scenario with a fixed number of files (which was the case for results of Figure 20), a higher density means a higher probability of finding what each device wants in the neighborhood. Consequently, the positive effects of applying a cache selection technique decrease. We complement our observations with Figure 12, where we present the same behavior for the complete model, along with the baselines. Therefore, the results presented in Figures 20 and 12 show that a real-time monitoring of the density of the network has key a role when deciding the right time to apply the cache selection technique. We leave the study of the tradeoff between our model and the network density as future work.

5.4 Use Case: Chunk-Based Adaptive Video Streaming

We now apply the proposed priority model in a chunk-based adaptive video streaming application. We compare it with the baselines in a similar way as it was done in Section 5.2.2, by varying the number of different video files, as well as varying the number of users.

Instead of having a content server and the devices sharing entire files composed by only one segment (chunk), we have them sharing chunks of different qualities for each video, with fixed size and duration for the chunks of each quality. In this way, each device is able to download, for each desired video, part of the chunks through D2D communications or read them from its own cache, and download the missing parts from the server.

For the sake of simplicity, and with the goal of aligning the results of this section with those obtained in the previous sections, we consider that the device receives the whole video, independently of the chunk download order or their respective quality, and the user consumes the video during the time described in the *Consumption Time* (Table 1). Therefore, in our experiments all chunks are downloaded before the consumption by the user. For each video download, the device follows the steps below:

1. If there are chunks available in the cache, the device reads them, and downloads the missing chunks from the server.

2. If no chunk is available in the device's cache, it searches for chunks in the D2D neighborhood, during the *D2D File Search Delay* (see Section 5.2.1 for more details). If a neighbor is found, the device downloads all the available chunks through D2D communications and download the missing chunks from the server.

3. If the device does not find chunks in the D2D neighborhood after the file search delay, it downloads the entire video from the server.

It is important to recall that the main goal of this work is to reduce the burden on the content server, by optimizing the use of D2D communications and the number of readings in the device's cache, at the same time with a small control overhead. Therefore, in the download steps above, we prioritize the consumption of chunks that are available in these two communication ways, independently of their respective quality.

It is also important to highlight that the download steps presented above are not the only possible way to schedule the download of chunks for one video. A finer grain evaluation on the effects of the priority model, considering other user consumption behaviors, as well as other chunk download scheduling algorithms, is left here as future work.

We consider three different qualities of videos: 480p, 720p and 1080p. Since we intend to run an adaptive video streaming application, we modify our simulation environment in order to assume that the connection quality between the user and the server, over time, follows a Normal distribution [9]. Therefore, we also assume that the server provides chunks with variable quality according to the same distribution.

In the device cache, we group chunks in *chunk blocks* of the same quality and belonging to the same video. For each video, the device may have stored in its cache up

to three chunk blocks: one block for chunks in quality 480p, and another two for 720p and 1080p.

Regarding the cache replacement procedure, the evaluation occurs for each chunk block. We do this basing on the fact that every chunk belonging to the same block has the same size and duration, and belongs to the same video. Therefore, all them should have the same evaluation results and may be selected (or removed) at once by the cache replacement procedure.

In the case of the priority model, priority values are computed for each chunk block, based on their respective size and the popularity of the respective video. In the case of LRS, we consider the storage time of a chunk block, thus removing the oldest ones in order to store a new chunk block when the cache is full. On the other hand, in the case of SACC, the original model does not specify how to compute popularity for devices that only have parts of a video [18]. Therefore, in order to compute SACC's popularity metric for a video in the closest way as possible to the original model, described in the beginning of Section 5.2, we consider the ability of a device to provide at least one chunk of that video. Following this premise, in some cases, SACC computes the same popularity value when all the devices have different chunks of the same video. In order to provide a "tie-break" for these cases, we defined a second criterion for SACC, removing chunk blocks from the biggest to the smallest one, considering the fact that bigger chunk blocks are less likely to be transmitted through D2D communications.

Due to the nature of chunk-based video streaming, we must add two new parameters: the chunk size for each quality (480p, 720p, and 1080p), and the duration of each video. We define the size of each chunk based on previous studies of Vaca-Rubio et al. [16], where the authors characterize the chunk size distribution of DASH-based video streaming services. The authors evaluate chunks with duration of one and six seconds. For convenience, we consider the chunk size distribution for chunks with duration of one second, and since the authors provide the average chunk size for each video quality in more than one bitrate, we chose the chunk size associated to the highest bitrate, for each quality. Therefore, for the quality of 480p the chunk size is defined as 53.42 KB, and for the qualities of 720p and 1080p the chunk sizes are defined as 217 KB and 326 KB, respectively [16].

The video duration is another missing parameter in the YouTube Trending dataset. In this section, we assume that the original file size (see Section 4.1 for more details) belongs to the highest video quality, i.e., 1080p, in order to compute the video duration we divide the total size by the size of one chunk. For example, if the video file has 3,260 KB, and considering a one-second 1080p chunk size of 326 KB, the duration of the respective video will be of 10 seconds, and for the quality of 1080p the video will be composed of 10 chunks.

For all simulations, we adopt the same simulation parameters from the previous experiments, described in Section 4.2. However, for simplification purposes, we present results only for the synthetic mobility model, since we observed similar behavior for the experiments with contact traces.

Figure 21 presents the global chunk ratio when we vary the number of different video files available in the network. In the same way as observed in Section 5.2.2, where we vary the number of complete files, the priority model is superior to the baselines for scenarios with more videos. For the scenario with 200 videos, the priority

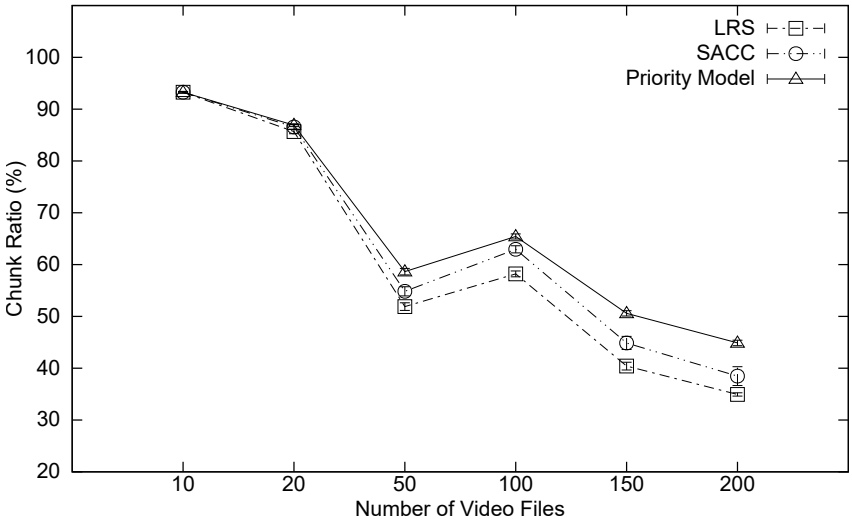

Figure 21: Global chunk ratio: varying the number of video files

model outperformed LRS and SACC by 28.46% and 16.64% respectively, a fact that demonstrates the ability of the priority model in keeping chunks that are more likely to be transmitted through D2D communications and/or to be read by the device in its cache.

Similarly to what was observed in the experiments with complete files, there is a decrease in the chunk ratio with the increase in the number of different video files in the network, since this reduces the probability of finding the desired video in the cache or in the D2D neighborhood. However, we also observe a slight increase in the chunk ratio between scenarios with 50 and 100 videos, for all solutions. Such an increase can be explained due to the joint nature of the size distribution for the whole video (according to a Weibull as described in Section 4.1) and the chunk quality delivered by the server according to a Normal distribution. Since in this experiments we allow the device to download a video through more than one download option (for example, part of the chunks through D2D and the missing part from the server), there will have some scenarios where more chunks will be found. Nevertheless, the decrease tendency on the chunk ratio, with the increase on the number of video files remains, as stated above and observed in Figure 21.

Another interesting point to observe in Figure 21 is the fact that the superiority of our solution over the baselines, was smaller in comparison to the superiority values observed with the experiments with complete files (Section 5.2.2). This occurs also due to the possibility of using more than one download option, the dynamic nature of the video size and the video quality provided by the server, and the chunk size defined for each quality. These factors, when combined, ended up favoring the performance of the baselines to some extent, however, even in these conditions the combination of content popularity and mobility attributes by the priority model, for a chunk-based scenario,

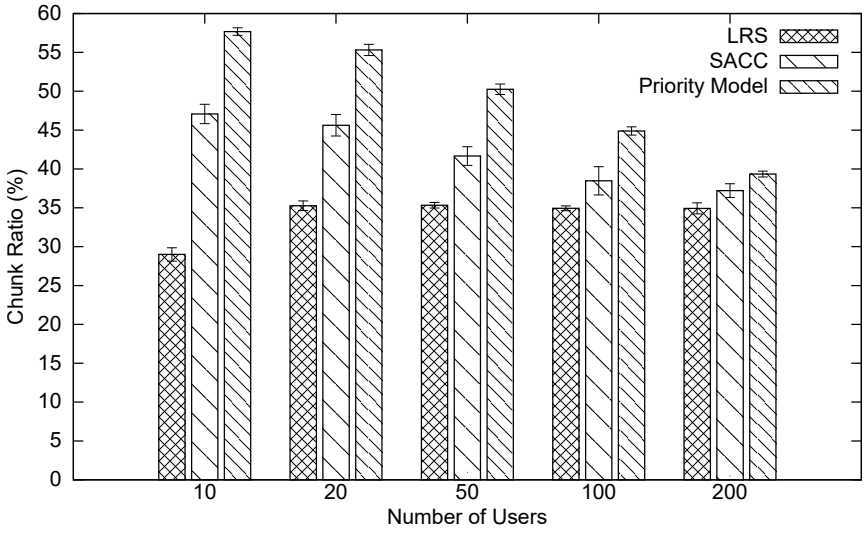

Figure 22: Global chunk ratio: varying the number of users

was able to provide improvements on the chunk ratio.

Figure 22 presents the global chunk ratio when we vary the number of users in the network. In the same way as observed in Section 5.2.2 with the experiments with complete files, we observe a gradual decrease in the chunk ratio for the priority model and SACC, and a gradual stabilization for LRS, with the increase in the number of users. For the same reasons presented for the experiments of Section 5.2.2, this behavior happened due to the mobile nature of the nodes, as well as the limited number of D2D sharings. As observed in the previous experiments, a larger number of users with limited D2D communication range, as well as the possibility of sharing chunks through D2D communications with only one neighbor at a time, cause a "dispute" for the D2D channel, thus reducing the number of available neighbors and consequently the chunk ratio.

Independently of the "decrease behavior" on the chunk ratio observed above, the superiority of the priority model over the baselines remained positive in comparison with the previous experiments, mainly for scenarios with few users. For 10 users, the priority model outperformed LRS and SACC by 98.88% and 22.52% respectively. Moreover, in the same way as observed in the experiments varying the number of video files, the performance of the baselines were closer to the priority model, in comparison with the results for complete files. This demonstrates the direct influence of the application running in the network on the performance of the cache replacement procedure, which deserves a complete dedicated study, also left here as future work.

Figure 23 shows the chunk quality rates for baselines LRS, SACC and the priority model when we vary the number of video files. For scenarios with small number of files, for all solutions, it is possible to observe that the chunk quality rate is similar. This is due to the fact that the device cache is able to store most videos from the server and

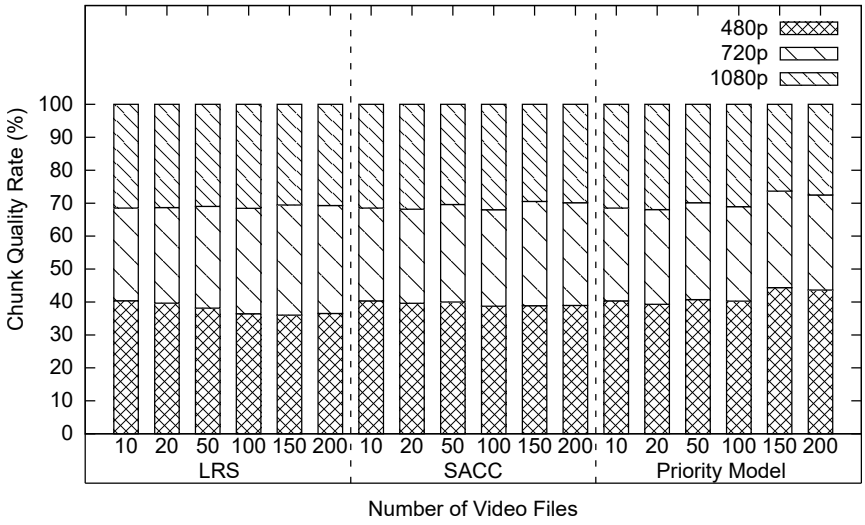

Figure 23: Chunk quality rate: varying the number of video files

almost everything is read from the cache right after, dismissing the device of searching for the video through D2D or download it again from the server.

For scenarios with more video files, it is possible to observe an approximate "balance" between the three video qualities (480p, 720p, and 1080p) for baselines LRS and SACC. On the other hand, for the priority model, there is an increase in the prioritization for chunks of the smallest quality (480p). This can be explained by the fact that the priority model computes higher scores for chunk blocks with higher popularity and smaller sizes. Therefore, the smaller the chunk block, and the higher the respective video's popularity, the higher the priority. Moreover, smaller chunk blocks tend to belong to the lowest qualities due to the smaller chunk size, a fact that also explains the highest percentages for the quality of 480p.

Although the video qualities delivered by the baselines are better balanced in comparison to those delivered by the priority model, it is important to remember that their respective chunk ratios are inferior to those measured for the priority model, as observed in Figure 21. Additionally, as also observed in the experiments with complete files, the control message overhead and the control bytes overhead measured by the baselines are considerably higher as well (see Section 5.2.3 for more details). Therefore, in order to enable and optimize the use of D2D communications and the number of readings in the device's cache, with small increases in the overhead, there will be the cost of reducing the quality of the video perceived by the user.

Figure 24 shows the chunk quality rates for baselines LRS, SACC and the priority model when we vary the number of users. For scenarios with 10 users, it is clear that the superiority of the priority model, in comparison with the baselines, in terms of the chunk ratio metric (Figure 22) is due to the prioritization of chunks with lower qualities. It is worth recalling that, for experiments varying the number of users, the number of

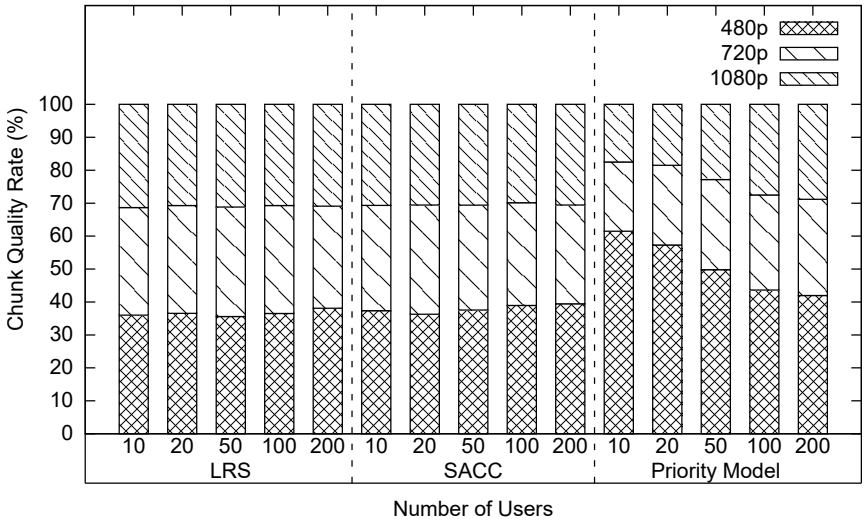

Figure 24: Chunk quality rate: varying the number of users

video files is fixed in 200, which means that each device must share its cache with a vast amount of videos in three different qualities. Therefore, in order to optimize the use of D2D as well as the readings in the cache, more chunk blocks with smaller qualities are preserved in the cache by the priority model. This explains a chunk quality rate of more than 60% for scenarios with 10 users.

In the same way as observed for the experiments varying the number of video files, the baselines provided a more balanced behavior for most scenarios, although with the higher overhead costs (see Section 5.2.3) and the smaller chunk ratios, as seen in this section. The priority model was able to do a similar balance with the increase in the number of users, but at the same time not denying the dynamics of the D2D contacts between devices, a fact that still forced the model to prioritize chunks of smaller qualities. Moreover, with the increase in the number of users, the priority model was also able to improve the quality of the video, a fact that can be observed with the increase in the chunk quality rates for 720p and 1080p.

6 Related Work

Computer architecture and Internet service providers researchers have been doing wide studies focusing on cache management. Classical caching replacement policies, such as the least frequently used (LFU) and least recently used (LRU), are simple to implement. However, caching to improve D2D communications brings new challenges, such as mobility of devices, user interest on each file, and limited storage [17]. Indeed, mobility has a dual role in D2D communication, while it may cause lower connection times between devices, it may provide cached content to more devices. Different

from web caching solutions based on content popularity, D2D communications must consider the popularity jointly with the characteristics of devices' contacts and content attributes. For instance, in a network characterized by shorter inter-contact time, popular but large files in the cache may be unfeasible to transmit between devices. The above-mentioned caching policies ignore the characteristics of mobility, popularity, and other content attributes.

The majority of solutions to enable caching in D2D communications, move files from centralized servers to the cache of selected devices, called cache-enabled *placement strategies* [7]. These proposed solutions are cluster-based, where centralized servers group mobile devices and define their cluster centers of each group as cache for other members of the cluster. On the other hand, several algorithms attempt to find the content at neighboring devices, called *delivery strategies*. In such a case, each device defines which files will remain in cache [7]. Interested readers may refer to [7, 17] for further discussion regarding mobile caching policies on D2D communications.

In our solution, each device has an independent caching process, requesting and consuming content in parallel. Moreover, the current proposal considers mobility and content characteristics at the same time, as well as the limited storage of each device.

As mentioned above, mobility in D2D communications allows higher coverage of the network, but it may cause short term contacts between devices [8]. As a result, forwarding larger files may be unfeasible. We highlight recent works that also consider the mobility characteristics and content download profile of neighbor devices, called *social and mobility-aware* caching policies [1, 4, 13, 14, 18–20].

Hosny *et al.* considers that users with the same mobility pattern must cache the same content and show that users with random mobility are efficient for caching more content in the network [4]. Song *et al.* explore users' mobility to model the placement of content on users' devices as an integer programming problem [14]. Shan *et al.* consider the mobility and the social structure of the nodes to propose a cache strategy basing on clusters of users with the same interests [13, 19]. Zhang *et al.* consider the mobility, social structure, and also, the content popularity [20]. Authors propose a user interest prediction to define content in the devices' cache.

In a similar way, Ali *et al.* consider user download profile and user mobility groups to optimize the device's cache. Their solution monitors users' mobility to define a mobility profile for each one, defining groups of users with similar profiles [1]. Wu *et al.* considers the social relations among users to propose a community-based content caching model. By forming user communities with similar content interests, the authors improve the accuracy of estimation of content popularity, optimizing the caching process and therefore raising the use of opportunistic D2D communications [18].

Our proposal considers a priority model for selecting content in the cache after exceeding its capacity, therefore increasing the chances of serving a file through opportunistic D2D communications. Most works mentioned above dismiss cache replacement or use classical algorithms when the cache is full. In this case, there is a probability of dropping shareable content with its neighbors.

Therefore, it is important to decide, with the arrival of new files greater than the available cache space, which files the device must maintain cached or drop. The solution proposed by Wu *et al.* is a step towards such direction, proposing a novel cache

Table 2: Related work summary

Work	Placement Strategy	Content Data Based	Mobility Based	Novel Cache Replacement
[4]	✓	✗	✓	✗
[14]	✓	✗	✓	✗
[19]	✓	✗	✓	✗
[13]	✓	✗	✓	✗
[20]	✓	✓	✓	✗
[1]	✓	✓	✓	✗
[18]	✓	✓	✓	✓
Our Solution	✗	✓	✓	✓

replacement strategy that relies on content popularity [18]. However, their solution estimates popularity basing only on information collected by devices in their respective community, as well as between communities. Our solution considers the history of content popularity and mobility for the whole network, thus enabling the caching of files that the device will be able to share with any neighbor, at any region of the network.

Table 2 summarizes the main features of the works discussed above, as well as our solution. In common, all these previous proposals consider the mobility of users, however, all them also rely on cache preloading, which requires centralized decisions. We provide a distributed solution, based on a priority model that considers content and mobility attributes for cache replacement.

Our proposal focuses on cache space optimization at each device by considering just content consumed by the owners' device, along with a real-time selection of files to keep cached. Different from cache-enabled *placement strategies* that preload content on devices' cache to optimize the cache hit rate, our solution reduces the overload in the core network by fetching only content requested by users, which is also one of the main goals when using opportunistic D2D communications.

7 Conclusion

We proposed a model to compute priority values to content in the cache of mobile user devices. The priority model combines content and mobility attributes, to determine in real time which content each device can transmit through opportunistic D2D communications. In this way, it optimizes the global cache hit rate of the network.

Simulation results showed that our solution provides a global cache hit rate almost 80% superior in scenarios with many different files, and around 420% superior for scenarios with few users, in comparison with the baselines. Compared to the state-of-the-art solution, the priority model reduces by 80% the overhead of control messages and by 90% the overhead of control bytes.

Additional results also suggested that there will have specific scenarios where variants of our solution offer bigger improvements in the hit rate, in comparison with the

complete model. There will also have scenarios where the complete solution and their variants will not offer a significant improvement in the hit rate. Therefore, the network should have a monitoring module that would trigger our solution in each user device in the proper moment, as well as switch to variants of it. We leave the study and development of such monitoring module as future work.

We also applied our solution in a chunk-based adaptive video streaming application, once again demonstrating the superiority of our solution over the baselines. We observed that our solution, with smaller control overhead, was able to improve the global chunk ratio which reduces the burden on the content server, although with a reduction in the video quality in some scenarios. A dedicated study focusing on video quality is left as future work.

Declarations:

Funding. This work received funds from the following Brazilian government agencies: *Coordination for the Improvement of Higher Education Personnel* (CAPES), *National Council for Scientific and Technological Development* (CNPq), *Foundation for Research of the State of Espírito Santo* (Fapes), and *Foundation for Research of the State of Minas Gerais* (Fapemig).

Conflicts of Interest. The authors declare that they have no conflict of interest.

Availability of Data and Material. The authors declare that all databases, as well as all raw output results from simulations can be provided online, if requested by the reviewers.

Code Availability. The authors declare that the source code used during the simulations can be provided online, if requested by the reviewers.

References

[1] Ali, A.S., Mahmoud, K.R., Naguib, K.M.: Optimal caching policy for wireless content delivery in d2d networks. Journal of Network and Computer Applications **150**, 102467 (2020). DOI https://doi.org/10.1016/j.jnca.2019.102467. URL http://www.sciencedirect.com/science/article/pii/S1084804519303273

[2] Eagle, N., Pentland, A.S.: CRAWDAD dataset mit/reality (v. 2005-07-01). Downloaded from https://crawdad.org/mit/reality/20050701 (2005). DOI 10.15783/C71S31

[3] Goian, H.S., Al-Jarrah, O.Y., Muhaidat, S., Al-Hammadi, Y., Yoo, P., Dianati, M.: Popularity-based video caching techniques for cache-enabled networks: A survey. IEEE Access **7**, 27699–27719 (2019). DOI 10.1109/ACCESS.2019.2898734

[4] Hosny, S., Eryilmaz, A., El Gamal, H.: Impact of user mobility on d2d caching networks. In: 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2016). DOI 10.1109/GLOCOM.2016.7841886

[5] Keränen, A., Ott, J., Kärkkäinen, T.: The ONE Simulator for DTN Protocol Evaluation. In: SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques. ICST, New York, NY, USA (2009)

[6] Krishnan, S.S., Sitaraman, R.K.: Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs. IEEE/ACM Transactions on Networking **21**(6), 2001–2014 (2013). DOI 10.1109/TNET.2013.2281542

[7] Li, L., Zhao, G., Blum, R.S.: A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies. IEEE Communications Surveys Tutorials **20**(3), 1710–1732 (2018). DOI 10.1109/COMST.2018.2820021

[8] Mota, V.F., Cunha, F.D., Macedo, D.F., Nogueira, J.M., Loureiro, A.A.: Protocols, mobility models and tools in opportunistic networks: A survey. Computer Communications **48**, 5 – 19 (2014). DOI https://doi.org/10.1016/j.comcom.2014.03.019. URL http://www.sciencedirect.com/science/article/pii/S0140366414001054. Opportunistic networks

[9] Mota, V.F.S., Macedo, D.F., Ghamri-Doudane, Y., Nogueira, J.M.S.: On the feasibility of wifi offloading in urban areas: The paris case study. In: 2013 IFIP Wireless Days (WD), pp. 1–6 (2013)

[10] Pang, Z., Sun, L., Wang, Z., Xie, Y., Yang, S.: Understanding performance of edge prefetching. In: MultiMedia Modeling, pp. 527–539. Springer International Publishing, Cham (2017)

[11] Reis, D.M., Lins, T.S., Nogueira, J.M.S., Mota, V.F.S.: Sharefile: Sharing content through device-to-device communication. In: 2018 IEEE Symposium on Computers and Communications, ISCC 2018, Natal, Brazil, June 25-28, 2018, pp. 736–741. IEEE (2018). DOI 10.1109/ISCC.2018.8538467. URL https://doi.org/10.1109/ISCC.2018.8538467

[12] Scott, J., Gass, R., Crowcroft, J., Hui, P., Diot, C., Chaintreau, A.: CRAWDAD dataset cambridge/haggle (v. 2009-05-29). Downloaded from https://crawdad.org/cambridge/haggle/20090529 (2009). DOI 10.15783/C70011

[13] Shan, G., Zhu, Q.: Sociality and mobility-based caching strategy for device-to-device communications underlying heterogeneous networks. IEEE Access **7**, 53777–53791 (2019). DOI 10.1109/ACCESS.2019.2912674

[14] Song, J., Choi, W.: Mobility-aware content placement for device-to-device caching systems. IEEE Transactions on Wireless Communications **18**(7), 3658–3668 (2019). DOI 10.1109/TWC.2019.2916781

[15] Stockhammer, T.: Dynamic Adaptive Streaming over HTTP–: Standards and Design Principles. In: Proceedings of the Second Annual ACM Conference on Multimedia Systems, pp. 133–144. ACM (2011)

[16] Vaca-Rubio, C., Gómez, G., Aguayo-Torres, M., López Martínez, F.: Statistical Characterization of the Chunk Size Distribution in DASH. In: XXXIV Simposio Nacional de la Unión Científica Internacional de Radio (URSI) (2019)

[17] Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., Wang, W.: A survey on mobile edge networks: Convergence of computing, caching and communications. IEEE Access **5**, 6757–6779 (2017). DOI 10.1109/ACCESS.2017.2685434

[18] Wu, D., Liu, B., Yang, Q., Wang, R.: Social-aware cooperative caching mechanism in mobile social networks. Journal of Network and Computer Applications **149**, 102457 (2020). DOI https://doi.org/10.1016/j.jnca.2019.102457. URL http://www.sciencedirect.com/science/article/pii/S1084804519303170

[19] Yuan, Z., Zhuang, W., Wei, X., Zhou, L.: Joint social-aware and mobility-aware caching in cooperative d2d. In: 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC), pp. 656–661 (2019). DOI 10.1109/IWCMC.2019.8766729

[20] Zhang, W., Wu, D., Yang, W., Cai, Y.: Caching on the move: A user interest-driven caching strategy for d2d content sharing. IEEE Transactions on Vehicular Technology **68**(3), 2958–2971 (2019). DOI 10.1109/TVT.2019.2895682