Multimedia Sharing over Opportunistic Networks

Radu Dragan, Radu-Ioan Ciobanu, Ciprian Dobre
University Politehnica of Bucharest
Sp. Independentei, 313
Bucharest, Romania
{radu.dragan, radu.ciobanu}@cti.pub.ro,
ciprian.dobre@cs.pub.ro

Constandinos X. Mavromoustakis
University of Nicosia,
Department of Computer Science,
46 Makedonitissas Avenue, 1700
Nicosia, Cyprus
mavromoustakis.c@unic.ac.cy

George Mastorakis
Technological Educational
Institute of Crete
Estavromenos 71500, Heraklion,
Crete, Greece
gmastorakis@staff.teicrete.gr

***Abstract*—The massive increase in the use of mobile devices lately led to the congestion of the mobile networks. One solution to offload part of the traffic is to send data, whenever possible, through direct phone-to-phone communication. Such opportunistic networks rely on wireless communication to support a decentralized model of communication where nodes can store, carry and forward data directly to others. Up-until-now most work focused on constructing algorithms designed to forward or disseminate data to nodes having a better chance of routing it to its destination. Here, we extend on our previous work, and propose *Opportunistic Multimedia Sharing*, an algorithm designed to support dissemination of multimedia content over an opportunistic communication model. The algorithm uses the predictability of the node encounters to determine which multimedia files a specific node can deliver closer to the interested nodes. Experimental results show the algorithm can deliver a high success rate (over 80% in same cases), while maintaining a low delivery latency and cost.**

I. Introduction

Opportunistic mobile networks (ONs) consist of human-carried mobile devices that communicate with each other in a store-carry-and-forward fashion, without any infrastructure [1]. In opportunistic networks, nodes act according to the store-carry-and-forward paradigm: Nodes may store a message, carry it around, and forward it, over wireless phone-to-phone communication protocols, when they encounter the destination or a node that is more likely to reach the destination. Such networks are formed between nodes spread across the environment, without any knowledge of a network topology. The routes between nodes are, thus, dynamically created, and nodes can be opportunistically used as a next hop for bringing each message closer to its destination.

We previously proposed SPRINT (Social PRedIction-based routing in opportunistic NeTworks) [2], a social-aware ON algorithm that relies on the history of contacts coupled with online social information about nodes, to statically determine social relationships between them. The algorithm combines several findings to increase the rate of successful delivery of messages to their destination, while maining a low congestion on the network and a faster delivery, when compared to similar solutions. First, it proposes the use of online social information about nodes, which today can be easily obtained through integration with social networks such as Facebook, Twitter of LinkedIn in various mobile applications. Moreover, previous results show that, in certain types of environments, contacts between opportunistic devices are highly predictable. When users follow rare events-based mobility patterns, we demonstrate that human mobility can be approximated as a Poisson distribution [3]. Finally, SPRINT adds an additional prediction component into the proposed routing algorithm. SPRINT was evaluated against other social-based routing approaches, in both real-world traces, as well as for synthetic mobility scenarios.

With the continuous increase in users network activities and the escalating amount of information that they generate within the Internet, , today we witness a new area of convergence between networks and media, paving the way towards the Future Media Internet [4]. Driving forces are the recent advances in connected media technologies and social networks, supported by the widespread deployment of broadband infrastructures and cloud computing facilities, along with a new breed of user-equipment that integrate communication and computation capabilities. In this evolving Future Media environment, it is normal for citizens to demand the kind of experiences that they are accustomed to in their daily/real lives. Towards conveying Media-related data, following a user/community-centric approach with the maximum possible Quality of Experience (QoE), *media delivery plays a key role*.

Towards achieving the maximum efficiency in Future Media Internet contexts, media delivery calls for new architectures, along with associated technologies (i.e. functionalities and mechanisms), supporting their synergies to meet emerging requirements and increase the quality that human-clients can experience. Moreover, with the increase in market penetration of smart portable devices, people expect more media content to be accessible anytime anywhere, and with high QoE demands. Several studies show, in fact, that smart mobile devices have surpassed for sometimes now the desktop devices that people use to access such content [5].

To meet such demands, in this paper we propose *Opportunistic Multimedia Sharing*, an algorithm designed as an application of opportunistic network design principles for the delivery of media content. The algorithm extends on our previous work (the SPRINT algorithm), and builds an approach where the opportunistic network supports an overlay for the decentralized discovery of media content and the rapid transfer of media files over phone-to-phone wireless communication protocols, thus relieving some of the congestion stress traditionally imposed over other broadband protocols (according to CISCO predictions, more than 50% of the broadband traffic, as in 3G/4G/5G, will consists, by 2018, of media content being transferred towards smart devices [6]). Experimental results show the algorithm can deliver a high success rate (over 80%), while maintaining a low delivery latency and cost.

The rest of the paper is structures as follows: We first present related work. In Section IV we present the *Opportunistic Multimedia Sharing* algorithm, followed by results showing its capabilities in Section V. Section VI concludes the paper.

II. Related Work

A detailed review of opportunistic networking is performed in [7]. Popular ONs routing algorithms in BUBBLE Rap [8], and dLife [9]. SPRINT was compared in our previous work with BUBBLE Rap, one of the most well-known and effective socially-aware opportunistic routing techniques [2]. BUBBLE Rap uses social knowledge about the nodes in the network to deliver messages. As such, it forwards data to nodes that are more popular than the carrier node, bubbling it up a hierarchical ranking tree using a global popularity level. A node's popularity is the value of its betweenness centrality, i.e. the number of times the node is on the shortest path between any other two nodes in the network. Communities in BUBBLE Rap are dynamically discovered using k-CLIQUE [10].

With SPRINT, we exploit the fact that mobility of the users can be predicted in terms of direct contacts, especially for academic environments [11]. We previously showed that the Poisson distribution can, in fact, be used to predict future encounters, based on the history of encounters. Social Prediction-Based Opportunistic Routing in Opportunistic Networks (SPRINT) [2] makes a step forward and, besides using the history of encounters and the social information, also considers the online social information (information provided by the social networks as Facebook, Twitter, LinkedIn), assuming that two users that are connected on one of these social networks tend to meet more often. In addition, a person that has many friends on the social networks might be more popular than others that have fewer friends.

Unlike previous work, we show here an application of SPRINT, where the ON algorithm is augmented to allow for opportunistic discovery of media content, and opportunistic spreading of media files to interested nodes. Unlike previous work, we assume here *large media files* being exchanged in the network, meaning we need to accomodate for files that might not be downloaded at-once when direct wireless communication opportunities arrise, due to node mobility. Thus, the download of files follows an approach similar to how media content is discovered and accessed in decentralized peer-to-peer networks. The challenge is that no node can learn or discover at any time information about all other nodes, thus the media transfer/routing algorithm has to accomodate for disconnections and highly variable delays caused by human mobility.

III. SPRINT: Social Prediction-Based Opportunistic Routing

SPRINT (Social PRedIction-based routing in opportunistic NeTworks) combines socially-aware routing, including both learned and offline social information about nodes, with a module capable to predict node behaviour. Each SPRINT node has a data memory and a cache memory. The former is used to store data objects (messages), while the cache memory contains the node's previous encounters with other ON participants. When two nodes meet, they exchange information about each message in their data memory, which includes a hash of the message's content (its unique ID), the source and destination, the generation time, and the number of hops traversed so far. Next, each node computes utility values for the messages in its own memory, as well as for the ones advertised by the encountered node. The formula used by a SPRINT node A to compute the utility of a message M is:

$$u(M, A) = w_1 * U_1(M, A) + w_2 * U_2(M, A) \tag{1}$$

Messages from both nodes are sorted according to their utility values. If some of the messages with high utility values belong to the encountered node, a download request is sent for each of them. The node then starts transferring these messages in utility order, until it has finished downloading all the required messages or the two encountering nodes are not in range anymore. In the formula, w_1 and w_2 are weight values which follow the conditions that $w_1 + w_2 = 1$ and $w_1 > w_2$. U_1 and U_2 are utility components computed according to the following formulas:

$$U_1(M, A) = freshness(M) + p(M, A) * (1 - \frac{enc(M, A)}{24}) \tag{2}$$

$$U_2(M, A) = c_e(M, A) * \frac{s_n(M) + hop(M) + pop(A) + t(M, A)}{4} \tag{3}$$

The $freshness(M)$ component of U_1 favours new messages.$p(M, A)$ is the probability of node A being able to deliver a message M closer to its destination. The term is based on predicting a node's behaviour, combined with the idea that a node has a higher chance of interacting with nodes it is socially connected with and/or has encountered before. $enc(M, A)$ is the time (in hours) until the destination of message M will be met by A according to the probabilities previously computed.

In U_2, $c_e(M, A)$ is set to 1 if node A is in the same community as the destination of message M or if it will encounter a node that has a social relationship with M. $s_n(M)$ is set to 1 if the source and destination of M do not have a social connection. $hop(M)$ represents the normalized number of nodes that M has visited, $pop(A)$ is the popularity value of A according to its social network information (i.e. number of Facebook friends in the opportunistic network), and finally $t(M, A)$ is the total time spent by node A in contact with M's destination.

IV. The Multimedia Sharing algorithm

A. An analysis of challenges and requirements

Multimedia Sharing extends SPRINT as an application for *multimedia content sharing over Opportunistic Networks*. Due to the opportunistic/stochastic nature of the communication infrastructure, and the specifics of media content, *Multimedia Sharing* faces two important challenges: First of all, because the network is mostly formed between resource-contrained mobile devices, the storage space is considered to be limited. Thus, unlike more traditional desktop-based peer-to-peer networks, we cannot assume a mobile device has much space to store for long-time large-size multimedia files (to share with other nodes). Second, there is no central entity that can hold the data which has to be spread among the devices. Thus, we have to rely on the opportunistic collaboration of devices[1] to handle the spreading of media content, and sharing the discovery of such content throughout the network.

Further, we assume mobile devices allocate a local caching space, where media messages can occupy space for a longer period of time. Due to lack of an end-to-end path from a source to its destination, nodes can carry in their cache data only to be forwarded when opportunity occurs to other nodes, maybe having a higher chance of succesful delivery to its destination. Also, several copies of the same content can co-exist in the network, traversing different paths, in order to increase the chances of succesful delivery. Some devices could also store a file even if it has been successfully delivered to the destination. With this in mind, the algorithm must make sure that the information about a file being successfully delivered is forwarded to all devices holding a copy of the file, so that the memory can be released and reused.

For the forwarding decision process, each device uses a history of data it collects about other nodes - information gathered during the previous encouters. Such information include social relationships, previous encounters with other nodes, information about each file requested by each node, information about all the files present in the network. We consider that such information can be reliable stored on the local memory of the device, considering hardware capabilities of modern such devices [12]. Still, because our algorithm need to work for all kinds of mobile devices, even old ones with smaller memory, we still consider the problem to be a challenge.

B. Data structures

Multimedia Sharing uses the two memory structures found on SPRINT: the *data memory* and the *cache memory*.

The cache memory holds a list of the most recent encounters with other nodes. For each encounter, we hold the *id* of that node, the *day* and *hour* of the encounter. This information is further used when computing the utility of each message, during a subsequent encounter.

[1]Mostly to keep things simple, here we assume devices are not acting selfish or malicious. We leave extensiosn to handle such situations for future work.

The data memory holds the media files that are carried by the current node. According to the protocol, we assume a message contains an *id*, by means of which a file is uniquely identified in the network, the *id of the source node* (the node that generated the file), and the actual *media content*. For multimedia content sharing, the structure of the file has also to be provided, for the purpose of dividing the file into smaller *chunks* that are more easily transferred. Thus, we assume the file contains a list of internal chunks, along with the total number of chunks (used for verifying if a file was delivered entirely or not).

As a data structure, we consider a chunk to be part of a video file of a maximum size of $1.5KB$. The size was chosen to correspond to the standard *Maximum Transmission Unit* of Ethernet [13]. That means that a chunks does not have to be further divided to fit into one standard communication packet.

A chunk contains several fields: the *id*, which uniquely identifies the chunk inside the media file, the *size* and the actual *content*.

Further, as most of the multimedia content sharing application, the algorithm is request driven, meaning that a file will be downloaded to a node only if the respective node previously requested the file. A *request* is made by a user, in the form of an interest the user expresses for a specific file. The *requests* includes the following fields: an *id of request*, which is unique for each request, the *id of the file* the request is intended for, a *list of solicitants* (nodes that requested the same file), and a *dictionary* containing the nodes where the file was already delivered. Each node holds a list of own requests (requests generated by itself), and a memory for requests generated by others. This requests is disseminated in the whole network since they represent the interest of the nodes in specific files and will be used for taking routing decisions.

A *solicitant* of a file represents a node that requested that particular file. As a data entity, a solicitant is characterized by the following fields: the *id of the node* that made the solicitation, a *list of chunk intervals*, and a *timestamp* that indicates the last modification of the solicitant. The *chunk intervals* are intervals involving only the *id* of the first chunk and the *id* of the last chunk requested. This implementation using chunk intervals was chosen because it diminishes the size of the information held for a solicitant (and consequently for a request), which, as mentioned before, is a very important gain for the algorithm, given the limited memory sizes of the mobile devices.

Solicitants are used to mark requests as delivered to some specific destinations. This way the notification about the delivery of a file to a specific destination ca be disseminated more easily, and the other nodes will not try to forward the message to that destination anymore. In this regard, the requests keep a list of node ids where the file was delivered. When two nodes meet, the delivered lists are checked and then updated. A solicitant marked as delivered will be deleted from the specific request and not taken into consideration anymore.

A timestamp is also incorporated into a solicitant suggesting the time of the latest update. This field is useful in the following scenario: a node delivers a part of the chunks (not the entire file) to a destination. In consequence, the node updates the specific solicitant of the specific requests, removing the chunks that have been delivered and updating the timestamp of the last update. When two nodes meet having the same request for the same solicitant, only the solicitant having the newest timestamp is kept.

When two nodes meet, the request are exchanged and if there are two requests for the same file they are merged into a new request containing the list of all distinct solicitants from both old requests.

Next, a node needs to find infomation about files available in the network. Because we do not assume a central entity to mediate the discovery of such files, we opportunistically use any contact between nodes to spread knowledge further in the network. As such, at first each node normally knows about the files carried around by the nodes he encounters. Further, an entity called *crumb* is used to advertise information about files available in the system (and exchanged when two nodes communicate).

The composition of a *crumb* is the following: an *id* for identification, the *id of the file* advertised, the *title* which should be relevant and should describe also the content of the file, a *list of topics* associated with the file (media descriptors), which may lead to a more specific description of the file, a *list of seeds* (identifiers of the nodes holding the file), and the *size of the file* in chunks. The list of seeds might be relevant in scenarios where a user wants to maximize the chances of a download and chooses a file with a large number of seeds, or when the user knows that they will meet one of the seeds in the near future.

C. Message Exchange

Deciding what files to keep and which ones to drop represents the core of the routing process, and can drastically influence the performance of the algorithm.

When two nodes meet, each of them executes the next steps:

1) Firstly, requests and crumbs are exchanged, since they might influence the routing decision.
2) As a second step, the node searches for files that it might be interested in, and generates a download request to the encountered node for that file. Assuming that the other node is not selfish towards the requested files, the download begins.
3) After fulfilling its interest, the node begins the actual routing process by gathering all the messages from its local data memory and encountered data memory and assign an utility value to them. Then, the node will try to keep the files with highest utilities from both memories (requesting downloads for the files with high utilities from other node's memory).

For computing *utility*, the nodes makes a prediction for future encounters of the node. Future encounters will specify for all other nodes the probability to be met by the current node, for each of the next 24 hours (we previoulsly showed in [11] the assumptions for making such a prediction).

Firstly, the cache memory is checked. If a node has been met recently, there is a high probability to be around and to be met again soon, so the probability of meeting the node is increased. If the node has been met in the past weeks in the same day the chances are that the node will be met again (the faculty timetable does not vary too much during a semester). Same principle is applied for a node met in the same hour interval in the past weeks (the nodes might have met at a course or seminar/laboratory). Under both circumstances, the probability is increased again. If the two nodes are part of the same community/ share a connection on the social networks, the probability of them meeting is doubled (people that share interests and have a lot of things in common tend to meet more often than random chosen people).

After each step of modifying the probability, the values have to be renormalized. As a consequence, the probability of two nodes meeting is a value strictly between 0 and 1.

Succeeding the measurement of the future encounters, the utility of a message will be computed using the following formula (similar to the one used by SPRINT, with a different meaning). The utility assigned by node A to a file M is the one presented in Equation 1.

The first component of the utility (Equation 2), highlights the chances of the node A to deliver the file M directly to a node interested in it. The interest of a node in a certain file is reflected by the existence of a request containing that node as a solicitant.

- The $freshness(M)$ will be 0.5 for messages that have been generated less than 24 hours ago, and 0 otherwise. Thus, new messages are favored over the old ones.
- B represents a node interested in the message M.
- $p(A, B)$ represents the the probability of node A to meet node B, considering the prediction of future encounters.
- $enc(A, B)$ represents the time (in hours) until node B is reached. Consequently, meeting an interested node faster will be translated into a greater utility value. If the node is not encountered in the next 24 hours the value of $enc(A, B)$ will be 24, so the product will equal to 0.

Messages with positive utility will be kept for downloading and sorted with regard to the utility value; messages with greater utility will be downloaded first, as they have the greatest chance of being delivered to an interested node.

Afterwards, for all those nodes who were assigned with a 0 utility using the first formula, utility value is recomputed. These nodes have a really small chance to deliver the file directly to any destination, but maybe they can carry the file closer to interested nodes.

The formula for computing the second part of the utility is similar to Equation 3. This component of the utility only considers the social relationship between users (provided by the social networks) or, if this information is lacking, communities of users that will be built using the k-clique method.

The meaning of the variables presented in this formula are very similar to the ones presented for SPRINT, having a different meaning for the word destination. Unlike SPRINT, where a message is generated for a specific destination, *Multimedia Sharing* considers a destination of a file any node interested in the file. As stated before, a node A considers node that B is interested in file M only if A contains a request for file M having B as one of the solicitants.

In consequence, the variables presented in Equation 3, for *Multimedia Sharing*, are as following:

- $c_e(M, A)$ equals the ratio of destinations in the same community as A, and the total number of destinations that A knows of. The more destinations in the same community, the greater the chance of delivering the file closer to them. If there is no destination in the same community as A the value of the second utility will be 0, the node having a very small chance of delivering the file towards any of the destination, so the file will be dropped with a high probability.
- $s_n(M, A)$ is the ratio of the number of the destinations of the file that are not in the same community with the source of the file and the total number of destinations. It is unlikely for a file to reach a destination which is in a different community that its source, so there is a big chance that the node will have to travel for a large number of nodes in order to reach the destination community.
- $hop(M)$ represents the normalized number of hops the file has traveled so far. The higher the number of hops traveled, the smallest the chance to find any of the destinations, so the file should be forwarded as much as possible.
- $pop(A)$ serves as the popularity of the node A. If there is social information available, the popularity will be considered as the number of friends (on the social platform used). If this is not the case, the popularity will be considered the number of nodes that are in the same community as A. In both cases, a popular node will have a greater chance of delivering the file to the destination, so the files should be forwarded towards more popular nodes as much as possible. If the node A is more popular than the encountered node, the value of the variable will set to 1, and 0 otherwise.
- $t(M, A)$ is the time spent by the node A in contact with the destinations of file M. This value will be normalized with the total amount of time spent by A in contact with other nodes. A node that spent a great amount of time with another node is expected to have other contacts with that node soon, so will have a good chance to deliver the file.

Having the two utility components, the transfer can begin. Firstly, the files are delivered according to the first utility value computed (files having greater utilities will be delivered first). Afterwards, the files remained (those having 0 for utility according to Equation 2), are delivered in the order of the second utility value. As a short reminder, the files are transferred only if the data memory was not exceeded. Then, the remaining files are added to the memory (if there is still room), until the memory is full.

Nevertheless, message exchanging faces a series of problems. Particularly, the algorithm cannot fully trust the good faith of the nodes. Having small capacity batteries on their mobile devices, users may choose not to participate in the routing process, only demanding the files they are interested in. This would be clearly not beneficial for the algorithm, which is fully dependent on the routing performed by the nodes.

Consequently, users should be persuaded to help each other and transport messages in benefit of others as often as they get the chance. Thereby, nodes that showed selfishness towards routing will be prevented by the opportunity to download files for themselves. When the battery level is under a certain level, battery should, though, be saved for important calls or emergencies, until the users get the chance to plug in the mobile device. In this case, the altruism level will not be affected by the decision to not participate in the routing process.

The mechanism for assigning altruism value is inherited from SPRINT and is computed for a specific node A towards a specific message/file M.

During the message exchange, it might happen that not all of the files get to be downloaded. However, the algorithm needs to take advantage of the entire contact so there is a possibility that some files might be only partially downloaded. In this case the message will be divided and the node that downloaded only a part of the message will generate a request for the chunks remained not downloaded, in order to restore the file as soon as possible.

When a node having a file comes into contact with another node having a file with the same *id*, it verifies if the two files have the same chunks. If this is the case, the second file is ignored (it will not be downloaded). Otherwise, each node will download only the chunk they do not contain, so that the file gets restored (at least to a closer state to the initial one).

Dividing the file in chunks can also lead to the following situation: multiple nodes demanding the same file from a node, but, due to a very short contact, neither node downloads the entire file. As any normal download, the transfer begins at the first chunk. This means that all nodes receive more or less the same chunks (some of the first chunks will surely be the same), and neither grabs the last chunks of the file. This is certainly an undesirable situation, giving that the first chunks will be heavily spread across the network and the last chunks will be very hard to find. In this case will be very difficult to restore the file later.

One simple solution for this issue is to send different chunks to the nodes that request the same file. As mentioned, the files hold a field that contains a chunk position. This field is used for implementing a round-robin algorithm for choosing the position where the next transfer will start. At each transfer, the position will be increased with the number of chunks transferred. This way a chunk can only be transferred for the second time if all other chunks were transferred at least once. Now, the chunks will be spread more equally among the nodes, so the files will be much easily restored.

V. Experimental Results

A. Experimental Setup

To evaluate the *Multimedia Sharing* algorithm we used the MobEmu [14] simulator. The choice was motivated by its ability to evaluate performances considering a mobility model based on real-world data. A trace for the simulator includes the time and duration of the contacts between nodes, the social relationship established between them, and also their interests.

The first trace selected is UPB 2012 [11], and contains contact data detected during an experiment conducted at the University POTELITEHNICA of Bucharest. The experiment included 66 users, laster 64 days, and in addition to the actual contact trace, it includes social information about the users, and their 6 topics they were interested in The trace information was collected by running an Android application on the mobile phones of the users which logged the contacts of the nodes.

The second trace selected is Sigcomm 2009 [15], a trace resulted from a conference held in Barcelona which implied 76 users, 154 topics, but lasted for only 6 days.

The experiments implied that each node would generate a given number of messages twice during the trace. With the messages, crumbs were generated and spread at every contact. Each node that discovers a file, or a crumb to a file that contains at least one of his topics, will generate a request for that file. Request are also disseminated until reaching a node that hold the requested file. Then the algorithm attempts to deliver the file to the solicitant node in an efficient manner.

B. Evaluation metrics

We use four metrics for testing the performance of our algorithm. The first one is *hit rate*, defined as the ratio between successfully delivered messages[2] and the total number of generated messages. Next, the *delivery latency* is defined as the time passed between the generation of a message and its delivery. Another metric we use is the *delivery cost*, defined as the ratio between the total number of messages exchanged and the number of generated messages, which shows the congestion of the network. We only count the actual data messages, and not the control messages sent when two nodes meet each other. The *hop count* is the number of nodes that carried a message until its destination on the shortest path, and should be low in order to avoid node congestion.

Obtaining the results shown in the Figure 1 implied the variation of a few features regarding the nodes and the files. First feature varied was the data memory size which received the following values: 100, 200, 500, and 1000 files. For each of these values a box plot was draw. The last box plot represents the results for running a modified version of Epidemic, an algorithm that just floods the network with messages with no regard to interest, algorithm that considers an unlimited size. For this reason, the Epidemic could be considered an almost ideal case (almost because the limited duration of the contacts is still considered).

[2]Each message we consider can carry a chunk from a multimedia file.

The second feature varied is the mean dimension of a file. For determining the size of each file being "launched" in the network, we used a normal distribution with the mean received as input parameter. The values considered as mean for the standard deviation were 500, 750, and 1000 chunks. Giving that a chunk has 1.5 KB, this leads to values between 250 KB and 1.5 MB per file. This sizes can be considered relatively small, but considering the latest improvements in the compression field, this values might become acceptable. Larger values were also tested, but the algorithm did not perform very well under these circumstances. The main reason for this poor behavior is that some contact could be really short, and that would induce a large amount of divided files which could be difficult to restore. Also there should be taken into consideration the memory congestion caused by larger files, since larger files would mean less files stored in the data memory and, consequently, some files that could have a considerable chance of being delivered would have to be dropped.

The number of files generated by each node was set to 5, 10, and 15 files (is hard to believe that a user can generate more than 15 files a day).

All the experiments were repeated, and we present the statistical results in Figure 1.

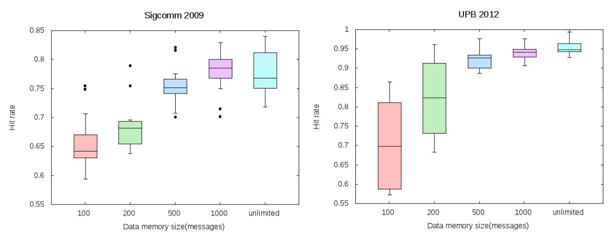
C. Discussion

Hit rate is highly dependent on the data memory size. When having a small memory size, files with high utility might be dropped due to the lack of storage and might not find the path to the destination. The fluctuation shown for the UPB 2012 trace demonstrates this fact and also that utility might not be computed so accurately when nodes have a small storage given that not so many files are exchanged. Considering that UPB 2012 trace last for 64 days, so the utility values gain some kind of stability, hit rate values are about 95% for a data memory size of 1000 messages, which is a promising value.

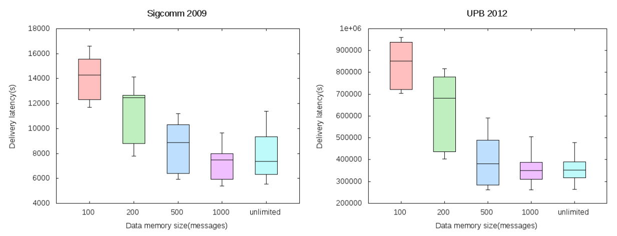
Considering the delivery latency, this decreases with the increase in the data memory size. This can be interpreted as a large memory size can contain more messages which can find their destination faster. For a small memory size, a lot of messages, maybe with high utility values might be dropped and might travel for a long time to find their destination.

The value of the delivery count is also directly dependent on the data memory size, this happening because a larger memory size induces a lot of message exchanges between nodes. The very low values for the Epidemic algorithm are demonstrated due to the fact that all nodes have an unlimited data memory size. For this reason, nodes close to each other will contain the same files, so not so many exchanges are needed. In addition, no file is dropped, so a situation when a file is dropped by a node and travels for a time and is transferred again to the same node is not possible.

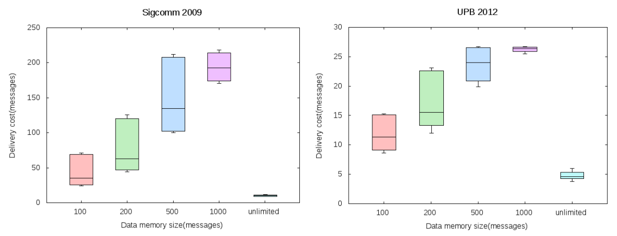
The values for hop count experience the same dependency as the ones for delivery cost. The reasons are also very similar. A larger memory size will induce a lot of data exchanges so a node might travel through many nodes. Also, for Epidemic



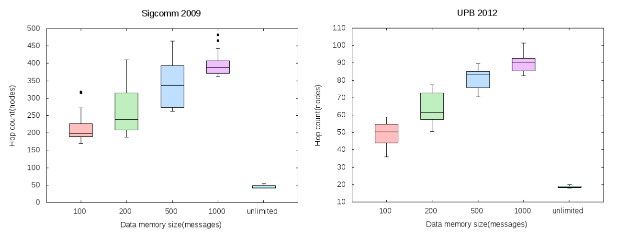
(a) Hit Rate



(b) Delivery Latency



(c) Delivery Cost



(d) Hop Count

Fig. 1. Multimedia Sharing results

algorithm, no file is dropped, so there is a good chance for files to find the shortest path to destination, this meaning the smallest number of nodes traveled.

VI. Conclusions and Future Work

In this paper, we have proposed *Multimedia Sharing*, an algorithm which deals with the sharing of multimedia content over Opportunistic Networks. Firstly, we presented SPRINT, the algorithm which was used as a base for Multimedia Sharing, then the additional features that Multimedia Sharing implements and the way it works and, finally, the results obtained for the evaluation of the algorithm in simulation.

A first step for future work would be the implementation of an Android application which uses the presented algorithm. The application should let the user see a list of the files shared by the other nodes and then choose some of them for downloading. We are really enthusiastic regarding the behavior of such an application in a real scenario.

Acknowledgment

The research presented in this paper is supported by projects: Data4Water, H2020-TWINN-2015 ID. 690900; and DataWay, PN-II-RU-TE-2014-4-2731. We would like to thank the reviewers for their time and expertise, constructive comments and valuable insight.

References

[1] K. Fall, “A delay-tolerant network architecture for challenged internets,” in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 27–34.

[2] R.-I. Ciobanu, C. Dobre, and V. Cristea, “Sprint: social prediction-based opportunistic routing,” in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*. IEEE, 2013, pp. 1–7.

[3] R. I. Ciobanu, C. Dobre, and V. Cristea, “Social aspects to support opportunistic networks in an academic environment,” in *Ad-hoc, Mobile, and Wireless Networks*. Springer, 2012, pp. 69–82.

[4] Y. Kryftis, C. X. Mavromoustakis, G. Mastorakis, E. Pallis, J. Mongay Batalla, J. J. Rodrigues, C. Dobre, and G. Kormentzas, “Resource usage prediction algorithms for optimal selection of multimedia content delivery methods,” in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5903–5909.

[5] D. Chaffey, “Mobile marketing statistics compilation,” http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/, Accessed on March 15, 2016.

[6] Cisco, “Cisco Visual Networking Index,” http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-forecast-qa.html, Accessed on February 10, 2016.

[7] M. Conti, S. Giordano, M. May, and A. Passarella, “From opportunistic networks to opportunistic computing,” *Comm. Mag.*, vol. 48, pp. 126–139, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866991.1867009

[8] P. Hui, J. Crowcroft, and E. Yoneki, “BUBBLE Rap: social-based forwarding in delay tolerant networks,” in *Proc. of the 9th ACM int. symp. on Mobile ad hoc networking and computing*, ser. MobiHoc ’08. New York, USA: ACM, 2008, pp. 241–250. [Online]. Available: http://doi.acm.org/10.1145/1374618.1374652

[9] W. Moreira, M. de Souza, P. Mendes, and S. Sargento, “Study on the effect of network dynamics on opportunistic routing,” in *Proceedings of the 11th international conference on Ad-hoc, Mobile, and Wireless Networks*, ser. ADHOC-NOW’12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 98–111. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31638-8_8

[10] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft, “Distributed community detection in delay tolerant networks,” in *Proc. of 2nd ACM/IEEE inter. workshop on Mobility in the evolving internet architecture*, ser. MobiArch ’07. New York, NY, USA: ACM, 2007, pp. 7:1–7:8. [Online]. Available: http://doi.acm.org/10.1145/1366919.1366929

[11] R.-I. Ciobanu, R.-C. Marin, and C. Dobre, “Interaction predictability of opportunistic networks in academic environments,” *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 8, pp. 852–864, 2014.

[12] Amazon, “64gb microsdxc memory card,” http://tinyurl.com/hnd3y22, Accessed on March 10, 2016.

[13] D. Murray, T. Koziniec, K. Lee, and M. Dixon, “Large mtus and internet performance,” in *High Performance Switching and Routing (HPSR), 2012 IEEE 13th International Conference on*. IEEE, 2012, pp. 82–87.

[14] R. I. Ciobanu, C. Dobre, and V. Cristea, “Social aspects to support opportunistic networks in an academic environment,” in *Proceedings of the 11th international conference on Ad-hoc, Mobile, and Wireless Networks*, ser. ADHOC-NOW’12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 69–82. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31638-8_6

[15] A.-K. Pietilainen and C. Diot, “CRAWDAD dataset thlab/sigcomm2009 (v. 2012-07-15),” Downloaded from http://crawdad.org/thlab/sigcomm2009/20120715, Jul. 2012.