INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS

Int. J. Commun. Syst. 2011; **00**:2–32

Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/dac

Social-Aware Delay Tolerant Networking for Mobile-to-Mobile File Sharing

Claudio E. Palazzi*, Armir Bujari

Department of Pure and Applied Mathematics

University of Padua

Padua, Italy

cpalazzi@math.unipd.it, abujari@studenti.math.unipd.it

SUMMARY

Peer-to-Peer (P2P) overlay networks and Mobile Ad-hoc Networks (MANETs) share many key characteristics such as self-organization and decentralization; they also face the crucial challenge of providing connectivity in a decentralized, dynamic environment. However, when considering ad-hoc networks composed by mobile devices such as smartphones, we cannot rely on the continuous end-to-end path between peers as for classic Internet P2P applications; rather, we have to deal with low node density that creates mobile disconnected networks. Porting the P2P paradigm into mobile networks to create a Mobile-to-Mobile (M2M) file sharing application will create a modern type of Delay Tolerant Network (DTN). In this context, we discuss our new approach for P2P file sharing that considers networks composed by mobile smartphones. As innovative feature, we leverage on peer mobility to reach data in other disconnected networks by implementing a DTN-like store-delegate-and-forward communication model, where a peer can delegate unaccomplished file download tasks to other peers. In order to increase the chances of eventually receiving the requested file while reducing the number of transmitted messages and data, social awareness is exploited by nodes to delegate unaccomplished tasks only to peers that are expected to be encountered again in the future. Copyright © 2011 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: *File Sharing; M2MShare; Mobile; Social-Aware Network*

1. INTRODUCTION

The mobile user is facing many options for wireless access with highly varying characteristics, including shorter and longer disconnection periods. Indeed, mobile phones have already evolved from simple voice communication means into powerful devices able to provide a variety of services to users, varying from complex multimedia documents management, personal productivity applications and all sort of connections to the Internet [1, 2]. They are growing in popularity and might eventually become the dominant mode by which users interconnect. It seems straightforward to export a popular application such as file sharing into the new scenario of smartphone networks. In this context, different from the classical wired P2P file sharing applications, mobile users could exchange data in proximity of each other by establishing opportunistic, proximity-based P2P ad-hoc connections [3, 4, 5].

These mobile opportunistic networks could be comprised of human-operated mobile devices moving in restricted physical spaces, such as conferences, university campuses, refectories, clubs and in many other social settings. For instance, they could include networks of commuters sharing every morning and evening the same train/bus. In essence, they are characterized by nodes with heterogeneous contact rates, unpredictable mobility and limited information; communication in such settings relies on both opportunistic multi-hop forwarding and physical carrying of messages by mobile nodes. This kind of scattered connectivity does not allow the continuous communication that would be needed by applications such as VoIP or videostreaming; yet, it would still be useful for appealing applications such as Video on Demand, ring tones exchange, profile transmission, file sharing, location/proximity-based information delivery, etc. [5, 6, 7, 8, 9].

While research into routing in mobile environments is not new, researchers have for many years assumed node encounters to be random. In reality, mobile nodes are of course used by people, whose behaviors are better described by social models. This opens up new possibilities for routing, since

*Correspondence to: Claudio Enrico Palazzi, Department of Pure and Applied Mathematics, University of Padua, Via Trieste, 63 - 35131, Padova, Italy

the knowledge of behavior patterns allows better routing decisions to be made [16, 17]. In this work, we exploit this idea of social relations between users operating mobile wireless devices and provide a proof of concept implementation of P2P file sharing application among mobile users. To this aim, we adopt a Delay/Disruption Tolerant Networking (DTN [10]) type of solution for the mobile disconnected networks. Indeed DTNs have been considered to support communication in situations with intermittent connectivity, long/variable delay, and high error rates: characteristics that common them with the wireless mobile world [9]. They use an asynchronous communication model and message replication techniques to maximize the probability of data delivery to the destination.

M2MShare is a P2P file sharing application for ad-hoc disconnected networks deployed on smartphones. It builds an application overlay network where routes are set up on demand by the search algorithm, maintained as long as necessary (e.g. transfer finished or mobile node out of reach area), closely matching network topology [8, 12]. Furthermore, M2MShare ports the DTN paradigm into the mobile world, addressing the node density issue by providing means for an asynchronous data exchange similar to that of DTNs. The idea of a DTN is modeled in an infrastructure-less environment where both source of the request and destination of the data are the same entities and where intermediary nodes (*servants*) can store-delegate-and-forward-back the requested user data toward the source.

Although DTN is not a synonym of bundle protocol, the great majority of DTN applications are based on it [13, 14, 15]. To avoid confusion we point out that M2MShare does not implement such protocol. Instead, we provide our own mechanism for data exchange. The reason for this lies in the fact that our application works differently from traditional data transfer protocols. Indeed, when delegating a task to servant(s), neither the node that generated the task nor the servant(s) knows which node will be the final destination of that particular task. Thereby, the use of classical DTN protocols such as the bundle protocol cannot be applied to our case. Yet, the DTN paradigm is taken as an inspiration for the use of servants in order to physically carry tasks from one network to another.

Routing in DTNs is concerned in building a forward route from source to destination and social-aware approaches have been considered to this purpose [16, 17, 18, 19, 20, 30]. In our application scenario the forward route consists of a single hop that is, from servant toward the request originator. Our focus is on exploring a new mechanism that allows peers to explore and download content available outside their reach area, provided in other local disconnected networks. Different from previous schemes, our idea is to reach data in other mobile networks by leveraging on node mobility and periodic encounters among users even if they are not aware of these social proximity (e.g. commuters utilizing the same train every morning even if they do not know each other personally).

Indeed, in our solution, a peer can delegate an unsatisfied or unaccomplished query or download task to another encountered peer; for the sake of clarity we refer to this operation as *delegation* in the rest of the paper. For example, if a node A, wants a certain file not available in the established network, then node A might delegate to some of these nodes (servants) the task of finding that file. However, it would be pointless to assign this task to a servant that will be never met again: it would not be able to forward back the output to A, even if found. Instead, M2MShare uses contact information to make routing decisions for delegations; nodes keep track of other nodes encountered in recent past and servants are elected among the frequently encountered devices.

In summary, main contributions of our work are:

1. a new paradigm of use of P2P solutions that matches file sharing with mobile users, fostering new applications;
2. a Bluetooth-based opportunistic overlay network capable of message routing between devices not in the in-reach area;
3. an example of DTN communication adapted for the mobile world;
4. a protocol that dynamically establishes forward routes (delegations) by exploiting users real life periodic encounters;
5. a smart criteria for download task delegation so as to speed-up file transfer through parallel operations, while reducing the transmission redundancy.

The remainder of this paper is organized as follows. Section 2 summarizes background information related to our work. In Section 3, we present M2MShare protocol stack, describing the duties and responsibilities of each individual layer. Section 4 introduces the *PresenceCollector* service which periodically gathers presence information of in reach area devices. Section 5 gives some insights inherent to the servant election strategy along with a study proving that the delegation technique serves its purpose. Section 6 surveys the file division strategy, demonstrating its efficiency compared to other possible division strategies. Finally, concluding remarks are given in Section 7.

2. BACKGROUND

During domain study we encountered some frameworks in the literature that inspired and guided us through our work. In this section we present some of these works that influenced some design aspects of M2MShare.

2.1. Delay/Disruption Tolerant Network

The TCP/IP protocol suite has been a great success at interconnecting communication devices across the globe [31]. Although new wireless technologies have appeared, connectivity on the Internet relies primarily on wired links which are continuously connected in end-to-end, low-delay paths between sources and destinations.

While Internet relies on TCP/IP at interconnecting devices, communication outside it, where power-limited mobile wireless, satellite and interplanetary communications are developing requires the invention of new protocols, each supporting specialized communication requirements. These networks have fundamental properties that make them incompatible with Internet: each is good at passing messages within its networks, but not able to exchange messages between networks. Communication characteristics inside one network are relatively homogeneous therefore, spanning two network regions requires the intervention of an agent that can translate between incompatible networks characteristics and act as a buffer for mismatched networks delays.

DTNs started as a network of regional networks [33] but now the association between nodes and territorial regions is no more strict [32]. DTN achieve interoperability by accommodating long delay between and within networks and translating between network communications characteristics. Therefore it can accommodate the mobility and limited power evolving wireless communication devices.

Many evolving and potential networks do not comply with the Internet underlying assumptions. These networks are characterized by:

- **Intermittent connectivity:** Due to mobility of nodes or their limit in wireless radio range there might be no end-to-end path between source and destination. Also, device heterogeneity may inhibit interworking; and radio range and interference may limit communications. Therefore, communication using the TCP/IP protocols does not work. Other protocols are required.
- **Long or variable delays:** In addition to intermittent connectivity, propagation delays between nodes and variable queuing delays at nodes contribute to end-to-end delays that can defeat Internet protocols and applications that rely on quick return of acknowledgements or data.
- **Asymmetric data rates:** The internet supports moderate asymmetries of bidirectional data rate for users with cable TV or asymmetric DSL access. But if asymmetries are large, they defeat actual protocols.
- **High error rates:** Bit errors on links require correction (added redundancy and processing time) or retransmission of the entire packet. For a given link-error rate, fewer retransmissions are needed for hop-by-hop than for end-to-end retransmission.

DTNs overcome these problems by using a store-and-forward message switching (Figure 1). This is an old method, used by pony-express and postal systems since ancient times. Whole messages or pieces of such messages are forwarded from a storage place on one node to a storage place on another node, along a path that eventually reaches destination.

Store-carry-and-forward methods are also used in today's voicemail and email systems, although these systems are not one-way relays (as shown in Figure 1). Furthermore, DTN routers need

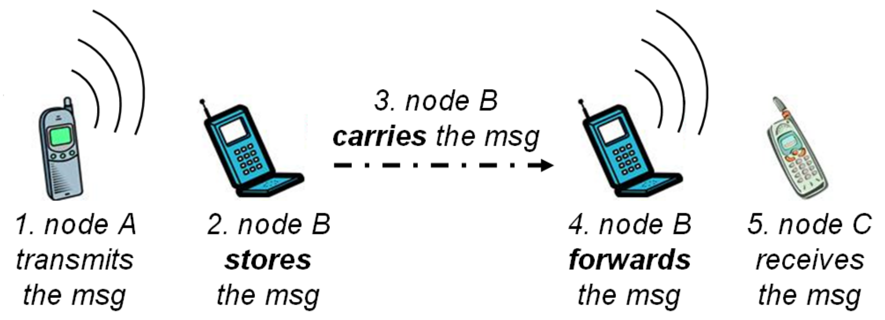

Figure 1. Store-carry-and-forward packet switching between DTN routers.

persistent storage for their queues as opposed to Internet routers that use short-term storage provided by memory chips. This, for the following reasons:

- a communication link to the next hop map not be available for a long time;
- one node in a communicating pair may send or receive data much faster or more reliably than the other node;
- a message, once transmitted, may need to be retransmitted if an error occurs at an upstream (toward destination) node or link, or if an upstream node declines acceptance of a forwarded message.

DTNs are still an area of research and a lot of other features would be worth mentioning but are outside of the scope of this work. For more insights refer to [10, 11].

2.2. *Optimized Routing Independent Overlay*

Optimized Routing Independent Overlay (ORION [12]), is a special-purpose approach for P2P file sharing tailored to MANETs. It comprises a framework for construction and maintenance of an application-layer overlay network that enables routing of various types of messages such as queries, responses and file transmissions.

ORION designers identify the maintenance of static overlay connections as the major bottleneck for deploying a P2P file-sharing system in MANET. Instead, the overlay connections are set-up on demand and maintained only as long as necessary (e.g. until file transfer is completed or disconnection occurs). It provides an efficient algorithm for keyword based file search by combining application-layer query processing with techniques known from Ad Hoc on Demand Distance Vector (AODV [34]) and Simple Broadcast Protocol for MANET. In the following we provide some insights on the ORION building blocks and their modus operandi.

- **Indexing:** Each mobile device maintains a local repository, consisting of a set of files stored in the local file system and provides searching capabilities for all files in the repository. The authors do not provide insights on how files are locally indexed and globally identified by the search algorithm; they simply state that files are associated with a unique identifier.
- **Routing:** ORION maintains two routing tables, a response routing table, and a file routing table. The response routing table, as in AODV is used to store the node from which a query message has been received as next hop on the reverse path. In this way, a node is able to return responses to the enquiring node without explicit route discovery. The file routing table stores the alternative next hops for file transfers based on the file identifier.
- **Transport:** The ORION transfer protocol utilizes the routes given by the file and response routing tables for transmission of control and data packets. The file routing table may store several redundant paths for copies of the same file. Due to changing network conditions, the sender of a file might change during a file transfer; thereby, control over the transfer is kept on the receiver side and, opposed to TCP, the ORION transfer protocol does not maintain an end-to-end semantic. For transfer, a file is split into several blocks of equal size. Since the maximum transfer unit of the mobile network is assumed to be equal between all neighboring nodes, the block size can be selected such that the data blocks fit into a single packet. The receiver sends a DATA-REQUEST message for one of the blocks along the path given by the file routing tables. Once the DATA-REQUEST reaches a node storing the file in the local repository, the node responds with a DATA-REPLY message, containing the requested block of the file.

The mentioned overlay organization technique of our M2MShare solution was borrowed by the ORION project: connections are established on demand and maintained as long as necessary. What ORION misses to address is the low node density issue, which is crucial for a smartphone based file sharing application. Using a DTN store-delegate-and-forward mechanism we address this problem by leveraging node mobility, reaching data content available on other disconnected networks.

2.3. Routing in MANETs

With the rapid and growing penetration of mobile hand-held devices with built-in wireless technology, more and more users use them to query and share interesting data among each other. Users could interconnect and exchange data just passing by each other, without the need of any infrastructure based network. Such a community-wide network formed by the mobile devices is an example of a mobile ad-hoc network.

In this networking environment communication links are transient and short in time, moreover in sparse mobile ad-hoc networks a path between source and destination might not even exist. To deal with these problems delay tolerant techniques have been exploited, where a mobile node physically carries data for some time until it moves within the communication range of some other node [17, 18, 20, 21, 22]. The decision whether to forward or not the data to the new contact is based on some algorithm running locally on each node.

Epidemic Routing [17] is one of the first routing schemes proposed for intermittently connected networks. Each node maintains a list of all messages it carries, whose delivery is pending. Whenever it encounters another node, the two nodes exchange all messages that they don't have in common. This way, all messages are eventually spread to all nodes, including their destination (in an epidemic manner). This method achieves lower delivery times but it is wasteful for network resources. One simple approach to reduce the overhead of flooding and improve its performance is to only forward a copy with some probability (p) [23]. A different, more sophisticated approach is that of *History-based* or *Utility-based Routing* [22, 21]. Here, each node maintains a utility value for every other node in the network and a node forwards message copies only to nodes with a higher utility by at least some pre-specified threshold value U_{th} for the messages destination. These schemes are still flooding-based in nature and are faced with an important dilemma when choosing the utility threshold. Too small a threshold and the scheme behaves like pure flooding. Too high a threshold and the delay increases significantly.

Single-copy schemes have also been extensively explored in [18, 19, 20]. These schemes are based on the idea that mobile nodes are operated by people, whose behaviors are better described by social models. This has opened up new possibilities for forward routing, since the knowledge that behavior patterns exist allows better decisions to be made. *SimBet* routing studies the small-world phenomenon of human society and uses ego-centric centrality and its social similarity to guide data forwarding. Messages are forwarded toward nodes with higher centrality. Similarly, *Bubble Rap* focuses on community and social centrality and nodes are structured into communities. High popularity nodes and community members of the destination are selected as relays. However, since source and destination may be faraway from each other, the delay for the destination to get the data from the source may be long.

Different from wired networks, MANETs exhibit unpredictable topology and as such the adopted mobility model has a crucial role in testing the performance of routing algorithms. Designing new algorithms or choosing between existing ones depends on different factors varying from the context the software is deployed and operable to the mobility patterns of users operating the software. Synthetic mobility models [24, 25] have been largely used to measure quantitative aspects of routing protocols but these are not sufficient as they do not capture reliably the properties of movement in the real life scenarios. Inter-contact times and contact durations are typical metrics for characterizing mobility in sparsely populated DTNs and their distribution does have practical implications. Musolesi et al. [26] show that simple mobility models have very different properties in terms of inter-contact time and contact durations compared to real user traces. [27] captures several different mobility characteristics at a lower level of abstraction than many other models have. This model is heterogeneous in both time and space and produces similar distributions of inter-contact times and contact durations as real user traces [28].

In our application scenario source and destination are the same entities whether the servant is an intermediary node along the path toward the destination. The forward path consists of a single hop,

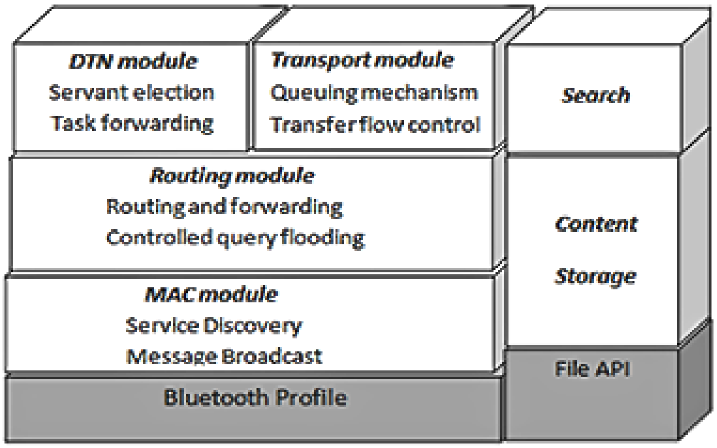

Figure 2. M2MShare protocol stack; gray blocks are proprietary.

from servant to destination; we have chosen this modus operandi to exploit social relation aiming at increasing the chances of eventually receiving the requested file while reducing the number of transmitted messages and data. Servants are hence selected by the expectation of encountering them again in the future. Therefore, the routing algorithms mentioned here do not suit our case, but their combination with our solution could be an interesting future extension, where servants could use multiple nodes in forwarding the task.

3. SOFTWARE ARCHITECTURE

Work in MANETs is diverse and touches various aspects of computer science and communication engineering. In this work we had to focus our attention on some aspects rather than all of them. A protocol stack was designed, providing core functionalities that a P2P system must have. The protocol stack is described with the help of Figure 2 and main modules are listed below.

- **Search module:** The searching scheme implemented follows the local indexing strategy where files and index are locally stored [39];. We use and inverted index list to index files under their description and provide a keyword query modeling scheme.
- **DTN module:** This module is responsible for servant election and task delegation. Studies in routing algorithms for challenging environments such as MANETs demonstrated that they

have a social dimension built in [35, 36]; knowing that behavior patterns exist allows better routing decisions to be made. We exploited the fact that certain users frequently encounter each other (e.g. by taking the same bus in the morning, by eating in the same cafeteria at lunch, etc.) in order to dynamically build a DTN path from source to destination.

- **Transport module:** It provides the task queuing mechanism and task lifecycle management. An important part of this module is the communication protocol for data packet exchange among nodes. Also, it provides a smart file division strategy, which allows for parallel and hence faster download while avoiding redundancy on downloaded data.
- **Routing module:** This module provides message forwarding capabilities to our overlay network and implements a controlled flooding technique alike AODV. It implements the overlay organization, i.e. connection establishment and maintenance, as done by ORION.
- **MAC module:** This module provides service discovery and message broadcast facilities for peers in our network. The heart of this module is a fundamental service called PresenceCollector which periodically gathers presence information about in-reach area devices so as to determine which nodes are frequently met and have a reasonable expectation to be met again in future (these nodes will be used as servants in order to propagate unaccomplished or unsatisfied tasks).

4. PRESENCE COLLECTOR SERVICE

M2MShare actively collects presence information of encountered devices that are in direct reach area of communication so as to exchange data and assign delegations. This job is handled by an active daemon of the system, called *PresenceCollector*. It is important to understand that two nodes A and B are not aware of each other immediately after they enter in communication range. Rather, while being in communication range, they learn the existence of the other as soon as one of the two initiate a scan phase by sending out presence beacons and the other node answers. To this aim, the PresenceCollector is modeled as an active daemon which periodically scans the network with a periodicity configurable by the user; a high frequency (e.g. a period of 1 s) is not reasonable from

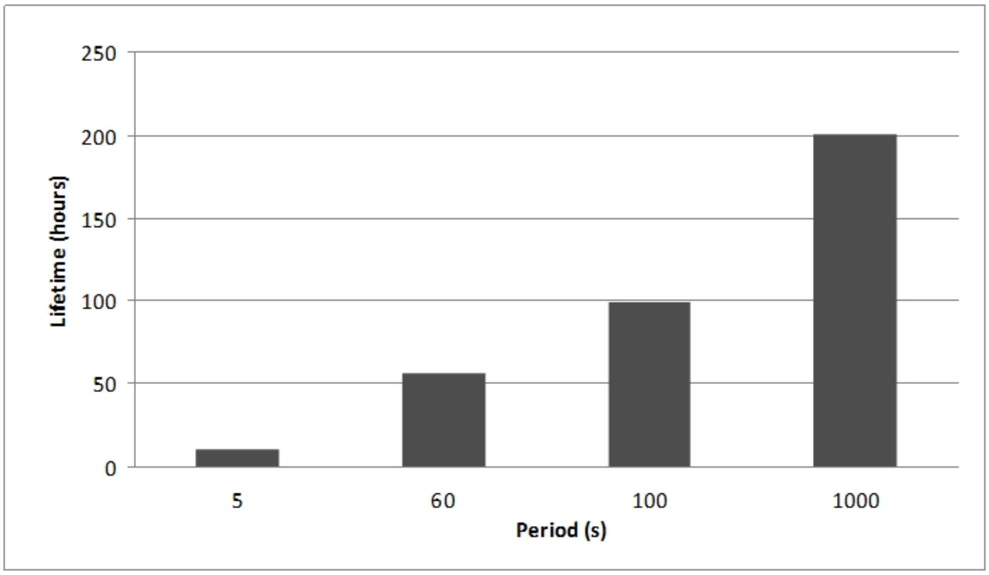

Figure 3. Battery consumption of a Nokia 5730 XpressMusic running PresenceCollector service with different beacon periods.

the energy preservation point of view as software is actively running, consuming network bandwidth and device energy.

Instead, having a low frequency (e.g. a period of 10 min), the device may miss an encounter with another node that lasts, for instance, just 3 min, and hence the chance to elect and delegate an unaccomplished task to a potential good servant. Also a servant might miss the chance to initiate an output forward of a previously delegated task while the delegator device is in-reach area but not yet discovered by its periodic discovery service. Both this situations refer to a particular class of tasks in our system whose creation and execution depends directly on the inquiry frequency.

To better understand the impact of beacon periods on energy consumption, the histogram in Figure 3 shows the battery lifetime of a Nokia 5730 XpressMusic cellphone running only PresenceCollector service with different periods between two consecutive scans (5 s, 60 s, 100 s, and 1000 s). The results are quite intuitive and expected.

Tackling the problem of data transfer, let's assume that device A has delegated to another device B some particular download task and that device B was able to accomplish it; then, the next time they will encounter each other, the servant B will notify the client A that it is ready to forward the

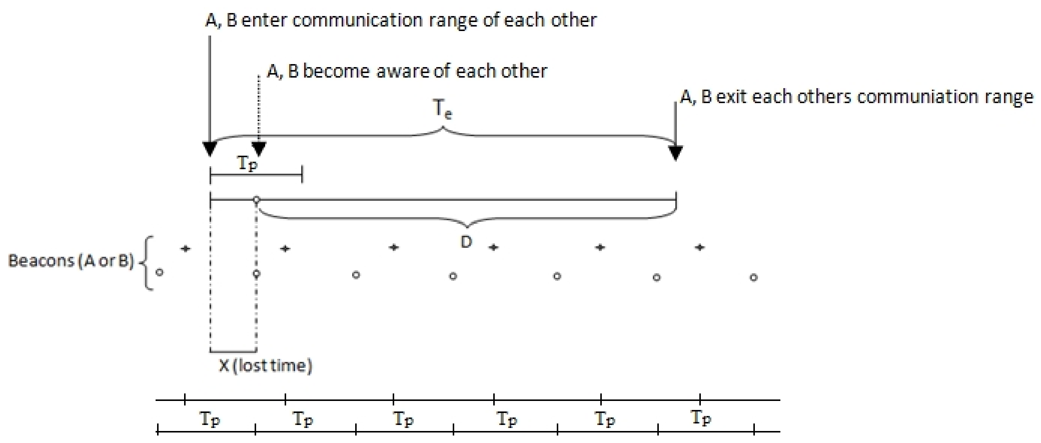
A, B enter communication range of each other
A, B become aware of each other
A, B exit each others communiation range
T_e
Tp
D
Beacons (A or B)
X (lost time)
Tp

Figure 4. Servant senses client presence and notifies (round ball) him that a previously delegated task output has been accomplished.

output of that particular task. In this case, the quantity of data that the client A will download from device B depends on different factors (Figure 4):

1. the time interval between two subsequent beacon transmissions *(T_p)*;
2. the duration *(D)* of the established communication link between node A and node B; this embodies the amount of time that might be actually used for data transfer and that is generally smaller than the physical encounter duration *(T_e)*;
3. the bandwidth available on the client side for data transfer *(B_w)*, which we consider to be constant during all link establishment time, neglecting factors such as interference, protocol activities and other possible on-going transfers (upload or download).

For the sake of simplicity, communication delays and queuing delays are not considered in the study; they are both considered as negligible amount of time. The two nodes may start the file transfer only after they become aware of each other; this happens when the first presence beacon is sent (by A or B) after the two nodes has become in communication range. Therefore, as Figure 4 shows, every time there is a lost time X before starting the file download.

Theorem: The average time lost for data transfer between devices A and B entering in communication range with each other, before becoming aware of each other, is

$$E[X] = \frac{1}{3} * T_p \tag{1}$$

Proof: Let X denote the interval between the moment at which the distance between two nodes A and B becomes smaller than the transmission range and the moment when the first node between A and B broadcasts a beacon message. Let X$_A$ be the interval between the moment at which node A and node B enter in the transmission range of each other and the moment when node A transmits its beacon message. Similarly, X$_B$ is the interval between the moment at which node A and node B enter in the transmission range of each other and the moment when node B transmits its beacon message. As node A and node B periodically transmit their beacon messages independently from each other, then X$_A$ is independent from X$_B$ and X = min (X$_A$, X$_B$).

From the single node perspective the probability P(X$_A$ $\leq$ t), where t ϵ [0, T$_p$] denotes the lost amount of time, is:

$$P(X_A \leq t) = \frac{t}{T_p} \Leftrightarrow P(X_A > t) = 1 - \frac{t}{T_p} \tag{2}$$

Since the two nodes independently transmit their beacons, the lost amount of time is characterized by the following probability function:

$$\begin{aligned} F(t) = P(X \leq t) &= P(min(X_A, X_B) \leq t) \\ &= 1 - P(X_A > t) * P(X_B > t) \\ &= 1 - (1 - \frac{t}{T_p})^2 \end{aligned} \tag{3}$$

In order to compute the expected time lost, we need to integrate the product with its density function. To this aim we know that *f(t) = dF(t)/dt*, where F(t) and f(t) denote the Cumulative Distribution Function (CDF) and the Probability Density Function (PDF), respectively. The resulting formula for the expected time lost is hence expressed by (4).

$$\begin{aligned} E[X] &= \int_0^{T_p} f(t) * t \, dt \\ &= \int_0^{T_p} \frac{2}{T_p} * (1 - \frac{t}{T_p}) * t \, dt \\ &= \frac{1}{3} * T_p \end{aligned} \tag{4}$$

From this result, we can derive the average data quantity that can be transferred when two nodes enters in the transmission range area of each other.

Corollary: Given T_e the average time node A and node B stay in the communication range of each other, T_p the frequency of periodic inquiry of each node and B_w the bandwidth available at each node, then the average data quantity transferred is

$$T_d = (T_e - \frac{T_p}{3}) * B_w \tag{5}$$

<u>Proof:</u> Given the overall expected time of the encounter (T_e) and the expected lost time (T_p/3) after which both devices can initiate the transfer, then the remaining time for data transfer is:

$$E[D] = T_e - T_p - T_a - T_c \tag{6}$$

Since we are under the assumption that T_e (queuing delay) and T_e (communication delay) are negligible and we know the average link duration between the two nodes from (4), then we can compute the average data quantity transferred after link establishment, which is:

$$T_d = (T_e - \frac{T_p}{3}) * B_w \tag{7}$$

To understand the impact of this lost time on downloads, depending on the beacon frequency, we have run a simulative experiment based on the introduced mathematical modeling and the outcome is reported in Figure 5 and Figure 6. In our simulations, we have fixed the physical encounter duration between nodes A and B; moreover, we have considered different periods of time between

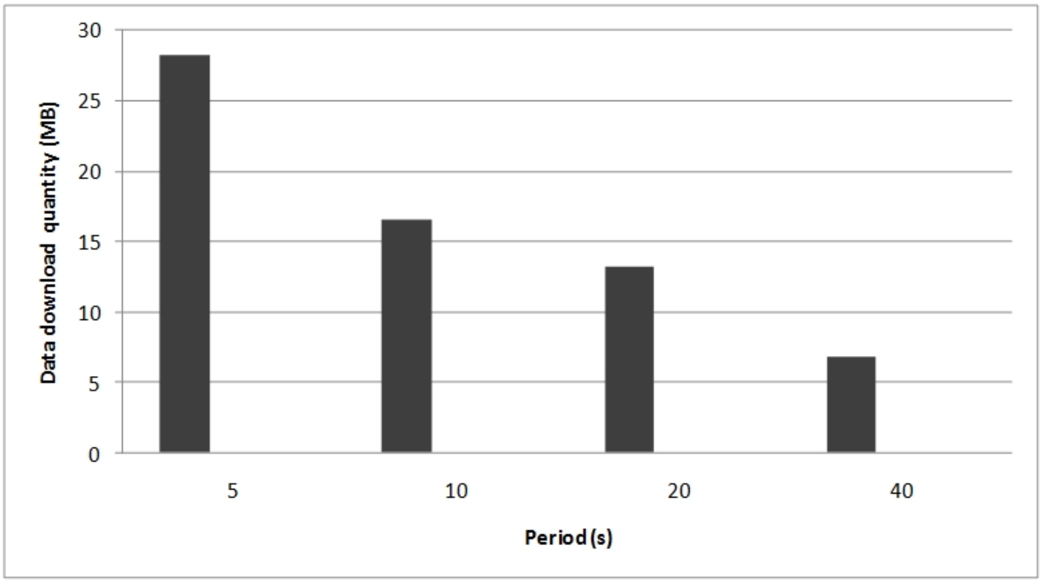

Figure 5. Average data download quantity (MB) between servant and client with an encounter period of 20 s.

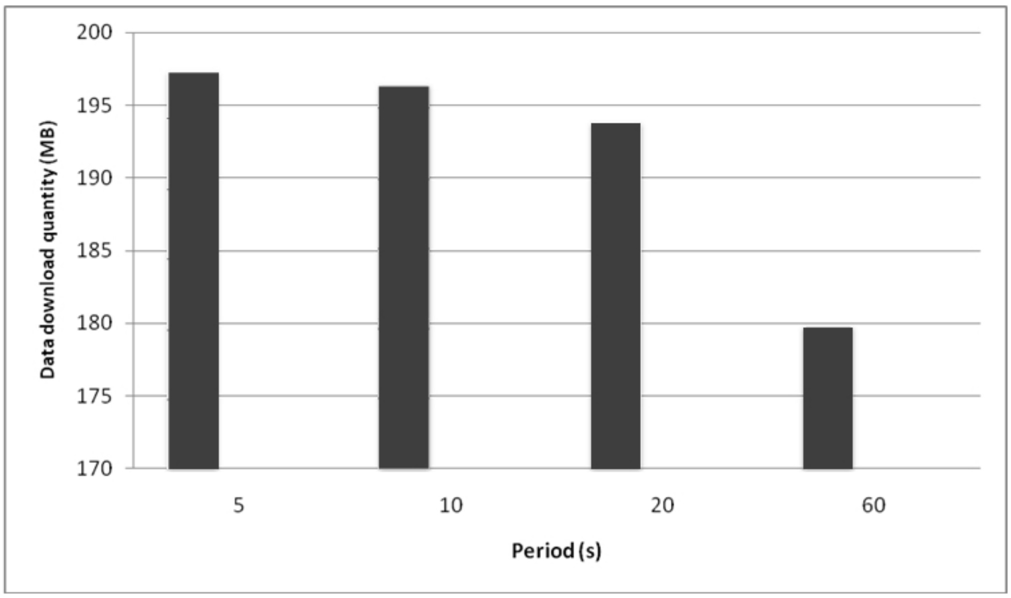

Figure 6. Average data download quantity (MB) between servant and client with an encounter period of 200 s.

consecutive beacons (i.e. 5 s, 10 s, 20 s, and 40 s). In Figure 3 we have seen that to a lower beacon frequency corresponds a lower energy usage; but, as expected, we see now in Figure 5 and Figure 6 that it also corresponds to a lower transfer performance.

5. SERVANT ELECTION STRATEGY

Our software makes use of delegations to extend a peers reach area to data files available in other disconnected networks. Delegating unnsatisfied/unnacomplished tasks to all peers in the system is bandwidth and energy consuming, therefore a criterion is needed in choosing one peer instead of others. In this section we describe in dettail the process of electing a servant peer among peers encountered during software run time. Also, a study proving that the delegation technique serves its purpose is provided.

5.1. Election Strategy

M2MShare implements an asynchronous communication mode between peers where a client peer can delegate an unsatisfied, unaccomplished task to a servant peer. By task delegation, is meant that task is locally encoded by the client and communicated to the servant, which locally stores it for later execution. When a servant accomplishes the task, it is ready to forward the output to the client peer that request the task accomplishment. The forward takes place the next time they encounter each other. In essence, we leverage on peer mobility to reach data in other disconnected networks where they might be available. Obviously, each delegated task has a TTL (time to live); the task is stored in the servant's local storage and can be forwarded to the client that delegated it only until the TTL is not expired. The servant does not re-schedule a task that is unaccomplished at TTL expiration.

While in DTNs there are pre-deployed entities that store-and-forward data along the destination path (routers), M2MShare achieves this functionality in an infrastructure-less environment, where this forward route is established dynamically along the path to destination. In other words, at each hop a client peer, which in turn might be acting as servant for another peer along the chain, dynamically chooses its servant to which delegate the task. Unlike DTNs where source and destination are different entities in our case both source of the request and destination of the data (task output) are the same entities, while servants are intermediary nodes along the chain, which store-delegate-and-forward back the task output.

As stated earlier, mobile devices are used by people whose behaviors are better described by social models and the fact that behavior patterns exist allows better routing decisions to be made [18, 21]. Therefore, the underlying assumption of our solution is that each user has a routine of his own that matches other user's routines. For instance, a user staying at his office could be in communication range with colleagues during working hours, a user traveling by bus or train to go to work frequently encounters other commuters, the same every day.

Delegating an unaccomplished task to all the peers in the established overlay is bandwidth and energy consuming therefore a criterion is needed to choose one peer instead of others. Also, it is sound to delegate tasks to peer devices operated by users whom we expect to encounter again in the future. In this way we augment the chances of output return, in case the servant found the desired content. In the current implementation those devices that exceed the *Frequency Threshold ($Freq_{Th}$)*, number of encountered times, are elected as servants to whom the system might delegate a particular unaccomplished or unsatisfied task.

A servant device is a frequently encountered device and the concept of frequently encountered changes in time, adapting to the observed dynamics. This because the contact rate of a single device operating M2MShare might vary from day to day. Moreover, some devices frequently encounter many other devices, while others encounter a small number of them. In the first case we would want to higher the expectations of a frequently encountered device in order to choose the best devices from those repeatedly encountered. In the second case, in order to have a certain number of frequently encountered devices we should be less selective by lowering the system expectations (parameters).

The servant election algorithm, at the beginning of each day imposes a goal that needs to be achieved during that day. This goal is the *Expected Ratio (E_r)*, the number of elected peers (servants) expected during one day. Since one day's activity might differ at some level from the others, the systems tries to adapt the configuration parameters to the observed dynamics in order to achieve a better performance (Expected Ratio). Essentially, the algorithm tunes the configuration parameters using past history of observed encounters.

```
1.   PwAdapt()
2.     // calculate number of expected delegations (expected ratio)
3.     Er = L / Pw
4.     // at the end of the probation day check if ratio was achieved
5.     if (Ar >= Er) then
6.       // expectations achieved, lower probation window
7.       Pw = Pw - 1
8.       // frequently encountering, electing servant/s:
9.       // is FreqTh  low?
10.      if (Pw < 2) then
11.        Pw = 2
12.        FreqTh = FreqTh + 1
13.    else
14.      // duplicate monitoring period, lower system expectations
15.      Pw = Pw * 2
16.      // frequently encountering, electing a small number of servant/s:
17.      // is FreqTh  high?
18.      if (Pw > 30) then
19.        Pw = 30
20.        FreqTh = FreqTh - 1
```

Initially the ratio is computed by the default configured values, no history seen before and at some point in time it is expected that the algorithm will reach an equilibrium where the configuration parameters *(E_r, $Freq_{Th}$)* will be stable or will not be subject to frequent change.

To better explain how the algorithm works let us refer to the pseudo code shown below:

At **line 3**, E_r is computed, except day 0, using the information gathered the day before; we impose a goal for today's activities based on data gathered the day before. The underlying assumption is that user has its own routine and habits which do not change radically from day to day. The Expected

Ratio is computed by performing the division between the number of servant slots in the servant list *(L)* and the current probation window value *(P_w)*. There might be users operating the software that have a high number of encountered devices per day and others whom have only few of them. By dividing with P_w the algorithm can tune the parameters *(P_w, $Freq_{Th}$)* imposing a sound goal for tomorrow's activity based on the user's capability of encountering other devices.

At **line 7**, E_r is achieved and we lower the monitoring period, decrementing it, imposing a higher goal for next time. A monitoring period P_w=2 means that a peer is considered periodic if it is seen *$Freq_{Th}$* times in 2 days. In this case the device is frequently encountering nodes and electing them, so everything seems going well. Frequently encountering and electing servant/s does not necessary mean that the they are returning back the output of the delegated tasks and we do not have any instrument or criteria to determine whether this is the case or not. When the monitoring period goes below its minimum value (i.e. when it is less than 2) we increment the *$Freq_{Th}$* and probe whether this high frequency of election is induced by a probable low *Frequency Threshold*.

At **line 15**, we use a conservative approach, doubling the monitoring period and by doing so we lower the system expectations (ratio halved) for the next probation window. Ratio could not be achieved either because *Frequency Threshold* is too high or effectively we're encountering a small number of devices (e.g. the software missed all of its active sessions but one). A monitoring period P_w = 30 means that a peer is considered periodic if it is seen *$Freq_{Th}$* times in 30 days. If the monitoring period exceeds its maximum value we decrement *$Freq_{Th}$*, imposing a lower threshold for election and probe whether a low frequency of election is induced by a probable high *$Freq_{Th}$*. Incrementing or decrementing the *Frequency Threshold* might seem mind troubling and initially difficult to comprehend. We do so as we do not have any other feedback other than the number of active servants *(Active Ratio)* and Probation window *(P_w)*.

5.2. Delegation Efficiency

In this section we demonstrate that the delegation technique serves its purpose. We do so by comparing the efficiency of our system, employing delegations, against a regular system with no

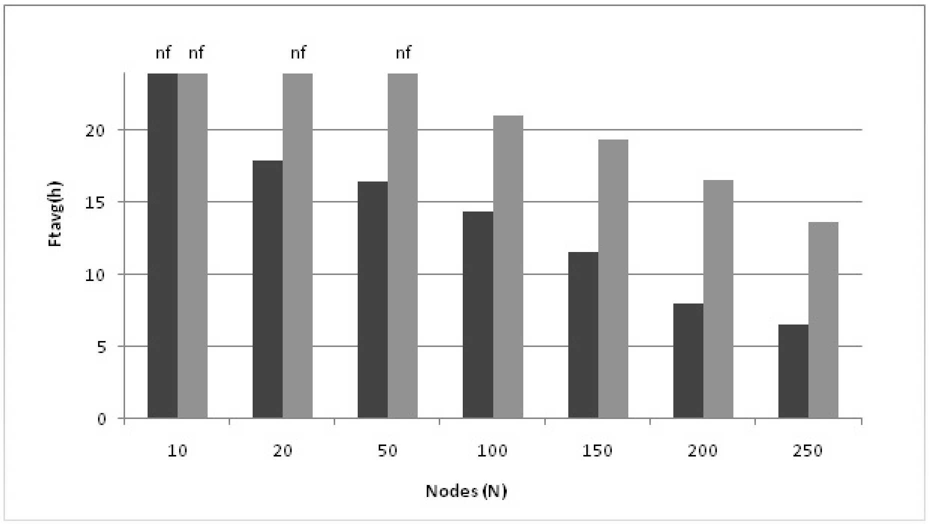

Figure 7. Comparison between the trivial strategy (black) vs. the strategy employing delegations (gray) and Fp= 10%.

delegations, which only resorts to direct exchange with nodes possessing the requested file (if found). The metrics we employ to this aim is the average found time (Ft_{avg}) for a specific file. The found time (Ft) is the time interval between the first delegation made and the time an output return for that specific file is received. If no delegation is made and the file request is directly satisfied by a file possessor, then the Ft_{avg} is equal to zero.

To this purpose we implemented the two protocols in THE-ONE [43], a DTN simulation environment for delay tolerant networks. For the sake of simplicity, delegations can be only one hop, which means that a servant peer cannot further delegate the task to other frequently encountered peers of his own.

We stated before that synthetic mobility models are not realistic and to this purpose we try to simulate a more realistic scenario provided by the *Working Day Movement Model* [27] implemented for THE-ONE. In our scenario we have a population of nodes (N), which emulate people operating M2MShare and are involved in their daily activities according to the cycle home-work-event-home. A node at home is inactive, thus the software is not operative. Nodes are uniformly distributed between the available districts in the default map available in THE-ONE and the simulation time is

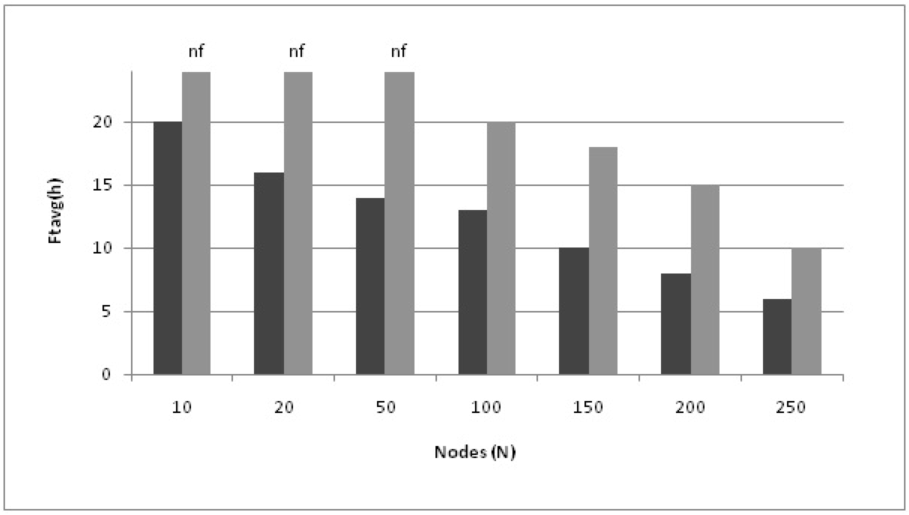

Figure 8. Comparison between the trivial strategy (black) vs. the strategy employing delegations (gray) and Fp= 20%.

set to two days, during which there are two full cycles home-work-event-home; each cycle home-work-event amounts at 12 h. The file population (*Fp*) parameter denotes the percentage of nodes that posses the required file. The node requesting the file is randomly chosen between the population and we repeat the experiment 50 times in order to achieve more accurate results, independent from the initial user's starting point.

In the first scenario (Figure 7) we consider *Fp = 10%*. The protocol not employing delegations (in gray in the chart) is able to find the requested file in a limited number of cases (when considering a number of nodes greater than or equal to 100). This is due to the trivial strategy employed by the protocol and the sparse environment of the network. The labels *nf*, associated in the chart to values equal to 24, actually means that the requested file was not found during simulation time if employing the number of nodes reported on the X-axis. As demonstrated by the chart, the protocol employing delegations outperforms the trivial one. Increasing the population (*N*), clearly corresponds to also increasing the number of nodes that posses the file; consequently, the time needed to find the requested file decreases. Similar, Figure 8 shows that by increasing the file population (*Fp* = 20% in the chart), the time to find the requested file decreases as, again, we have an increment in the number of nodes in the population that posses that file.

6. FILE DIVISION STRATEGY

The majority of file transfer applications in the market follow a client-server paradigm where devices pair with each other for all the duration of data transfer. If a disconnection takes place (e.g. because devices are not in-reach area anymore) file transfer has to restart from the beginning and already downloaded data are of no use. Pairing for all the duration of file exchange is desirable if possible, but taking in consideration the mobility of users in our scenario and that established connections are opportunistic and short in time, the chances of this happening reduce drastically. As said earlier we would like that software be entirely user transparent and as such a file download should run automatically whenever possible. For instance, user might enter a coffee shop; drink his coffee in say 2 min and during this time software is actively running and a file exchange might have started. Also, part of user daily routine may involve taking the subway while the software is running and operative. A borderline situation might be that of a user walking in the street where mobility is continuous and a file transfer might initiate even for a brief period of time.

Some P2P software deployed on the wired Internet divide the file into data chunks (BitTorrent, Gnutella [38, 37]), which are the atomic transferable parts. Here, we have real time vision of what is happening, which data is being downloaded and from whom. This is a good starting point but taking into consideration the possibility of overlapping data piece delegations and in order to increase the chances of eventually receiving the requested file while reducing the number of transmitted messages, we require a more flexible file division strategy.

M2MShare provides a new file division strategy where a file can be downloaded in pieces and a piece size varies. The file is seen as map of non overlapping intervals of variable length that need to be downloaded (Figure 9). For the remainder of this section we are using the term file server to denote both the device which is in posses of the file and the device to which we delegate a file piece download.

When a user chooses to initiate a file download, a task is created and scheduled for execution. Initially there is only one interval to be downloaded that is the entire file *[0, fileLength]* (Figure

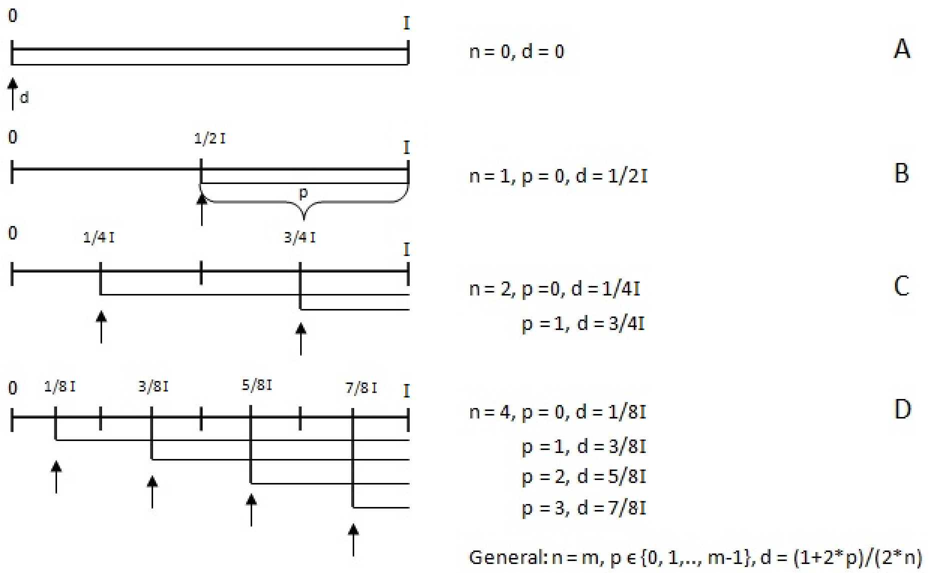

Figure 9. M2MShare file division strategy.

9-A). Once a file server is in reach area, a DATA-REQUEST is issued containing the missing data interval, in this case *[0, fileLength]*. If there is more than one file server in reach, a transfer might be initiated with each one of them; in this case, to each execution flow is assigned an interval that needs to be downloaded and each interval corresponds to one potential file piece.

The starting point of the next interval to be requested is calculated by the following formula *d = (1+2p)/(2n)*fileLength*, where *n* denotes the current number of pieces the initial interval *[0, fileLength]* is composed off and *p* denotes the next interval on the current partitioning to be fetched.

To better illustrate how these intervals are computed, let's consider some potential scenarios, referring to Figure 9. In each case the whole file may be downloaded, yet the starting point for the download varies as follow:

- **Case Figure 9-A and case Figure 9-B.** Two file servers are in reach and there are available resources to launch two parallel execution flows:
  - data is requested from file server 1 starting from the beginning of the file;
  - data is requested from file server 2 starting from the middle of the file, i.e. point *(1/2)*fileLength*.

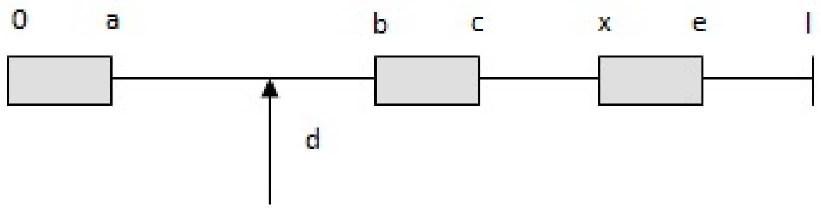

Figure 10. Potential download map; some data pieces have been downloaded; the resulting map in this case is d;[a+1,d-1],[d,b-1],[c+1,x-1],[e+1,l].

- **Case Figure 9-C.** Four file servers are in reach area and there are available resources to launch four parallel execution flows:
  - data is requested from file server 1 and 2 as stated above;
  - data is requested from file server 3 starting from point *(1/4)* fileLength* of the file;
  - data is requested from file server 4 starting from the point *(3/4)* fileLength* of the file.

In essence, the starting point of the requested interval is calculated so as to halve the largest interval left undivided on the original interval. Clearly, when the end of the file is reached, the download continues from the beginning until the whole file is downloaded. However, in case all the parallel downloads are prematurely interrupted by disconnection, they will all have downloaded different parts of the file thus maximizing the possibility to have cumulatively downloaded the whole file instead of having redundantly downloaded the same limited part of the file.

Once the assigned data interval is downloaded from the file server we would desire to use this source to download other missing intervals while in reach. On the other side, it is possible that a data request is not entirely satisfied (transmission was interrupted for some reason) and between the next starting point *(d)* and interval end point *(I)* may exist downloaded parts interleaved with missing ones (Figure 10). To represent this, *download map* is used and provided to future servants, with the format specified in Figure 11. This format includes the indication of the starting point *d* and missing intervals, those not yet downloaded.

As mentioned earlier, DTNs often use message replication techniques along the destination path in order to increase the probability of data reaching the destination. In our case the file division strategy might add redundancy during data transfer as it can happen that at concurring file servers are requested overlapping data intervals. However, if two file possessors come in different points in time, e.g. a disconnection occurred and transfer was not finished, the next requested data interval consists of only free, missing intervals.

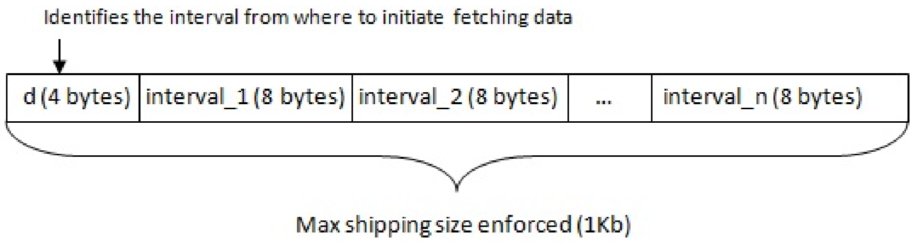
Identifies the interval from where to initiate fetching data
d (4 bytes)
interval_1 (8 bytes)
interval_2 (8 bytes)
...
interval_n (8 bytes)
Max shipping size enforced (1Kb)

Figure 11. M2MShare download map format.

The chart in Figure 12 shows a simulative comparison of our file division strategy against two other division strategies:

- **iM:** a strategy which requests from each file server the entire file, always starting from its first byte;
- **rM:** a strategy that randomly chooses the initial download point in the file request.

In the tested scenario, file possessors are in-reach area. If n is the number of file possessors, the task handling the file download can initiate n simultaneous transfers. Each experiment was repeated 40 times for each division strategy. As we can see from the chart above, our division strategy has the least redundancy (overlap among simultaneous file transfer) during data transfer, thus increasing the amount of useful data transfer while reducing overhead.

7. CONCLUSION AND FUTURE WORK

Mobile users increasingly find themselves in different types of networking environments, spanning from globally connected networks such as cellular networks or the Internet to the entirely disconnected networks of stand-alone mobile devices, environments that impose different forms of connectivity. Due to mobility, communication links between mobile nodes are transient and

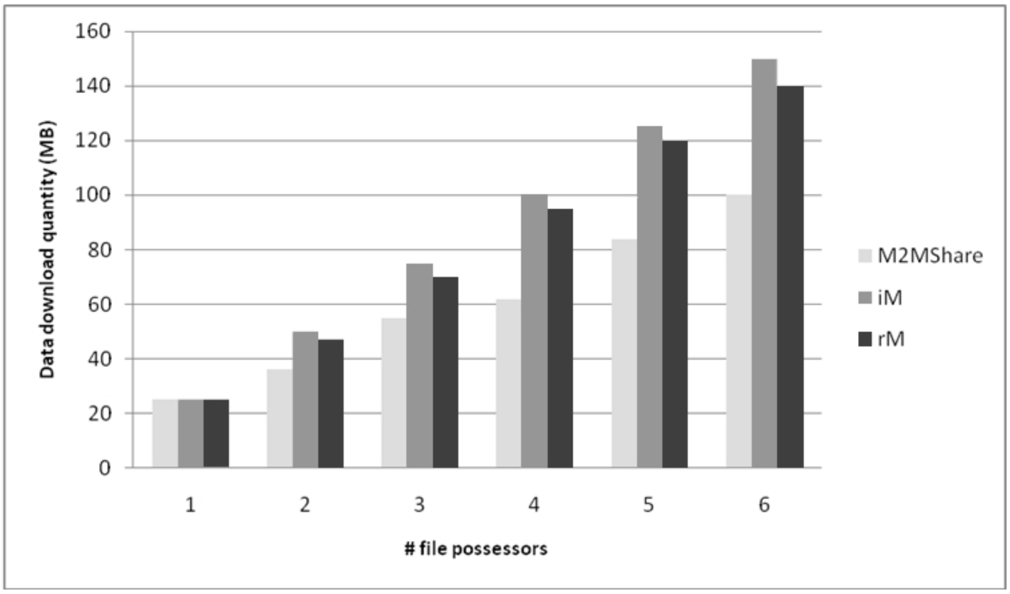

Figure 12. Data quantity downloaded by each division strategy, varying the number of file possessors in the in-reach area.

temporarily connected, thus impeding a continuous end-to-end path between a source and a destination. This is a modern, increasingly common type of DTN, which was originally intended to be used for communication in outer space, but is now directly accessible from our pockets.

To this aim, we examined the experimentation that we found in literature and devised a special purpose delay/disruption tolerant solution for P2P file sharing in mobile networks. We do not see mobility as an obstacle; instead, we exploit peer mobility to reach data in other disconnected overlay networks, implementing a mechanism like DTN (store-delegate-and-forward) where peers in the network delegate tasks to other peers (store) and wait back for their output (forward).

We argued that delegating tasks to all other peers in the established network proves bandwidth and energy consuming. Therefore a metrics was defined, by which individual peers select their servants to whom they delegate tasks. The criterion of selecting one peer instead of others is based upon the frequency with which one device encounters other devices. As stated previously this frequency of election changes and adapts to the observed dynamics; there might be days when the device encounters a small number of other devices and others when it encounters a large number of them.

This said, there is a lot of work that can be done to further improve and extend the functionalities provided by the software. As first, we need to test the protocol stack under synthetic and real trace mobility scenarios and see how it performs. To this purpose we are currently studying the feasibility of testing the software in The Opportunistic Network Environment (THE ONE), a simulation environment which seems to serve our purpose.

An interesting extension would be that of electing a servant for a particular download task not just by considering the frequency of encounter but also considering the servants possibility of finding that particular thematic data file. The work [16] shows how this could be accomplished, by using an information retrieval similarity metrics between servant's files of interest and the questioned task subject to delegation. In this way, we might increase the chances of finding that particular data file as we delegated the task to a servant who is interested in or has relations with other peers interested in similar contents. Concluding, other possible extensions of this work include the design of a file division strategy and local congestion control to increase the quantity of file downloaded per time unit [40, 41].

ACKNOWLEDGEMENT

We wish to express our gratitude toward Dr. Carla De Francesco and Dr. Luigi De Giovanni for the insightfull discussions we had on topics related to this paper. This work has been partially funded by the University of Padua through its research funding program.

REFERENCES

1. Marfia G., Pau G., Di Rico P., Gerla M., "P2P Streaming Systems: A Survey and Experiments", *in Proc of the 3rd STMicroelectronics STreaming Day (STreaming Day'07)*, Genoa, Italy, Sep 2007.
2. Amoroso A., Roccetti M., "On the Making of an Ubiquitous and Altruistic Application for Medical First Responses", *in Proc. 2009 IEEE International Workshop on Ubiquitous Multimedia Systems and Applications (UMSA'09)- International Conference on Ultramodern Telecommunications (ICUMT 2009)*, St. Petersburg, Russia, Oct 2009.
3. Palazzi C. E., "Buddy-Finder: A Proposal for a Novel Entertainment Application for GSM", *in Proc of IEEE International Workshop on Networking Issues in Multimedia Entertainment 2004 (IEEE NIME 2004), GLOBECOM*

2004 Dallas, TX, USA, Nov 2004.

4. Di Ferdinando A., McKee P., Amoroso A., “A Policy Based Approach for Automated Topology Management of Peer To Peer Networks and a Prototype Implementation”, *in Proc of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, Como, Italy, Jun 2003.
5. Jung S., Lee U., Chang A., Cho D.-K., Gerla M., BlueTorrent: Cooperative Content Sharing for Bluetooth Users, *Pervasive and Mobile Computing* 3(6): 609-634, 2007.
6. Zhang T., Li Z., Cheng X., “Multi-Task Downloading for P2P-VoD: An Empirical Perspective”, *in Proc of the 16th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2010)*, Shanghai, China, Dec 2010.
7. Woodbridge J., Nahapetian A., Noshadi H., Kaiser W., Sarrafzadeh M., “Wireless Health and the Smart Phone Conundrum”, *in Proc. of the 2nd Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS/MDPnP 2009)*, San Francisco, CA, USA Apr 2009.
8. Palazzi C. E., Bujari A., “A Delay/Disruption Tolerant Solution for Mobile-to-Mobile File Sharing”, *in Proc of the 3rd IFIP/IEEE Wireless Days 2010*, Venice, Italy, Oct 2010.
9. Gaito S., Maggiorini D., Pagani E., Rossi P. G., “Distance Vector Routing for Public Transportation Vehicular Networks: Performance Evaluation on a Real Topology”, *in Proc of the 2nd IFIP/IEEE Wireless Days 2009*, Paris, France, Dec 2009.
10. Cerf V., Burleigh S., Hooke A., Torgerson L., Durst R., Scott K., Fall K., Weiss H., *Delay-Tolerant Networking Architecture*, http://tools.ietf.org/html/rfc4838 Apr 2007.
11. McMahon A., Farrell S., Delay- and Disruption-Tolerant Networking, *IEEE Internet Computing*, **13**(6):82–87, 2009.
12. Klemm A., Lindemann C., Waldhorst O., “A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks”, *in Proc of IEEE Semiannual Vehicular Technology Conference 2003 (IEEE VTC 2003*, Orlando, FL, USA, Oct 2003.
13. Scott K., Burleigh S., *Bundle Protocol Specification*, http://tools.ietf.org/html/rfc5050 Nov 2007.
14. Whitbeck J., Conan V., “HYMAD: Hybrid DTN-MANET routing for dense and highly dynamic wireless networks”, *in Proc of IEEE World of Wireless, Mobile and Multimedia Networks Workshops 2009 (WoWMoM 2009)*, Kos, Greece, Jun 2009
15. Caini C., Cornice P., Firrincieli R., Livini M., Lacamera D., “DTN Meets Smartphones: Future Prospects and Tests”, *in Proc of IEEE International Symposium on Wireless Pervasive Computing 2010 (IEEE ISWPC 2010)*, Modena, Italy, May 2010.
16. Zhang Y., Zhao J., “Social Network Analysis on Data Diffusion in Delay Tolerant Networks”, *in Proc of ACM Symposium on Mobile Ad Hoc Networking and Computing 2009 (MobiHoc 2009)*, New Orleans, LA, USA, May 2009.
17. Vahdat A., Becker D., “Epidemic Routing for Partially-Connected Ad Hoc Networks”, Duke University Technical Report, Jul 2000.

18. Hui P., Crowcroft J., Yoneki E., “Bubble Rap: Social Based Forwarding in Delay Tolerant Networks”, *in Proc of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2008)*, Hong Kong, China, May 2008.
19. Daly E. M., Haahr M., “Social network analysis for routing in disconnected delay-tolerant MANETs”, *In Proc of the 8th ACM international symposium on Mobile ad hoc networking and computing*, Montreal, Canada, 2007
20. Burgess J., Gallagher B., Jensen D., Levine B. N., “MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks”, *In Proc of INFOCOM 2006. 25th IEEE International Conference on Computer Communications*, Barcelona, Spain, April 2006
21. Lindgreny A., Doria A., Scheleny O., “Probabilistic Routing in Intermittently Connected Networks”, *in Proc of Service Assurance with Partial and Intermittent Resources 2004 (SAPIR 2004)*, Fortaleza, Brazil, Aug 2004.
22. Chen X., Murphy L. A., “Enabling disconnected transitive communication in mobile ad hoc networks”, *In Proc of Workshop on Principles of Mobile Computing*, Newport, RI (USA), Aug 2001.
23. Tseng C.Y, Ni S. Y., Chen S. Y., Sheu J., The broadcast storm problem in a mobile ad hoc network, *Wireless Networks*, **8**(2/3):153–167, 2002.
24. Bettstetter Ch., Resta G., Santi P., The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks *IEEE Transactions on Mobile Computing*, **2**(3):257–269, 2003.
25. Gowrishankar S., Sarkar S., Basavaraju, T. G., “Simulation Based Performance Comparison of Community Model, GFMM, RPGM, Manhattan Model and RWP-SS Mobility Models in MANET”, *In Proc of the 2009 First International Conference on Networks & Communications*, DC, USA, 2009
26. Musolesi, M., Mascolo, C., “A community based mobility model for ad hoc network research”, *In Proc of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, Florence, Italy, May 2006
27. Ekman F., Kernen A., Karvo J., Ott J., “Working Day Movement Model”, *Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models*, NY, USA, 2008
28. Hsu W., Helmy A., Investigation of Mobile-User Patterns Across University Campuses using WLAN Trace Analysis, Technical report, University of South California, 2005
29. Srinivasan S., Moghadam A., Hong S., Schulzrinne H., “7DS - Node Cooperation and Information Exchange in Mostly Disconnected Networks”, *in Proc of IEEE International Conference on Communications 2007 (IEEE ICC’07)*, Glasgow, Scotland, Jun 2007.
30. Konstantinidis A., Zeinalipour-Yazti D., Yang K., “Socially-aware Query Routing in Mobile Networks”, *in Proc of the 9th Hellenic Data Management Symposium (HDMS 2010)*, Ayia Napa, Cyprus, Jun-Jul 2010.
31. Stevens R. W., *TCP/IP Illustrated.* (3rd edn), vol. 1, Addison-Wesley.
32. Fall K., Farrell S., DTN: An Architectural Retrospective, *IEEE Journal on Selected Areas in Communication*, **26**(5):828–836, 2008.
33. Warthman F., *Dealy Tolerant Networks (DTN)-A Tutorial*, http://www.dtnrg.org/docs/tutorials/warthman-1.1.pdf [April 2010]
34. Perkins E. C., Belding-Royer M. E., Chakeres D. I., Ad hoc On- Demand Distance Vector (AODV) Routing, *IETF Internet Draft 2004.*

35. Rhee I., Shin M., Hong S., Lee K., Chong S., "Human Mobility Patterns and Their Impact on Delay Tolerant Networks", *in Proc of the 6th ACM Workshop on Hot Topics in Networks (HotNets VI)*, in Atlanta, GA, USA, Nov 2007.

36. Samal S., Mobility Pattern Aware Routing in Mobile Ad Hoc Networks, MS Thesis, Virginia Polytechnic Institute and State University (May 2003).

37. Ripeanu M., Iamnitchi A., Foster I., Mapping the Gnutella Network, *IEEE Internet Computing*, **6**(1):50–57, 2002.

38. Cohen B., "Incentives Build Robustness in Bitorrent", *in Proc of the 1st Workshop on Economics of Peer-to-Peer Systems 2003 (P2PECON 2003)*, Berkeley, CA, USA, Oct 2003.

39. Buford J., Yu H., Lua K., *P2P Networking and Applications*, (1st edn), vol. 1, Elsevier 2009 166–169.

40. Marfia G., Lutterotti P., Eidenbenz S., Pau G., Gerla M., "Fair Multi-Media Streaming in Ad Hoc Networks through Local Congestion Control", *in Proc of the 11th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (ACM MSWiM 2008)*, Vancouver, Canada, Oct 2008.

41. Marfia G., Pau G., Giordano E., De Sena E., Gerla M., "VANET: On Mobility Scenarios and Urban Infrastructure", *in Proc of the 1st IEEE INFOCOM Workshop on Mobile Networking for Vehicular Environments (MOVE 2007)*, Anchorage, AK, USA, May 2007.

42. Ferretti S., Roccetti M., "Fast Delivery of Game Events with an Optimistic Synchronization Mechanism in Massive Multiplayer Online Games", *in Proc of ACM International Conference on Advances in Computer Entertainment Technology 2005 (ACM ACE 2005)*, Valencia, Spain, Jun 2005.

43. Kernen A., Ott J., Krkkinen T., "The ONE simulator for DTN protocol evaluation", *In Proc of the 2nd International Conference on Simulation Tools and Techniques*, Rome, Italy, 2009