Proactive Data Replication Using Semantic Information within Mobility Groups in MANET

Hoa Ha Duong and Isabelle Demeure

Institut TELECOM - TELECOM ParisTech - CNRS LTCI, 46 rue Barrault, 75013 Paris, France
hoa.haduong@telecom-paristech.fr, isabelle.demeure@telecom-paristech.fr

Abstract. In this article we propose a distributed data replication algorithm to be used for data sharing in Mobile Ad hoc NETworks (MANETs). Our system replicates data before users access them. To this purpose, it uses a predictive algorithm based on semantic information about the user and the data and previous access patterns. It also aims at creating enough replica to prevent data loss in case a peer unexpectedly disappears or a partition occurs. To this end, we also propose a stable group creation algorithm based on long lasting connectivity. While data sharing systems for MANET already exist, both the use of semantic information and of temporal stability are new in this domain. We illustrate the interest of the proposed algorithms by showing how a wiki service on MANETs would benefit from them.

Keywords: MANET, data sharing, intelligent replication, mobility, wiki.

1 Introduction

Mobile Ad hoc NETworks (MANETs) are networks established spontaneously (with no pre-existing infrastructure) between mobile terminals. In MANETs, terminals may act as routers as well as end users terminals. The toplogy of the network evolves as the nodes move. These networks therefore create new challenges for distributed systems and applications designers: they must design algorithms that adapt to the dynamic network topology; they must also take into account that there is no guarantee of a persistent access to a given terminal. Algorithms should avoid the use of central coordinators and should therefore be conceived as decentralised; they should also introduce redundancy to prevent faults created by peer disappearance and network partition. The communications are more scarce than in a wired network and the amount of communication directly impacts the energy consumption and therefore the amount of time a user can work before the battery "dies". Most mobiles terminals also have less storage space than desktops. Algorithms should thus be aware of the potentially limited capacities of the terminals.

Collaborative services over MANETs are of interest to academics and industries as demonstrated by the existence of projects such as the POPEYE european

C. Giannelli (Ed.): Mobilware, LNICST 0007, pp. 129–143, 2009.

research project [5] or the One Laptop Per Child (OLPC) project [2]. A classic collaborative data sharing service is a distributed file system, such as adHocFS [3], but other services may be useful, such as a distributed editor, or a social networking application. These services may be used as an extension of a centralized service, when its users are out of reach of the server but still in group, or even as a totally independent application.

In this article we focus on the problem of data replication to increase data availability in a MANET. To each data we associate metada giving information about data content and attributes. Users also share information about themselves, advertising their interests and the resources of their terminals. We use this semantic information to provide an efficient and adaptable replication mechanism. Since we aim at offering a middleware for collaborative applications, we postulate that while users are mobile, they are moving around in group. To optimize the storage space, we enforce collaborative replication within these groups. One of our contributions with respect to other existing data sharing systems for mobile ad hoc networks, such a the file system adHocFS[3] or the middleware XMiddle[11], is the introduction of semantic information that allows intelligent management of data replication by predicting users acesses. Systems such as Bayou[22], or Coda[21] allow users to work when they are disconnected, but they use replicated servers to store the data. Since we are in a MANET environment, a decentralized peer to peer solution, not based on reliable peers to act as servers is better suited to our problem.

The algorithms presented in this paper are currently being implemented in a data sharing system that will be run on top of a middleware for MANETs that was developed within the framework of the Transhumance projet [15].

The remainder of this paper is organised as follows. In Section 2, we present a wiki service on MANETs that is used throughout this paper to illustrate the interest of the proposed algorithms. In Section 3, we survey the current replication mechanisms for MANETs. In Section 4, we present our proposal that consists in an algorithm for creating stable neighbourhoods and a replication algorithm that reduces the latency of data access, and prevents the loss of data in case of partition. In Section 5, we discuss the validation of the proposed algorithms. Finally, in Section 6, we conclude and discuss the work that still needs to be done.

2 A Target Application: Wiki over MANETs

In the work presented herein, we focus on a wiki service suited to MANETs. A wiki is a web application allowing its users to share information by creating and editing articles in a quick and simple way [1].

Let us consider a scenario taking place in a school where students are encouraged to take a scientific approch to knowledge by the deployement of a wiki where they describe new knowledge acquired by experimentations and observations. Each student has his/her own personal laptop, with wireless capability and ad hoc network protocols. When a class leaves for a field trip, the wiki content

is distributed on the students laptops so that they can access it even if the field trip location does not provide network access. The students can therefore gather new information and contribute to the school knowledge base. Upon their return to school, the new articles produced during the trip are added to the school wiki and the modifications are merged.

In this article we focus on the management of the wiki when users are working in ad hoc mode. We use this service to demonstrate our algorithms. The wiki articles are structured by hypertext links and categories. Data may be simultaneously modified by several peers hence the need for a coherence protocol. Data are associated with correlated data via hypertext links, and with semantic information about their content, via categories, such as Science, or Birds of Southeast Asia. For a given data, few users edit the data and most of them are only readers.

The algorithms presented below take into account semantic information to enforce intelligent replication.

3 State of the Art: Replication Mechanisms for MANETs

Replication may be done with two objectives: a data may be replicated to reduce the latency of accesses, or to enforce the data presence in the network, even if a fault occurs. We describe 3 kinds of algorithms below. The first kind of algorithm tries to solve the issue of fault, and more precisely partition, without overloading the storage space. The second kind of algorithm aims at minimising the latency without overloading the network. The third kind of algorithm attempts to solve both issues.

In algorithms such as those proposed by Wang [12] and Chen [4], the authors create the minimum amount of replica such that when the mobility causes the network to split, all the parts of the network still hold a copy of every data. To enforce this property, they predict partitioning using the users terminals position to extrapolate their future position. These methods require GPS (Global Positioning System) information or equivalent and complex costly computation.

The algorithms proposed by Jing [14] and Yin[13] inspect the data accesses to decide if a replica should be created. Jing postulates that data often read should be replicated to lower the access time while a data often written should not be replicated as the replication would create additional coherence traffic. He counts the accesses made in a neighbourhood and decides based on the read/write ratio. Yin examines the traffic and counts the requests for a data. Depending on the number of requests and the nature of the data (often edited or not), a peer may cache the data, or a path to the closest copy.

Takahiro Hara has proposed several algorithms to determine how data should be replicated. Depending on the information we have about the data, a quantifier is created to evaluate the pertinence of replicating the data. In [7], Hara considers that for each user i and each non mutable data j, we know the access frequency ${}_f i, j$. In [8], Hara considers data periodically updated, with the period t_j and the probability $p_{i,j}$ a peer accesses a data in this period. He defines the quantifier

$pt_{i,j} = p_{i,j} * t_j$, with t_j denoting the remaining time to the next update; $pt_{i,j}$ is the probability that the peer accesses the data before it is updated and the local copy becomes invalid. In [10], Hara considers correlated data. For each pair of data i and j, the correlation $c_{i,j}$ is the probability that the two data are accessed together is known. For each data, a priority is calculated in order to reflect the frequency of access to it. In [9], Hara considers data updated aperiodically, with the probability for each peer of reading the data $pR_{i,j}$ and writing the data $pW_{i,j}$ known. He defines the ratio $RWR_{i,j} = \frac{pR_{i,j}}{pW_{i,j}}$.

For each of those quantifiers, three algorithms are proposed. In the first algorithm, each peer replicates data for itself only, using the quantifier to decide which data should be replicated first. This algorithm is simple but two neighbours may end up hosting the same data. The second algorithm tries to fix this problem by having each peer broadcast how they quantify the data to their neighbours. For each data, the neighbour with the highest quantifier creates a copy. This algorithm fixes the problem of neighbours hosting the same data but since the users are mobile, neighbours may be separated between two periods. The third algorithm tries to fix the problem caused by mobility by creating stable neighbourhoods: a peer considers another peer to be part of his neighbourhood if it can be attained by two paths. Within this neighbourhood, the peers have the same behaviour as in the second algorithm.

Hara's algorithms use precise information about the data and the accesses, such as the update frequency or the access frequency. This approach is suitable for some types of data, such as those produced by sensors, but this information does not seem to be relevant for human accessed and edited data.

Our solution introduces semantic information about the data and the user in order to predict the user accesses on the fly by using keywords describing the content of a data and the user interests. By observing the real accesses, we correct our prognosis.

4 Proposal: Stable Neighbourhood and Replication Algorithms

4.1 Asumptions

In our proposals, we make some assumptions about the target MANET system, namely:

- First we assume that the terminals used are laptops. They have good capacity and are scarcely mobile but nevertheless they may move and disappear.
- We do not make any specific assumptions about the communication protocols that are out of our the scope of this research, except that the routing layer may be queried to provide the number of hops between any two peers (which is possible in a number of implementations such as, for example, The UniK OLSR implementation [16].)
- Each user has a profile, in which the user's interests are described by a list of keywords.

– Each data has related metadata, also in the form of keywords, outlining data content. A data may also have a set of correlated data.

In our algorithms, the targetted data granularity is that of a memory page with size ranging from a few kiloByte to a megaByte.

4.2 Stable Neighbourhood Algorithm

To share the load of hosting data and reduce the number of adjacent replica, we propose to do collaborative replication: a peer should host data for itself as well as for its neighbours. This should be done among peers staying in view of each other on the long term, else the data would be stored without being accessed.

Since the topology is not fixed, a peer needs to discover and maintain a stable neighbourhood. Here we propose an adaptable algorithm 1 for a peer to construct a neighbourhood that is stable over time.

In a nutshell, this algorithm 1, executed periodically, behaves as follows:

– For each peer
  – If the peer is present this round, PresenceCounter is incremented, unless the value is already at MaxPresence.
  – If the peer is absent this round, PresenceCounter is decreased, unless the value is 0 in which case we do not keep trace.
  – It also stores the average distance between peers, in case an overlaying algorithm needs this information.
– It decides on a group by setting a minimum threshold for the presence counter value.

How we set the period of execution for the algorithm 1 is discussed in the Validation Section (we use the OLSR algorithm route maintaining a period of 2 seconds).

The parameters of the algorithm 1 have the following meaning:

– maxPresence (line 11): the PresenceCounter value must have an upper bound so that when a peer present for a long time disappears, it is removed from the neighbourdhood.
– α (line 20) reflects how our evaluation of the average distance taking into account the instant value. α must belong to [0,1]
– thresholdStable (line 26) reflects how volatile a peer in the neighbourhood is allowed to be.
– thresholdDistance (line 26) reflects the acceptable width of the neighbourhood

The use of a maximum (MaxPresence) and a minimum (thresholdStable) value for the presence counter instead of letting it increase indefinitly allows us to eliminate a peer from the group if it disappears, while tolerating a few intermittent disconnections. They can be understood as time spans: e.g. for a polling period of 10 seconds, a thresholdStable of 60 and a MaxPresence of 72,

we expect a peer to stay 60*10s=10 minutes in our vinicity before considering it a neighbour. We tolerate at most (72-60)*10s=2 minutes of absence before we remove it from the neighbourhood.

```
1  Init
2  Set Neighbourhood, Potential, Reachable= ∅;
3  This part of the algorithm is repeated periodically until line 29 to take
   into account the mobility;
4  Reachable = { (peer reachable this period, distance) };
5  maxDist = distance to the furthest peer;
6  foreach (peer, distance) ∈ Reachable and peer ∉ Potential do
7      add (peer, distance, 1) to Potential;
8  end
9  foreach (peer, distance, PresenceCounter) ∈ Potential do
10     if (peer, distanceNew) ∈ Reachable) then
11         PresenceCounter = max(PresenceCounter++, maxPresence);
12         distance = α*distanceNew+(1 − α)*distance;
13     end
14     else
15         if (PresenceCounter =1) then
16             remove peer from Potential;
17         end
18         else
19             PresenceCounter- -;
20             averageNumberHops = α*maxDist+(1- α)*
               averageNumberHops;
21         end
22     end
23 end
24 Neighbourhood =∅;
25 foreach (peer, distance, PresenceCounter ∈ Potentials) do
26     if PresenceCounter ≥ thresholdStable and averageNumberHops ≤
       thresholdDistance then
27         add peer to Neighbourhood;
28     end
29 end
```

Algorithm 1. A stable group formation algorithm

As we have seen, our algorithm periodically checks which peers are within reach. This can be done in different ways, depending of the underlaying routing algorithm:

- a proactive routing algorithm such as OLSR creates and maintains routes between nodes even when their is no traffic.
- a reactive routing algorithm such as AODV creates routes only when requested.

We do not discuss the pros and cons of each type of algorithm which is out of the scope of this paper. The availability of the Transhumance middleware [15] runing OLSR in our lab, led us to choose a proactive routing algorithm (OLSR); we can therefore obtain the list of reachable terminals by "peeking" in the OLSR routing tables.

This algorithm does not use location information to construct groups and may be executed independently by each terminal. While it may not be as accurate as GPS based algorithms, it is less compute-intensive and network greedy. Some algorithms use an average distance between peers and assume that if peers are close for long enough, they belong to the same group; we prefere to base our algorithm on the time peers spend within reach of each other.

4.3 Replication Algorithms

Replicating data is a mechanism used to accelerate data access by caching it. It is also used to introduce redundancy and provide alternate access when a fault occurs, rather than loosing the data.

In this section we present two complementary replication algorithms to solve both problems.

Replication Based on Interest. We propose to create local copies of potentially interesting data before the user tries to access them. To do so, we need to predict the user accesses.

For a data D, described by a set of keywords K_D, and a user U, whose interests are summed up by a set of interests I_U, we define in 1 the interest U may have in the data as $P_{U,D}$:

$$P_{U,D} = \frac{1}{2} * (\frac{card(K_D \cap I_U)}{card\ K_D} + \frac{card(K_D \cap I_U)}{card\ I_U}) \quad (1)$$

$P_{U,D}$ calculates which proportion of the keywords are interesting for the user with respect to the number of keyword and the number of interests.

Since we want to enable collaborative sharing, a peer should take into account the interests of its neighbours when replicating data. As terminals may have heterogeneous capacities, a peer should only provide this service to peers with less resources.

With K the set of peers in the stable neighbourhood, each peer $k_i \in K$ evaluates the resources it possesses to produce a resource representant *res*, $res \in [0, 1]$. All the peers must use the same scale to establish their resources. res_i is then used to calculate how many peers may be generous in sharing their resources with peer i. Interests and *res* are broadcasted in the stable neighbourhood. For each keyword A found in the neighbourhood (ie present in at least one set of interest), each peer $k_p \in K$ determines the weight of A.

First of all, it aggregates the interest of other peers $k_{i,i\neq k} \in K$ with less resources than it has by summing the differences between its *res* and the one of peers with less resources for which A is an interest as in 2 :

$$SumWeight(k_p, KW) = \sum_{k_i \in K, k_i\ interested\ by\ A} max\big((res_p - res_{k_i}), 0\big), \quad (2)$$

To each keyword A we then associate a weight, AWeight, using formula 3:

$$AWeight(k_p, KW) = SumWeight(k_p, KW) + \begin{cases} 1 \ if \ KW \ interest \ k_p \\ 0 \ else \end{cases} \quad (3)$$

Thus, each peer creates a set of aggregated interests, AG, containing a list of pairs interests with their weight(A, AW).

For a dataD with associated keyword KW, and a user U, with aggregated interests AG, the interest formula becomes 4:

$$c = \sum_{(A,AW) \in AG \ \wedge \ A \in KW} AW, P_{U,D} = \frac{1}{2} * (\frac{c}{card \ KW} + \frac{c}{\sum_{(A,AW) \in AG} AW}) \quad (4)$$

In the data sharing system that we are currently specifying, a message is broadcasted to all peers to advertize the creation of a new data. This message will also carry the keywords associated with this data. When a peer receives such a creation message, it calculates its interest in the data. If P is superior to a threshold the data is replicated. The values of the interests are also aggregated: for each percent P between 0 and 100, we count how many data have an interest rounded to P. If a data interest is superior to the N^{th} centile, it may be replicated.

Example
interestThreshold : 0,8
data Keywords =A,B,C,E
if user Interests =A,B,C,D, $P_{U,D} = 0,75$, the data is not replicated
if user AG=(A,1.3),(B,1),(C,2),(D, 1),(E,0.1), $P_{U,D} = 0,99$, the data is replicated

Replication to Prevent the Loss of Data. We have seen how to decide if a data may be interesting for the user and how to replicate it. Even with this mechanism, since a peer is limited by its storage, some data may not be replicated. We propose an algorithm to create enough replica so that, should a crash happen, such as a crash or a network partition, no data would be lost.

This algorithm is executed when a peer learns of the existence of a new data but is not interested enough to replicate it.

```
1 repeat
2     wait to be informed of replica creation in the previous step, ie the
      RTT to the farthest node in the group ;
3     K = the number of copies that must be created);
4     N = the number of potential hosts for the data (i.e., hosts in the
      neighbourhood with no copy);
5     if rand[0,1] ≤ K/N then
6         Create a replica and broadcast the information;
7     end
8 until enough replica ∨ a local replica has be created ;
```

Algorithm 2. An algorithm to enforce the creation of a minimum number of replica

Combining the Algorithms. In the beginning, each peer establishes his stable neighbourhood. We consider that all peers use the same parameters to tune the algorithm. The relationship "Belongs to neighbourhood" is thus symmetrical. Each peer broadcast its profile (resources level and interests) to its neighbours.

The stable neighbourhood is maintained by periodically checking the state of the neighbours. When a modification occurs in a peer profile, it is broadcasted and the neighbours integrate the modifications.

When information about a new data is received, a peer calculates its potential interest for the data. If it is interested, the peer replicates the data and broadcasts this information (first algorithm). Else, it caches the metadata in case a neighbour would later need the data. The peer then checks if there is enough replica in the neighbourhood and if it is not the case, it attempts to create a new one.

4.4 Learning New Interests

The user profile includes a list of interests as keywords and associated metadata; data are also described by keywords. In the preceeding section, we have described how this information is used to predict the user accesses and gather data which may interest him. This allows the user to access data even when the creator of the data is out of reach.

As the user accesses data, we want to analyze the keywords associated to these data to see if new interests may be detected.

In a wiki, editors are encouraged to structure the articles into categories. We take those categories as keywords; the keywords are thus organized in a graph of categories and subcategories.

This graph is a Direct Acyclic Diagram (DAG), and not a tree; e.g. an article about tourism in France should be part of both the France and Tourism categories. The uppermost categories are under a category that we call root. Furthermore, a category cannot be a subcategory of itself, so we do not have cycles in the graph.

Each data comes with its DAG representing its categories and each peer stores a DAG representing all the data it has accessed so far. Each node has a counter indicating how many times this keyword has been meet.

When an user accesses a data for the first time, the data DAG is merged with the user DAG. To determine if a node may constitute a potential interest, we distinguish the case of the leaf nodes and the case of the other nodes. If the node is a leaf, we check if its value is superior to $\overline{x} + 2 * \sigma$, with $\overline{x}$ being the average of the value of the leaves, and σ the variance. If the node is not a leaf, we check how many children it has, and how the counter values are distributed: if the variance is low enough, we consider the node as a potential new interest.

5 Validation

As we have seen in Section 3, existing algorithms have tried to solve the problem of data access latency and data loss by replication. Hara's algorithms attempt

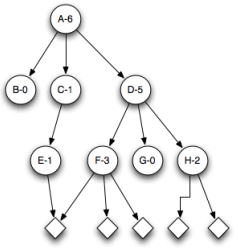

Fig. 1. Structured keywords space

This example references Figure 1.
For the leaves E, F and H $\overline{x} = 2, \sigma = \approx 0.57$
None of those nodes is considered as a potential new interest.
For the nodes D, $\overline{x} = 3, \sigma = \approx 0.5$
D children have an evenly distributed load, so we consider D as a potential new interest.

to fix those two problems but his hypotheses are not adapted to an application where data are edited by humans.

Our solution introduces semantic information about the data and the users to predict their accesses. We also adapt to the user by extracting patterns from his previous accesses.

A first version of our system has been implemented in the POPEYE project [5]. This version implements a simple replication mechanism based on interests, and uses clusters created at a lower level as stable neighbourhoods. A more advanced stand alone version that could be run on top of the Transhumance middleware, as mentioned before, is under development.

Validation and evaluation of algorithms for MANET is a complex problem. Since the terminals are mobile, a live experience requires moving participants and thus cannot be reproduced ad lib. Furthermore, since terminals have wireless connectivity, the reproductibility of an experiment is heavily dependant of the environment (e.g. how many other wireless networks are presents that can create interferences). We now explain how we intend to validate our algorithms.

5.1 Validating the Group Algorithm

Since we are building our prototype on top of a proactive routing algorithm, OLSR, a peer just takes a peek at the content of its routing table to update it stable neighbourhood. Thus, the cost in messages is in effect null. We will therefore validate our algorithm pertinency, by running it in a context where we know the existing groups, and checking if the algorithm comes up with the effective groups. We want to see how volatile the peers within a group can be, with our algorithm still detecting the effective groups.

In order to validate and adjust our group algorithm, we have implemented it in the NS-3 simulator [19]. NS-3 is an open source discrete event simulator intented to eventually replace the NS-2 simulator. While NS-2 is much more widely used, we choose NS-3 for several reasons. NS-2 has had a lot of contributions over the years that make it more complete but the code is less homogeneous and more complicated to grasp. NS-3 and NS-2 both implement the models that we need to simulate our algorithm (WiFi, IP and OLSR routing), but while we need to

patch NS-2 to use OLSR, it is already part of the NS-3 simulator. Finally, since NS-3 aimed at replacing NS-2, we believe that its designers learned from the mistakes made in NS-2, and it should be, in the long term, a better tool. Even if NS-3 is not as well documented and known as NS-2, we prefere to use it as we think it should become more used in the future.

Since there is no Group Mobility model in NS-3, we have implemented a simple Group mobility model that allows us to predefine groups of nodes sticking together, that peers may join and leave from time to time. These peers move at pedestrian speed and stop at times. We set the period of the algorithm at 2 seconds, namely the period of route refreshing in OLSR. We ran different simulations with differents values of parameters *thresholdStable* and *MaxPresence*, and we made the size of the area and the number of peers vary. Then, we computed the correlation between the actual groups as defined by the mobility model and the groups built by our algorithm. We simulated 20 minutes of execution.

In tables 1(a) and 1(b) we can see results for two particulars simulations. In the first one 1(a), thresholdStable=3 and MaxPresence=10 : a peer is part of a neighbourhood as soon as it has been present for 6 seconds (thresholdStable*period); it leaves the neighbourhood if it is absent for more thant 14 seconds ((MaxPresence-thresholdStable)*2). In the second table 1(b), thresholdStable=90 and MaxPresence=150: a peer is part of a neighbourhood as long as it as been present for 3 minutes; it is ejected when it is absent for more than 2 minutes. We simulated for 16, 25, 36 and 50 peers and for an area side length of 500m, 1000m, 1500m, 2000m and 2500m.

The columns represent length of the size of the simulated area while the lines represent the number of groups by the number of peers in each group. The values in the table are the percentage of accuracy of our algorithm: for each round and for each peer we have checked if the calculated neighbourhood is the effective group; we have then computed the portion of rounds where the neighbourhood is accurate.

As we can see in 1(a) and 1(b), the accuracy of the algorithm seems to grow when the simulated area is larger.

Figures 2 and 3 plot the percentage of accuracy, this time in relation with the density of the terminal : $\frac{nb\_groups * nb\_peers\_per\_group}{area\_side\_length^2}$. These figures show the algorithm is more accurate if the density is low. Thus, those errors are due to the

Table 1. Simulation results. Columns: size of the simulated area; lines: number of group by number of peers in each group

	500	1000	1500	2000	2500
6x3	0,03	47.23	73.10	72.62	94
5x5	1.02	23	54.61	74.61	75
6x6	0	17.96	37.58	65	71
5x10	NA	NA	26.2	50	58.38

(a) thresholdStable= 3 and a MaxPresence=10

	500	1000	1500	2000	2500
6x3	0,03	16,88	55,95	71,23	68,55
5x5	0	24,75	58,96	81,23	62,88
6x6	0	23,3	50,63	59,31	57,78
5x10	NA	9,42	30,12	53,92	51,41

(b) thresholdStable = 90 and MaxPresence = 150

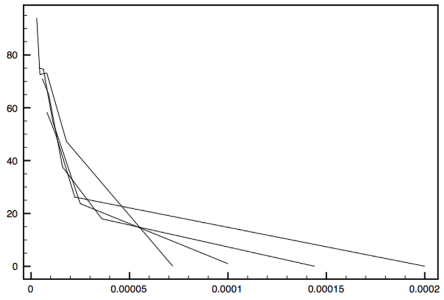

Fig. 2. Thresholdstable=3, MaxPresence=10, accuracy as a function of density

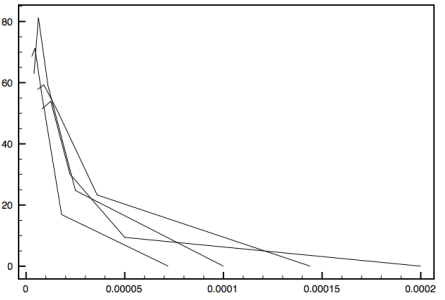

Fig. 3. Thresholdstable=90, MaxPresence=150 , accuracy as a function of density

number of peers per square meter : if a lot of peers are present in a small space, they cover enough ground to provide stable connectivity even with peers not in their group. Other simulation results, not included, here confirm this result.

The right values of thresholdStable and MaxPresence depend of the size of the area, the volatility and the speed of the terminals. We intend our system to work and terminals moving to pedestrian speed but the other parameters will depend on the situation.

While these tests underline this issue, they also show that our algorithm does recognize the effective groups as being part of the neighbourhood, even is some other peers happen to be connected for a time long enough to be included, mainly because the density of peers is high enough to provide connectivity.

5.2 Validating the Replication Algorithm

We are now in the process of evaluating our replication algorithm, which leads us to answer the following questions: What is the cost in terms of computing power and number of messages of these algorithms? Is a replica accessed? How many data must be fetched? How many requests cannot be satisfied because the data is not available?

For the cost of the preemptive replication, we expect our results to be close to the results obtained by the algorithms proposed in Hara's works: similarly to his proposal, the peers exchange information about their interests within a group to decide what they should replicate.

Let us consider that we have M peers in total. How many messages are exchanged in a stable group of N peers? First of all, interests and resources coefficient are exchanged once within the group, for a cost of N*(N-1) unicast messages. If our group algorithm manages to detect stable groups, this step should be rarely done. We also postulate that peers seldom update their interest. When a new data is shared anywhere, this information is broadcasted to all the peers, with either M-1 unicast messages, or 1 broadcast message. The cost for a group of N peers is therefore N. Peers decide if they intend to replicate.

When a peer creates a replica, it send a message to ask for the data, receives it (2 unicasts messages, one whose payload is the data) and informs its neighbours (N-1 messages). Our algorithm creates K replica when a new data is shared, in a group of N peers, the number of messages exchanged is therefore N+K(2+N-2) messages.

The algorithm has been implemented in the Popeye project[5] and was tested in real conditions, as can been seen on video available on the Popeye website[18]. While our algorithm has been tested functionally within Popeye, we are missing evaluation of the scalability and the performance of the algorithm. We also would like to be able to execute the algorithm several time while we change a parameter to adjust it. As we have seen before, this must be done with a simulator. We are currently conducting more evaluations.

Our problem with evaluating this algorithm lies in deciding if a replica has been wrongly created, and if a replica should have been created: we propose a predictive replication algorithm that estimates the behaviour of the user based on a set on predefined interests and its previous accesses.

We do not have for the moment a non biaised solution to validate our replication algorithm, as we do not have traces of accesses to a set of data with semantics, by a set of users with established interests. We cannot use artificial users either since it would give us strongly biaised results: our system itself simulates the behaviour of the users to predict which data should be replicated.

To evaluate the replication algorithm, we are currently implementing our proof of concept application, a P2P wiki for MANETs. This system will then be deployed and used and we will collect real traces of usage to reinject in a simulator.

6 Conclusion and Future Work

Collaborative services over MANETs need to take into account the particularities of MANETs, such as the dynamic topology, or the lack of a reliable server. In this article, we focused on the problem of data replication in MANETs, to provide small latency data access, and to prevent the loss of data in case of a fault (e.g. peer becoming out of reach).

First of all, we propose an algorithm to build stable peers group. Rather than using location information or topological proximity to outline a group, we shape up a group by surveying the continuous connections between peers over time.

Then we propose to solve both replication problems by using a two phases algorithm using those stable groups. First of all, data are replicated by peers they interest. Then, if not enough replica exist to guarantee the survival of the data in case of fault, new copies are created randomly within a stable neighbourhood. Our solution differs from existing algorithms because we introduce semantic information about the data and the user to predict the user accesses on the fly and manage a more intelligent replication.

In future work, we will modify our algorithm to take into account editable data in the replication algorithm. Let it be noted that editable data raises the problem of coherence. While this is not addressed in this paper, we are handling

this issue; we offer eventual consistency by using the commutative replicated data type Treedoc[20].

We are currently developing a distributed wiki engine integrating the aforementioned algorithms. We will use this application to demonstrate the validity of our algorithms and evaluate their performance. To this end, we will also have to decide on a positioning algorithm and a search algorithm.

References

1. http://en.wikipedia.org/wiki/wiki
2. OLPC Association, http://www.laptop.org
3. Boulkenafed, M., Issarny, V.: Adhocfs: sharing files in wlans. In: Second IEEE International Symposium on Network Computing and Applications, 2003. NCA 2003, pp. 156–163, April 16-18 (2003)
4. Chen, K., Shah, S.H., Nahrstedt, K.: Cross-layer design for data accessibility in mobile ad hoc networks. Wirel. Pers. Commun. 21(1), 49–76 (2002)
5. Popeye Consortium, http://www.ist-popeye.eu/
6. Inc. Google. picasa.google.com
7. Hara, T.: Effective replica allocation in ad hoc networks for improving data accessibility. In: INFOCOM, pp. 1568–1576 (2001)
8. Hara, T.: Replica allocation methods in ad hoc networks with data update. Mob. Netw. Appl. 8(4), 343–354 (2003)
9. Hara, T., Madria, S.K.: Dynamic data replication using aperiodic updates in mobile adhoc networks. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 869–881. Springer, Heidelberg (2004)
10. Hara, T., Murakami, N., Nishio, S.: Replica allocation for correlated data items in ad hoc sensor networks. SIGMOD Rec. 33(1), 38–43 (2004)
11. Mascolo, C., Capra, L., Zachariadis, S., Emmerich, W.: Xmiddle: A data-sharing middleware for mobile computing. Wirel. Pers. Commun. 21(1), 77–103 (2002)
12. Wang, K., Li, B.: Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks (2002)
13. Yin, L., Cao, G.: Supporting cooperative caching in ad hoc networks. IEEE Transactions on Mobile Computing 5(1), 77–89 (2006)
14. Zheng, J., Su, J., Yang, K., Wang, Y.: Stable neighbor based adaptive replica allocation in mobile ad hoc networks. In: International Conference on Computational Science, pp. 373–380 (2004)
15. Demeure, I., Paroux, G., Hernando Ureta, J., Khakpour, A.R., Nowalczyk, J.: An Energy Aware Middleware for collaboration on small scale MANets. In: Autonomous and Spontaneous Networks Symposium Telecom ParisTech, Paris, November 20-21 (2008)
16. Tannesen, A., Hafslund, A., Kure, O.: The UniK - OLSR plugin library. In: OLSR interop workshop, San Diego, August 6-7 (2004)
17. Ha Duong, H.D., Melchiorre, C., Meyer, E.M., Nieto, I., Arrufat, M., Pelliccione, P., Tastet-Cherel, F.: POPEYE: a simple and reliable collaborative working environment over mobile ad-hoc networks. In: The 3rd International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007) Crowne Plaza White Plains, New York, USA, November 12-15 (2007) IEEE Catalog Number: 07EX1828C. ISBN: 1-4244-1317-6

18. http://www.ist-popeye.eu/
19. http://www.nsnam.org/
20. Shapiro, M., Preguica, N.: Designing a commutative replicated data type (2007)
21. Kistler, J.J., Satyanarayanan, M.: Disconnected Operation in the Coda File System. In: Proceedings of the 13th Symposium on Operating Systems Principles, pp. 213–225. ACM Press, New York (1991)
22. Demers, A.J., Petersen, K., Spreitzer, M.J., Terry, D.B., Theimer, M.M., Welch, B.B.: The Bayou architecture: Support for data sharing among mobile users. In: Proceedings IEEE Workshop on Mobile Computing Systems & Applications, Santa Cruz, California, September 1994, pp. 2–7 (1994)