

Distributed Estimation of Global Parameters in Delay-Tolerant Networks

Alessio Guerrieri^b, Iacopo Carreras^a, Francesco De Pellegrini^a, Daniele Miorandi^a,
Alberto Montessoro^b

^a CREATE-NET

Via alla Cascata 56/D - I-38100 – Povo, Trento (Italy)

`name.surname@create-net.org`

^b University of Trento

via Sommarive 14 - I-38100 – Povo, Trento (Italy)

`name.surname@unitn.it`

Abstract

Distributed estimation of global parameters in intermittently connected mobile environments is a challenging problem. In this paper, we introduce a set of methods, based on gossip techniques and population protocols, for performing such task. The applicability of such techniques to various environments, characterized by different mobility patterns, is evaluated through numerical simulations and discussed extensively. Guidelines are provided to help practitioners choosing the right method for their specific application problem.

Key words:

1. Introduction

Distributed estimation of global parameters is a relevant problem in many distributed computing applications. In general, connectivity of the underlying network is taken as granted and attention is focused on optimizing the accuracy of estimation while minimizing the incurred communication costs. Dropping the connectivity assumption, methods need to be adapted (if not re-thought) in order to be able to cope with possible disconnections. In this paper, we focus on the problem of estimating global parameters in the class of intermittently-connected mobile wireless networks usually referred to as delay-tolerant networks (DTNs) [1].

In order to better understand the problem, let us consider the following two examples (one more technical, one more application-oriented):

- In a given DTN deployment, we want to optimize the routing algorithm to minimize a given cost function. To do so, we include adaptation strategies in the protocol itself, so that it changes its behavior in order to enhance its performance. The cost function may include a term that depends on the total number of copies of a packet made in order to reach the destination. Estimating such number is one of the problems addressed by our framework.

- A company developed a new service (e.g., exchange of MP3 files) running on smart-phones which exploit phones' Bluetooth interface. The service can spread "virally", so that a user having the software can disseminate it among her friends (again, through the Bluetooth interface). The company wants to estimate the number of users running its service: this problem fits our framework.

Problems like these – where a *local* summary of some *global* system property must be obtained – are well-known in distributed systems, and are often generically referred as *aggregation* [2]. Aggregation allows local access to global information in order to simplify the task of controlling, monitoring and optimizing distributed applications. In this paper, we discuss algorithms for computing *distributive* and *algebraic* aggregate functions, such as min, max, sum, average, counting, etc. [3]; we do not consider *holistic* functions like median, mode, and rank. However, our numerical analysis will focus on counting, which has proven to be the most sensitive function w.r.t. to failures and lack of connectivity.

Our work started from a study of the applicability of two classes of algorithms (pair-wise averaging [4] and population protocols [5]) to a DTN environment. In general, such algorithms do not perform satisfactorily, especially when considering the typical features (in terms of mobility patterns) of real-world DTN deployments. Hence the need for inferring new methods specifically tailored to such environments:

The contributions of our work are as follows:

- To provide algorithms and methods to efficiently estimate global parameters in intermittently connected networks; the term efficiency relates to fast convergence and good scalability properties;
- To provide mechanisms for termination of the estimation processes presenting a good trade-off between accuracy and speed of convergence;
- To provide numerical results, based on the use of both synthetic and real-world mobility traces, on the performance of the methods introduced;
- To provide guidelines for practitioners on which method to choose depending on the specific application needs and deployment features.

The remainder of the paper is organized as follow. Sec. 2 presents a quick introduction to distributed techniques for estimation of global parameters, together with a description of the system settings and assumptions used throughout the paper. Sec. 3 describes efficient estimation methods and termination algorithms. Sec. 4 evaluates the performance of the proposed methods and reports outcomes of numerical simulations. Sec. 6 overviews related works. Sec. 7 concludes the paper discussing guidelines for applications to real-world DTN deployments.

2. System Model and Problem Description

We consider a *delay-tolerant network* composed of a collection of cooperating, mobile nodes communicating wirelessly. Communication is based on *meetings*, i.e., encounters during which nodes come within mutual radio range [6].

Our work is based on the following assumptions:

- *Homogeneity*: nodes participate equally in the aggregation computation, following the same set of rules/algorithms.
- *Cooperation*: nodes are trustworthy (i.e., they do not provide wrong/fake information to other nodes).
- *Sparsity*: at any time instant, nodes are isolated with a probability close to 1; meetings are thus sporadic events and connectivity is never guaranteed.
- *Failures*: nodes may fail, either by abruptly leaving the system or by stopping operations.
- *Meeting duration*: meetings last enough to ensure the transmission of all data needed for a message exchange. This corresponds to assuming atomicity of the transactions performed by the algorithms (§4 provides an analysis of the robustness with respect to message loss).

The goal of this paper is to compute, in intermittently connected network, generic aggregation functions of the type:

$$f [X(1), X(2), \dots, X(N)], \quad (1)$$

where $X(i)$ is known at node i and the function is computed over all values. Playing with both the function and the values, such definition is general enough to include several applications of practical interests:

Example 1: Let us take $X(i) = 1$ for all nodes running a given software, and $X(i) = 0$ otherwise. If f corresponds to the sum function, we can easily measure the number of nodes having installed and running such software.

Example 2: Take $X(i) = 1$ if the battery level of device i exceeds a given threshold, and 0 otherwise. Further, we take, $Y(i) = 1$ for all nodes in the system. We can therefore compute the fraction of nodes whose battery level exceeds a given threshold as $\sum X(i) / \sum Y(i)$.

Example 2. Consider the use of a routing protocol that makes multiple copies of a given message (e.g., epidemic routing, spray-and-wait). If we denote by $X(i)$ the number of copies made by node i of a given message in the delivery process, we can estimate the resource usage (in terms of number of copies disseminated) associated to the message delivery process. This, in turn, can be used to estimate the performance of the routing protocol used.

The actual performance of the various mechanisms we will study may depend on the type of function considered in (1). As in this paper we are interested in comparing the performance of various mechanisms and understanding their strengths/shortcomings, we have decided to focus on counting, where the goal is to evaluate the total number of nodes in the system. Such assumption is used *only* for the performance evaluation part, and it is motivated by the inherent difficulty of counting. The mechanisms introduced in the following section will be presented with reference to the general class of problems represented by (1).

Algorithm 1 PAIRWISEAVG($v(j)$)▷ @ node i **Init:** $v(i) \leftarrow X(i), \widehat{X}(i) \leftarrow X(i)$ ▷ Default to $X(i)$

- 1: $v(i) \leftarrow (v(i) + v(j))/2$
 - 2: $\widehat{X}(i) \leftarrow v(i)$
 - 3: **return** $\widehat{X}(i)$
-

3. Algorithms for Distributed Estimation

The algorithms presented here are adaptations of well-known aggregation algorithms for connected networks: *pairwise averaging* [4] and *population protocols* [5].

The former belongs to the general class of gossip/epidemic protocols. Since the seminal work of Demers on the epidemic spreading of database updates [7], the gossip paradigm has gone far beyond dissemination, solving a large and diverse collection of problems – including aggregation [4].

In the population protocols framework, mobile agents interact with each other to carry out a computation. Interactions between agent pairs cause them to update their states; they are unpredictable but subject to a fairness constraint. Population protocols can be profitably used to model algorithms over DTNs, where meetings are caused by mobility.

3.1. PAIRWISEAVG algorithm

This algorithm is an adaptation to DTNs of the algorithm originally introduced in [4] to compute distributed averages, i.e., $f(X(1), \dots, X(n)) = \frac{1}{N} \sum_i X(i)$. The PAIRWISEAVG algorithm is well suited to the DTN scenario. In particular, the variant based on a random node matching described in [4] can be ported naturally to sparse mobile ad-hoc networks; the required matching, in particular, is here induced by pairs of nodes coming into radio range.

Alg. 1 reports the pseudocode. It works as follows. Every node i stores variable $v(i)$, initially set to $X(i)$. At each meeting, nodes i and j exchange their current values $v(i)$ and $v(j)$, and update the stored variable as $v(i) = v(j) = (v(i) + v(j))/2$. The current estimate $\widehat{X}(i)$ is obtained according to $\widehat{X}(i) = X(i)$ when $v(i) = 0$ and $\widehat{X}(i) = v(i)$ otherwise.

Node count. At the beginning, one node i stores $x(i) = 1$ and all the remaining nodes store 0. It is easy to see that with this values, the protocol converges to $1/N$ as long as the resulting contact graph [4] is connected. The number of nodes is obtained as the inverse of the estimate. To solve the problem of identifying a single node and to provide a more robust estimate, several concurrent instances of the basic version can be run, each started by a single node.

3.2. POPULATION Algorithm

In this algorithm, the estimate is calculated based on *tokens*. At the beginning of the run, every node i is assigned a set of $t(i) = X(i)$ tokens (notice that $X(i)$ may

Algorithm 2 POPULATION($t(j), a$)▷ @ node i **Init:** $t(i) \leftarrow X(i)$

▷ Initialize tokens

-
- 1: $\widehat{X}(i) \leftarrow \max\{\widehat{X}(i), \widehat{X}(j), t(i) + t(j)\}$
 - 2: **if** $a < 0.5$ **then**
 - 3: $t(i) \leftarrow t(i) + t(j)$
 - 4: **else**
 - 5: $t(i) \leftarrow 0$
 - 6: **end if**
 - 7: **return** $\widehat{X}(i)$
-

be real valued though). At each meeting, node i and node j toss a fair coin: the node winning the ballot, say node i , gathers the overall tokens. The counters are updated accordingly: $t(i) \leftarrow t(i) + t(j)$ and $t(j) \leftarrow 0$. The estimate produced by the meeting pair is then given by the maximum value between the old estimates and the sum of the tokens. At the increase of the number of inter-meetings, tokens gather on a single node j which possess the exact estimate $\widehat{X}(j)$.

Alg. 2 contains the pseudocode. For the ease of reading, the coin tossing procedure is assumed to generate an input variable $0 \leq a \leq 1$ at node i , whereas at node j the input argument is $1 - a$.

Node count. At the beginning, $t(i) = X(i) = 1$ for all nodes i ; i.e., each node owns a token. This variant applies also to the two protocols described in the following.

3.3. C-POPULATION Algorithm

This variant of POPULATION takes advantage of non-uniform meeting patterns among nodes. In this protocol, the input variable x is generated according to the relative fraction of the node meetings experienced by the two nodes. The rationale is that tokens should be gathered at those nodes which are able to perform and diffuse the estimate faster.

This variant works as follows. A local meeting counter $m(i)$ is maintained at each node, initialized at 1 at the beginning of a run. When a meeting occurs, node exchange their $t()$ and $m()$ variables. Variable a is now computed as $a \leftarrow m(i)/(m(i) + m(j))$; note that one node will get a value $a = v$, and the other the value $a = (1 - v)$, meaning that only one of them will get all the tokens, as in POPULATION. Finally, the meeting count is incremented, $m(i) \leftarrow m(i) + 1$.

In the following, we will refer to this variant as C-POPULATION algorithm, where the C reads “clustered”.

The complete pseudo-code is shown in Algorithm 3.

3.4. Two-Phases Algorithm

This algorithm builds on C-POPULATION adding a further phase in order to speed up the final stages of the computation - where most of the tokens are concentrated in few nodes that are unlikely to meet. The idea is that if a node i cannot improve the estimate for a given number MIN-TOKEN of consecutive meetings, and it possesses

Algorithm 3 POPULATION($t(j), m(j)$) ▷ @ node i

Init: $t(i) \leftarrow X(i), m(i) \leftarrow 1$ ▷ Initialize tokens

- 1: $\widehat{X}(i) \leftarrow \max\{\widehat{X}(i), \widehat{X}(j), t(i) + t(j)\}$
- 2: $a \leftarrow m(i)/(m(i) + m(j))$
- 3: **if** $a < 0.5$ **then**
- 4: $t(i) \leftarrow t(i) + t(j)$
- 5: **else**
- 6: $t(i) \leftarrow 0$
- 7: **end if**
- 8: **return** $\widehat{X}(i)$
- 9: $m(i) \leftarrow m(i) + 1$

Algorithm 4 2-PHASES($t(j), m(j), E_j$) ▷ @ node i

Init: $ac \leftarrow 0, E_i \leftarrow \emptyset$ ▷ Aggregation counter

- 1: $E_i \leftarrow E_i \cup E_j$
- 2: $\widehat{X}_{old}(i) \leftarrow \widehat{X}(i)$
- 3: $\widehat{X}(i) \leftarrow \text{C-POPULATION}(t(j), m(j))$
- 4: $ac \leftarrow ac + 1$
- 5: **if** $\widehat{X}(i) > \widehat{X}_{old}(i)$ **then**
- 6: $ac \leftarrow 0$ ▷ Reset the aggregation counter
- 7: **end if**
- 8: **if** $ac > \text{MAX-AGGR}$ **and** $t(i) > \text{MIN-TOKEN}$ **then**
- 9: $E_i \leftarrow E_i \cup \{(randID(), t(i))\}$
- 10: $t(i) \leftarrow 0$
- 11: **end if**
- 12: $\widehat{X}(i) \leftarrow \max\{\widehat{X}(i), t(i) + \sum_{(id,s) \in E_i} s\}$
- 13: **return** $\widehat{X}(i)$

a number of tokens larger than a reference threshold MAX-AGGR, it start to spread epidemically the news that node i possess a given amount of tokens. Nodes collect news coming from different nodes and compute a more precise estimate of the size.

The pseudocode of this algorithm is reported in Alg. 4. At each meetings, nodes exchange the information required for C-POPULATION ($t(j)$ and $m(j)$) plus a cache of “signed” estimates E_j (stored at each node). The received estimates are merged to the local cache (line 1) and a new estimate is computed through C-POPULATION (line 3). The ac counter of meetings is increased by 1, unless the new estimate is larger than the previous one, in which case it is reset to zero (lines 4–7). If ac is now larger than MIN-TOKEN and the number of local tokens is larger than MAX-AGGR, a new pair (*identifier, number of tokens*) is inserted in E_i , while the token counter is set to zero (lines 8–11).

Note that for the sake of simplicity we referred to the case of nonnegative $X(i)$ s.

Trace	PAIRWISEAVG	C-POPULATION
RWP	$8.00 \pm 0.00 \cdot 10^3$	$2.06 \pm 0.24 \cdot 10^5$
Reality	$2.47 \pm 0.16 \cdot 10^8$	$2.16 \pm 0.19 \cdot 10^7$
Haggle1	$1.23 \pm 0.01 \cdot 10^7$	$1.85 \pm 0.14 \cdot 10^6$
Haggle2	$1.80 \pm 0.02 \cdot 10^7$	$2.11 \pm 0.19 \cdot 10^6$
Haggle3	$1.63 \pm 0.06 \cdot 10^6$	$9.67 \pm 0.76 \cdot 10^5$
NUS	$1.06 \pm 0.01 \cdot 10^8$	$5.35 \pm 0.87 \cdot 10^7$
CN	$3.25 \pm 0.25 \cdot 10^4$	$1.61 \pm 0.82 \cdot 10^5$
Trace	POPULATION	2-PHASES
RWP	$2.10 \pm 0.23 \cdot 10^5$	$2.21 \pm 0.09 \cdot 10^4$
Reality	$2.19 \pm 0.40 \cdot 10^9$	$2.24 \pm 0.05 \cdot 10^7$
Haggle1	$2.20 \pm 0.30 \cdot 10^7$	$1.66 \pm 0.18 \cdot 10^6$
Haggle2	$4.15 \pm 1.21 \cdot 10^7$	$1.66 \pm 0.10 \cdot 10^6$
Haggle3	$7.52 \pm 1.21 \cdot 10^6$	$7.43 \pm 0.42 \cdot 10^5$
NUS	$1.54 \pm 0.30 \cdot 10^9$	$3.29 \pm 0.13 \cdot 10^7$
CN	$9.02 \pm 6.50 \cdot 10^4$	$8.92 \pm 4.75 \cdot 10^4$

Table 1: Convergence time (w/o termination), in seconds; 95% confidence intervals computed over 30 runs.

4. Performance Evaluation

4.1. Evaluation Methodology

DTNs are characterized by a fully distributed architecture, where the information is conveyed by exploiting the physical mobility of nodes. As such, the mobility pattern of nodes plays a crucial role in the performance evaluation of these networks. In order to deal with this issue, two standard approaches exist. The first one leverages synthetic mobility models which mimic the real behavior of mobile nodes. The second one is based on empirical studies, in which nodes' encounters are monitored by tracing their proximity for a given period of time: collected traces are then used to reproduce the meeting pattern of nodes. In this work, we have considered both approaches. Each mobility model has been characterized by the corresponding meeting pattern of the nodes. In real-world traces, meetings are simply the result of an empirical study, whereas in the case of synthetic mobility models the contact patterns have been measured through a preliminary experiment.

With respect to synthetic mobility traces, we considered the *Random Waypoint* (RWP) model, where nodes select a destination at random (usually according to a uniform distribution) and move, on a straight line, till they reach it, and the *Community Model* [8], which reflects the non-homogeneous nature of meetings among people.

As concerns real world traces, we utilized the Haggle traces [6], which report the results of three experiments conducted for tracing the meeting pattern of people inside Intel Research Cambridge CorporateLab (*Haggle 1*), people inside the Computer Lab of the Cambridge University (*Haggle 2*) and people attending the IEEE Infocom 2005 conference (*Haggle 3*); the MIT *Reality* measurement campaign (2004-2005), involving approximately 100 faculty members and students at MIT; the *NUS* [9] dataset, which contains Bluetooth contact traces collected in Singapore (2005-2006), and the

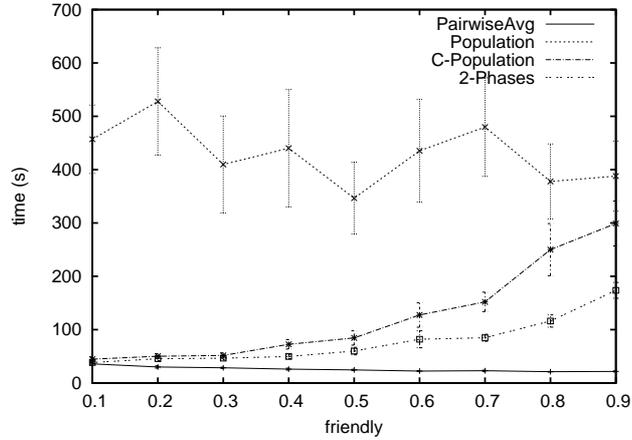


Figure 1: Convergence time vs value of the *friendly* parameter, community model, $N = 200$, $mov = 0.2$, $k = 40$.

Create-Net (CN trace, obtained by monitoring 21 employee for a 4-week period. Additional details can be found in Appendix A.

In what follows, results were obtained by a trace-driven simulator based on Peer-Sim [10]. The simulator reproduces the contact pattern of nodes and, at each meeting, one of the algorithms described in Sec. 3 is executed.

Each experiment has been repeated several times and results are reported together with their confidence intervals. Different runs are obtained by randomly choosing the contact from which the simulation starts. Each simulation ends when the termination condition is satisfied. Due to the limited length of some measurements, traces were arranged in a cyclic fashion, and their pattern was iterated until the end of the experiment.

The performance evaluation of our algorithms is divided in two parts. First, we consider the *asymptotic* performance of the algorithms, in the sense of their behavior when no termination mechanism is employed. For such a case, we evaluate the convergence time and the ability of the algorithms to scale well with the number of nodes. Second, we consider the performance of the algorithms when the termination mechanisms are applied: for this case, we analyze the tradeoff between convergence time and estimate accuracy, and the ability of the algorithms to perform well in the presence of message losses.

4.2. Without Termination: Convergence

In the absence of termination mechanisms, the main metric we consider is the convergence time, defined in this case as the time needed to converge to the actual value of the number of nodes in the system.

We started by considering the community model. We took a scenario with 200 nodes and varied the three parameters describing such model, mov , k and *friendly*. We found that, PAIRWISEAVG is the one offering the best performance in terms of convergence time. POPULATION conversely, performs rather poorly, showing large

convergence time. A more detailed analysis revealed that such a behavior is due to the slow dynamics at the end of the execution, where meetings of the few nodes having tokens become less frequent.

We notice that C-POPULATION behaves slightly better than POPULATION, whereas 2-PHASES offers some advantages over C-POPULATION. Also, performance turns out to be only loosely dependent on the value of the parameters mov and k . Conversely, things change drastically when varying the parameter $friendly$. The results are depicted in Fig. 1 for $k = 40$, $mov = 0.2$ and inter-meeting intensity $\lambda = 0.1 \text{ s}^{-1}$. For low values of $friendly$, in fact, meetings are driven by the Zipf’s law regulating meetings among non-friends. In such case, clusters appear, enabling C-POPULATION and 2-PHASES algorithms to significantly enhance their performance, approaching closely the performance of PAIRWISEAVG. As the value of the parameter $friendly$ increases, on the other hand, their performance decreases quite sharply and approaches that of POPULATION. This is due to the fact that, for high values of $friendly$, the resulting meeting pattern is driven by the Watts–Strogatz small-world model, presenting a rather regular (i.e., memoryless) pattern. Hence, the enhancements introduced by C-POPULATION and 2-PHASES are quite ineffective and, in fact, the gain over pure population protocol is very small.

Next, we evaluated the scalability of the four considered algorithms. We used again a community model, with parameters $mov = 0.2$ and $friendly = 0.2$. We considered a number of nodes ranging from 10^3 to 10^5 . The scaling of the inter-meeting intensity is performed considering nodes moving in an area with constant density of nodes per square meter [11].

We considered a fixed number of friends, $k = 150$. This corresponds to the value known in social sciences as Dunbar’s number [12], which is supposed to represent a limit to the number of individuals with whom people can maintain stable social relationships. The results are shown in Fig. 2. It can be seen that, as the number of nodes increases, the performance of C-POPULATION increases and exceeds that of PAIRWISEAVG. This is due to the fact that, under a constant number of friends k , large values of N lead to the formation of a number of (small) clusters. In such conditions, C-POPULATION offers an advantage over PAIRWISEAVG. Finally, the best scalability is attained by 2-PHASES (MIN-TOKEN = 150, MAX-AGGR = 50 in the experiment).

These first results tell us that PAIRWISEAVG behaves best in the presence of uniform meeting patterns. When clusters are present, C-POPULATION and 2-PHASES offer competitive advantages in terms of convergence time.

We next evaluated the case of real-world traces. The results are presented in Tab. 1. Neglecting RWP, it can be seen that 2-PHASES and C-POPULATION present the best performance among the algorithms considered (the former over-performing the latter). This comes from the fact that real-world traces tend to show a high degree of clustering, typical of real-world mobility patterns. This indicates that such algorithms fit well the features of deployments in which nodes are personal devices and the contact pattern is driven by human mobility.

4.3. With Termination: Convergence and Accuracy

To measure the relative performance of the our algorithms, we adopted a *global* termination condition that stops the simulation whenever the average estimate of nodes

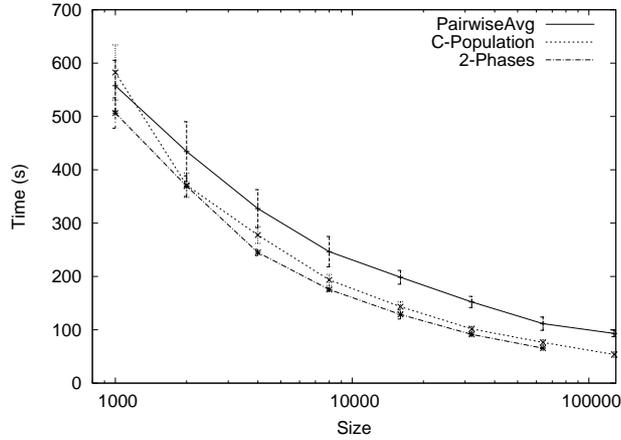


Figure 2: Convergence time vs. total number of nodes, community model, $mov = 0.2$, $k = 150$, $friendly = 0.2$.

approximates the true value by a nominal threshold, e.g. 95% in our experiments. In practice, this simulation trick cannot be employed, and each node requires a *local* termination condition to validate the estimate and proceed to further computations.

We now study the performance of the aforementioned algorithms when a termination condition is enforced. We evaluated the algorithms in terms of the tradeoff between convergence time (time at which the termination rule stops the estimation algorithm) and accuracy (ability of the algorithm to achieve the exact value of N without incurring in premature stops).

4.3.1. Termination condition

The termination condition proposed here works as follows. Every node stores a meeting counter, which is incremented at each meeting. The counter starts from 0, and is reset to 0 whenever the estimate changes. When the counter reaches a given threshold, the computation is considered terminated at that node and the output value is validated. Note that the node continues to perform the computation, in order to support the distributed algorithm and to detect subsequent changes.

In the case of node count procedures, the termination condition is inherently different for POPULATION and C-POPULATION, compared to PAIRWISEAVG. In fact, in the former case the estimate is increasing, so that changes are simply positive integer increments, whereas in the latter case, the estimate of the number of nodes is the inverse of a fraction. Hence, in the first case, the counter is reset when the estimate increases, whereas in the latter, the counter is reset when the estimate changes of a conventional percentage (1% in the following experiments).

4.3.2. Convergence vs. Accuracy

We first performed extensive numerical simulations to understand how the values MAX-AGGR and MIN-TOKEN should be dimensioned in order to let the termination algorithm behave efficiently. We found that setting $MIN-TOKEN = 1$ and

Trace	PAIRWISEAVG	C-POPULATION	2-PHASES
Reality	$1.35 \pm 0.01 \cdot 10^9$	$5.86 \pm 0.32 \cdot 10^8$	$5.52 \pm 0.32 \cdot 10^8$
RWP	$2.55 \pm 0.00 \cdot 10^4$	$3.50 \pm 0.33 \cdot 10^4$	$3.46 \pm 0.14 \cdot 10^4$
Haggle1	$3.02 \pm 0.03 \cdot 10^7$	$1.41 \pm 0.09 \cdot 10^7$	$1.30 \pm 0.09 \cdot 10^7$
Haggle2	$1.05 \pm 0.09 \cdot 10^8$	$7.65 \pm 0.68 \cdot 10^7$	$7.07 \pm 0.80 \cdot 10^7$
Haggle3	$1.34 \pm 0.10 \cdot 10^8$	$1.10 \pm 0.13 \cdot 10^8$	$1.11 \pm 0.13 \cdot 10^8$
NUS	$6.40 \pm 0.22 \cdot 10^8$	$5.02 \pm 0.28 \cdot 10^8$	$4.33 \pm 0.20 \cdot 10^8$
CN	$3.70 \pm 0.01 \cdot 10^5$	$3.35 \pm 0.34 \cdot 10^5$	$3.47 \pm 0.27 \cdot 10^5$

Table 2: Convergence time (with termination), MAX-AGGR = 10, MIN-TOKEN = 1 for RWP, Haggle1/2/3, and CN, MIN-TOKEN = 10 for Reality and NUS; confidence intervals computed over 30 runs.

MAX-AGGR = 10 leads to a good compromise for a wide range of settings. We used such parameters for all considered traces, apart from the Reality and the NUS ones, for which a MIN-TOKEN = 10 was used. We evaluated the performance of PAIRWISEAVG, C-POPULATION and 2-PHASES algorithms using both synthetic mobility traces (RWP) as well as real-world ones. The results are reported in Tab. 2 in terms of convergence time and in Tab. 3 in terms of average estimated value at termination (averages are computed over 30 runs). For all real-world traces C-POPULATION and 2-PHASES offer faster convergence than PAIRWISEAVG. 2-PHASES overcomes C-POPULATION, in terms of estimate accuracy, when applied to regular scenarios, e.g., RWP trace.

As RWP presented a large gap of performance for the considered algorithms, we studied the case more in detail by tracking the dynamics of the algorithms (in terms of value of estimated size vs. time) for 30 runs. The results are shown in Fig. 3. It can be seen that PAIRWISEAVG converges quickly and in a very regular way (i.e., logs from all runs get superimposed) after an initial overshooting. The convergence of 2-PHASES is somehow noisy, but it attains the correct value for all runs. On the contrary, C-POPULATION algorithm shows a very large variability in both the value at which the estimate converges as well as in the time at which the estimate stops. We can conclude that such algorithm is not suitable for application to regular patterns. We may also conclude that the 2-PHASES algorithm represents an interesting choice for deployment, as it is able to achieve a good accuracy while converging quickly in all considered settings.

In order to complete the picture, we also studied the number of signed values produced by 2-PHASES. Indeed, as one node needs to keep track of the signed values it received/sent, such parameter describes well the additional resources needed by 2-PHASES over C-POPULATION. The results are presented in Tab. 4. These numbers give an idea of the additional resources needed to run 2-PHASES.

4.4. With Termination: Robustness

One important aspect when dealing with wireless networks is the impact of lost packets. Packet losses may be due, e.g., to interference, noise at the receiver or simply the fact that the mobility of nodes led them out of mutual communication range before completion of a message exchange. Each considered algorithm is based on the exchange of a request/reply pair of packets between meeting nodes. If the request

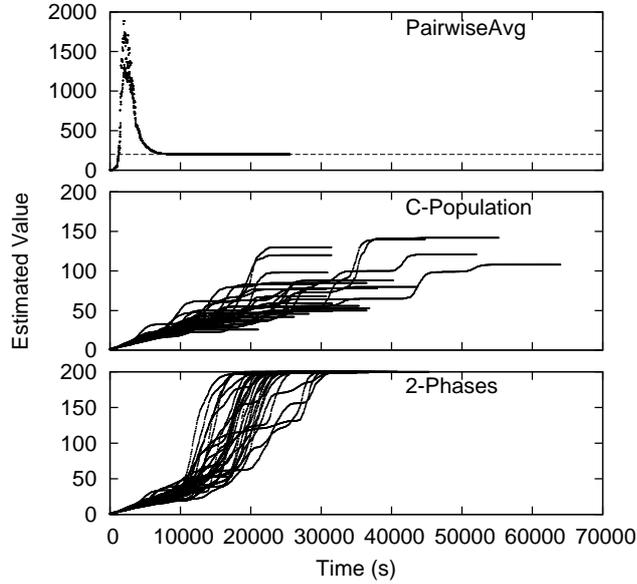


Figure 3: Dynamic behavior of the various algorithms for a RWP mobility trace, $N = 200$, constant speed $v = 5$ m/s, playground size 5000×5000 m^2 , no pause time.

is lost, nothing happens. Here, we provide an analysis of what happens if the reply is lost.

The situation is depicted in Fig. 4. At the beginning of the exchange, both nodes maintain a variable Y_i , a function of the current estimate of the global parameter. During the execution of the algorithm, the values of such variable get exchanged. If the second message is missing, only one of the nodes updates its value. This may affect the final estimated value. This lack of atomicity cannot be avoided in a systems subject to omission failures.

Example: consider PAIRWISEAVG. Y_i represents in this case the inverse of the estimated size. Consider the case when $Y_1 = 0.1$ and $Y_2 = 0.5$. The average of their estimated size before the meeting equals 6. Upon the reception of the first message, the second node updates its estimate as $Y_2 = 0.3$. As the second message gets lost, Y_1 stays at 0.1. After the meeting, the new average of the estimated size equals 6.67. If we reverse the initial estimates, we get a new estimated size (after the meeting) equal to 2.67. So the average estimated size may increase or decrease, depending on which message gets lost.

Example: consider POPULATION with tokens $Y_1 = 1$ and $Y_2 = 3$. Upon the reception of the message, node 2 sets $Y_2 = 0$ and returns its tokens to node 1. As the message does not get received, Y_1 stays at 1 and three tokens get lost. In this case, message losses always lead to a decrease in the average estimated size.

We considered for this case the community model, with parameters $N = 1000$, $friendly = 0.1$, $mov = 0.2$ and $k = 200$, and varied the probability that the return

Trace	Real N	PAIRWISEAVG	C-POPULAT.	2-PHASES
RWP	200	200	76.66	199.73
Reality	2135	2135	2135	2135
Haggle1	110	110	110	109.96
Haggle2	187	187	187	187
Haggle3	214	214	214	214
NUS	841	841	839.42	841
CN	21	21	20.86	20.93

Table 3: Average estimated value at termination.

Trace	N	MAX-AGGR	MIN-TOKEN	# signed values
RWP	200	10	1	15.63
Reality	2135	10	10	168.06
Haggle1	105	10	1	29.70
Haggle2	187	10	1	56.03
Haggle3	215	10	1	82.73
NUS	842	10	10	72.50
CN	21	10	1	6.26

Table 4: Details on the performance and resource consumption of 2-PHASES.

packet is lost during an exchange. We used a loss probability of 0.1%, 1% and 10% in our experiments. In general, the loss of a packet may lead to two types of problems. The first one is loss in accuracy, as the algorithm may stop at a value different from the real one. The second one is loss in convergence speed, as it may be necessary now to undergo additional exchanges before reaching convergence. The results are depicted in Fig. 5. As it can be seen, PAIRWISEAVG is very robust to noise: message losses impact the convergence time but most of the runs converge to the expected size (or values very close to it). On the other hand, C-POPULATION suffers quite heavily in the presence of message losses. These affect both the convergence time (which gets noisy, while still being lower than that of PAIRWISEAVG) as well as the estimated network size. 2-PHASES, on the other hand, offers a good compromise. It offers the quickest

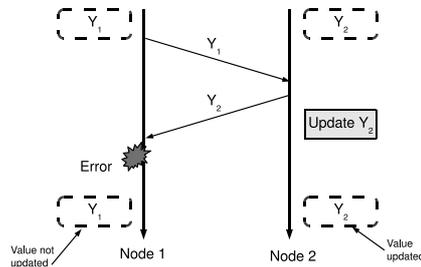


Figure 4: Message loss pattern considered.

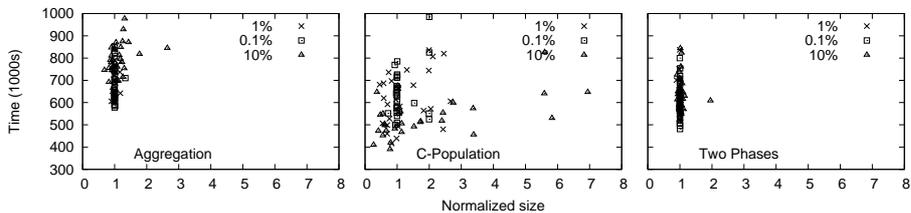


Figure 5: Convergence time and final estimated value for Community Model, $N = 1000$, $friendly = 0.1$, $mov = 0.2$, $k = 200$, 30 runs.

convergence and in general the estimated value is quite close to the real one. We only experienced one case in which the estimate was far from the actual one: the cause was the loss of a key message during the final phase of the algorithm. We may conclude that 2-PHASES offers again a good compromise between performance and accuracy, even in case of message losses.

5. Online Protocol Tuning

The distributed estimation of parameters is a very powerful tool, especially in the case of dynamic scenarios, where network conditions change over time (e.g., the number of nodes). In this case, the result of the estimation can be used in order to let nodes behavior of the system adapt dynamically. In this section, we describe the application of node counting to the dynamic tuning of the parameters of a specific message forwarding algorithm.

In the following, we assume an opportunistic communication system to be deployed in an urban environment, and the density of nodes to be following the daily patterns of people. In particular, we assume the density of nodes to be low (20 nodes) until 8 AM in the morning, when people leave their homes for going to school, work, etc.. The city remains “busy” (200 nodes) between 8 AM and 19 PM, before slowing in the evening time (90 nodes) and finally going back to sleep (20 nodes) after midnight.

5.1. Dynamic Node Counting

As a first experiment we measured the ability of the proposed counting schemes to keep the pace of such daily patterns, and the accuracy of the estimates over time. Starting from the algorithms described in Sec. 3, we introduce a *restarting mechanism* which periodically restarts the counting process. The newly produced estimate of the number of nodes is then used by the different algorithms such as the forwarding mechanism. Clearly, the more often the counting process is restarted, the faster is able to detect the change in the environment conditions. At the same time, as confirmed in Tab. 2 and in Fig. 3, too small restarting periods correspond to a bad accuracy of the algorithm, due to the convergence time of the algorithm.

Fig. 6 presents the PAIRWISEAVG estimate of the number of nodes over time, for a restarting period of 10, 20 and 30 minutes, respectively. As it can be easily observed, for a restarting period of 10 and 20 *min.* the PAIRWISEAVG algorithm is not able to adapt to the system dynamics. This is due to the premature stop of the algorithm, which

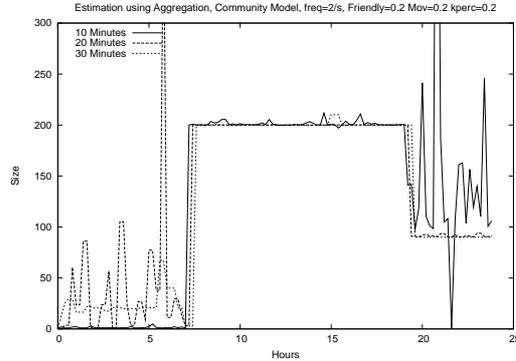


Figure 6: Dynamic estimation of the number of nodes in the case of Community Model mobility model ($friendly = 0.2$, $mov = 0.2$, $k_{perc} = 0.2$), utilizing the PAIRWISEAVG algorithm.

is not yet converged to the correct value of nodes. In the case of 10 *min.*, PAIRWISEAVG is estimating a lower value of nodes, and this is reflected in either a lower (as in the case of 20 nodes) or in a noisy (as in the case of 90 nodes) estimate. Differently, with a restarting period of 20 *min.*, PAIRWISEAVG is probably in its “overshooting” phase, which leads to a much higher estimate of the number of nodes, as evident in the case of 20 nodes.

5.2. Dynamic Adaptation of Forwarding Mechanisms

As a second step, we have evaluated how such dynamic estimate of the number of nodes in the system can be used by other algorithms. In particular, we have focused on the dynamic tuning of forwarding mechanisms in DTN networks [1], which use opportunistic forwarding for achieving network-wide communications. The forwarding mechanism is highly dependant on the density of the network, as it regulates the number of copies that each node is allowed to generate for each message. Clearly, the larger the number of copies of a message in the system, (i) the faster it reaches its destination (ii) the more it is robust with respect to the nodes mobility and node/link failures. On the other hand, in order to have more copies of the same message traveling in the network at the same time, a larger amount of network resources has to be exploited. Given this tradeoff, the knowledge of the number of nodes in the network can be of great help in fine tuning the forwarding mechanism.

In the following, we will refer to the Spray-and-Wait [13, 14] protocol, since it represents a reference case in literature due to its provable high efficiency in trading off number of messages and end-to-end delay. The Spray-and-Wait protocol has a budget of L copies that can be released in the network; initially the overall budget is retained at the source. Depending on the strategy used to release copies, the protocol shows different performances.

As depicted in Fig 7, the effect on the choice of L has an impact on both the delay experienced by messages to reach the intended destination, and the overall number of copies generated in the network. As it is intuitively clear, the higher the value L , the

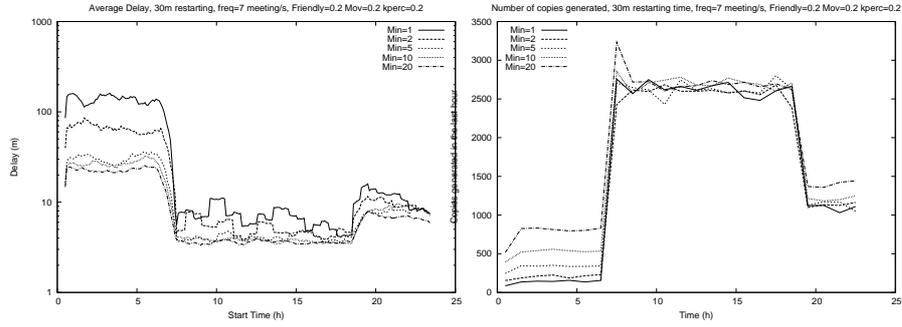


Figure 7: Delay and number of copies over time, for various copies budget L , in the case of Community Model mobility model ($friendly = 0.2$, $mov = 0.2$, $k_{perc} = 0.2$) and the PAIRWISEAVG algorithm for the estimation of the nodes number.

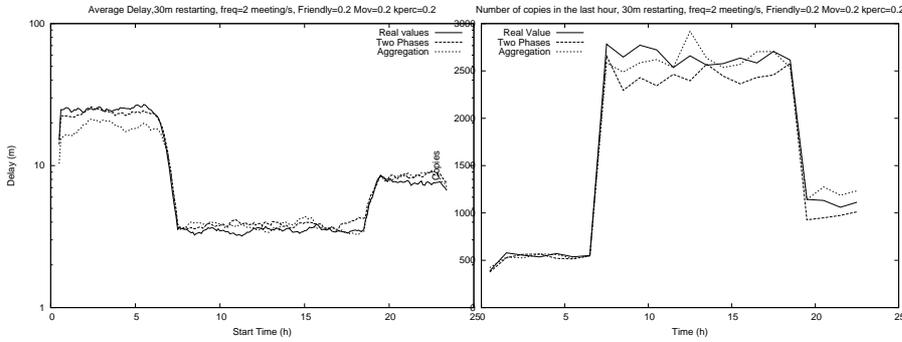


Figure 8: Delay and number of copies over time, for the various nodes estimation algorithms, in the case of Community Model mobility model ($friendly = 0.2$, $mov = 0.2$, $k_{perc} = 0.2$).

more copies are generated. This leads to a better diffusion of data in the network and, consequently, a lower delay. Such difference is evident when increasing L from 1 to 5, while is less significant for higher values.

We performed a set of experiments that demonstrate how the tuning of the algorithm works in the dynamic case.

6. Related Work

Aggregation is an hot problem in distributed systems; it has been studied in the most diverse environments, including both wired and wireless settings. In wired networks, the possibility of building structured and semi-structured topologies allows for several different approaches, such as tree-based [2] and gossip-based [4, 15]. PAIRWISEAVG is derived from the work of Jelasity et al. [4], where the algorithm is applied to random topologies maintained through a peer sampling service [16]. At the best of our knowledge, this is the first time that PAIRWISEAVG is applied to DTNs.

In wireless networks, the problem of distributed averaging (also known as the distributed consensus problem, or the agreement algorithm), has been introduced by Tsitsiklis [17], and it is concerned with letting a distributed set of processors converge to some common value. This problem has been studied in the context of sensor fusion by Spanos et al. [18], as well as by Xiao et al. [19], and in the context of vehicle formation control by Fax and Murray [20].

The other algorithms described in this work are derived from the literature on population protocols [5]. The population protocol framework can be used to model mobile, ad hoc sensor networks consisting of very limited agents with no control over their own movement. Agents are identically programmed finite state machines that interact with one another to carry out a computation. Meetings between pairs of agents cause them to update their state.

Examples of mechanisms for aggregating information over DTNs are included in the work of Spyropoulos et al. [13, 14] and Walker et al. [21]. In particular, as already described before, the Spray-and-Wait algorithm [13, 14] is based on the knowledge of the number of nodes included in the network; the authors propose a complex mechanism to evaluate this parameter, which assumes approximately exponentially distributed meeting times. We have shown already that for the online tuning of that protocol, our algorithms do not require this assumption to be true. The work of Walker et al. [21] describes a particular application of an aggregation technique; the goal is to limit the *carrier fraction*, i.e. the ratio of the number of nodes who carry a message w.r.t. the total number of nodes.

7. Discussion and Conclusions

In this paper, we have presented methods for estimating global parameters in DTNs. Starting from techniques developed for distributed computing applications (PAIRWISEAVG, POPULATION), we have developed variants thereof (C-POPULATION, 2-PHASES), which achieve better performance by exploiting features commonly present in real-world DTN mobility patterns (in particular: clustering). Termination algorithms have also been introduced and described. Validation has been performed through extensive simulations, carried out using a variety of contact traces, both synthetic and experimental.

Our study can lead to the following recommendations for practitioners dealing with real-world DTN deployments:

- If the meeting pattern is regular (i.e., memoryless), PAIRWISEAVG offers the best performance in terms of convergence speed and robustness to message losses;
- If the meeting pattern is irregular (i.e., clustered) and nodes have stringent memory requirements, C-POPULATION offers a good trade-off in terms of performance and resource usage;
- If the meeting pattern is irregular and no stringent memory constraints are present, 2-PHASES offers very good performance in terms of convergence time, accuracy of estimation and robustness to message losses.

The counting algorithms presented here can be easily adapted to compute other functions. E.g., it is possible to compute how many unique nodes have a copy of a given message by creating tokens only in those nodes; PAIRWISEAVG is clearly suitable for any kind of average; etc.

Future work includes extending the presented mechanisms to situations in which the quantities $X(i)$ vary dynamically. In this case, one would like to track the evolution over time of a given global parameter. This could be achieved by, e.g., by periodically restarting the counting algorithms introduced. However, more sophisticated techniques can be envisaged, leading to a better and smoother tracking of the variation in the network status.

References

- [1] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proc. of SIGCOMM'03, ACM, 2003, pp. 27–34. doi:<http://doi.acm.org/10.1145/863955.863960>.
- [2] R. Van Renesse, K. P. Birman, W. Vogels, Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining, ACM Trans. Comput. Sys. 21 (2).
- [3] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, H. Pirahesh, Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals, Data Min. Knowl. Discov. 1 (1) (1997) 29–53.
- [4] M. Jelasity, A. Montresor, O. Babaoglu, Gossip-based aggregation in large dynamic networks, ACM Trans. Comput. Syst. 23 (1) (2005) 219–252.
- [5] J. Aspnes, E. Ruppert, An introduction to population protocols, Bulletin of the European Association for Theoretical Computer Science 93 (2007) 98–117.
- [6] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, J. Scott, Impact of human mobility on the design of opportunistic forwarding algorithms, in: Proc. of INFOCOM'06, Barcelona, Spain, 2006.
- [7] A. J. Demers et al, Epidemic algorithms for replicated database maintenance, in: Proc. of the ACM PODC'87, 1987.
- [8] A. G. Miklas, K. K. Gollu, S. Saroiu, K. P. Gummadi, E. de Lara, Exploiting social interactions in mobile systems, in: Proc. of UBIComp'07, Innsbruck, Austria, 2007.
- [9] V. S. W. Wang, M. Motani, Adaptive contact probing mechanisms for delay tolerant applications, in: Proc. of MobiCom'07, Montreal, Quebec, Canada, 2007.
- [10] M. Jelasity, A. Montresor, G. P. Jesi, S. Voulgaris, PeerSim - Peer-to-Peer simulator, <http://peersim.sf.net>.

- [11] A. Guerrieri, A. Montessor, I. Carreras, F. De Pellegrini, D. Miorandi, Distributed estimation of global parameters in delay-tolerant networks, Tech. Rep. 200800023, CREATE-NET (Aug. 2008).
- [12] R. Dunbar, Coevolution of neocortical size, group size and language in humans, *Behavioral and Brain Sciences* 16 (1993) 681–735.
- [13] T. Spyropoulos, K. Psounis, C. S. Raghavendra, Spray and wait: An efficient routing scheme for intermittently connected mobile networks, in: *Proc. of ACM SIGCOMM WDTN'05*, 2005.
- [14] T. Spyropoulos, K. Psounis, C. S. Raghavendra, Efficient routing in intermittently connected mobile networks: the multiple copy case, *IEEE Trans. on Networking* 16 (2008) 77–90.
- [15] D. Kempe, A. Dobra, J. Gehrke, Gossip-based computation of aggregate information, in: *Proc. of FOCS'03*, IEEE, 2003.
- [16] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, M. van Steen, Gossip-based peer sampling, *ACM Trans. Comput. Sys.* 25 (3) (2007) 8.
- [17] J. N. Tsitsiklis, Problems in decentralized decision making and computation, Ph.D. thesis, Massachusetts Institute of Technology (1984).
- [18] D. Spanos, R. Olfati-Saber, R. M. Murray, Distributed sensor fusion using dynamic consensus, in: *Proc. of IFAC'05*, 2005.
- [19] L. Xiao, S. Boyd, S. Lall, A scheme for asynchronous distributed sensor fusion based on average consensus, in: *Proc. of IPSN'05*, 2005.
- [20] A. Fax, R. M. Murray, Information flow and cooperative control of vehicle formations, *IEEE Trans. on Automatic Control* 49 (2004) 1465–1476.
- [21] B. D. Walker, J. K. Glenn, T. C. Clancy, Analysis of simple counting protocols for delay-tolerant networks, in: *Proc. of CHANTS'07*, Montréal, Québec, Canada, 2007.
- [22] J.-Y. L. Boudec, M. Vojnovic, Perfect simulation and stationarity of a class of mobility models, in: *Proc. of IEEE Infocom'05*, 2005.
- [23] Machine perception and learning of complex social systems.
URL reality.media.mit.edu/
- [24] R. Groenevelt, P. Nain, Message delay in MANETs, in: *Proc. of ACM SIGMETRICS'05*, Banff, Canada, 2005, pp. 412–413, see also R. Groenevelt, *Stochastic Models for Mobile Ad Hoc Networks*. PhD thesis, University of Nice-Sophia Antipolis, April 2005.

A. Mobility Models

In the present work, we considered the following synthetic mobility models:

- *Random Waypoint (RWP)*: in the RWP model, nodes select a destination at random (usually according to a uniform distribution) and move, on a straight line, till they reach it. When reaching the destination, nodes pause for a random interval of time, and then repeat the process. In our implementation of RWP, MNs move at a constant speed and without pausing. We reproduced a perfect simulation [22], sampling the initial location of nodes according to the corresponding stationary distributions (which is not uniform). Subsequent destinations are then sampled from the uniform distribution. This approach eliminates the time needed for the simulation to reach the stationary regime. In particular, we considered the case of 100 nodes moving over a $2500 \text{ m} \times 2500 \text{ m}$ playground with no pausing and a speed of $v = 5 \text{ m/s}$. Two nodes are assumed in communication range if their distance falls below $R = 25 \text{ m}$.
- *Community Model*: in [8], a mobility model capturing key social and temporal aspects of mobile environments has been introduced. In particular, authors considered those application scenarios where the mobile nodes are constituted by the mobile devices of people. In this case, the mobility pattern of nodes is determined by the social dynamics of people. In order to capture this aspect, authors proposed a mobility model where meetings are distinguished between friend and strangers encounters. In particular, a Watts-Strogatz small-world model is assumed to regulate the contact among friends, while a Barabasi scale-free model is used to generate encounters between strangers. Clearly, this makes friends meetings more frequent than those with strangers, as occurring in real social dynamics. The parameters of this mobility model are the number of friends per node k , the Zipf parameter α regulating the stranger distribution, the probability *friendly* of encountering a friend.

and the following real-world mobility traces:

- *Haggle*: in [6], the authors report extensive experimentations conducted in order to trace the meeting pattern of mobile users. A slightly modified version of iMotes, equipped with a Bluetooth radio interface, was distributed to a number of people, each device collecting the time epoch of meetings with other Bluetooth devices. Due to technical problems, some of the traces have not been collected. Three experiments have been conducted:
 - *Haggle 1*: 12 out of 16 iMotes, staff of Intel Research Cambridge CorporateLab, 1 week;
 - *Haggle 2*: 12 out of 12 iMotes, students of the Computer Lab of the Cambridge University, 1 week;
 - *Haggle 3*: 42 out of 50 iMotes, people attending the IEEE Infocom 2005 conference, 3 days.

- *Reality*: the MIT Reality Mining experiment [23], carried out throughout academic year 2004/05, involved approximately 100 people (faculty and students) at MIT. A software for Nokia Symbian Series 60 Phone, able to record all meetings exploiting a Bluetooth interface, was developed. The result is a single, extremely large, trace file, covering about 350,000 hours. We used for our numerical analysis the first 200,000 entries of the file.
- *NUS*: the *NUS* dataset [9] contains Bluetooth contact traces collected in Singapore from end 2005 to early 2006. The measurements campaign consisted of 12 devices, 3 static and 9 mobile. The static devices were line powered, and placed in three of the busiest lecture theaters on National University of Singapore campus. The 9 mobile probes were distributed to various people with a different social role, including 5 students on campus, 2 faculty members and 2 students who lived off campus.
- *CN*: the Create-Net dataset [] has been obtained by monitoring 21 employee — playing different roles within our organization and working on different floors of the same building — for a 4-week period. Employee were asked to carry a mobile running a java application, and relying on Bluetooth connectivity for exchanging data. The application periodically triggers (every 60 seconds) a Bluetooth node discovery. Whenever another device is detected, its Bluetooth address, together with the current timestamp is saved in the permanent storage of the device for a later processing.

B. Scalability Measurements

In section 4.2, we addressed the scalability of the proposed algorithms, characterizing the convergence time with respect to the number of nodes N . The intensity of the base contact process, λ , was scaled with reference to a set of N nodes moving on a square playground of size $\sqrt{N} \times \sqrt{N}$, i.e. we considered a scenario with constant node density. For synthetic mobility models such as Random Waypoint, Random Direction and Brownian Motion, in particular, it was shown in [24] that the inter-meeting intensity, i.e., the number of meetings per second of any pair of nodes, can be approximated as $k \cdot v/L^2$, where L is the playground side, v is the node speed and k is a positive constant. Thus, since distances increase as fast as \sqrt{N} with the area, for a given constant speed v , the inter-meeting intensity decreases as \sqrt{N} . Finally, the meeting process intensity for all pairs of nodes is $\lambda = k \binom{N}{2} v/L^2$, so that the required scaling for $\lambda(N) \sim N^{3/2} s^{-1}$.