J. Parallel Distrib. Comput. 96 (2016) 95–105

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Count on me: Reliable broadcast and efficient routing in DTNs through social skeletons

Alessandro Mei, Natascia Piroso, Julinda Stefa *

Department of Computer Science, Sapienza University of Rome, Italy

HIGHLIGHTS

- Building distributively a sparse connected overlay of a DTN called Social Skeleton.
- Mathematically proving the connectivity properties of the Skeleton.
- Building upon it COM—the *first* reliable broadcast mechanism for DTNs.
- Exploiting the opportunistic Skeleton routes to deliver SR—a new routing mechanism.
- Assessing the performance of COM and SR through experiments on large network traces.

ARTICLE INFO

Article history:
Received 30 July 2015
Received in revised form
8 January 2016
Accepted 12 May 2016
Available online 19 May 2016

Keywords:
Reliable Broadcast
Efficient multi-hop routing
Social mobile networks

ABSTRACT

This paper challenges the belief that reliable broadcasting and efficient routing primitives are not possible when DTNs are involved. Firstly, we present COM, a reliable broadcasting mechanism for hybrid networks where nodes can rarely use long-range and costly communication links (e.g. 3G) to complete missing opportunistic links. COM is based on the Social Skeleton, a compact and connected subgraph, computed in an efficient and distributed way, that best represents the strongest social links among nodes. COM exploits the Social Skeleton to guarantee reachability of 100% of nodes with the lowest number of long communications. Then we empirically prove that the Social Skeleton can be used to build routing mechanisms upon it. We deliver SR (Skeleton Routing), which involves at most 3 copies per message, and yields delivery rates up to 5.5 times higher than state-of-the-art forwarding protocols.

1. Introduction

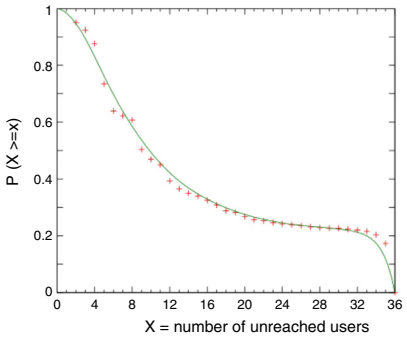
Pocket Switched Networks (PSNs) [14,3] are a special type of Delay Tolerant Networks [9] where short-range communicating nodes are carried around by people. Smartphones, laptops, or body-sensors with built-in Bluetooth or WiFi communication technology enable the Pocket Switched networking paradigm. The nodes communicate opportunistically as the human owners get in proximity of each-other. The social-related features, brought in by the involvement of the people, have given rise to sophisticated and elegant routing [22,5,15,8,24] and security-related [2,21,25,29,12] mechanisms for PSNs.

To the best of our knowledge, *Reliable Broadcast* – spreading important information to *all* network nodes within a given message time-to-leave (TTL) – has never been addressed in PSNs. Reliable broadcast is a foundational primitive for every kind of distributed networked system. In particular it is so for PSNs. Without reliable broadcast, PSN nodes cannot even know who is and is not part of the network or simply agree on a given protocol to use. Let alone to implement other important fundamental mechanisms like definition and spread network community structure information, distributed snapshots of global states, leader election, report on node failure or misbehavior, and others. In PSNs it is not possible to perform a reliable broadcast only on the basis of short-range opportunistic communications. As we empirically show in Section 2, even a pervasive distributing strategy like Epidemic [30] fails to achieve 100% delivery. Of course, one can use costly remote communication technologies (e.g. cellular network) to reach those nodes that Epidemic fails to. But, besides from being more expensive, this requires having a global vision of the network state (who received what), impossible to achieve without a reliable broadcast mechanism.

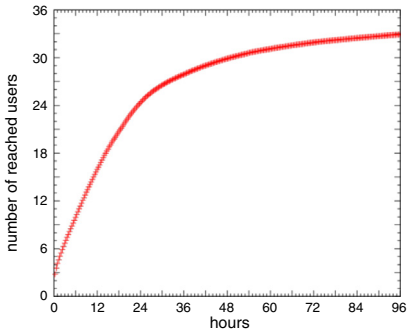
In this scenario the issue becomes: Is it possible to build a distributed and efficient reliable broadcast primitive that guarantees reachability of 100% of nodes? Distributed in the sense that it only

* Corresponding author.
E-mail addresses: mei@di.uniroma1.it (A. Mei), piroso@di.uniroma1.it (N. Piroso), stefa@di.uniroma1.it (J. Stefa).

http://dx.doi.org/10.1016/j.jpdc.2016.05.011
0743-7315/



(a) Tail distribution of the number of *unreached* users with Epidemic. 1 day TTL.



(b) Average number of users *reached* by a message starting at a random source and time. 4 days TTL.

Fig. 1. Unreliability of Epidemic on the Cambridge dataset.

relies on local information of nodes to spot the unreached members of the network. Efficient so that it requires as few remote messages as possible to deliver the information to the unreached nodes before the TTL elapses. In this paper we attack this issue and show how to implement such a broadcast primitive. Our contribution is as follows:

- We show how to build, in a distributed way, what we call a *Social Skeleton*—a sparse connected opportunistic overlay of the network (Section 5). The Social Skeleton is made up of strong social links, thus underlining couples of nodes that are most likely to have opportunistic contacts. Links also define a mutual commitment between nodes that take on the responsibility of transmitting messages to their counterpart.
- We prove the connectivity properties of the distributively built Skeleton in Section 5.4.
- Upon the Social Skeleton, we build a reliable and distributed broadcast primitive called Count-On-Me (COM). The primitive exploits the strength of the links in the Skeleton: If they are carefully measured, opportunistic dissemination between a linked couple is usually possible. But, in case of failure, nodes resort to the long-range communication infrastructure, right before the TTL is reached, to disseminate the broadcast message to the neighbors they are responsible of according to the Skeleton. Our experiments with real traces made of up to thousands of nodes presented in Section 6.2 show that the COM broadcast primitive generates only a small, near minimal, number of long-range connections to guarantee 100% of delivery.
- Weak links are as important as strong ones in the Skeleton. In fact, they are links that most likely generate remote messages by the end of the TTL. In Section 6.3 we make use of this feature by employing anticipated remote forwarding over these links, as soon as a message is received by one of the relative nodes. Anticipated forwarding eliminates local trapping of messages in well-connected components. Thus, it enables free opportunistic dissemination over links impossible to exploit otherwise. The results show that this process lowers considerably the number of the remote messages.
- Lastly, a further contribution of this work is to show how the Social Skeleton can help to efficiently route messages in a pure opportunistic way. We build upon it the SR (Skeleton Routing) primitive that makes use of distributively built Skeleton end-to-end routes to push messages towards destination. The comparison of SR with well-known state of the art routing mechanisms like Delegation, BUBBLE, and SimBet [8,15,5], shows its prominence in delivery, particularly in large networks, despite its limited-copy nature.

2. Unreliability of opportunistic broadcasting

We start off with evaluating the reliability of broadcasting in opportunistic networks. We aim at empirically evidencing the limits of epidemic forwarding in terms of guarantee to delivery a broadcast message to all users in a network. For this we consider a small network consisting of 36 users, the *Cambridge dataset*, whose short-range contacts have been collected and logged for 11 days [19,20]. We repeatedly chose a random source s and a random starting time t_s from the 11 days real trace. For each drawn starting time, we simulate epidemic forwarding up to a given TTL. At time t_s the message is ready at s and starts to spread through the network according to the encounters recorded in the trace. The results are included in Fig. 1. Firstly, we focus on the complementary cumulative distribution of the number of users that are not reached by the message at the end of the simulation time (Fig. 1(a)). As the results show, after 24 h of forwarding, the probability that more than 50% users remain unreached is almost 0.3. The curve also shows a rather heavy tail. The situation is left practically unchanged by the extension of the forwarding period. In Fig. 1(b) the TTL of the message is extended to 4 days. Most of the users are rapidly reached after a period ranging between 24 and 30 h. Afterwards, the curve increases at a slower pace, showing that only a handful of users are further reached by the message in nearly three days time. This means that, for three days, the message is retained by the users that have already received it – over 75% of the total users – and keeps circulating in multiple copies in order to reach less than 15% of the total users.

Cambridge is a rather dense and small trace. So, results showed in Fig. 1 are representative of structured systems where users all know each other and have frequent encounters. The problem can only be exacerbated in larger networks (we see this in the final experimental results of this work). As a result, a reliable broadcasting for opportunistic networks can only be implemented if, when necessary, the unreached users are contacted by more costly remote messages.

3. System requirements and problem definition

We consider a network consisting of n nodes (or *users*). Broadcast messages are generated by users and are to be delivered to *all* other users in the cheapest possible way. Nodes exchange messages using the short-range technology (e.g. Bluetooth or Wi-Fi) if in physical proximity, or long-range wireless technology (e.g. cellular network) otherwise. When a message M is transmitted through the short-range technology we call M a *proximity message*; when M is transmitted through the long-range technology we call it a *remote message*. The problem we consider is to efficiently implement a primitive of *reliable broadcast* on top of this

hybrid network. Remote messages are more flexible than proximity messages. In fact, they enable communication almost at any time, but they are also much more expensive. Indeed, wireless service providers typically ask for a fee to use the cellular network infrastructure. Even when it is part of a fixed monthly plan, it is actually a cost for the providers since the infrastructure has to be large enough to sustain the load. As a simple approximation, we assume that proximity messages are free (there is no infrastructure or service involved). The goal is then to guarantee that each single user receives any broadcast message generated in the network, by making use of the smallest possible number of remote messages.

We assume that nodes can fail, but failure is transient. In other words, after a certain time failing nodes are recovered and fully operational. Faults can be associated with hardware or software failures, battery exhaustion or simply with the intention of the user of being off-line. During failures, nodes can neither send nor receive proximity and remote messages, but remote message that are sent to them are buffered by the infrastructure and delivered after recovery.

Messages are created with an associated time-to-leave (TTL). If there are no faults in the network, a message is expected to reach all other nodes within the given TTL. However, since faults are asynchronous, in the presence of faults we still guarantee that all nodes will eventually receive the message, but we cannot guarantee all nodes to be reached before the TTL.

4. The social skeleton

In a structured system, such as an academy institution, a business company or a city-wide community, people have well defined roles. The role of an individual involves the definition of a behavioral pattern inside the system he is part of: His time schedule, the locations he visits, the people he encounters are all events that are predictable up to a certain extent. Of course, sometimes our behavior deviates from our habits. All we assume is that there is some degree of regularity in the behavior of our users, as suggested by intuition and confirmed by well-known sociological studies [28,11].

When we want to make an information available to all the members of the society we are part of, we naturally turn to the dynamics of our social behavior: We can mentally review our acquaintances in order to delegate the duty of passing the information by *word of mouth*. The spread of information through an opportunistic mobile ad-hoc network is quite similar to the spread of information through a structured system where information is passed by word of mouth. Nodes physically close to each other can exchange data and store it in order to pass it to other nodes in the near future.

4.1. Word of mouth, opportunistic forwarding, and the responsibility graphs

We assume that a message M, starting at a *source* user s, spreads in the network through Epidemic: Users retain self-generated messages or messages received from others, and transmit them opportunistically to other users whenever possible through proximity messages. In our setting, Epidemic is an effective way to disseminate a message. It is the technological alter ego of what word of mouth is in society. However, as empirically shown in the first part of this paper and confirmed by all the literature in the area, Epidemic gives no guarantees that messages are actually received by all nodes in the network.

Our idea is to define a *responsibility graph* that users can look up in order to make sure every other user is informed. A responsibility graph is a graph where the nodes are the users of the network and where an edge $u \xrightarrow{r} v$ indicates that node u commits to make sure that node v receives all broadcast messages that u receives by other nodes. On a high level, our protocol works as follows: When a node u receives a new message M, it opportunistically forwards it by proximity communication to all the nodes it subsequently meets. In particular, during the activity, u keeps track of the possible interaction had with nodes v it is responsible of according to the responsibility graph. I.e., it registers that either u passes the message to them or that they already received it through earlier opportunistic contacts with other nodes. When the TTL expires, u uses this information to spot nodes it is responsible of – it exists an edge $u \xrightarrow{r} v$ in the responsibility graph – that did not meet with u before the TTL. As u has no information on whether they retain the message or not, it sends them M through a remote message. In turn, nodes v receiving a remote message, do the same by propagating it remotely to their own responsibility links. It is easy to see that it makes sense to consider only undirected responsibility graphs. Indeed, when u gets to know that v has received the message, also v gets to know that u has received the message, without overhead. Therefore, assuming that the responsibility graph is undirected does not increase the number of remote messages.

If the responsibility graph is connected, we know that the broadcasting protocol will reach all the nodes of the network. Therefore, by guaranteeing its connectivity, we get the correctness of our reliable broadcast primitive.

One possible solution for the definition of the responsibility graph is to take a star configuration, with a central node u. However, aside from putting a large overhead on node u, most of the broadcasts will incur a high overhead caused by the high number of required remote messages. In fact, more than likely u will physically meet, and thus, get information on, just a handful of nodes in the network before the TTL expires. For all other nodes it will not know whether they have received the message or not. As a result, it will have to send many useless remote messages to nodes that have already received it opportunistically from others (u just does not know it!). So, our goal is to build a responsibility graph that is *connected*, *sparse* (with a small number of edges) and consists of *strong social edges*—edges that most probably will *not* generate useless remote messages. We call such a graph a *Social Skeleton*.

5. How to construct a social skeleton

Here we describe how to build a Social Skeleton through the definition of a series of correlated responsibility graphs. Users collect data regarding contacts with other users during a training period of appropriate length. Then, they evaluate the strength of the relations in terms of encounter probability. Subsequently, each user u selects a small number of *friends* independently. Node u then commits to make sure that its friends get every broadcast message received by u. All this is achieved through several distinct stages. Every stage refines the responsibility graph that has been defined in the previous stage, until the final Social Skeleton is built.

5.1. Best friends selection

Given two users u and v, let $I = I_0 \cup I_1 \cup \cdots \cup I_q$ be the collection of intervals corresponding to the continuous maximal timespans in which they physically see each other, as shown in Fig. 2. Suppose that at time t_0 a message M is received by user u and let t_M be the residual TTL associated to the message. When the TTL elapses, user u storing M sends a remote message containing M to user v only if:

1. $u \xrightarrow{r} v$ and
2. u did not see v during the interval $[t_0, t_0 + t_M]$.

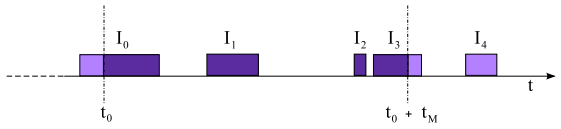


Fig. 2. Blocks on the time axis t represent continuous contact periods between two fixed nodes.

Clearly, if u and v do meet in the interval $[t_0, t_0+t_M]$, then v gets M directly through a proximity message from u, unless v already has received M. In both cases, u records that v detains M and vice versa.

To evaluate the chance of u sending a remote message to v, we are interested in defining the following probability:

$$P\left(I \cap [t_0, t_0 + t_M] \neq \emptyset\right), \tag{1}$$

where t_0 is uniformly sampled at random in the timeframe of the training period. The value t_M represents how long, *at most*, node u can wait for a meeting with v. As all nodes have to be reached by the time of expiry of message M, it is useful to set t_M much smaller than the residual TTL associated to the message, as originally stated instead. t_M should depend on the TTL value and on the approximate number of users in the network. Our experiments have shown that setting $T_M = \mathrm{TTL}/n$ leads on average to the expected result.

Each user chooses the k *friends* that maximize the probability expressed by Eq. (1) by using the indicated t_M. Of course, this is an approximation. Our experiments however show that this is indeed a good one. Note that Eq. (1) is different from the frequency of meetings and it is also different from the cumulative length of the meetings, in the sense that it induces different choices of friends. Indeed, Eq. (1) is exactly what we need to maximize the probability of not sending a remote message through the link (u, v), assuming that the training process predicts well the future encounters.

At this point, we denote with $f_i(u)$ the ith best friend selected by node u. In this first stage, the Social Skeleton is represented by the directed responsibility graph $S = (U, R)$ where U is the set of users and R are the responsibility relations derived from the best friend selection process described above. Formally we have,

$$R = \{u \xrightarrow{r} v \mid u \in U \text{ and } v = f_i(u), \text{ for } i = 1, \ldots, k\}. \tag{2}$$

5.2. Links consolidation

As we discussed earlier, we can make the responsibility graph undirected without having to pay any overhead in terms of induced remote messages. Our second stage then simply requires to turn the responsibility graph defined in Section 5.1 into an undirected graph by replacing all directed edges with undirected ones, so that if $u \xrightarrow{r} v$, then also $v \xrightarrow{r} u$ holds.

5.3. Sparsification process

Social networks often exhibit the *triadic closure* property [7]. I.e., if a strong tie exists between a node u and a node v and between node u and a node w, then there is an increased likelihood that v and w are also connected. In this case, the responsibility graph S constructed through the previous stages is likely to present a large number of small cycles. Cycles are highly undesirable as they can easily generate useless remote messages. Therefore, this third stage aims at eliminating unnecessary responsibility links and is central to our procedure for its effects on costs reduction. As a matter fact, in a structured system of users, as the ones we are considering in this work, the triadic closure property is strongly present. Our experiments on real traces, as well as previous works on the same traces, confirm this assertion.

Each user erases unnecessary links contained in cycles independently from other users, while the sum of all users decisions leads to an agreed *sparsified* version of the responsibility graph. The sparsification process consists of three steps which we describe from the point of view of single node u.

5.3.1. Step 1: Neighborhood discovery

In this first step, node u discovers the local topology of the responsibility graph S built through the previous stages. Given $N_h(u)$ to be the set of nodes at distance at most h from u, we assume that u is able to reconstruct the subgraph $lS_u := S[N_h(u)]$ induced by $N_h(u)$ on S, which we call the *local responsibility graph* of u. Experimental simulations presented later in this paper show that the neighborhood discovery step is not a demanding task at all: Nodes just need to exchange small proximity messages and the size of the explored neighborhood is usually also very small. For the traces considered in this work, that consist of up to thousands of nodes, it is enough to set $h = 3$ or $h = 4$.

5.3.2. Step 2: Cycle detection

The subgraphs lS_u build independently by each node will most certainly contain cycles due to the triadic closure property of social networks [7]. These cycles, however, could generate useless loops: Messages departing from a node v of a given cycle would again return to v during a broadcast process. Clearly, the last transmission is useless, as v itself started the diffusion of the message. Therefore, it becomes important to detect and break cycles in the local responsibility graphs lS_u. But, as we have a choice on how to break cycles, we would like to use a process that keeps the strongest edges and deletes the weakest ones. Recall that weak edges correspond to couple of nodes with the smallest probability of passing messages through proximity contacts. Thus, they are more likely to generate remote messages. A convenient way to eliminate cycles from a graph is by computing its spanning tree. Since we want to drop the weakest edges in the process, we make user u compute a *maximum spanning tree* of the local responsibility graph, which we call $lMST_u$ (local Maximum Spanning Tree of u). This can efficiently be achieved through the well-known Kruskal's algorithm.

By construction, the local responsibility graph built so far is sparse. Thus, so will be the local maximum spanning tree.

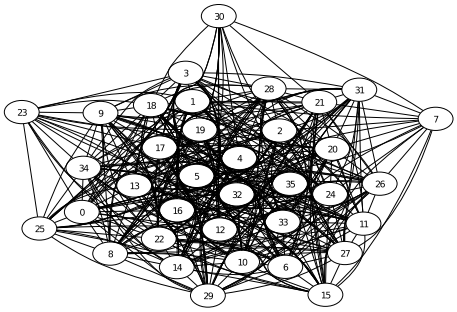
5.3.3. Step 3: Confirmation/deletion of responsibility links

Once node u has found its $lMST_u$, it can proceed to the assessment of its outgoing responsibility edges in lS_u. The decision node u is called to make is simple: Given a responsibility relation $u \xrightarrow{r} v$ in the local responsibility graph, if $u \xrightarrow{r} v$ belongs to the $lMST_u$ calculated by u, then u maintains the link in lS_u. Otherwise, u breaks its commitment to v deleting $u \xrightarrow{r} v$ from lS_u. In the next section we prove that the decision is coherent for adjacent nodes, though taken independently.

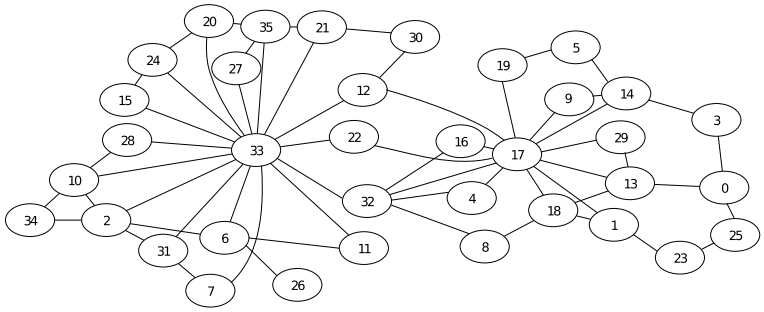
At this point, the responsibility graph built in Section 5.1 is modified according to the results of the confirmation/deletion step followed independently by each user. Precisely, the social links of the new responsibility graph are the (undirected) links of Eq. (2) that have been confirmed in Step 3.

5.4. Local bones are global bones

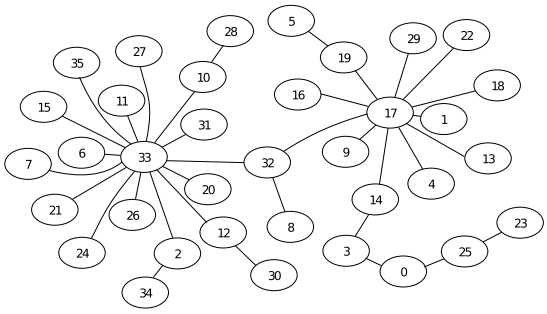
The sparsification process described in Section 5.3 is run by each user locally and independently from other users, and aims at removing links of the responsibility graph that are unnecessary. In this section we prove that nodes agree on the deletion or confirmation of a shared link, so that there is no ambiguity on the definition of a new responsibility graph. Further, we prove that the sparsification process does not create disconnections among nodes



(a) Contact graph.



(b) Responsibility graph after best friend selection and link consolidation.



(c) Social skeleton obtained after the sparsification.

Fig. 3. Generation of the social skeleton for the Cambridge dataset.

connected in the responsibility graph. To this end, let S be the Social Skeleton before sparsification and S' be the Social Skeleton after sparsification. At the local level, in accordance to the neighborhood discovery step in Section 5.3, we have that $lS_u := S([N_h(u)])$ and $lMST_u$ are respectively the local skeletons of u before and after sparsification. This holds for all $u \in U$.

S can be viewed as a weighed graph, where weights are given by the strength of the social relations among users measured during the training process. Each edge can be considered to have a distinct weight. If this is not the case, we break ties according to a lexicographic order on the edges. Under these assumptions, it is well known that the maximum spanning tree of a given graph is unique. In the following, we prove that it is obtained by breaking the cycles by deleting the edges with the minimum weight.

Lemma 1. *Let G be an undirected graph with distinguished weights and let MST be its unique maximum spanning tree. Then $e \in E(G)$ belongs to MST if and only if e is never the edge of minimum weight in any cycle in G.*

Proof. By the well known *cycle property*, if $e \in E(G)$ is the edge of minimum weight of an arbitrary cycle in G, then e cannot belong to a maximum spanning tree of G. Therefore, the deletion in *all* cycles of G of the edge of minimum weight, returns a spanning tree which is also the unique maximum spanning tree. □

So far we know that users act independently. I.e., they generate their local maximum spanning trees by breaking cycles in their local graphs by deleting the edges with minimum weight. This means that each user decides to let go of the responsibility links that are more likely to generate remote messages. But, how do we know that the decision on the retaining or not of a link is coherent for adjacent users? I.e., how do we know that the independent decisions that two users u and v have on whether to keep their responsibility towards each other are mutual? We prove this in the following proposition:

Proposition 1. *In the confirmation/deletion step of the sparsification process, two adjacent users u and v in S agree on the confirmation or deletion of their common edge (u, v).*

Proof. lS_u and lS_v satisfy the hypothesis of Lemma 1. Therefore, (u, v) can be deleted by both u and v if and only if (u, v) is the edge of minimum weight of a cycle C_u of lS_u as well as the edge of minimum weight of a cycle C_v of lS_v. As a cycle C containing (u, v) appears in S_u if and only if it appears in S_v, it follows that nodes u and v always agree on the possible deletion of edge (u, v). □

Finally, we need to prove that the local sparsification process does not disconnect the responsibility graph. We do so in the following theorem:

Theorem 1. *The sparsified version S' of the responsibility graph S has the same number of connected components of S.*

Proof. First, note that if the global S has k connected components, then its MST has also k connected components (spanning trees on a graph do not introduce disconnections). Now, let C be a cycle of lS_u, the local graph of u. Because lS_u is simply the subgraph of S induced by the neighbors of u, C also belongs to S. From Lemma 1, if an edge $e \in lS_u$ does not appear in $lMST_u$, then it must be the edge of minimum weight of a cycle C of lS_u. As C also belongs to S, then e does not belong to the maximum spanning tree MST of S. Therefore, the sparsification process only deletes from S edges that do not appear in MST, so that S' must also have k connected components. □

In Fig. 3 the reader can visually discern the transition from the initial contacts of the training phase (Fig. 3(a)), to the responsibility links (Fig. 3(b)), and finally, to the sparsified Social Skeleton (Fig. 3(c)) for the Cambridge dataset.

5.5. Connecting up the responsibility graph

Unlike Cambridge, large networks with thousands of nodes might result in unconnected responsibility graphs, and thus, unconnected skeletons. Testing connectivity is a hard problem in distributed computing. Indeed, it is not possible to test connectivity in time less than the diameter of the network. Fortunately, social networks are believed to have a small diameter, and this is actually confirmed by our experiments.

To connect up the responsibility graph, it is enough to run a distributed protocol to detect and label its connected components by a distributed version of a standard visit of a graph. Once connected components have been detected, each component elects a leader by taking the node with larger activity (we can use any of the well-known measures used in the literature, like the information dissemination function [26]). If a connected component is small (less than half of the network), the leader chooses a random node and, by using a remote message, connects to that node, provided it belongs to a different component (this can be checked by looking at its label and happens with probability of at least 0.5). However, the newly connected nodes setup the strength of the link to be 0, in order to record a link between components. The process is iterated until the protocol detects one single component that spans the whole network. It is easy to see, by standard probabilistic techniques, that the number of iterations is with high probability logarithmic in the number of components.

At this stage, our responsibility graph is connected, extremely sparse (as our experiments will confirm) and made up of social links that favor short-range communications against long-range expensive communications. We have a fully connected Social Skeleton!

5.6. Dynamic joins and leaves

It is interesting to see how nodes dynamically joining or leaving the network can be handled. It is reasonable to divide time into *epochs*, and to have the training period performed again at the beginning of every epoch. This is important to optimize the system according to the evolution of social relationships in the network. Every epoch can be seen as an independent instance of the network. Therefore, if nodes join or leave the network at the beginning of an epoch, then they do not need special attention.

It is also easy to handle a join occurring during an epoch: The new node only needs to perform a training period in order to connect to the node of the Social Skeleton that maximizes the social strength of the relative edge, according to Eq. (1). The Social Skeleton is trivially still connected and this greedy choice does not considerably reduce the quality of the solution.

Handling dynamic *graceful* leaves in an epoch is also easy. The leaving node is expected to look at the Social Skeleton locally, and connect up the possible disconnections due to its leave. Neighbors can be readily updated through remote messages. Handling sudden leaves or permanent crashes is harder. One possible solution is to let the infrastructure deal with these cases. The infrastructure can detect the permanent failure of a node (by using a long enough timeout) and impose the re-construction of the Social Skeleton. We are currently working on more efficient solutions based on the idea that these failures can be locally detected and recovered.

6. CountOnMe: Performance evaluation

In Section 5 we have explained how to construct a Social Skeleton to realize a reliable broadcast. We recall that a broadcast is defined *reliable* when, departing from any node in the network, it reaches *all* other nodes within a given TTL. A link in the Social Skeleton is a commitment between two nodes to make sure that the counterpart in the link is reached by any broadcast message. On the basis of this commitment, we refer to the broadcast procedure with the name `CountOnMe`, abbreviated in the following to `COM`.

In this section we evaluate the performance of `COM`. Since it is important to validate the protocol in real environments, we choose to perform our experiments on real-life traces. We have run a wide range of tests over traces of different size: Cambridge, a small dataset of 36 nodes already described in the problem evaluation (Section 2), *Reality Mining*, a medium-size dataset, and *Dartmouth*, a large-size one, described here below:

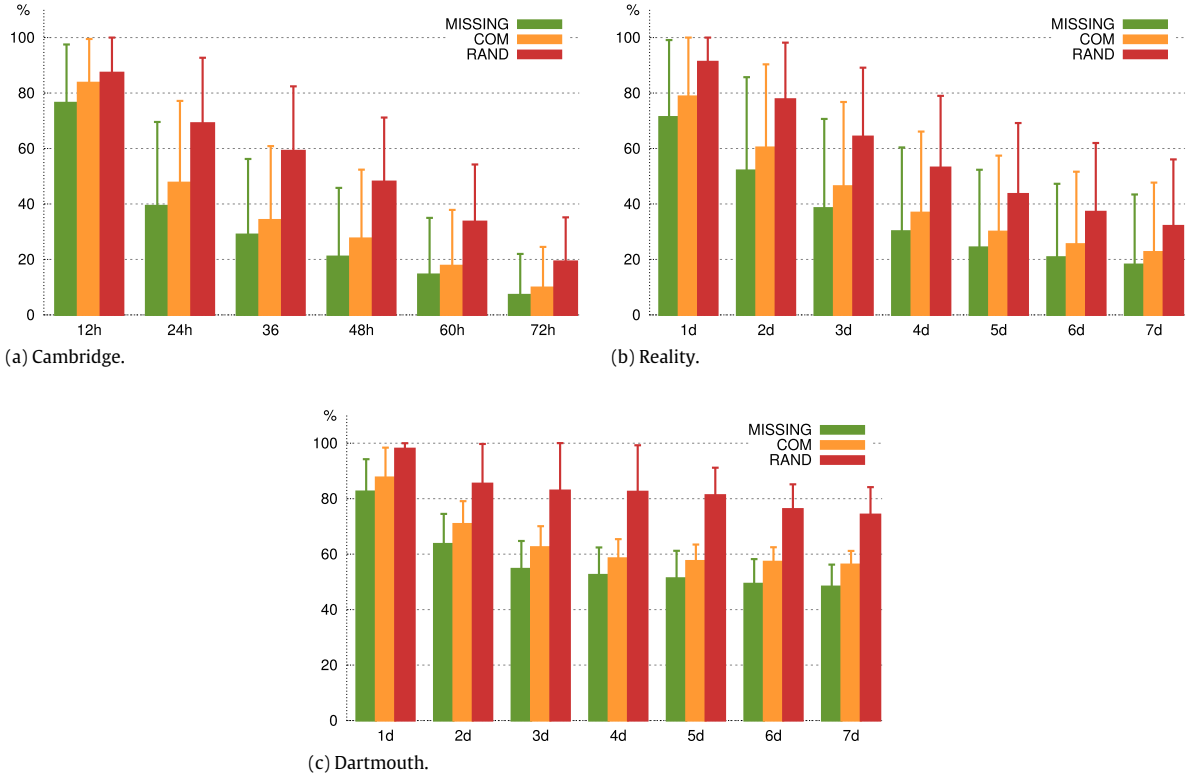
Reality mining dataset. This dataset [28] consists of Bluetooth contacts occurring among smart phones distributed to 96 subjects of the MIT institute. The Bluetooth logs extend across a period of 9 months (Sept. 2004–May 2005). 68 users were colleagues working in the same building on campus (90% grad students, 10% staff), the remaining 26 were incoming students at the MIT Sloan business school. The Reality dataset has portions of time in which only a few contacts have been recorded and some users hardly ever appear throughout the whole trace or only appear in part of it. We isolated data recorded from September 2004 to December 2004, when the activity of users showed to be quite stable. We also excluded those users who failed to appear in at least one-third of the days in the considered timeframe. This left us with 76 users.

Dartmouth dataset. This data was collected at Dartmouth College campus between April 2001 and June 2004. It consists of SMNP logs from over 450 access points installed across the campus [18]. It recorded nearly 6000 different users. In order to make the data suitable for our purpose, we followed a conventional approach [3,23,1]: We consider two mobile users in contact when they are associated to the same access point. We consider activities during a period of eight weeks in which the academic campus life is reasonably consistent, spanning from January 2004 to March 2004. We chose to work with the set of nodes that have a fairly stable activity in time: Those that appear in at least half of the considered days. This results in a set of 1145 nodes.

For all datasets we imposed the selection of only two best friends per users ($k = 2$) and each users has accomplished the sparsification process on its neighborhood at a distance h ranging from two (Cambridge) to five (Darthmouth). These values always generate a Social Skeleton not containing cycles—a tree. If we chose lower values for h, the Social Skeleton could have few long cycles, but often not generating overhead. Finally, the Social Skeleton is built exploiting a training period that lasts the first half of each trace. Then, we use the second half to evaluate the performance of our solution.

6.1. A competitor for COM

The performance of `COM` depends on the number of remote messages that it generates with respect to the actual number of nodes that have not been reached by the broadcast message by the end of the TTL. To the best of our knowledge, this is the first time that a reliable broadcasting is being presented in literature. However, we intend to provide the reader with a meaningful benchmark against which the efficiency of the proposed solution may be assessed. We do so by comparing `COM` with a method that, like `COM`, exploits remote messages towards nodes connected from responsibility links to achieve the 100% of delivery when Epidemic fails to. The two methods, however, differ in the way in which they decide the responsibility links. `COM` exploits the Social Skeleton, as discussed in the previous section. The benchmark method, that we call `Rand`, generates responsibility paths through a random permutation of the set of the users. In this way, each user within the path (besides the extremes) is linked through responsibility relations to strictly two users. The two extremes of the path are linked to only one user, in order to avoid loops that generate unnecessary messages. As in `CoM`, each user detaining a message is requested to inform its neighbors according to the responsibility links. When the message elapses, a user u sends a remote message to the user(s) v that he is responsible of, whenever he cannot tell if v has already received the information opportunistically through a proximity message.



(a) Cambridge.

(b) Reality.

(c) Dartmouth.

Fig. 4. MISSING: Unreached users with `Epidemic`. COM: Remote messages generated by `COM`. RAND: Remote messages generated by `Rand`. All graphs depict the average (per message) and standard deviation of the values.

6.2. COM: Efficiency results

As we already showed by experimental means in Section 2, even epidemic is not able to reach all network nodes within the message TTL in opportunistic scenarios. But, if the mechanism needs to be reliable, these nodes need to be reached as well. An optimal remote scheduling would, therefore, know exactly which nodes are left out right before the message TTL, and send a remote message only to these nodes. However, without a single global entity tracking all interaction among nodes, clearly missing in a distributed scenario like the one considered here, the optimal remote scheduling is impossible to achieve. Nonetheless, it represents a benchmark for a reliable broadcast mechanism: The number of remote messages exceeding the optimal is a measure of the efficiency of any reliable broadcast procedure. In fact, it corresponds to the number of remote messages sent to users that were already informed opportunistically.

In our evaluation, we compare the number of the remote messages scheduled by both `COM` and `Rand` to the optimal one: That of the number of users that are not reached by Epidemic when the TTL elapses. The basal experiment involves the choice of a random source user s and a random starting time t_0. At time t_0 the message is created at s and starts circulating through the network according to the contacts recorded in the selected trace. When the TTL elapses, the required remote messages are sent and counted. Each simulation includes a high number of basal experiment that varies from 1000 to 10,000, according to the trace. The results are then averaged.

Fig. 4 depicts the average and standard deviations of the messages scheduled by `COM` and `Rand` in dependence of the message TTL w.r.t. the optimal remote message scheduling for the three traces considered. Note that Cambridge, the first trace, is considerably shorter than the two other traces. Thus, in this case the TTL is at the order of hours. The first observation is that in all the traces the number of the nodes unreached is considerable, and it increases with the size of the network under examination. This is another proof that remote messages are necessary in order to achieve a reliable broadcast. Second, we observe that `COM` always outperforms `Rand` independently from the network size and TTL. In particular, we notice how the number of remote messages scheduled by `COM` is at most 10% higher (for lower TTLs) than that of the optimal. Consider that, though predictable, user behavior is not always exactly the same. So, it is evident that the Social Skeleton generated observing the behavior during the first part of the trace is reflecting quite well the strong relationships among users. From the other side, the messages that `Rand` schedules are considerably more: They vary from a minimum of 20% (12h TTL on the Cambridge trace) to more than 35% higher than the optimal (2–4 day TTL on the Reality Trace). These results give further evidence of the importance of considering social interactions between users for a 100% reliable broadcast.

6.3. Going beyond optimality with anticipated forwarding

In large or sparse social structures the number of social communities – small groups of people frequenting each-other often and having just a few contacts with other groups – increases. This phenomena makes it more difficult for a message to reach nodes that are (socially) further away from the source. An empirical proof is the considerably larger number of unreached nodes in Dartmouth (1145 nodes) with respect to Cambridge (36) and Reality (76). The same holds for the messages scheduled by `COM`. The reason is that messages remain trapped in well-connected components of the source for very long and unreached nodes of further away components generate lots of remote messages at the end of the TTL. However, in this case we can use the strength of the links within the Social Skeleton to improve its efficiency: Indeed, the weakest links are most likely the ones that cause remote messages. But, at the same time, the weakest links are also the ones that typically connect distinct components. Thus, by

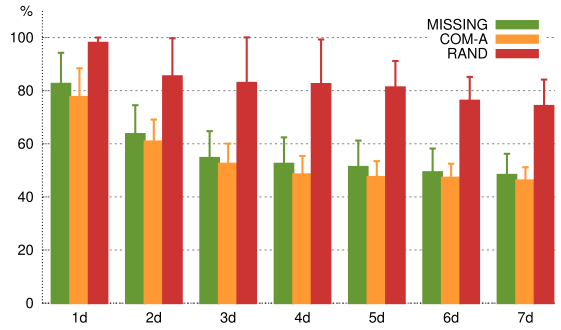


Fig. 5. Results with anticipated forwarding on Dartmouth. MISSING: Unreached users with `Epidemic`. COM-A: Remote messages generated by `COM` with anticipated forwarding. RAND: Remote messages generated by `Rand`. All graphs depict the average (per message) and standard deviation of the values.

anticipating the sending of a remote message over these links, as soon as the message is received, we have a good chance to impede a message to be trapped within a well-connected component. Recall indeed that in Section 5.5 we saw that component leaders are bound together in the Social Skeleton by links of strength zero. Therefore, if anticipated forwarding is correctly applied, leaders would directly push a given message towards another component much earlier before its expiration. Then, messages would start to circulate in it through opportunistic forwarding and reach more nodes by short and cheap links.

That said, we modify our protocol accordingly: Nodes that have links with weight 0 send a remote message to the other counterpart of that link as soon as they receive a new message M. We present the results of `COM` with anticipated forwarding in Fig. 5. The first thing to notice is that there has been an improvement w.r.t. pure `COM` of more than 20% (compare with Fig. 4(c)). Most importantly, anticipated forwarding through weak Skeleton links makes so that the number of remote messages scheduled by `COM` is lower than the number of users not reached by Epidemic—the optimal one. Indeed, choosing to send a few remote messages from one network component to the other much earlier than TTL elapse avoids local trapping. As a result, opportunistic links on the other component are given now the chance to be exploited for message spreading, avoiding otherwise necessary remote communications.

7. Skeleton based routing

For how it is constructed, the Social Skeleton is a sparse graph of the network made of links that represent the strongest relations among nodes. In pure opportunistic networks, these relationships are the most valuable for hop-to-hop routing. By exploiting the links of the Social Skeleton, we can build an opportunistic routing primitive that makes use of the most performing routes to push a message towards destination. In this section we show how to build such a primitive, and we compare it with state-of-the-art routing protocols for PSNs. For a fair comparison, routing does not make use of remote messages, just opportunistic links.

7.1. Skeleton routing: The Protocol

Once the nodes have built the local responsibility graphs and applied the sparsification process described in Section 5, they start exchanging their responsibility links with other peers they meet. During this process, nodes start locally building and incrementally updating routing tables towards other peers in the network. The routing tables contain also the number of hops and the cost to reach a certain peer. The cost is calculated as the sum of the inverse of the strengths of the hops (given by Eq. (1)) the route is made of. So, the stronger the relationship, the lower the cost of a certain hop. Our empirical evaluation shows that this process, exploiting opportunistic contacts only, successfully ends in about 3 days in the Cambridge trace, 7 days in the Reality trace, and 14 days in the Dartmouth trace. (However, to speed up the process, or in cases of particular large networks, our `COM` primitive can always be applied by the first node that completes the routing table to reliably spread it out to the rest of the network.)

The routing primitive, that we call Skeleton Routing, works as follows: A message M for a destination d follows possibly three routes starting from the source s:

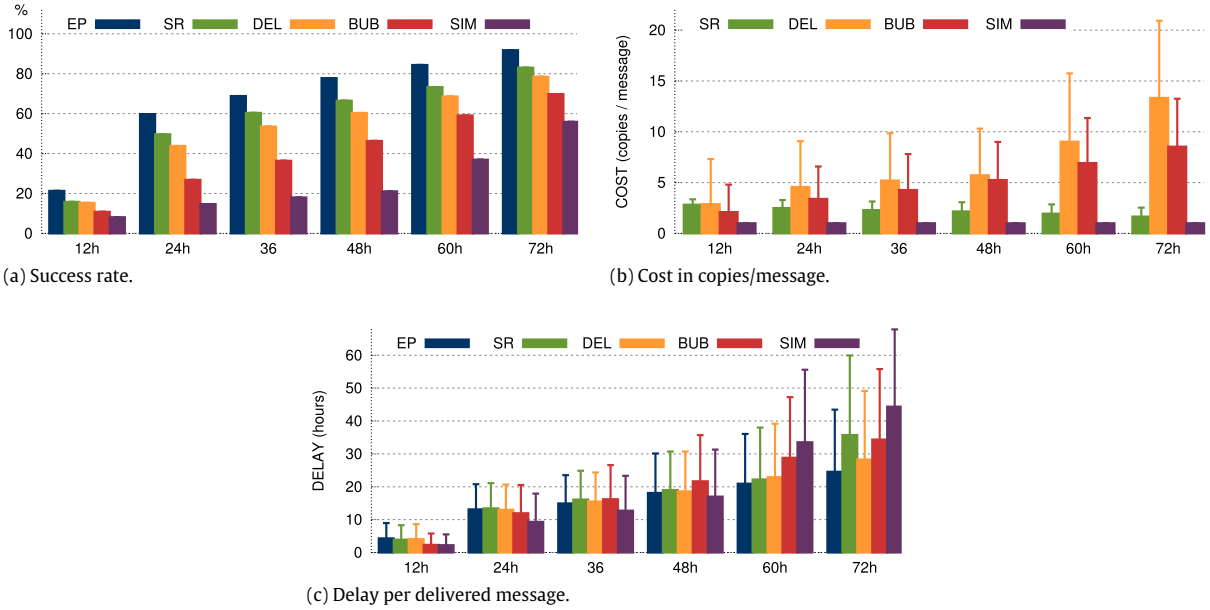
1. through nodes v of the route $s \ldots d$ as defined by the Social Skeleton;
2. through nodes v outside the path $s \ldots d$, whose route-cost to d is lower than that of the current node;
3. through nodes v outside the path $s \ldots d$, whose route-cost to d is higher but the route-length (hops) is lower than that of the current node.

The source s passes a copy of M to the first three nodes met that fulfill each of the rules above. From that moment on, the single-copy travels following the type of the rule specified by the source. Note that, while almost certainly node s encounters a node v that fulfills the first rule (its best-friend in the route towards d), it is not certain if the other two conditions are ever met. Therefore, every message generates at most 3 independently traveling copies in the network.

7.2. Experimental evaluation

To evaluate our routing primitive we compare it to SimBet [5], Bubble [15], and Delegation [8]—three of the most well-known social-based protocols in the literature. SimBet [5] exploits a mix of the node betweenness in its ego-network and its similarity with the destination in order to forward single-copy messages. We have selected it because it outperforms Prophet [22], another well-known protocol, and because it is a limited-copy protocol like ours. Bubble [15] is a multi-copy protocol. It makes use of global and local ranking of nodes, within respectively the global network and the communities they belong to, to push the message towards the destination. To detect the communities we have exploited the k-clique algorithm [27] for the two smaller networks of Cambridge and Reality, and the modularity algorithm [16] for the larger Dartmouth network (k-clique does not support large networks). Delegation [8] is another multi-copy protocol. It is based on a quality rank, and nodes forward a copy of a given message to nodes that have a higher rank than any other node they have seen till then. We have implemented Delegation frequency – the quality is the frequency of the meetings with the destination – and Delegation last seen—the quality is the last time a node has seen the destination. The latter, Delegation last seen, outperforms the former in all traces. Therefore, we opt to present only results with Destination last seen. Protocols are also compared to the very costly benchmark Epidemic.

The results, for the three traces, are presented in Figs. 6, 7, and 8. Note that, unlike in the broadcasting case, the success rate does not present error intervals: A message either arrives to destination or not. In addition, we also present the results on delivery time (delay) and overhead (copies per message generated in the network) with the respective confidence intervals. The first observation that we make is that, in the three traces, SR outperforms all its social-aware protocols in terms of success rate (see Figs. 6(a), 7(a), and 8(a)), despite its limited-copy nature (see Figs. 6(b), 7(b), and 8(b)). The gap increases for longer TTLs. The delay, from the other side, is comparable to that of the other social-aware protocols. However, recall that the delay is measured for the messages that are actually delivered. This means that SR pushes more messages faster to destination than its competitors.

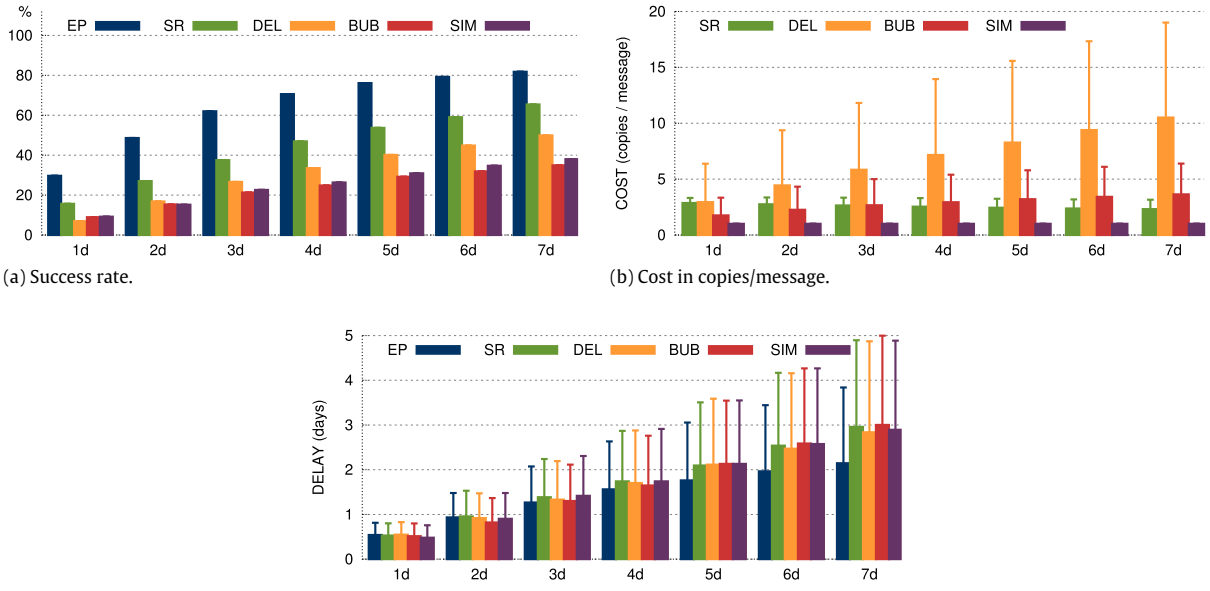


(a) Success rate.

(b) Cost in copies/message.

(c) Delay per delivered message.

Fig. 6. Cambridge trace: Performance of Epidemic (EP), Skeleton Routing (SR), Delegation (DEL), BUBBLE (BUB), and Simbet (SIM). Cost and delay results depict the average and standard deviation of the values. The (very high) cost values for EP are omitted to make possible visual comparison of the other protocols.



(a) Success rate.

(b) Cost in copies/message.

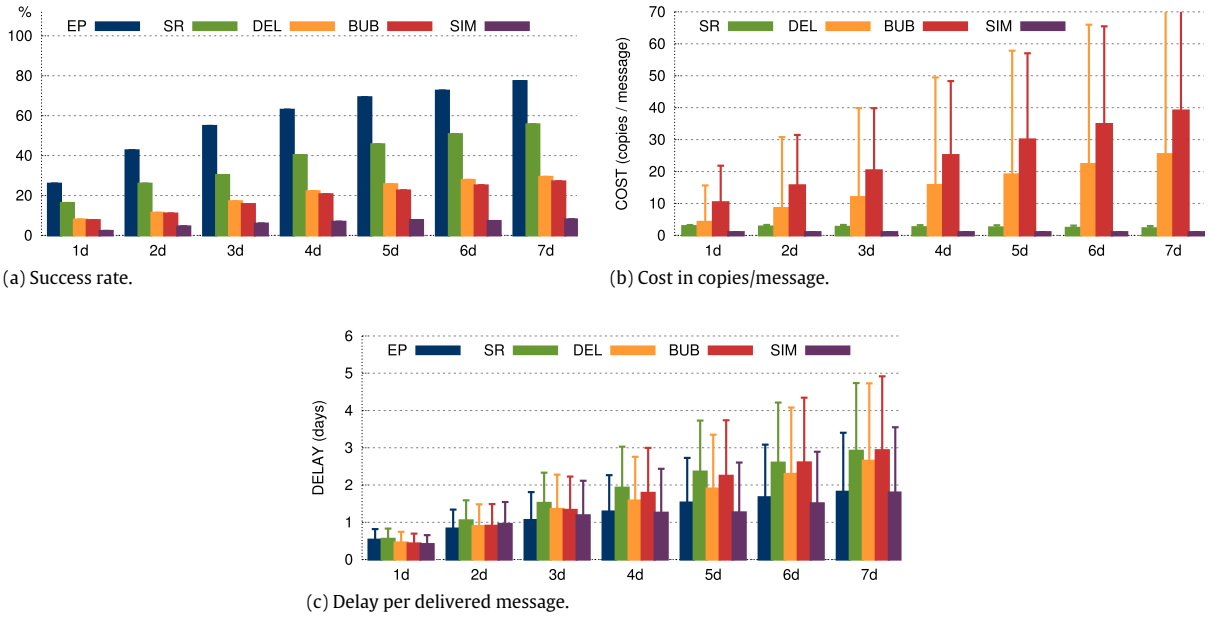
(c) Delay per delivered message.

Fig. 7. Reality trace: Performance of Epidemic (EP), Skeleton Routing (SR), Delegation (DEL), BUBBLE (BUB), and Simbet (SIM). Cost and delay results depict the average and standard deviation of the values. The (very high) cost values for EP are omitted to make possible visual comparison of the other protocols.

It is important to observe that SR performs particularly better than its competitors in large networks. Indeed, in the Dartmouth network made of 1145 nodes, SR delivers up to two times more messages than the two multi-copy competitors Delegation and BUBBLE, and up to 5.5 times more messages than SimBet (see Fig. 8(a)). All at the cost of at most 3 copies per message (Fig. 8(b)). This is another evidence of the Skeleton's capability to capture correctly the relationships among network nodes, and to allow us to build the most performing routes upon them.

8. Related work

Opportunistic mobile ad-hoc networks have attracted the attention of many researchers in the last few years. There is now a large and consistent body of research work in this area. In this context, the idea of using information regarding social ties in networking is not new, it is actually common to a good part of the literature. Much work has been done in the analysis of data collected during real-life experiments, to compute statistical properties of human mobility, and to uncover its structure in sub-communities [3,14,17]. Later on, a good part of the work in the field focused on message-forwarding and to find the best strategy to relay messages in order to route them to a destination as fast as possible (see [22,5,15,8,24], among many others). Excellent recent works have investigated how, although very effective, existing routing protocols suffer from a redundancy problem [10]. Also security problems have been considered; problems like node capture [4] and detection of selfish behavior and other security issues [2,21,25,29,12]



(a) Success rate.

(b) Cost in copies/message.

(c) Delay per delivered message.

Fig. 8. Dartmouth trace: Performance of Epidemic (EP), Skeleton Routing (SR), Delegation (DEL), BUBBLE (BUB), and Simbet (SIM). Cost and delay results depict the average and standard deviation of the values. The (very high) cost values for EP are omitted to make possible the visual comparison of the other protocols.

have been solved by exploiting social relations among the network nodes.

All these works introduce clever ideas, novel notions and elegant ways to exploit the social structure of the network. In the area of routing, which is the closest to our present work, the above contributions take into consideration *pure* opportunistic networks, networks in which the only mean of communication is based on local communication technologies like Bluetooth or WiFi. In these works routing is never reliable, in the sense that it is just impossible to implement a reliable broadcasting service: If a node is disconnected from the network (or just connected to a small disconnected subset of the network) then it will not be able to receive all messages. Besides from that, all mechanisms that rely on global knowledge of structural properties of the network, even though distributively computed, are not possible without a reliable broadcasting of this information to the nodes first. Most importantly, the security mechanisms that require to report on network misbehaviors, node captures, banning of members from the system [4,2,21,25,12], cannot do so without a reliable broadcast mechanism.

Fortunately, every smartphone and many of the other portable devices that are commercially available, are equipped with some kind of remote communication technology like those provided by a cellular network. The drawback is that this technology is way more expensive than using local free Bluetooth communication. This is the reason why short-range opportunistic communication is preferred whenever possible, even to communicate traffic of overloaded cellular networks to partially alleviate them [13,1,31]. The approach is to select a small set of key nodes to which to send through cellular traffic the data. Then, the key nodes disseminate the message in the network through opportunistic links. Nonetheless, the protocols are not distributed (the selection of the initial key nodes is done by the cellular service provider), and broadcasting is not reliable—none of the solutions achieve 100% coverage.

Our work is also related to the area of computing overlays and dominating sets in both static and mobile wireless ad-hoc networks. In [6], for example, the idea is to compute a small, weakly connected dominating set (a skeleton) by using a sparsification process. However, the network is static, and the dominating set has nothing to do with the social interactions of the nodes. In [32], the authors compute a virtual backbone in a mobile ah-hoc network by using two mechanisms: Clustering and adjustable transmission range. Again, this work does not consider a social network, does not consider the possibility of using remote cellular communication, and broadcasting is not reliable. These two papers are good starting points to explore the literature in the area.

To the best of our knowledge, this work is the first that addresses – in a completely distributed way – the problem of reliable broadcasting in a hybrid wireless mobile network of *people*. In addition, our opportunistic routing primitive based on the distributively constructed Social Skeleton, outperforms state of the art social-aware routing protocols for PSNs despite its limited-copy nature.

9. Conclusions, discussion, and future work

In this work we have introduced the notion of Social Skeleton. The Social Skeleton is a compact subset of a social mobile network consisting of socially strong links, that can be used to implement reliable broadcast in a very efficient way. We have presented several experiments with real-life mobility traces that show that our solution guarantees that all the nodes of the network receive broadcast messages even when some of the nodes are disconnected from the social network. In addition, we have shown how Social Skeleton paths can be used to route messages to destination in a pure opportunistic way cheaply, quickly, and with a very low overhead.

The construction of the Social Skeleton depends on a long-enough training period. During this period the nodes "learn" their strongest links in the network from which they deduct the local social graphs. Nonetheless, the requirement of the training period makes the broadcast and routing mechanisms not readily employable in the network. However, this is a limitation of all mechanisms proposed for opportunistic, infrastructure-less scenarios [16,15,22,5,25,1], and often worthy to be payed in order to achieve services similar to those in networked ones but with very limited costs.

The local neighborhood level explored by users in our evaluation was of $h = 3$ and $h = 4$, in dependence of the scale of the trace. However, we believe that for larger networks with sizes of up to hundreds of thousands of nodes, the value of h considered

should be slightly larger. Although we are not able to perform experiments with larger traces, as they are not available, our intuition is that the value of h should be of the order of $O(\log n)$, where n is the number of network nodes.

We believe that our contributions are useful to move the large body of deep and meaningful results on mobile social network to a more practical setting, and are a first step to implement other important primitives of distributed computing on this type of networks. As future work we would like to study the coupling of these mechanisms to already existing protocols that require them, like detection of communities [16,15], alerting nodes about misbehaving members of the network [25], and many more. In particular, it would be interesting to assess the overheads of these protocols in a more complete way, in view of these unexplored costs in their evaluation.

References

[1] M. Barbera, A. Viana, M.D. Amorim, J. Stefa, Data offloading in social mobile networks through VIP delegation, Ad Hoc Netw. 14 (2014) 92–110.
[2] L. Buttyán, L. Dóra, M. Félegyházi, I. Vajda, Barter trade improves message delivery in opportunistic networks, Ad Hoc Netw. 8 (1) (2010) 1–14.
[3] 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 IEEE INFOCOM, 2006.
[4] M. Conti, R.D. Pietro, A. Gabrielli, L. Mancini, A. Mei, The smallville effect: social ties make mobile networks more secure against node capture attack, in: Proc. of ACM MobiWac, 2010.
[5] E.M. Daly, M. Haahr, Social network analysis for routing in disconnected delay-tolerant manets, in: Proc. of ACM MobiHoc, 2007.
[6] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, A. Srinivasan, Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons, in: Proc. of ACM-SIAM SODA, 2003.
[7] D. Easley, J. Kleinberg, Networks, Crowds, and Markets: Reasoning About a Highly Connected World, Cambridge University Press, New York, NY, USA, 2010.
[8] V. Erramilli, M. Crovella, A. Chaintreau, C. Diot, Delegation forwarding, in: Proc. of ACM MobiHoc, 2008.
[9] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proc. of ACM SIGCOMM, 2003.
[10] W. Gao, Q. Li, G. Cao, Forwarding redundancy in opportunistic mobile networks: Investigation and elimination, in: Proc. of IEEE INFOCOM, 2014.
[11] M.C. Gonzalez, C.A. Hidalgo, A.-L. Barabasi, Understanding individual human mobility patterns, Nature 453 (2008) 779–782.
[12] L. Guo, C. Zhang, H. Yue, Y. Fang, A privacy-preserving social-assisted mobile content dissemination scheme in dtns, in: Proc. of IEEE INFOCOM, 2013.
[13] B. Han, P. Hui, V. Kumar, M. Marathe, J. Shao, A. Srinivasan, Mobile data offloading through opportunistic communications and social participation, IEEE Trans. Mob. Comput. 11 (5) (2012) 821–834.
[14] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, C. Diot, Pocket switched networks and human mobility in conference environments, in: Proc. of ACM WDTN, 2005.
[15] P. Hui, J. Crowcroft, E. Yoneki, Bubble rap: social-based forwarding in delay tolerant networks, in: Proc. of ACM MobiHoc, 2008.
[16] P. Hui, E. Yoneki, S.Y. Chan, J. Crowcroft, Distributed community detection in delay tolerant networks, in: Proc. of ACM/IEEE MobiArch, 2007.
[17] T. Karagiannis, J.-Y. L. Boudec, M. Vojnović, Power law and exponential decay of inter contact times between mobile devices, in: Proc. of ACM MobiCom, 2007.
[18] D. Kotz, T. Henderson, I. Abyzov, CRAWDAD data set dartmouth/campus (v. 2009-09-09), http://crawdad.cs.dartmouth.edu/dartmouth/campus.
[19] J. Leguay, A. Lindgren, J. Scott, T. Friedman, J. Crowcroft, Opportunistic content distribution in an urban setting, in: Proc. of CHANTS, 2006.
[20] J. Leguay, A. Lindgren, J. Scott, T. Riedman, J. Crowcroft, P. Hui, CRAWDAD trace upmc/content/imote/cambridge (v. 2006–11–17), Downloaded from http://crawdad.cs.dartmouth.edu/upmc/content/imote/cambridge (November 2006).
[21] Q. Li, S. Zhu, G. Cao, Routing in socially selfish delay tolerant networks, in: Proc. of IEEE INFOCOM, 2010.
[22] A. Lindgren, A. Doria, O. Schelén, Probabilistic routing in intermittently connected networks, SIGMOBILE Mob. Comput. Commun. Rev. 7 (3) (2003) 19–20.
[23] M. McNett, G. Voelker, Access and mobility of wireless pda users, SIGMOBILE Mob. Comput. Commun. Rev. 9 (2) (2005) 40–55.
[24] A. Mei, G. Morabito, P. Santi, J. Stefa, Social-aware stateless routing in pocket switched networks, IEEE Trans. Parallel Distrib. Syst. 26 (1) (2015) 252–261.
[25] A. Mei, J. Stefa, Give2Get: Forwarding in social mobile wireless networks of selfish individuals, IEEE Trans. Dependable Secure Comput. 9 (4) (2012) 569–582.
[26] G. Nemhauser, L. Wolsey, M. Fisher, An analysis of the approximations for maximizing submodular set functions, Math. Program. 14 (1978) 265–294.
[27] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (2005) 814–818.
[28] N.E.A. Pentland, D. Lazer, Inferring social network structure using mobile phone data, in: Proc. of PNAS, 2009.
[29] L. Shi, S. Yu, W. Lou, Y. Hou, Sybilshield: An agent-aided social network-based sybil defense among multiple communities, in: Proc. of IEEE INFOCOM, 2013.
[30] A. Vahdat, D. Becker, Epidemic Routing for Partially Connected ad hoc Networks, Tech. Rep. CS-200006, Duke University, 2000.
[31] X. Wang, M. Chen, Z. Han, D. Wu, T. Kwon, Toss: Traffic offloading by social network service-based opportunistic sharing in mobile social networks, in: Proc. of IEEE INFOCOM, 2014.
[32] J. Wu, F. Dai, Virtual backbone construction in manets using adjustable transmission ranges, IEEE Trans. Mob. Comput. 5 (9) (2006) 1188–1200.

Alessandro Mei is a full professor at the Computer Science Department of Sapienza University of Rome, Italy. He received the laurea degree in Computer Science summa cum laude from the University of Pisa, Italy, in 1994. He continued his studies as a Ph.D. student at the Department of Mathematics of the University of Trento, Italy, and as a visiting scholar at the Department of EE-Systems of the University of Southern California during 1998 and part of 1999. He received the Ph.D. in Mathematics at the University of Trento in 1999. After a postdoctoral position at the University of Trento, in 2001 he joined the faculty of the Computer Science Department at Sapienza University of Rome, Italy. His main research interests include computer system security and parallel, distributed, and networked systems. He was presented with the Best Paper Award of the 16th IEEE International Parallel and Distributed Processing Symposium in 2002, the EESystems Outstanding Research Paper Award of the University of Southern California for 2000, and the Outstanding Paper Award of the Fifth IEEE/ACM International Conference on High Performance Computing in 1998. He is a member of the ACM and the IEEE, a past associate editor of the IEEE Transactions on Computers (20052009), and the general chair of IEEE IPDPS 2009, Rome, Italy. Alessandro Mei was a Marie Curie Fellow (at the CSE Department, University of California San Diego, from Aug. 2010 to Aug. 2011, and at the CS Department, Sapienza University of Rome, from Aug. 2011 to Aug. 2012).

Natascia Piroso got her Bachelor and Master degree in Mathematics from "Roma Tre" University in respectively 2005 and 2007. She got her Ph.D. in Computer Science from Sapienza University of Rome in 2012. Her research interests include algorithms for parallel and distributed systems, network security, fault-tolerant computing, resource management, optimization, computer networks and communication, wireless networks, analysis of social mobile networks and dedicated applications. In spring 2011 she was a graduate visiting scholar at the CS&E department, University of California San Diego. Now she is a Software Developer at Codin Spa, Rome, Italy.

Julinda Stefa is an Assistant Professor at the Computer Science Department of Sapienza University of Rome, Italy. She received the Laurea degree in Computer Science, summa cum laude, and the PhD in Computer Science from Sapienza University of Rome respectively in July 2006 and February 2010. In 2005 she joined Google Zurich for 3 months as an engineering intern. She was a visiting scholar at the CS Dept. of UNC-Chapel Hill, USA, from November 2008 to April 2009, and a Research Intern at Microsoft Research, Cambridge, UK, from January to April in 2011. Her research interests include computer systems and network security, parallel and distributed systems, and analysis and modeling of social mobile wireless networks. She was the recipient of the Working Capital PNI Research Grant and offered by Telecom Italia (30 winners out of 2138). Julinda was the co-winner of the Best Demo Award of IEEE INFOCOM 2013 and IEEE SECON 2013.