Collaborative Service Discovery in Mobile Social Networks

Michele Girolami (Corresponding author)
Istituto di Scienza e Tecnologie dell'Informazione – Consiglio Nazionale delle Ricerche, Via G. Moruzzi 1, 56124 Italy, Pisa, Tel. +39 050 3152040, michele.girolami@isti.cnr.it

Dimitri Belli
Department of Computer Science, University of Pisa, Largo Bruno Pontecorvo 3, 56127 Italy, Pisa, Tel. +39 050 2223179, dimitri.belli@di.unipi.it

Stefano Chessa
Department of Computer Science, University of Pisa, Largo Bruno Pontecorvo 3, 56127 Italy, Pisa, Tel. +39 050 2213122, stefano.chessa@di.unipi.it
Istituto di Scienza e Tecnologie dell'Informazione – Consiglio Nazionale delle Ricerche, Via G. Moruzzi 1, 56124 Italy, Pisa

Collaborative Service Discovery in Mobile Social Networks

Michele Girolami[a], Dimitri Belli[b], Stefano Chessa[a,b]

[a] ISTI-CNR, Via G. Moruzzi 1, 56124 Italy, Pisa, michele.girolami@isti.cnr.it

[b] Department of Computer Science, University of Pisa, Largo Bruno Pontecorvo 3, 56127 Italy, Pisa, ste@di.unipi.it, dimitri.belli@di.unipi.it

Abstract:

Mobile social networking is a recent paradigm arisen from the wide spread of mobile and wearable devices. Based on the short-range communication interfaces of these devices it is possible to establish opportunistic communications among them and build networks independent to the global one. Challenges introduced by this new type of networks are related to the sharing of resources and services and to the exploitation of the communication opportunities among devices. Limit of existing algorithms, that have sought to fill these shortages, is the lack of attention on the main actor of this service-oriented chain, the user. To this purpose, we introduce the COllaborative seRvice DIscovery ALgorithm (*CORDIAL*) that leverages both mobility and sociality of the users. We evaluate the performance of *CORDIAL* combined with different routing protocols for opportunistic networks, and we compare it with a benchmark algorithm (*S-Flood*) based on flooding and another service discovery algorithm designed to leverage mobile social network features, namely, ServIce DiscovEry in Mobile sociAl Networks *(SIDEMAN)*. Our results show that the performance of *CORDIAL* remains stable with the different routing algorithms and that, in function of the query forwarding strategy triggered, *CORDIAL* matches the performance of *S-Flood* in terms of Query Response Time, achieving a better proactivity score with respect *S-Flood* and *SIDEMAN* as well.

Keywords: Social Mobility, Ad-Hoc Networking, Service-oriented Architectures, Delay-tolerant Communications, Opportunistic Routing, Mobility Datasets, Community Detection.

1. Introduction

Mobile social networks (MSN) [1] are a novel paradigm of networks that exploits point-to-point communications among users' devices (usually smartphones or smartwatches) to establish social networks of users that are dependent on mobility and location of the users themselves. Differently than on-line social networks (such as Facebook or Twitter), MSN do not rely on virtual relationships among users, rather they rely on physical relationships. For example, two users may be linked in a MSN because they often meet (because they frequent the same places or use the same bus, etc.) or because they often talk to each other and/or do the same activities together [2,3,4,5]. Under this respect, MSN aim to build cyber-physical social networks in which people are connected both in the virtual and in the physical worlds.

MSN are enabled by the unbridled diffusion and progress of smartphones technologies in the last years. In particular, spurred by technological advances in terms of memory, processing, sensing, and short-range communication capabilities, such devices are now able to interact among themselves and with the physical world without the intermediation of remote cloud services.

In this work, we thus consider a scenario in which people carry their smart devices (from the networking point of view we will refer these devices as nodes) and establish meaningful social relationships with other people. Nodes form a point-to-point network (exploiting their short-range

communication infrastructure) are part of a service-oriented architecture [1] to share services and information. On the other hand, mobility of people affects the communication opportunities among nodes. Indeed, mobility is important in our scenario: it is not strictly functional to networking because people move according to their own specific objectives, but it follows patterns that reflect the social relationships among people. In turn, this gives the underlying network the opportunity to share information and services according to such social relationship patterns. In particular, three main key aspects characterize human mobility: common activities (e.g., going work in the morning and back home in the evening), visiting a limited number of locations (e.g., home, work office, gym etc.), and traveling more frequently along short paths instead of long routes. Furthermore, concerning sociality, longer encounters (and also co-activity) among people usually reflect some common interests [2,3,4,5,6]. Observing the patterns of movements and encounters, it is then possible to identify *communities* of people, for example people that work in the same office or students attending the same classes, who share interests for similar topics and that tend to meet regularly and periodically. A MSN in this context would provide to such communities the capability of sharing the (hardware and software) device's resources of their devices in a service-oriented approach. This problem is commonly known as *service discovery* [7], whose primary goal is to efficiently provide the nodes with the services they need, without relying on any specific infrastructure. Specifically, service discovery builds upon the two main phases of service advertisement and service query which make possible to conciliate together the search for a service with its offer. However, in the context of opportunistic communications given by MSN, where message latencies are usually high and where it is important to keep the overhead low, conventional message diffusion approaches for advertisements and queries fail to optimize both these aspects for two reasons: social aspects of the users are often treated as marginal characteristics, and advertisements and queries require different diffusion strategies. Our approach focuses on communication opportunities among devices through the study of the social aspects of the users, leveraging their ties to detect communities in accordance with the dissemination of advertisements and queries. On the basis of their respective competences, each entity operates according to a specific information dissemination strategy: proactive for advertisements and reactive for queries. Specifically, we propose *CORDIAL* (COllaborative seRvice DIscovery ALgorithm), a novel algorithm for service dissemination and discovery in MSNs. *CORDIAL* is a social and opportunistic algorithm that exploits the human behavior concerning mobility and community membership. Especially, *CORDIAL* implements the query forwarding phase that looks for available services in the network by leveraging on the communities of nodes within the network (such communities are built based on the mobility of the nodes – and then of their users) and by forwarding the query to the nodes that have higher probability of answering. Furthermore, *CORDIAL* implements the advertisement dissemination phase (which communicates to the other nodes the availability of a service in the network) by proactively disseminating service advertisements among devices that might be more interested in accessing the service. In our preliminary work [8], we evaluated the performance of *CORDIAL* with respect to metrics like Proactivity, message delivery rate, Query Response Time and number of answered queries in a simplified simulation scenario and against a social-based flooding strategy and *SIDEMAN* (ServIce DiscovEry in Mobile sociAl Networks) [9,10]. In this work, we provide a detailed description of *CORDIAL* and an extended evaluation of its performance.

The paper is organized as follows. Section 2 describes some needed background material, giving a summarization of the major related works in the field of MSN, introducing the opportunistic routing strategies considered, and reporting the service discovery algorithms in highly distributed and mobile environments. Section 3 gives an overview of *CORDIAL* and describes the three phases that characterize the forwarding strategy of this service discovery algorithm, namely, reactive phase, proactive phase, and message reception. Section 4 introduces the experimental settings describing the metrics, datasets employed, and the behavior of the nodes. It also briefly introduces a strategy based

on the technique of flooding used to assess the performance of *CORDIAL*. Section 5 reports the experimental results, whereas Section 6 ends the paper by presenting conclusions and giving a perspective for future works.

2. Background and Related Works

2.1 Mobile Social Networks

We assume a fully decentralized architecture of a MSN, where all communications among nodes (usually smartphones) in the network happen in an opportunistic way. Specifically, we consider a community of mobile nodes, each representing an individual who moves within a bounded region where there are also other mobile nodes that may be with common purposes or interests. Occasionally, a node establishes a contact with another node by using a short-range communication interface. These nodes constitute a MSN.

In our work, we assume that nodes in a MSN cooperate by offering and using each other's services. A service can be any resource that can be shared by a node, for example a data, the connection to an access network, the capability to store and process data etc. To this purpose, the nodes in the MSN implements service discovery mechanisms that include four protocols of service advertisement, query, selection, and access [1]. In particular, we focus on the first two steps of *advertisement dissemination* and *query forwarding*, since we consider them highly challenging for our scenario. The goal of the advertisement dissemination phase is to allow service providers to advertise a service to nodes encountered along the time. A service advertisement is a compact data structure describes the most key features of a service, i.e. its functional and non-functional characteristics (note that the actual formalism used for the service advertisement is out of the scope of this paper). We assume that every advertisement adv_j is described with the set of interests $SI_j \subseteq I$, where $I = \{y_{1, \ldots, y_m}\}$ is the set of all possible interests. The interests classify the advertisements, queries and nodes according to some pre-existing categories. Some notable example can be sport and games, entertainment, news, forecast, messaging or social services, navigation and so forth. The query forwarding phase allows a node to retrieve available services. During this phase, a node first creates the query $q \in Q$ that it injects in the network in order to find a node carrying an advertisement matching with q. The query is modeled by a message that contains the list of interests describing the required service. To the purpose of query forwarding, each node maintains a service cache *CH*, and it implements a service lookup function, denoted with $\lambda : Q \rightarrow CH$. Given an input query q, function λ returns a set of matching services in *CH* (i.e. services whose advertisements match with the interests of q). The pending queries originated by node n_i are stored in the pending query set (denoted PQ_i) until a service matching with the query is found.

From the networking point of view, we model the MSN at time t as a directed graph $G_t = (V, E_t)$, where $V = \{n_{1, \ldots, n_v}\}$ is the set of nodes and $E_t = \{e_{ij} = (n_i, n_j) : n_i, n_j \in V\}$ is the set of links among the nodes at time t. Links in E_t are directed, indicating one-way communications typical of opportunistic networks. The neighborhood N_i^t of n_i at time t is the set of nodes $n_j \in V$ such that $e_{ij} \in E_t$. In other words, the neighborhood of n_i consists of all nodes that can communicate with n_i at time t.

Since the nodes are carried by their human owners, they have the same mobility of their owners. Specifically, they move, meet with other nodes (i.e. become neighbors) and thus form communities according to the social pattern of their owners. Based on this observation, algorithms and applications designed for MSN exploit the existence of communities to optimize the diffusion of information, the

routing and the access to the available resources in the MSN [1]. For this reason, nodes in the MSN use a community detection algorithm to determine the community to which they belong.

To our purposes, we consider four distributed community detection algorithms, namely *Simple*, *Modularity*, *K-Clique* [11] and *Draft* [12]. The first three algorithms belong to the same family and they assume that each node n_i maintains a *familiar set*, which contains nodes whose cumulative contact duration with n_i exceeds a given threshold. The community of node n_i is thus a superset of the *familiar set* of n_i that also includes nodes identified by the community detection algorithm. In particular, the community detection algorithm analyses the familiar sets of the nodes: if two nodes have similar familiar set they are added to the each other's community. The three algorithms differ in the way they merge the communities: *Simple* merges the communities of two nodes if they overlap over a given threshold; *K-Clique* merges the communities of nodes n_i and n_j if there are at least k–1 nodes in the familiar set of n_j that are in the community of n_i; *Modularity* merges the communities of nodes n_i and n_j if the familiar set of the community members of n_j, which is not included in the familiar set of n_i, belongs to the community of n_i.

Note that in all these three algorithms the community of a node is an increasing set, since there are no mechanisms to evict nodes from a formed community. On the other hand, they can be easily combined with a simple aging mechanism of contacts to this purpose, as presented in [13].

Similarly to the previous ones, also the *Draft* algorithm uses a cumulative contact duration among nodes with a threshold to add nodes to a community. On the other hand, it also includes a decay mechanism to evict nodes from a formed community, which decays the cumulative encounter duration of a factor δ (the decay ratio) among each pair of nodes every t seconds (the frame length parameter).

2.2 Opportunistic Routing Strategies

In this work, we assess the performance of *CORDIAL* combined with four routing protocols for opportunistic networks, namely *Epidemic* [14], *Spray & Wait* [15], *Prophet* [16] and *BubbleRap* [17] that we briefly summarize here.

Epidemic exploits a simple flooding-based strategy for dissemination of messages in the network. Its messages diffusion mechanism works as follows: when two nodes n_i and n_j enter in contact, they exchange their respective *summary vector*, which contains the list of messages they have generated or received so far. Then the node n_i requests to n_j the messages that are in the list of n_j but that are not in its own list, and n_j does the same. To reduce the overhead due to the exchange of the summary vector each message is represented by a unique, network-wide identifier. Furthermore, each message is associated to a hop count that limits the number of times it can be forwarded to other nodes. When the hop count of a message becomes 1 the message can be only delivered to its destination. Finally, the destination can confirm the reception of a message by sending an acknowledgement to the node that originated the message.

Spray & Wait is an extension of *Epidemic* that limits the number of copies of a message in the network. It operates in two phases: spray phase and wait phase. When a node n_j originates a message, it switches in spray phase, in which it makes L copies of the message and it forwards up to L–1 copies to other rely nodes that did not receive the message yet. When a relay node n_j receives a copy of the message, it switches to the wait phase. If during the spray phase n_i encounters the destination of the message, then it concludes the diffusion of the message. Otherwise, it remains with the last copy of the message and switches to the wait phase.

Prophet is another extension of the *Epidemic* algorithm which exploits some assumptions about the mobility of the nodes to increase the probability of message delivery success. In particular, since nodes move according to the mobility of their human owners, they present repetitive mobility patterns

because humans visit various places with different probabilities and frequencies (for example, they tend to visit the place where they live or work frequently and periodically). Based on this observation, *Prophet* introduces the concept of *delivery predictability* $P(n_i, n_j)$ which estimates the probability that node n_i is able to deliver a message to node n_j.

Similarly to *Epidemic*, when two nodes n_i and n_j encounter they exchange their summary vectors. However, in *Prophet* this vector also includes the estimation of the (well-known) delivery predictability values for all the destination nodes of the messages. This information lets a rely node to forward a message to other rely nodes that have a higher chance to deliver the message to its destination.

The delivery predictability between any pair of nodes decays over time with a decay factor, and it increases of a given factor whenever they encounter again. The delivery predictability is transitive: if node n_i meets frequently node n_j, and node n_j meets frequently node n_k, then the delivery predictability between nodes n_i and n_k is also high.

Differently than the other algorithms, *BubbleRap* leverages on the social behavior of the users (and then of the nodes) to deliver messages. In particular, it exploits the centrality of a node in the network and the communities of nodes to take forwarding decisions. *BubbleRap* computes the communities using the *K-Clique* algorithm and it estimates the centrality of a node based on the number and frequency of encounters of the node with other nodes, which is expressed in terms of two ranks: one global and one computed within communities.

When a node n_i originates a message for a destination n_j, it forwards the message to nodes with a higher global rank, and it is forwarded with the same rule by the relay nodes, until it reaches a node in the same community of the destination. From this point on, the message is forwarded to nodes that have higher rank in that community until it reaches its destination. Note that this mechanism can be implemented in a distributed way and it does not require n_i to know the rank of every other node in the network but, rather, to compare its own rank to that of other nodes it encounters.

2.3 Service Discovery Algorithms

Service discovery in highly distributed and mobile environments, similar to that of MSN, have been investigated for over a decade [7,18]. In particular, service discovery protocols addressing mobile ad-hoc networks [19,20], delay-tolerant networks [21], or opportunistic networks [22,23,24] are particularly relevant for our work since these networks have many similarities with MSN. In particular, [19] and [21] offer service discovery and delivery by means of a peer-to-peer model which, however, is not suitable to tolerate the frequent and prolonged disconnections typical of MSNs. Furthermore, they disregard the aspects related to the mobility of users and of their sociality. The approach proposed in [20], which also addresses service discovery in mobile ad hoc networks, exploits a reliable broadcast mechanism and assume low mobility of nodes. The services are described based on their input/output parameters (rather than a service identifier), and their advertisements are spread by means of the proactive exchange of tables that each node stores locally and that contain a mapping between the service and the I/O parameters needed. The query diffusion is based on a flooding mechanism, and, differently than *CORDIAL*, does not exploit any knowledge about sociality and mobility of users.

User sociality and interests are instead considered in [24,25], whose main focus however is in energy efficiency and in privacy, and in [22] that devises a strategy for discovery of resources (which are classified based on the user interests) in a delay-tolerant network. Similar to our approach, the discovery of resources in [22] is community-based, however, if the resource is not found, then the protocol reverts to a flooding-based strategy. Differently from this approach, *CORDIAL* also

implements a proactive strategy for the diffusion of the service advertisements with the goal of proactively store advertisements of interest for the user.

The authors of [23] propose a proxy-based, OSGi-based middleware for service provisioning in opportunistic networks. Its idea is to make a service *proxable* if it can be offered by more than one node in the network. However, this work focuses on how the services are managed by the proxies, where the actual strategy for service discovery can be arbitrary. This idea is further refined by the same authors in [26], which also introduces an opportunistic location-aware algorithm for service discovery and invocation. The node location is exploited during the forwarding message to select the rely nodes that are closer to the destination during the service advertisement, in order to limit the advertisements' propagation only to regions close to the provider itself, and to forward the service requests during the service invocation only through nodes close to the provider. Due to the heterogeneity of the nodes, *CORDIAL* does not rely on the assumption that all nodes are aware of their position. Indeed, such information could be not available on resource-constrained devices, thus making the location-based algorithms not always effective.

In our previous works [9,10] we proposed *SIDEMAN*, a service discovery protocol for MSN that, similarly to *CORDIAL*, leverage on human behavior, in particular on membership to communities and mobility, to build opportunistic and social-aware mechanisms of service discovery. In particular, both algorithms diffuse service queries and advertisements in the network by exploiting the similarity of the users' interests and the membership to common communities, aiming at limiting this diffusion to nodes that are not interested in. However, they differ in the forwarding strategy (specifically in the selection of forwarding nodes): *SIDEMAN* forwards query and advertisements to members in the current community of a node that have interests matching with the query, while *CORDIAL* chooses members in the community with highest social centrality metric. Another significant difference in the two algorithms is in the algorithm for selection of advertisements and queries to store in cache. In particular, *SIDEMAN* selects advertisements and queries whose interests match those of the community members, while *CORDIAL* selects advertisements and queries with highest similarity index.

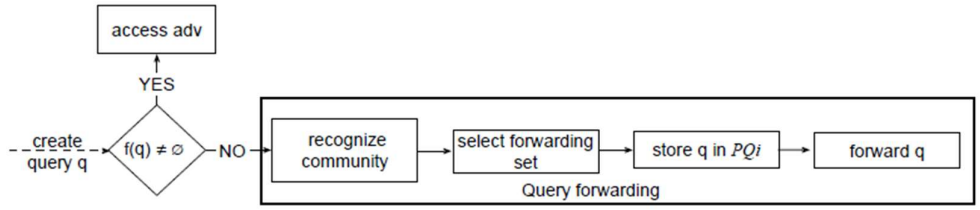
3. Overview of *CORDIAL*

CORDIAL is a service discovery algorithm executed by each node in the MSN. Each node in the MSN plays three roles: provider of services, broker and user of the services. When a node decides to offer a service to the other nodes in the network, it forwards the corresponding advertisement to other nodes in its community that may be interested. In turn, these nodes cooperate to the dissemination of the advertisement by forwarding it to other nodes in their respective communities. At the same time, if a node is looking for a service, it disseminates the corresponding query to all nodes in its community, which in turn, forward the query to their communities. In this respect, each node also acts as a broker: if it happens that it receives a query and an advertisement that match, it informs the node that raised the query.

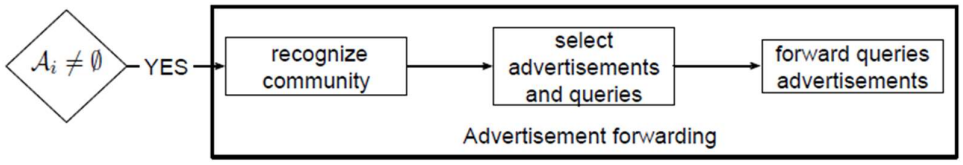
The two phases of query dissemination and advertisement dissemination are independent, and they are executed in two different ways by *CORDIAL*, namely the *reactive phase* (for query dissemination) and the *proactive phase* (for advertisement dissemination). The reason for using a reactive approach for query dissemination lies in the need of providing a fast response to the query (i.e. to reduce the Query Response Time), while advertisements, that do not need to be answered, are diffused in a proactive way. Another important feature of *CORDIAL* is that, to reduce the communication overhead, each node restricts the forwarding of queries and advertisements to other nodes that are in its own

community. To this purpose each node independently computes its own community of nodes based on parameters like the inter-contact time and the last time of encounter with other nodes.

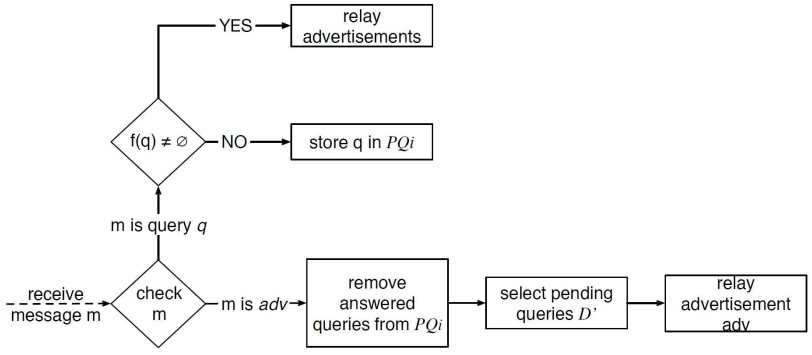
The schemas of the reactive phase and proactive phase of *CORDIAL* are shown in Figure 1.a and Figure 1.b respectively. When a node n_i requires a service, it executes the reactive phase (Figure 1.a) that creates the query q and executes the function $f(q)$ to look for any advertisement matching with the query that is stored in the local service cache CH_i of n_i. If there is at least one advertisement in CH_i that matches q, then n_i selects among these the one that best matches with the query and it accesses the service. Otherwise, n_i starts the query forwarding. The query forwarding is limited to the nodes in the same community of n_i to limit the network traffic while keeping high chance of finding the service, in fact, as discussed in [27], nodes with similar interests tend to meet more frequently than nodes with non-overlapping interests. To this purpose, *CORDIAL* exploits a community detection algorithm, which is executed in background, and that guarantees an updated community set of n_i when needed. This set includes nodes that have significant social ties with n_i.



(a) Reactive phase



(b) Proactive phase



(c) Message Reception

Figure 1. The three phases of the *CORDIAL* algorithm.

To the purpose of query forwarding, n_i builds the *query forwarding set* (FS_i) that is a subset of the community of n_i. The nodes in the community of n_i are selected with the goal of reducing both the Query Response Time and the number of copies of the same query. In particular, FS_i includes nodes with interests matching the query q or nodes that can forward q to friends whose interests match with

q. Once the node carrying the query finds a matching advertisement, it sends back the advertisement to n_i.

The proactive phase of *CORDIAL* is periodic. Its goal is to disseminate advertisements and queries carried by n_i to nodes potentially interested in such messages (Figure 1.b). The proactive phase uses different strategies to forward advertisements and queries. For the dissemination of queries stored in the pending query set PQ_i which still remain to be answered (and that can either be generated by n_i itself or by another node), *CORDIAL* adopts the same strategy used in the reactive phase. In particular, it builds the forwarding set and delivers q to any node in it (whenever it has the chance). The dissemination of advertisements instead is used to proactively disseminate any advertisement adv_j in its cache CH_i to other nodes that might be interested. To this purpose, n_i selects as relay nodes its neighbors that in the past did not already receive adv_j and whose interests match with those of adv_j. As a consequence of the proactive phase, it may occur that n_i receives an advertisement adv_j not directed to itself but to another node. In this case, n_i stores adv_j in a local cache, called FC_i (it is the forwarding cache), and it acts as relay node for the advertisements.

The reception of messages is described in Figure 1.c. Node n_i can receive asynchronously a message from another node. If the message received is a query q, then n_i executes the function $f(q)$ in order to check if it carries an advertisement matching with q. If this is the case, n_i acts as relay node in order to forward the advertisements matching with q toward its final destination (relay advertisements). Otherwise, n_i stores the query in the pending query set PQ_i waiting for a matching advertisement. If the message m received is the advertisement adv, n_i first removes all the pending queries now answered from adv. Then, n_i selects the queries received from other nodes that are now answered from adv, namely the set D'. After this step, n_i checks if the requester of the queries in D' is in contact with ni, if this is the case n_i delivers the advertisement to the final destination otherwise n_i selects one of the nodes to which it is currently in contact with and that can quickly deliver the advertisement to the final destination.

Data structure	Description
Forwarding Cache FC_i	contains advertisements of services (either generated by the node n_i or received from other nodes) that still have to be forwarded by n_i to their destination.
Pending Query PQ_i:	contains the queries received from other nodes and the queries generated by n_i for which that n_i did not find a matching advertisement yet.
Service Cache CH_i:	contains the service advertisements received from other nodes that are of interest for n_i.
$E_i(j)$	the (sorted) list of past encounters between n_i and node n_j,
$ict_i(j)$	the inter-contact times that contains the average time between two successive contacts of node n_i with node n_j,
$lt_i(j)$	the last time of encounters, which contains the last time n_i encountered the node n_j

Table 1. Main data structures managed by *CORDIAL.*

3.1 Forwarding Algorithm of *CORDIAL*

The MSN is modeled as a temporal graph G_t that evolves over time due to mobility of nodes. Note that the temporal graph is not built explicitly by the nodes, but, rather, each node keeps a few data structures that implicitly represent its own view of the graph. From the point of view of the node n_i, that cannot observe the entire network but only its neighborhood, the graph G_t changes when its neighbor set changes (we call this event a local change). Hence, in each of such local change events, n_i may acquire new neighbors (which means that it starts some contacts with new nodes) and it may lose some other neighbors (which means that the contacts with those neighbors are terminated). Since

the information about contacts and their duration is necessary to run the community detection algorithms, each node records the history of contacts. In particular, node n_i maintains three data structures (refer to Table 1):

- $E_i(j)$: the (sorted) list of past encounters between nodes n_i and n_j, for each $n_j \in V$; encounter k is stored in a pair $\langle t'_k, t''_k \rangle$, where t'_k and t''_k are the start and end times of a single encounter between n_i and n_j, respectively.
- $ict_i(j)$: the inter-contact times that contains the average time between two successive contacts of node n_i with node n_j, for each $n_j \in V$;
- $lt_i(j)$: the last time of encounters, which contains the last time n_i encountered the node n_j, for each $n_j \in V$;

Note that these three data structures are local to each node and that they represent the only explicit information that each node keeps about the temporal graph G_t. The inter-contact time $ict_i(j)$ is defined as:

$$ict_i(j) = \begin{cases} \infty & if\ E_i(j) = \emptyset; \\ E[\{t'_{k+1} - t''_k \ \ \forall 1 \le k < |E_i(j)|\}] & if E_i(j) \neq \emptyset; \end{cases}$$

and $lt_i(j)$ is defined as:

$$lt_i(j) = \begin{cases} \infty & if\ E_i(j) = \emptyset; \\ \max\limits_{\langle t'_k, t''_k \rangle \in E_i(j)} t''_k & if E_i(j) \neq \emptyset; \end{cases}$$

Each node updates the contact history in a seamless way, any time a local change event occurs. Since contacts lost interest after a certain period, elements in sets $E_i(j)$ are evicted when their age exceed a given age. Furthermore, each node n_i maintains the three caches forwarding cache FC_i, pending query PQ_i, and service cache CH_i to manage queries and advertisements either generated by the node itself or received from other nodes.

The reactive phase of *CORDIAL* is shown in Algorithm 1. Node n_i first generates the query q containing the set of interests of the needed services. If the service cache CH_i already contains some advertisements matching with q, i.e. $f(q) \neq \emptyset$ (line 2 in Algorithm 1), then n_i can select and access the service advertisement. Otherwise, n_i executes the community detection algorithm *DetectCommunity*.

Algorithm 1 Reactive phase

```
1: q = generateQuery(SI_i)
2: if f (q) ≠ ∅ then
3:     select and access one of the services selected by λ (q)
4: else
5:     C = DetectCommunity(N_i^t , CT^i, τ )
6:     Ĉ = FS(C, q)
7:     forward q to top k nodes in Ĉ
```

Note that *CORDIAL* does not constraint the use of a specific community detection algorithm, instead, it can be configured with any existing distributed community detection algorithm [12]. Given the community C_i (line 5 Algorithm 1), n_i builds the query forwarding set FS_i and assigns to every node $n_j \in C_i \cap FS_i$ a score ρ_j, which provides an indication of the capability of n_j in answering to q. The score ρ_j assigned to n_j is given by:

$$\rho_j = \frac{s(q, I_j)}{1 + ict(i,j)} + \sum_{\substack{w \in C_j - C_i \\ ict(i,w) > ict(j,w)}} \frac{s(q, I_w)}{1 + ict(j,w)} \qquad (1)$$

In particular, the first term of Eq. 1 is the ratio between the similarity of the interest's query and the interests of n_j with respect to the inter-contact time between n_i and n_j. In the second term of Eq. 1, we consider the set difference between the communities C_j and C_i and we apply the constraint that the inter-contact time between $w \in C_j - C_i$ and n_i is higher than the inter-contact time between every $w \in C_j - C_i$ and n_j. In this way, the second term of Eq. 1 selects those nodes that belong to C_j only and such that n_j visited them more frequently with respect to n_i. The second term is the ratio between the similarity of the interest's query and the interests of w with respect to the inter-contact time between n_j and w. This last part measures the capability of node n_j to enter in contact with nodes that might answer the query q. The node n_i then selects the top k nodes with the highest score ρ.

The proactive phase of *CORDIAL* is described with Algorithm 2. The proactive phase manages the forwarding of queries and advertisements to nodes potentially interested in such messages. The first part of Algorithm 2 exchanges the advertisements stored in the service cache CH_i (lines 1– 3).

Algorithm 2 Proactive phase

```
1:  for all adv ∈ CH_i do
2:          V_t = { n_j ∈ N_i^t | s(I_j , adv) > τ ∧ adv ∉ CH_j }
3:          Forward adv to V_t
4:  for all adv ∈ ForwardingCache_i do
5:          if final destination of adv is in contact then
6:                  Forward adv to final destination
7:          else
8:          n_k = TemporalForwarder(adv)
9:          Forward adv to n_k
10: for all q ∈ PQ_i do
11: C = DetectCommunity(N_i^t , CT^i, τ )
12: Ĉ = FS(C, q)
13: forward q to top k nodes in Ĉ
```

For every *adv* in CH_i, n_i builds the set V_t composed by the nodes in the neighborhood N_i^t of n_i at time t whose interests match with *adv* and that did not already receive *adv*. Once V_t has been computed, node n_i multicasts *adv* to V_t. After this first round of message exchanges, the node n_i checks if some of the advertisements stored in the forwarding cache FC_i can be delivered to the final destination (lines 4–9). If *adv* is in FC_i, then it is directed to a node n_j, if n_j is currently in contact with n_i then n_i directly delivers the advertisement to n_j. Otherwise, n_i executes the *TemporalForwarder* function, which selects the relay node $n_w \in N_i^t$ with the lowest remaining inter-contact time between n_w and the final destination of *adv*, namely the node n_k. To this purpose, the remaining inter-contact time (*r-ict*) between nodes n_w and n_k is defined as $r\text{-}ict = |ict(w,k) - [t - lt(w,k)]|$ and it measures the difference between the average inter-contact time between n_w and n_k and the time elapsed since the last encounter between n_w and n_k. The smaller the *r-ict*, the more likely n_w will meet n_k again, and n_w will deliver *adv* to the final destination. If the *TemporalForwarder* function finds $n_w \neq n_i$, then n_i forwards the *adv* to n_w, otherwise n_i stores *adv* in its forwarding cache FC_i. The last part of Algorithm 2 exchanges the pending queries carried by n_i (line 10–13). For every $q \in PQ_i$, node n_i applies the same mechanism described in Algorithm 1 lines 5–7.

Algorithm 3 distinguishes between the reception of queries from the reception of advertisements. If n_i receives a query q (lines 1–10), then it executes the function *f(q)* to find all the advertisements in the service cache CH_i matching with q. In this case n_i checks if the node requesting the query, namely r, is in contact. The node r is the node waiting for an answer for the query q. If r is in contact, then n_i forwards the set of matching advertisements *f(q)* to r, otherwise n_i selects the nodes n_k with the lowest *r-ict* from r. If $f(q) = \emptyset$ then n_i stores the query in its pending query set. This last case implements the collaborative strategy adopted by *CORDIAL*: nodes that cannot answer to a query take care of finding an answer to it. If n_i receives an advertisement *adv* (lines 11–28) then n_i checks all pending queries q

$\in PQ_i$ that match with *adv* (i.e. such that $s(q, adv)$ is greater than threshold τ; such set is named D (lines 13). Set D is removed from PQ_i, because it contains queries that are now answered.

Moreover, n_i checks all pending queries carried on behalf of other nodes (queries whose requester is not n_i) that are answered with *adv*, such set is named D' (lines 15). For every query q in D', if the requester r of q is in contact with n_i, then n_i forwards *adv* to r; in this way r will receive an answer to at least one of its pending queries. If r is not in contact, then n_i selects the node n_k with the lowest *r-ict* and it forwards *adv* to n_k. After this round of checks, the node n_i checks which is the final destination of *adv*. If the final destination is not n_i, then n_i received an advertisement addressed to another node. In this case, n_i has been selected as relay node. Node n_i forwards *adv* to its final destination (if it is in contact with), otherwise n_i forwards *adv* to the node with the lowest *r-ict*. The last operation in Algorithm 3 is to store the advertisement *adv* in CH_i only if the interests of m match with the interests of n_i.

Algorithm 3 On messages reception

```
1: On reception of a query q:
2:     r is the requester of q
3:     if f(q) ≠ ∅ then
4:         if r is in contact then
5:             Forward f(q) to r
6:         else
7:             n_k = TemporalForwarder(r)
8:             Forward f(q) to n_k
9:     else
10:        store q in PQ_i
11: On reception of advertisement adv:
12:    let dst be the final destination of adv
13:    D = { q ∈ PQ_i | requester(q) = n_i ∧ s(q, adv) > τ}
14:    Remove D from PQ_i
15:    D' = { q ∈ PQ_i | requester (q) ≠ n_i ∧ s(q,adv) > τ}
16:    for all q ∈ D' do
17:        r requester of q
18:        if r is in contact then Forward adv to r
19:        else
20:                n_k = TemporalForwarder(r)
21:                Forward adv to n_k
22:    if dst ≠n_i then
23:        if dst is in contact then
24:                Forward adv to dst
25:        else
26:                n_k = TemporalForwarder(dst)
27:                Forward adv to n_k
28:    if s(adv, I_i > τ ) then add adv to Cache CH_i
```

4. Experimental Settings

4.1 Mobility Datasets

The performance of *CORDIAL* is evaluated through simulations using two datasets made up by co-location traces obtained studying two different sets of individuals in the physical world. While the first set acts in a bounded environment, the second, on the contrary, acts in an external one. The datasets are respectively Cambridge [28] and MDC Nokia[1] [29,30].

The Cambridge dataset is the result of the work of Scott et al. [28]. The experiment was carried out for 12 days and involved 36 individuals; everyone carrying an iMote device (from Intel). The devices, which detected other devices within a radius of 30 meters, were calibrated to scan (i.e. to detect contacts) every 120 seconds. Contacts were stored in a flash memory with a structure showing MAC address, start time contact, and end time contact. MDC Nokia dataset was collected by Lausanne's Nokia Research Centre from 2009 to 2011. The experiment involved 185 volunteers who were provided with a Nokia N95 smartphone. Data were collected in a fully transparent way before the participant's eyes. Scans were carried out every 600 seconds along with the readings of GPS, other sensor data, and parameters of the smartphone. The data collected, locally stored, were automatically sent to the server each time a device hooked up to a WLAN. These include data pertaining to social interactions (calls, messages, and scan results); data pertaining to localizations (GPS and WLAN access); data pertaining to multimedia content and their use; data pertaining to users' behavior (what kind of applications are installed on their devices, their frequency use, the activities monitored by the accelerometer, etc.).

Co-location traces are fragments of information, which are detected through short-range communication interfaces, describing contacts taking place among people. We got the co-location traces from either contact or localization traces of the two datasets, using them in the simulation of the service discovery algorithms. We initially analyzed the co-location traces based on social and temporal metrics to extract some parameters regarding the users' behavior.

To analyze social characteristics of the co-location traces we used *the average cardinality of the recognized communities*. Thanks to this measure, it is possible to notice how the mobility in two different areas and with two different samples affects the tendency of the nodes to interact with each other. Concerning the temporal characteristics of co-location traces, we analyzed the number of contacts between pairs of nodes, the inter-contact time, and the contact duration metrics. Specifically, the number of *contacts between pairs of users* is an indicator of the mobility pattern since the amount of contacts the users realize over time highlights their social activity. The *inter-contact time* measures the time elapsed between two subsequent contacts of nodes. This metric indicates the regularity of the contacts, and, consequently, it is also an indicator, based on the mobility of nodes, of the frequency of encounters. The *contact duration metric* is a measure of the average distribution of the contacts duration among nodes. Since long contacts mean deeper social relationships, the contact duration also indicates the familiarity degree among nodes. A more detailed explanation of the temporal metrics can be found in [31].

4.2 Simulation

For our purposes, we used the ONE (Opportunistic Network Environment) simulator [32], which is developed at Aalto University to study communications among mobile nodes in opportunistic scenarios. The ONE is designed with a plug & play mechanism, which simplifies the addition of the

[1] Portions of the research in this paper used the MDC Database made available by Idiap Research Institute, Switzerland and owned by Nokia.

needed components to run a new simulation scenario. It is Java-based, and its strength concerns routing and application protocols which can be evaluated upon customizable settings by the users.

The ONE can define the model of mobility using either co-location traces of external datasets or a generator of simulated movements (Map-Based Model, Random Waypoint). It models external events as generation of messages and simulates their transmission and receipt by means of various routing strategies.

4.2.1 Nodes behavior

Nodes (i.e. devices) in our service-oriented MSN operate the main phases of service discovery: advertisements generation, query generation, and spread of services to other nodes. These phases are carried out by exploiting the opportunity of communications among nodes that, in turn, depend on the social relationships, habits and interests of the nodes themselves.

The advertisement generation rate is relative to the services that are assigned and stored by nodes within their own service cache without the help of any service discovery algorithm. A constant number of nodes, called *service providers*, is established at the early stage of the trial. These nodes are chosen in a completely random fashion, and each of them generates a constant number of advertisements. About our simulation, the number of *service providers* is set to 20. Each service provider generates 5 advertisements and stores them in its cache. At every advertisement is assigned one interest, which, in turn, is selected by permuting the set of interests associated to the node and choosing the first one.

The query generation rate is modelled as a Poisson process that generates an average of λ queries in a time period t. Specifically, each t minutes are generated λ queries to be assigned to λ nodes randomly selected from the set of nodes. In our simulations, we set $\lambda = 2$ for all the experiments.

Advertisements and queries diffusion strategy leverages the relations among nodes with similar interests. However, all datasets used in this simulation do not provide information about interests of the nodes, and for this reason it was necessary to model them. In particular, we assumed that the set of interests is fixed at the beginning of each execution of the simulation, and each node is associated with a constant number of interests. The arrangement of the interest depends on Zipf's distribution, which is associated to a *skew* parameter. If the skew parameter is equal to one, the distribution reproduces the human behavior under the aspect that many people have few common interests and very few people have lots of common interests. This implies that there will be a higher gap between nodes with many common interests and nodes with few common interests. Otherwise, if the skew parameter is near to 0, it means that the interests all have the same popularity. In our simulation, the total number of interests is 100 and at each node is initially assigned 10 interests randomly chosen, whereas one single interest is randomly assigned to each query or advertisement.

Each node exploits its own mobility (as it results from the mobility traces of the datasets used in the simulations) and the mobility of other nodes for spreading messages. Since there cannot be direct paths between the sender and the recipient, each node cooperates forwarding messages by using the store-carry and forward model.

4.2.2 Simulation parameters and evaluation metrics

We perform two kinds of simulations. One simulation conducted over a brief time period of 144 hours aimed at analyzing in detail the performance of *CORDIAL*. The first simulation (presented in Section 5.1) concerned the use of a large number of traces to achieve a 97.5% confidence limits and a 0.025%

probability of error for all the parameters under study. The second simulation (presented in Section 5.2), is aimed at analyzing the behavior of *CORDIAL, S-Flood* and of another service discovery algorithm called *SIDEMAN* [10] over a long time period of 12 days in a specific sample trace of the datasets.

All the parameters of the simulations and their settings are briefly summarized below.

The *data transmission rate* of communications is set to 2Mbps, whereas the transmission radius is set to 10 meters (which is compatible with the Bluetooth standard). The *number of nodes* is fixed to 36 for Cambridge, and 185 for MDC Nokia. The *similarity index* is set to one (i.e. $\tau = 1$). The value of the *familiar threshold* T_{th} is set to 700, whereas the k value is equal to 5. The parameters of the routing algorithms are those of default. Specifically, the TTL of *BubbleRap* is set to 24 hours; the maximum number of copies that can be spread into the network by *Spray & Wait* is set to 10, and *Prophet* has the initialization constant P_{init} set to 0.5, the ageing constant γ of the delivery predictability set to 0.999885791, and the transitive constant β set to 0.9.

In order to evaluate the performance of our service discovery algorithm, we evaluate the following metrics:

- *Accuracy* is the ratio between the number of useful advertisements and the total number of advertisements. Both measures are stored within the cache of a node. Formally, it is given by $\frac{|A'|}{|A|}$. Such metric shows the efficacy of a service discovery algorithms in spread of services of interest instead of virtually useless services. Its value falls within the range [0,1], where 0 represents the worst-case scenario and 1 the better one.
- *Proactivity* measures how efficient is a service discovery algorithm in reducing the number of queries to obtain an advertisement. Proactivity is the result of the ratio between the number of time the node finds an advertisement as useful service (already stored in the cache) and the number of time in which that node generates a query to ask for that service. Formally, it is given by $\frac{|Q_{CH}|}{|Q|}$. Its value falls within the range [0,1], where 0 represents the worst-case scenario and 1 represents the better one.
- *Query Response Time* measures the average of time elapsed between the generation of a query and the reception of a matching advertisement. For a given query $q \in Q_A$ generated at time t', the response time for q is defined as $t'' - t'$; where t'' is the time in which we receive an advertisement corresponding to q (with $t'' > t'$). This metric measures the efficacy of the algorithm to provide a service for a node as quickly as possible.
- *Service Cache and Forwarding Cache* inform about the memory space that nodes require to perform a service discovery algorithm. They respectively measure the average number of advertisements that nodes retain in their memory over time, and the average number of advertisements that nodes receive but that are not addressed to them. The latter are stored in a special space of memory labelled forwarding cache.

4.2.3 Flooding-based service discovery

In addition to *SIDEMAN, CORDIAL* has been compared with a strategy based on the technique of flooding to assess its performance. The reason for this choice is that a flooding-based strategy is really aggressive in terms of delivery and latency (and thus it is a good benchmark from our purposes),

while it is not social-aware, which gives us the possibility to assess the social-aware diffusion strategy of *CORDIAL*. This technique, henceforth named *S-Flood* [33], exploits the mobility of the nodes to implement the phases of research and publication of services. Differently from *CORDIAL* and *SIDEMAN*, *S-Flood* does not exploit the social characteristics of the nodes either for advertising its own services or to research services needed.

Particularly, *S-Flood* leverages the mobility of the nodes for allowing the spreading of both queries and advertisements to nodes with which it comes into contact. Consequently, no community detection algorithm is necessary.

Another significant difference of *S-Flood,* with respect the other service discovery algorithms, is the absence of cooperation among nodes. As a result, two structures are affected by variations: the forwarding cache, which is not foreseen, and the pending query, which in *S-Flood* only contains the queries generated by node n_i that do not have received any answer.

The reactive and proactive phases of *S-Flood* are similar to that of *CORDIAL* and *SIDEMAN*, with few differences. In particular, in *S-Flood* both the queries and the advertisements are forwarded to the whole node's neighborhood rather than to the neighbors in the relay nodes' community.

5. Experimental Results

We describe the experimental results of *CORDIAL* and *S-Flood* based on the previously introduced metrics and on the two datasets, namely Cambridge and MDC Nokia. It is worth to recall that the *Accuracy*, *Proactivity*, *Query Response Time*, *Service Cache*, and *Forwarding Cache* metrics have been analyzed by combining *CORDIAL* and *S-Flood* with 4 different routing algorithms.

5.1 Performance of Service Discovery Algorithms

Accuracy

As introduced in Section 3, *CORDIAL* spreads the advertisements to nodes only if they are interested in accessing such advertisements (the proactive phase). Therefore, the service cache of nodes running the *CORDIAL* algorithm contains only the advertisements of interest for such node. As a result, the Accuracy metric for *CORDIAL* is always optimal (Accuracy = 1) during the whole duration of the simulations.

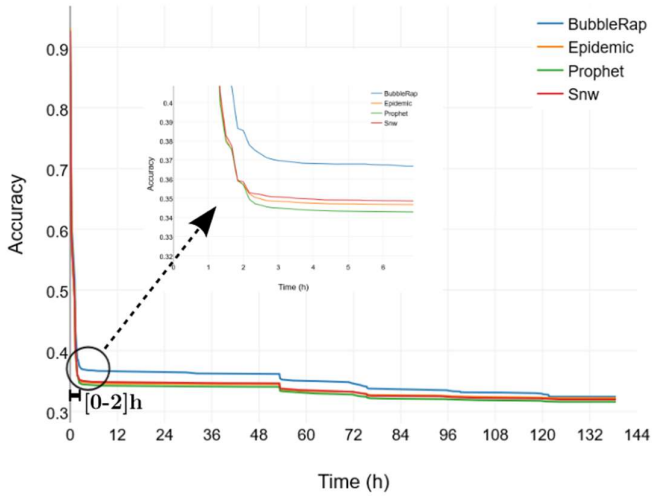


Figure 2. Accuracy of *S-Flood* with Cambridge.

Differently, *S-Flood* spreads the advertisements to all the nodes that are in contact to each other without any filter, as a result the accuracy of *S-Flood* tends to decrease with the time. Figures 2 and 3 show the accuracy of *S-Flood* respectively for the Cambridge and MDC Nokia datasets. In particular, Figure 2 shows a rapid decrease of the accuracy during the interval [0,2]*h*, during which the accuracy quickly decreases to 0.36. After this threshold, the accuracy reduces progressively (the inset graph in Figure 2 shows such trend). We consider that such decrease is due to the high number of advertisements exchanged from the nodes during the first two hours of simulation, leading the service cache of nodes to store advertisements off-topic.

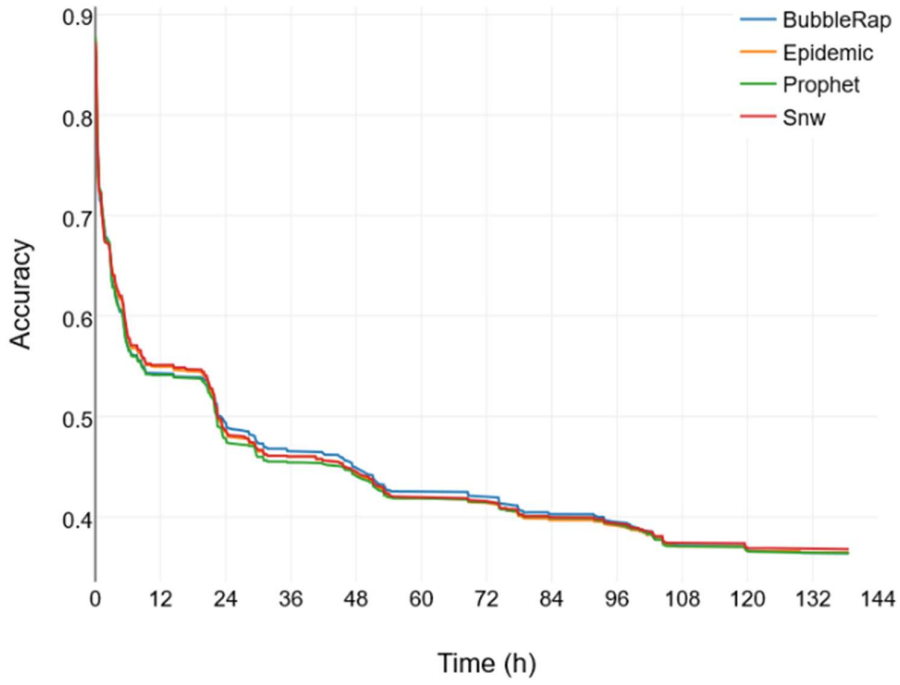


Figure 3. Accuracy of *S-Flood* with MDC Nokia.

Figure 3 shows the Accuracy metric of *S-Flood* measured with the MDC Nokia dataset. It is possible to notice a slower decrease than that the results with the Cambridge dataset. Such effect is motivated by the lower number of interactions with respect to the Cambridge dataset. Consequently, nodes tend to exchange a small number of advertisements. We measured that, independently of the trace and of the routing algorithms used, only the 30% of the services stored within the cache of nodes is of their own utility.

Proactivity

Results concerning the proactivity of *CORDIAL* with the Cambridge and MDC Nokia datasets are shown in Figures 4 and 5 respectively.

By observing the curves reported in both figures, it is possible to observe a growing trend. Such trend grows remarkably during the interval [0 - 7]*h* for Cambridge and during the interval [0 - 3]*h* for MDC Nokia. By the end of the observation period, *CORDIAL* performs with Cambridge a proactivity value around 0.8, whereas the values with MDC Nokia stands at 0.3. The rapid, initial growth of the trend is explained by the fact that the cache of the nodes is initially empty. Once that nodes enter in contact

to each other they start to exchange services filling out their caches, and, consequently, increasing the possibility to find the service required within their own caches. After the initial interval, the trend grows much more slowly because the nodes tend to encounter communities already met, declining the exchange of services.

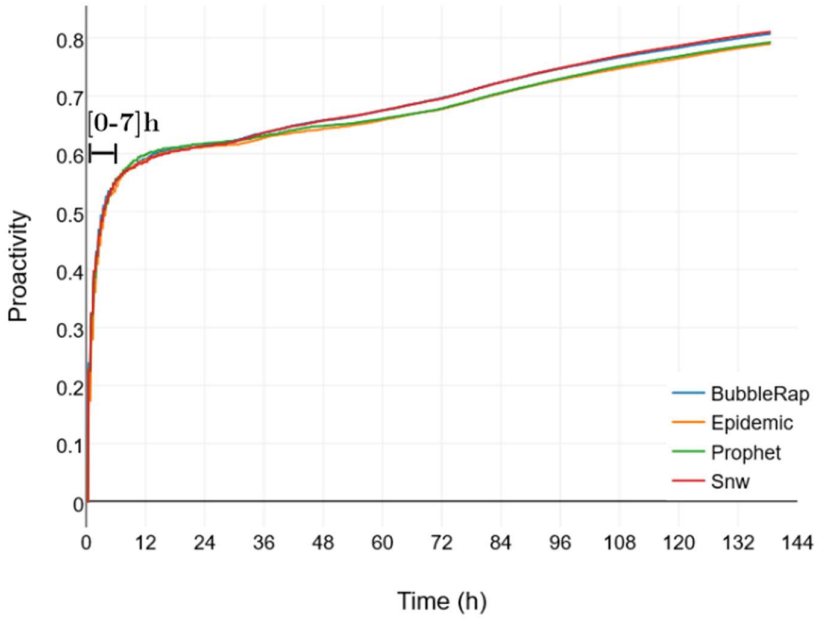


Figure 4. Proactivity of *CORDIAL* with Cambridge.

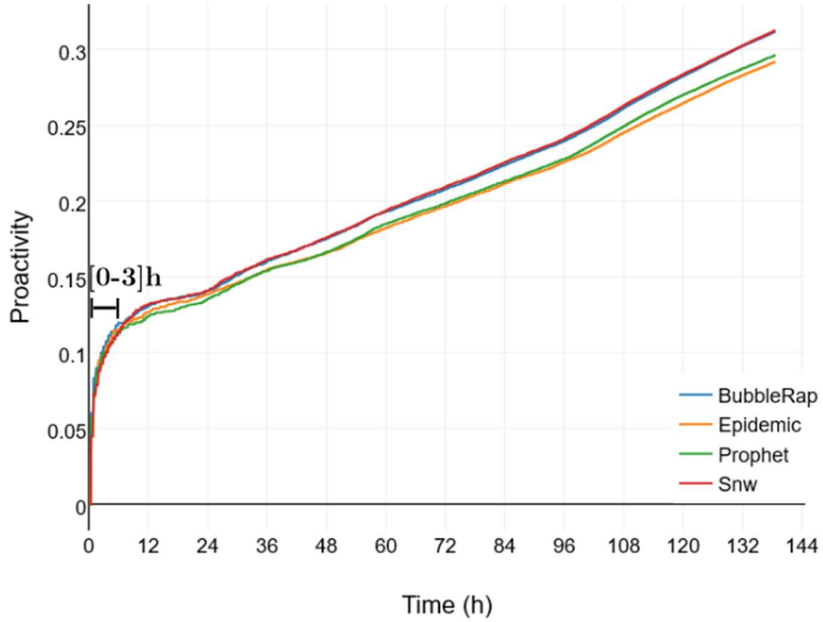


Figure 5. Proactivity of *CORDIAL* with MDC Nokia.

Although showing the same trend between the two datasets, it is possible to notice a clear distinction of the obtained values. This difference is justified by the fact that Cambridge has been collected in a restricted and constrained environment, leading nodes to exchange a higher volume of advertisements with respect to the simulation obtained with the MDC Nokia dataset.

The Proactivity metric is slightly affected by the routing algorithms we used. In particular, we observe from Figures 4 and 5, that *Epidemic* achieves the worst performance of all, even though it owns the most *aggressive* spread strategy. Its proactivity score stands at 78% and 29% for Cambridge and MDC Nokia respectively. *BubbleRap* and *Spray & Wait* stand at 81% with Cambridge and at 30% with MDC Nokia. *CORDIAL* limits the behavior of the *BubbleRap* routing algorithm since its spread strategy is based on social characteristics of the nodes. Against this background, an algorithm as aggressive as *Epidemic* is more restricted than others routing algorithms. It reduces the number of services propagated, and, therefore, the likelihood of find the desired service within its cache. Finally, it is possible to see that *BubbleRap*, which is a more selective algorithm in the diffusion strategy, achieves the same performance of the others. Despite such feature, it does not come into conflict with *CORDIAL*, whose strategy is based on the cooperation among nodes to leverage their shared interests for spreading services.

Results concerning the Proactivity metric obtained with *S-Flood* using Cambridge and MDC Nokia are shown in Figures 6 and 7 respectively.

Differently from *CORDIAL*, nodes running *S-Flood* exchange messages with the whole neighborhood. The effect is that there is a higher exchange of services with respect to *CORDIAL*. Nonetheless, the proactivity scores of *S-Flood* are not less significant than the *CORDIAL* ones. This is also because most exchanged services are not useful for nodes. However, *CORDIAL* obtains the highest value of proactivity with Cambridge and MDC Nokia both.

By analyzing the effect of the routing algorithms, we notice that *BubbleRap* performs less than the others. More specifically, *BubbleRap* performs a proactivity score of 75% with Cambridge and 27% with MDC Nokia at the end of the observation time. Differently, *Epidemic*, *Prophet* and *Spray & Wait* perform a proactivity score of 79% with Cambridge and 29% with MDC Nokia.

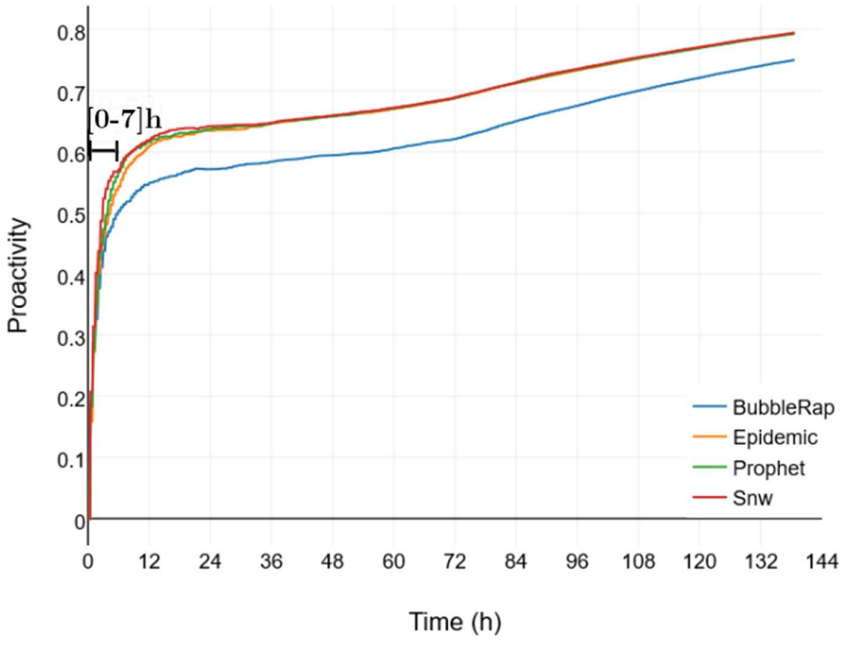


Figure 6. Proactivity of *S-Flood* with Cambridge.

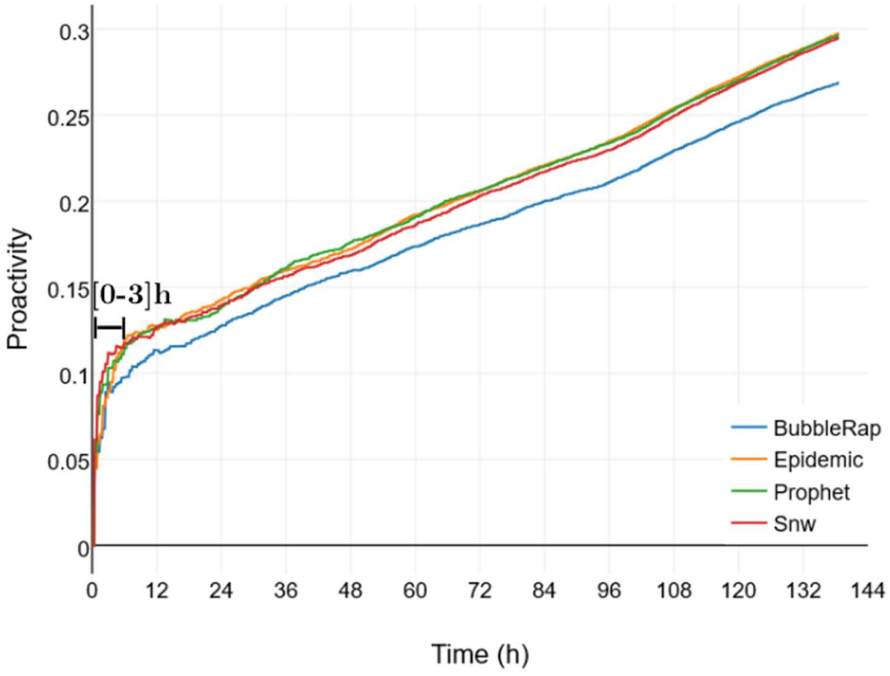


Figure 7. Proactivity of *S-Flood* with MDC Nokia.

An algorithm as *S-Flood*, which does not leverage the social characteristics of the nodes, generates a lower exchange of messages, thereby decreasing the likelihood of find a useful service in its cache, and, hence, the proactivity score.

Query Response Time

Results concerning the Query Response Time obtained with *CORDIAL* are shown in Figure 8 and Figure 9. Both of the graphs show an upward trend. At the beginning of the observation time, it is possible to notice that the Query Response Time is about 1.5 hour, but it tends to increase over time.

This behavior is a consequence of the fact that people roaming in the environment tend to visit the same people, and therefore they tend to exchange the same set of the service advertisements. Specifically, if a node has previously required a service without receiving any answer, even if it keeps moving around, the probability of encountering another node carrying an answer to its services remains low. As a consequence, the Query Response Time tends to increase over time.

Despite MDC Nokia is characterized by few contacts and very small communities, the selectivity of *CORDIAL* query diffusion strategy allows to forward queries uniquely within their own community deciding the nodes based on a score assigned to each node. Such strategy allows to mitigate the negative effects of the MDC Nokia's performance.

Varying the co-location traces, it is possible to notice that the behavior of *BubbleRap*, when is used with *CORDIAL*, does not adversely affect performance, even if the service discovery algorithm is equipped with a widely selective query diffusion strategy.

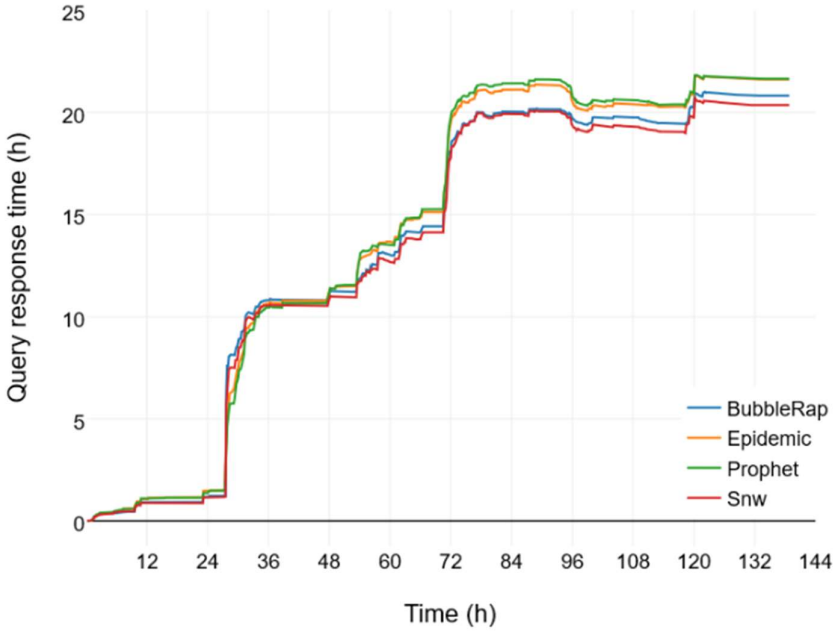


Figure 8. Query Response Time of *CORDIAL* with Cambridge.

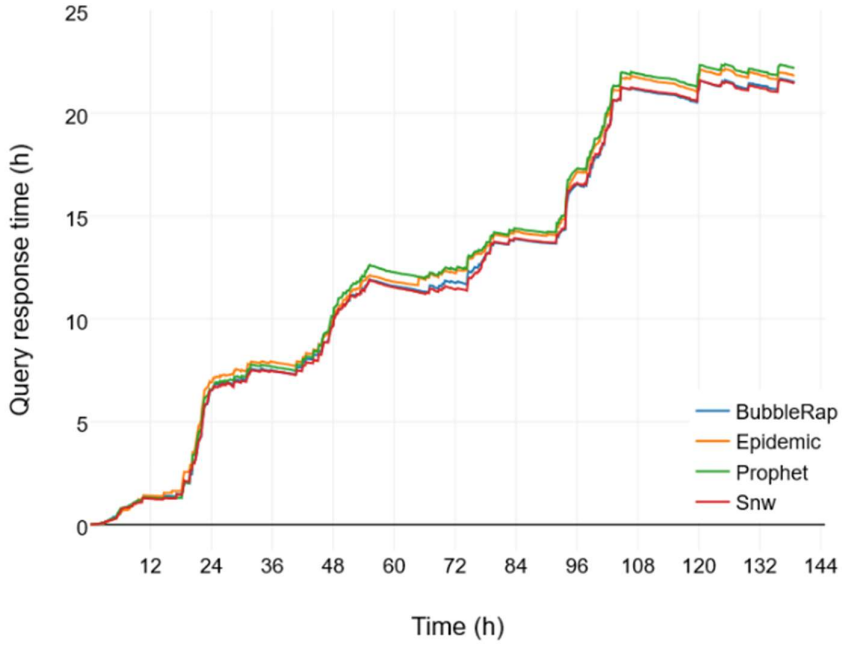


Figure 9. Query Response Time of *CORDIAL* with MDC Nokia.

Figure 10 and Figure 11 show the Query Response Time to the query obtained by the *S-Flood* algorithm by using the Cambridge and MDC Nokia.

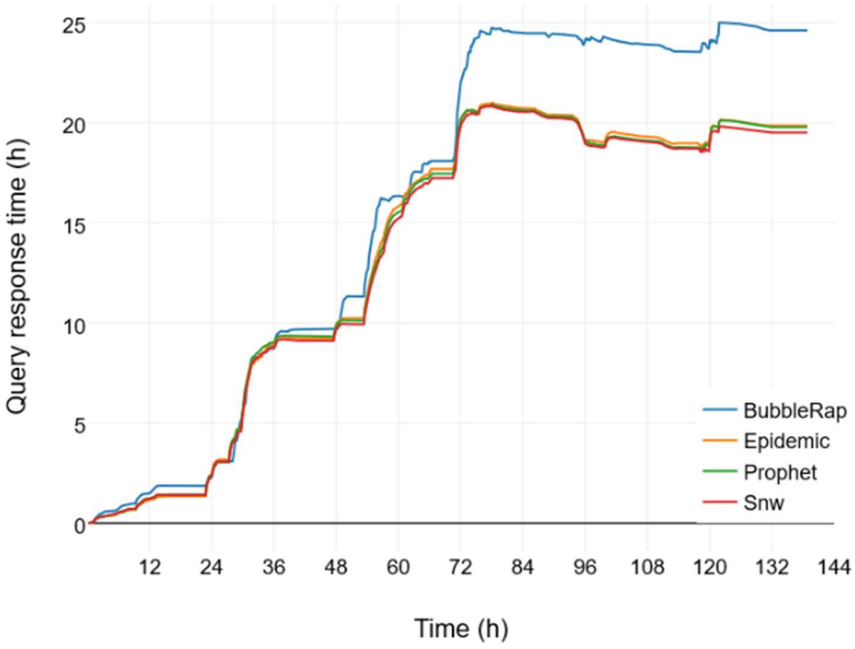


Figure 10. Query Response Time of *S-Flood* with Cambridge.

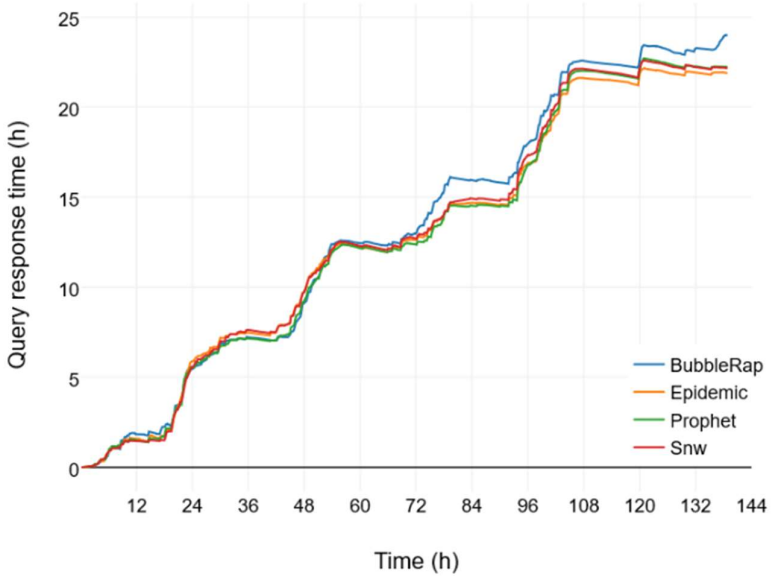


Figure 11. Query Response Time of *S-Flood* with MDC Nokia.

The trend of both traces is the same as that obtained using the previous algorithm, except for the values obtained at the end of the observation time. Specifically, the value of Query Response Time is 25 hours for *BubbleRap* and 20 hours for the other routing algorithms regardless of the considered trace. Contrary to *CORDIAL*, *S-Flood*'s query dissemination takes place between a node and all other nodes with whom it comes into contact. This behavior entails a higher diffusion of the requests, and, consequently, an increase in the probability of finding an answer briefly. This difference is only noticeable using the Cambridge trace. At the end of the observation time (see Figure 10) the Query Response Time value is 20 hours.

Service Cache

Results for the average size of the service cache obtained with *CORDIAL* are shown in Figures12 and 13. In the Figure 12, the trend increases rapidly in the interval [0-4]*h*. After the initial interval, it

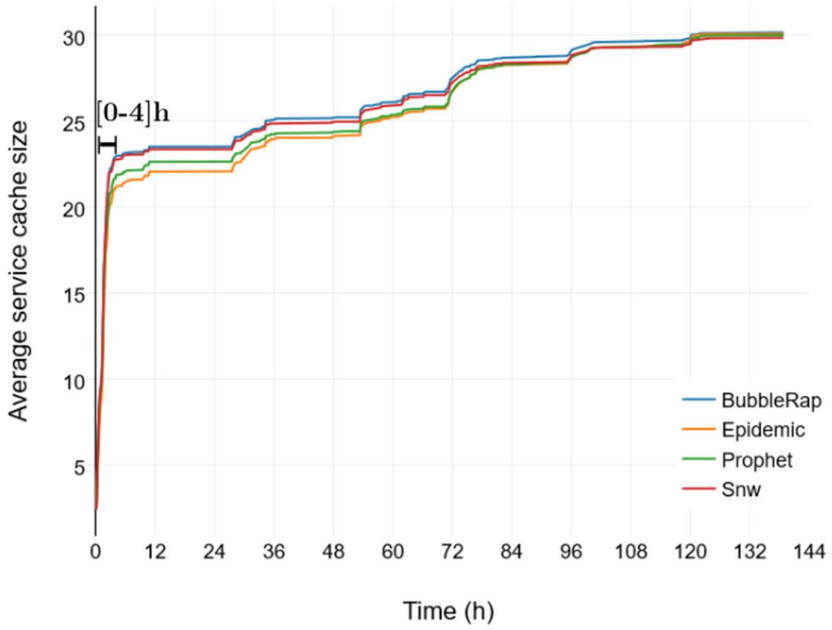


Figure 12. *CORDIAL* Service Cache with Cambridge.

increases much more slowly by alternating intervals whose dimension remains constant to intervals whose dimension grows again. By the end of the simulation time, the average size of the service cache is of 30 services out of 100. This upward trend is justified by the fact that the service cache of each

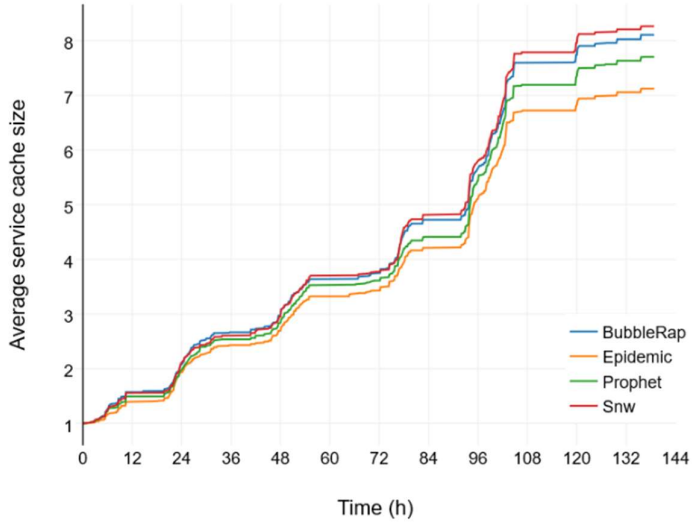


Figure 13. *CORDIAL* Service Cache with MDC Nokia.

node is initially empty. Gradually, the nodes meet each other, and their respective service caches fill up with services exchanged. In the early hours of the Cambridge trace is possible to see a large number of contacts. This number corresponds to the rapid, initial growth. It is possible to notice that when the number of contacts decreases, the average number of services stored into the service cache remains constant.

Figure 13 shows a wavelike, upward behavior that remains constant roughly all the time. At the end of the observation time, the average size of the service cache is of eight services out of one hundred. Once again, the trend is correlated to the number of contacts establishing among the nodes of the MDC Nokia dataset. Comparing the charts of both Figures, it is possible to notice differences between the average sizes of their respective caches.

Results concerning the average size of the service cache obtained by *S-Flood* algorithm using the Cambridge dataset are shown in Figure 14, whereas the results using the MDC Nokia dataset are shown in Figure 15.

In the Figure 14 the trend grows rapidly during the interval [0-4]*h*. After the initial interval, it increases much more slowly by alternating intervals whose dimension remains constant to intervals whose dimension grows again. In other words, it is the same trend obtained by *CORDIAL*. By the end of the observation time, the average size of the service cache is of 94 services out of 100.

Same observations can be done for the Figure 15. It is the same trend obtained by *CORDIAL* using MDC Nokia. By the end of the simulation time, the average size of the service cache is of 22 services out of 100.

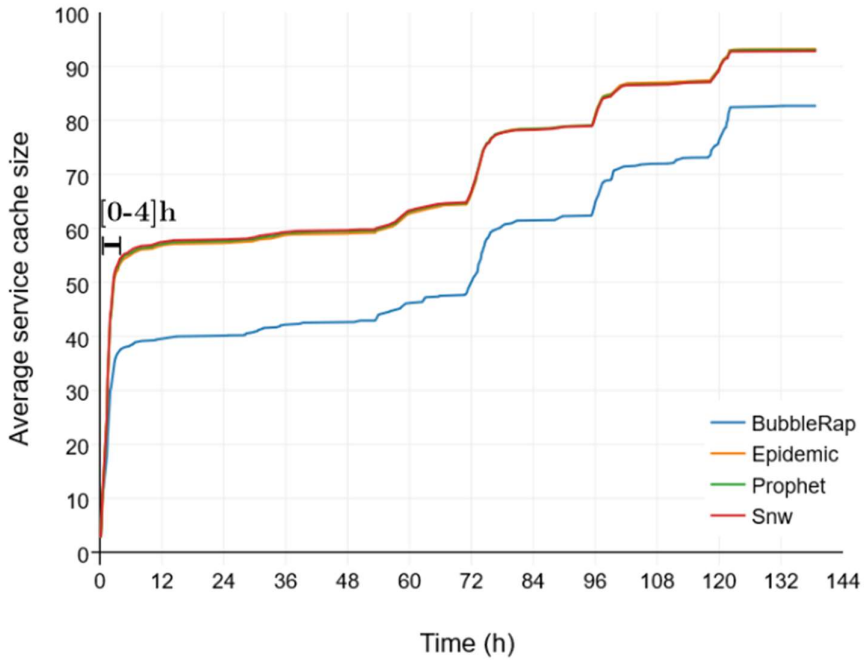


Figure 14. *S-Flood* Service Cache with Cambridge.

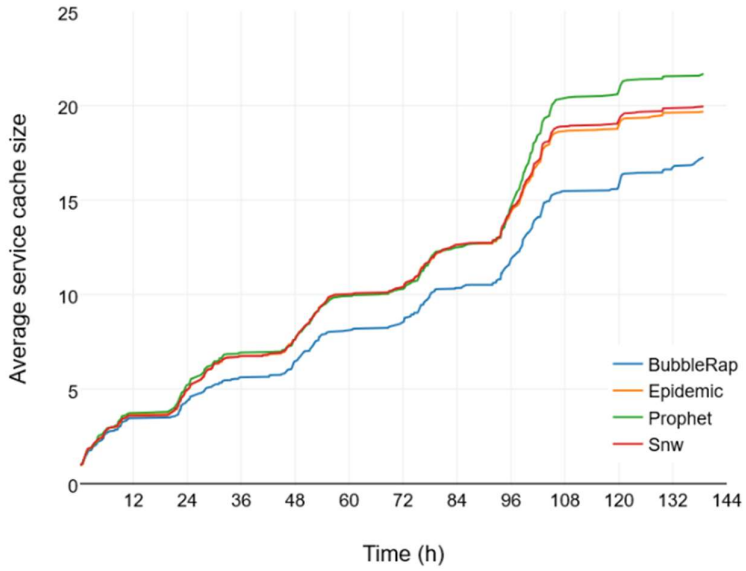


Figure 15. *S-Flood* Service Cache with MDC Nokia.

Comparing the routing algorithms, it is possible to notice the different behavior of *BubbleRap*. This algorithm is much more selective than the others, and, as a consequence, it obtains an average size of 84 services for Cambridge and 17 for MDC Nokia.

Forwarding Cache

The Forwarding cache is only measured for the *CORDIAL* algorithm because the *S-Flood* strategy does not provide the capability of storing the advertisements received by nodes. The advertisements must be delivered either to the final recipient or to the node with the lesser intercontact time.

Results concerning the average size of the forwarding cache obtained with *CORDIAL* using Cambridge are shown in Figure 16, whereas the same results using MDC Nokia are shown in Figure 17.

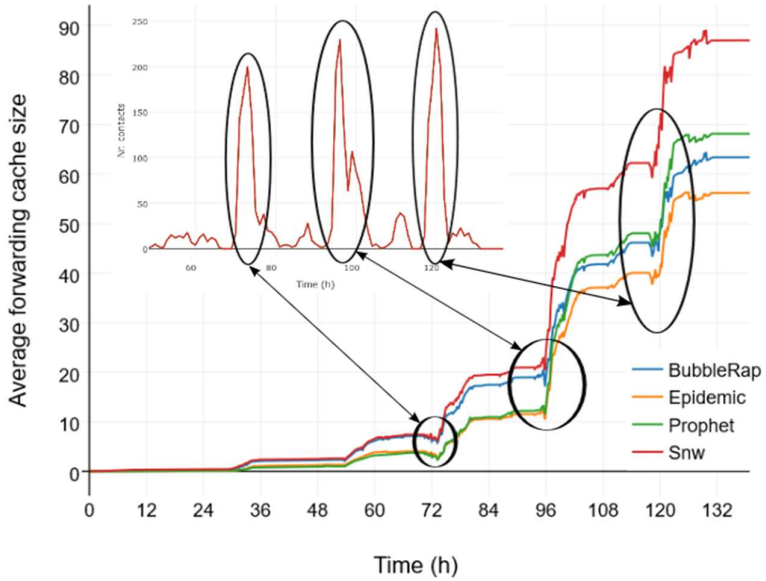


Figure 16. Average dimension of the *Forwarding cache* with Cambridge.

As can be seen from the first chart, we notice a constant trend where in the interval [0-30]*h,* the average size is zero. Subsequently, the dimension starts to grow with a stair-step effect before stabilizing by the end of the simulation. The values obtained vary according to the routing algorithm.

Spray & Wait obtains a final average size of 86 services, *Prophet* and *BubbleRap* nearly 66, and *Epidemic* 56. The forwarding cache of *Epidemic* obtains a lesser average size than that of the others. As a final point, although *Epidemic* is an aggressive algorithm its behavior is restricted by *CORDIAL*. The consequence is a lesser exchange of services among nodes which affects their subsequent storage into the forwarding cache. The development of the forwarding cache follows the trend of the number of the contacts per hour. The more grows the number of contacts, the less is the average size of the forwarding cache. In this way, the advertisements are delivered to the encountered nodes and removed from the forwarding cache. Whereas, each time that the number of contacts significantly decreases, we have an increase of the average dimension of the forwarding cache.

Despite the considerable number of contacts, the initial trend is linear. This behavior is justified by the fact that the forwarding cache is initially empty. Only when nodes start moving around the network they begin to receive the advertisements and to fill such cache.

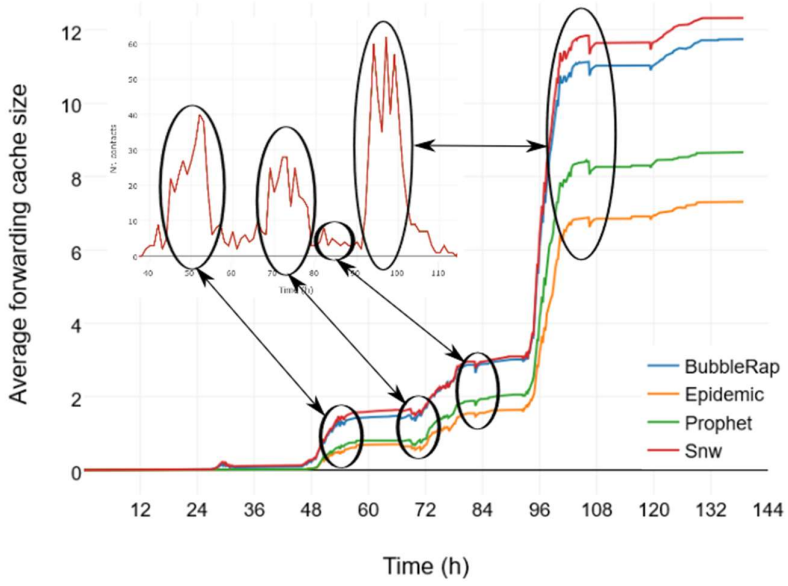


Figure 17. Average dimension of the *Forwarding cache* using MDC Nokia.

Figure 17 shows a trend similar to that obtained using Cambridge. Particularly, the trend of the forwarding cache follows the one of the number of contacts per hour of MDC Nokia.

Comparing Figure 16 with Figure 17, we can see that there is a noticeable difference about the average dimensions of each cache at the end of the observation time. As previously mentioned, this difference is caused by the number of nodes that meet each other in both traces. By the end of the observation time, values obtained vary from routing algorithm to routing algorithm: *Spray & Wait* and *BubbleRap* obtain an average forwarding cache size of 12, *Prophet* of 9, and *Epidemic* of 7.

From Figures 12 and 16 in conjunction, and from Figures 13 and 17 in conjunction, it is possible to derive the overheads of *CORDIAL* computed for both Cambridge and MDC Nokia respectively. Such results are roughly independent from the routing protocol in use. Even Spray & Wait, which shows the highest lack of accuracy in keeping the number of advertisements low into the forwarding cache, shows the same behaviour in both datasets. In addition, it is possible to observe that with MDC Nokia dataset the memory occupation of *CORDIAL* is much lower than that of *S-Flood* (about this algorithm

has been considered only the service cache because it does not need of any forwarding cache), and with the Cambridge dataset the memory occupation of *CORDIAL* is also lower than that of S-Flood, but the gap is much smaller than the previous case.

Concerning the processing costs, each node in *CORDIAL* has two kinds of costs: those inherent to the messages processing and those related to the construction and maintenance of the data structures it needs, in particular to keep its community updated. In both cases the computational costs for the device are mostly due to a few updates/look-ups of its caches/tables, which are not particularly expensive (they can be made very efficient with suitable indexes). Also note that, concerning message processing costs, these mostly depend on the number of messages exchanged, which depend on the number of services available and of queries generated (both independent on the specific service discovery protocol at hand), and only in part by the diffusion strategy of the protocol. In our case, since *CORDIAL* limits the number of messages (either advertisements or queries) it forwards to the devices that are potentially interested in, it can limit these costs more than algorithms like *S-Flood*.

5.2 Comparison of Service Discovery Algorithms

To demonstrate the effectiveness of *CORDIAL* in discovering and advertising services, we analysed its behaviour against *S-Flood* [33] and another algorithm for service discovery in MSN, namely *SIDEMAN* [10]. *SIDEMAN* represents a useful term of comparison for *CORDIAL* since it is also designed to cope with the social features of MSN. Specifically, *SIDEMAN* is characterized by two elements:

- The use of both reactive and proactive approaches to service discovery for actively submitting a query and for passively being notified with services of interests.
- The use of a community-based diffusion strategy for the propagation of queries and services. In particular, query and service messages are forwarded selectively to the members of a community whose interests match those of the message to be forwarded.

Moreover, *SIDEMAN* keeps track of communities visited in the past so to avoid keeping detecting communities already known. Differently from *CORDIAL*, nodes running *SIDEMAN* neither consider the friends of a node for forwarding queries and advertisements nor store-and-carry queries to be forwarded to other nodes.

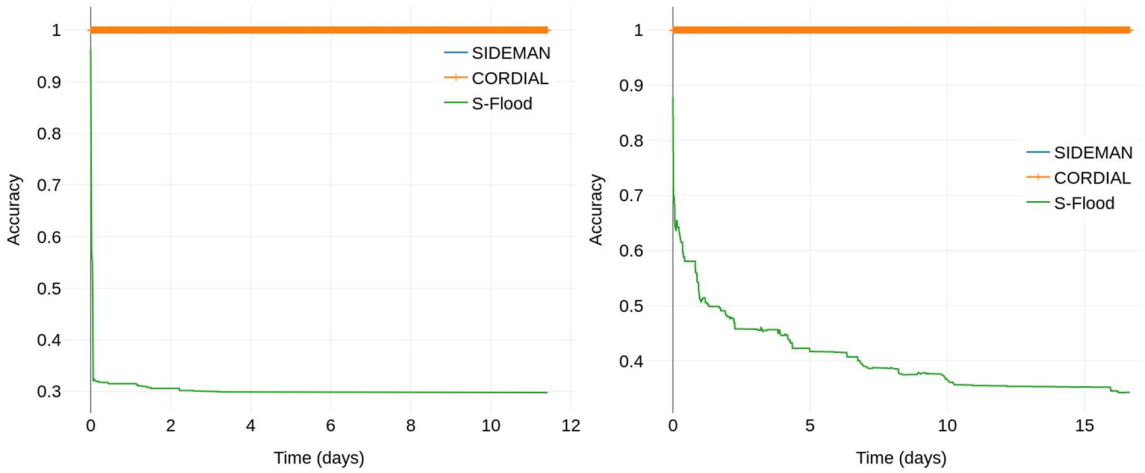


Figure 18. Accuracy in Cambridge (left) and MDC Nokia (right)

In order to clarify the difference among *CORDIAL*, *SIDEMAN* and *S-Flood* we report on Table 2 how such algorithm faces with the two key operations of most of the Service Discovery algorithms:

- To select the forwarding set, the nodes to which forward query and/or advertisement messages.
- To select the queries and advertisements to share with the forwarding set.

Table 2. Features of the Service Discovery Algorithms.

Data structure	**Select forwarding set**	**Select advertisements and queries**
CORDIAL	Members of the current community with highest score (See Eq. 1 in Section 3.1).	Advertisements and queries with highest similarity index.
SIDEMAN	Members of the current community with interests matching the query.	Advertisements and queries whose interests match those of the community members.
S-Flood	All members of its current community.	All advertisements and queries to all community members.

In the following, we analyse the behaviour of the three algorithms reported in Table 2 by means of the metrics Accuracy, Proactivity, Query Response Time and Service Cache (see Section 4.2.2), and by using the two datasets previously introduced, namely Cambridge and MDC Nokia (see Section 4.1) over a comparable period of time.

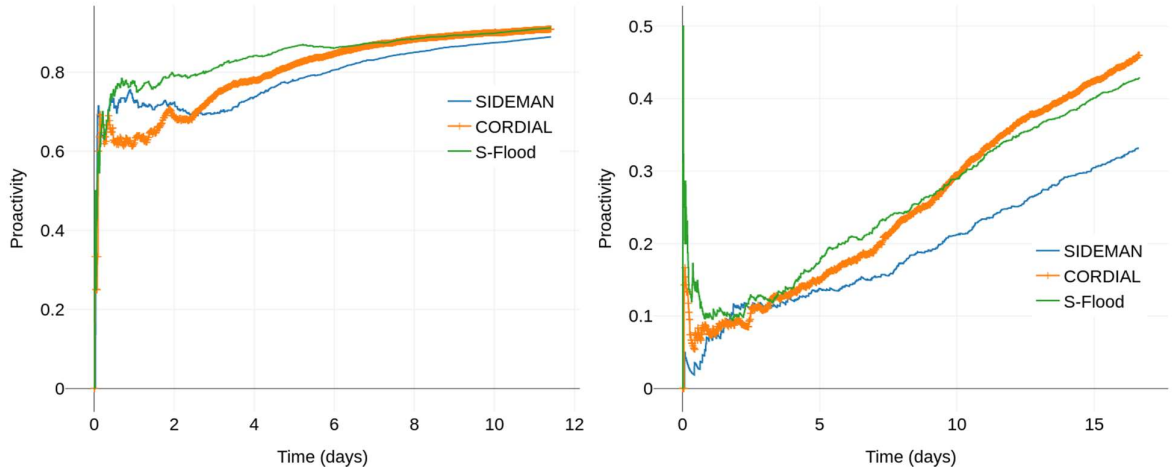


Figure 19. Proactivity in Cambridge (left) and MDC Nokia (right)

Figure 18 shows the results concerning the Accuracy metric both with Cambridge and MDC Nokia scenarios. *CORDIAL* and *SIDEMAN* algorithms report a perfect accuracy value, while *S-Flood* decreases the accuracy as time progresses. More precisely, *CORDIAL* and *SIDEMAN* implement a strategy for reducing the diffusion of advertisements off-topic for a node by matching the topics of the advertisements with topics of the nodes. Differently, *S-Flood* spreads the advertisements to all the nodes that are in contact to each other without any filter, as a result the accuracy of *S-Flood* tends to decrease with the time. Results of *S-Flood* differ in Cambridge with respect to the MDC Nokia scenario. Such difference is caused by the nature of the dataset we used. Nodes in the Cambridge dataset have a higher number of contacts with respect to nodes in MDC Nokia. As a result, nodes in Cambridge share with higher probability information off-topic. Differently, in MDC Nokia nodes encounter more slowly other nodes by reducing the probability of exchanging information off-topic.

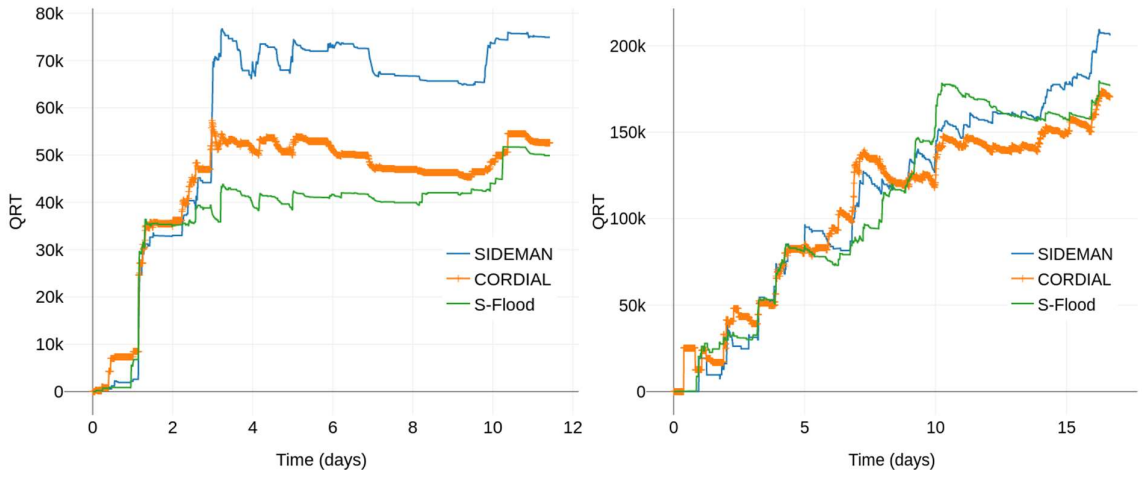


Figure 20. Query Response Time in Cambridge (left) and MDC Nokia (right)

Results concerning the Proactivity metric are reported in Figure 19. In both simulation scenarios, the proactivity increases with the simulation time. In particular, at the beginning of every simulation, devices start without any advertisement stored in their cache and, as time passes, they exchange advertisements with other devices. The proactivity quickly increases during the first 2 days of simulation, after which the curve still grows but with a slower slope. *CORDIAL* obtains a value of proactivity always comparable with respect to *S-Flood* algorithms (our benchmark) and always higher than that the *SIDEMAN* algorithm. By the end of the observation time, the proactivity of the three

algorithms is 0.95 meaning that a device willing to access a service finds, with high probability, in its cache the service advertisement needed. In MDC Nokia the Proactivity metric increases slower than that the Cambridge scenario, as reported in Figure 19. The MDC Nokia dataset reproduces a more

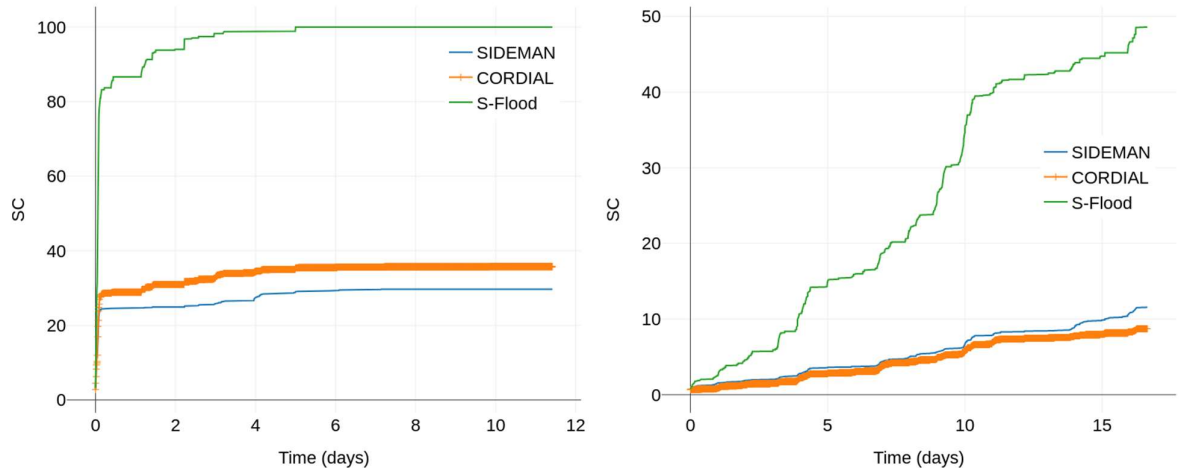


Figure 21. Service Cache in Cambridge (left) and MDC Nokia (right)

challenging and interesting scenario. In this case, the mobility of devices is not limited to a specific region, rather devices are free to roam in large area. However, devices meet only a small portion of the whole population and the number of contacts per hour is even smaller that Cambridge. These aspects affect the proactivity value, giving rise to a very slow increase during the simulation time. Also in this case, by the end of the simulation time, *CORDIAL* outperforms both *S-Flood* and *SIDEMAN*. Hence, the strategy of *CORDIAL* for the diffusion of advertisements is effective both in scenarios highly connected (e.g. Cambridge) and in scenarios highly disconnected (e.g. MDC Nokia).

Results concerning the Query Response Time are shown in Figure 20. The trend for the three algorithms is the same: the Query Response Time increases quickly during the warm-up period of the simulation after which the Query Response Time value remains stable until the end of the simulation. As previously discussed the *S-Flood* algorithm exchanges the highest number of advertisements among devices, hence devices have also the highest probability of answering to a query. However, the drawback of such strategy is an un-controlled diffusion of advertisements, giving rise to a value of accuracy very low in every scenario. *CORDIAL* obtains a value of Query Response Time never noticeable, worse than that of the *S-Flood* strategy (our benchmark), but with optimal values for both the Accuracy and the Proactivity metrics. The results of the Query Response Time metric demonstrate the effectiveness of *CORDIAL* in controlling the diffusion of advertisement, without meaningful effects on the responsiveness of the service discovery. When compared with *SIDEMAN*, *CORDIAL* has a Query Response Time always far lower, this provides a further indication of the improvement of *CORDIAL* with respect to *SIDEMAN*.

Finally, Figure 21 shows the Service Cache metric in the Cambridge and MDC Nokia scenarios. As expected, devices running the *S-Flood* algorithm store the highest number of advertisements. After approximately 2 days of simulation, devices running the *S-Flood* carry every advertisement available in the simulation. However, most of such advertisements are off topic for the device, as shown by the Accuracy metric in Figure 2. We notice that *S-Food* requires an excessive storage capacity for a device, and this represents a non-negligible constraint for the application scenario to which we refer. Differently, devices running *CORDIAL* and *SIDEMAN* control the number of advertisements stored their cache. More precisely, devices running *SIDEMAN* store an average of 30 advertisements while

devices running *CORDIAL* store an average of 37 advertisements. *CORDIAL* requires a bit more storage capacity with respect to *SIDEMAN* because of the advertisement forwarding strategy. Such strategy differs from *SIDEMAN* and it requires more storage capacity. However, the benefits deriving from the forwarding cache concern an increase in Proactivity and Query Response Time.

5.3 Summary of Simulation Results

The performance of *CORDIAL, SIDEMAN* and *S-Flood* depend on the routing protocols implemented and the dataset on which they are tested. Evaluation metrics highlight that the accuracy of *S-Flood* decreases rapidly after the first part of each simulation, whereas the accuracy of both *CORDIAL* and *SIDEMAN* remains stable throughout all simulations. Concerning the Proactivity metric, *CORDIAL* performs a better result with respect both *SIDEMAN* and *S-Flood* on both datasets. This behaviour is due to the advertisement dissemination strategy of *CORDIAL*, based on the cooperation among nodes through the exploitation of their social characteristics, in contrast with the *SIDEMAN* strategy, based on common queries interest, and the *S-Flood* strategy, based on flooding technique. A diversification in the algorithms behaviour from one dataset to the other is given looking at the Query Response Time metric. *CORDIAL* performs a better execution with respect *SIDEMAN*, highlighting that the control of the advertisement dissemination, does not negatively affect the responsiveness of the service discovery. Regarding the comparison with *S-Flood*, *CORDIAL* allows to forward targeted queries selecting nodes within the community of the node sender, even reducing poor performances on a dataset with a limited number of communities as MDC Nokia. Conversely, the higher probability of finding an answer to a request using *S-Flood* is overshadowed by the high cost in submitting requests to any node with which the sender node comes into contact. Concerning the Service Cache metric, which estimates the average number of advertisements stored locally in devices' cache, *CORDIAL* controls its dimension selecting advertisements based on proactivity phase, thus avoiding its overload. However, if compared with the performance of *SIDEMAN*, which is more selective, *CORDIAL* requires capacity of storing slightly larger. On the contrary *S-Flood*, that does not have a strategy for limiting the number of service advertisements, performs an excessive overload of the service cache.

For what has been ascertained, the Forwarding cache metric, which is limited to the performance of *CORDIAL* and *SIDEMAN*, returns (for the former) quite different outcomes on the basis of the routing algorithm implemented on each test carried out. Such results are more encouraging with the use of routing algorithms that limit the number of copies of the messages as Spray & Wait with respect others that use flooding-based techniques as Epidemic.

6. Conclusions

The ever-increasing number of mobile and wearable devices achieves short-range communications without precedents. The request of services as well as their dissemination are crucial factors for MSN entities, and the study of users' social aspects become vital for fully understanding the communication possibilities of their devices. In this paper, we presented *CORDIAL*, a collaborative service discovery algorithm for MSN users. Performances of our algorithm have been compared with *S-Flood*, a benchmark data dissemination technique and against *SIDEMAN,* a state of the art algorithm for service discovery in MSN. Both *CORDIAL* and *S-Flood* were performed with four routing protocols for opportunistic networks, namely *Epidemic*, *Spray & Wait*, *Prophet*, and *BubbleRap* (*SIDEMAN* has its own dissemination strategy). Evaluation metrics included *Accuracy*, *Proactivity*, *Query Response Time*, *Service Cache* and *Forwarding Cache*.

Our results show that the performance of *CORDIAL* remain stable with the different routing algorithms, whereas depending on the co-location trace in use, its behavior is deeply diversified as proof that disparate properties in terms of mobility/sociality of the users have a remarkable influence on it. Results also show that *CORDIAL* is able to match the performance of *S-Flood* in terms of Query Response Time and outclass the latter in terms of proactivity. As compared against *SIDEMAN*, *CORDIAL* performs a better result in terms of proactivity and Query Response Time, requiring a percentage of extra memory space for storing queries and advertisement with respect its antagonist.

The improvement of different diffusion strategies of queries and advertisements is an important aspect which we have considered for future works. In fact, queries should be distributed to nodes with the highest probability of finding an answer in the shortest possible time. The diffusion of advertisements could rely on relay devices to be more accurate, and the selection of such relay devices could be simplified by using community detection algorithms that join temporal, spatial, and social attributes.

References

[1] Girolami M., Chessa S., Caruso A., "On Service Discovery in Mobile Social Networks: Survey and Perspectives", Computer Networks, 88 (2015):51-71, DOI 10.1016/j.comnet.2015.06.006

[2] Álvarez-García J. A., Arcos García A., Chessa S., Fortunati L., Girolami M., "Detecting Social Interactions in Working Environments through Sensing Technologies", 7th International Symposium on Ambient Intelligence (ISAmI), University of Sevilla (Spain), 1-3 June 2016. Appears in Advances in Intelligent Systems and Computing, vol. 476, pp. 21-29.

[3] Eagle N., Pentland A., "Reality Mining: Sensing Complex Social Systems", Personal and Ubiquitous Computing 10.4 (2006): 255-268.

[4] Matic A., Osmani V., Mayora-Ibarra O., "Analysis of Social Interactions through Mobile Phones", Mobile Networks and Applications 17.6 (2012): 808-819.

[5] Wyatt D., Choudhury T., Bilmes J., Kitts J. A., "Inferring Colocation and Conversation Networks from Privacy-Sensitive Audio with Implications for Computational Social Science", ACM Trans. on Int. Systems and Technology (TIST) 2.1 (2011): 7

[6] McPherson M., Lovin L. S., Cook J. M., "Birds of a Feather: Homophily in Social Networks", Annual Review of Sociology, 27: 415–444, 2001

[7] Chakraborty D., Joshi A., Yesha Y., and Finin T., "Toward Distributed Service Discovery in Pervasive Computing Environments", IEEE Transactions on Mobile Computing, 5(2):97–112, 2006

[8] Girolami M., Ferro E., Chessa S., "Discovery of Services in Smart Cities of Mobile Social Users", Management of Cloud and Smart city systems (MoCS), Larnaca, Cyprus, 6-9 July 2015, pp. 1081-1086

[9] Girolami M., Chessa S., Basagni S., Furfari F., "Service Discovery in Mobile Social Networks", IEEE 25th Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC'14), Washington DC, USA, 2-5 September 2014, pp. 1464-1468

[10] Girolami M., Basagni S., Furfari F., Chessa S., "*SIDEMAN*: Service Discovery in Mobile Social Networks", Ad Hoc & Sensor Wireless Networks 34(2016):1-39

[11] Hui P., Yoneki E., Chan S.Y., Crowcroft J., "Distributed Community Detection in Delay Tolerant Networks", Proc. Of ACM/IEEE MobiArch, 2007, pp.7:1–7:8

[12] Orlinski M., Filer N., "The Rise and Fall of Spatio-Temporal Clusters in Mobile Ad Hoc Networks", Ad Hoc Networks 11(5) (2013): 1641–1654

[13] Borgia E., Conti M., Passarella A., "Autonomic Detection of Dynamic Social Communities in Opportunistic Networks", In Ad Hoc Networking Workshop (Med-Hoc-Net), 2011 The 10th IFIP Annual Mediterranean (pp. 142-149).

[14] Vahdat A., Becker D. "Epidemic Routing for Partially Connected Ad Hoc Networks", Technical Report, Duke University, 2000. doi:10.1.1.34.6151

[15] Spyropoulos T., Psounis K., Raghavendra C. S., "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks", Proc. ACM SIGCOMM Workshop on Delay-tolerant Networking (WDTN), pp.252-259, New York, NY, USA, 2005

[16] Lindgren A., Doria A., Schelén O., "Probabilistic Routing in Intermittently Connected Networks", SIGMOBILE Mob. Comput. Commun. Rev., 7(3):19-20, July 2003

[17] Hui P., Crowcroft J., Yoneki E., “Bubble Rap: Social-based Forwarding in Delay Tolerant Networks”, Proc. 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 241-250, New York, NY, USA, 2008
[18] Ververidis N., Polyzos G. C., “Service Discovery for Mobile Ad Hoc Networks: A Survey of Issues and Techniques”, IEEE Communications Surveys & Tutorials, 10(3):30–45, 2008
[19] Klein M., Konig-Ries B., Obreiter P., “Service Rings - A Semantic Overlay for Service Discovery in Ad Hoc Networks”, in DEXA 2003, pp. 180–185
[20] Aguilera U., López-de-Peña D., “A Parameter-based Service Discovery Protocol for Mobile Ad-Hoc Networks”, Proc. 11th international conference on Ad-hoc, Mobile, and Wireless Networks (ADHOC-NOW), 2012. Xiang-Yang Li, Simon Papavassiliou, and Stefan Ruehrup (Eds.). Springer-Verlag, Berlin, Heidelberg, 274-287
[21] Helal S., Desai N., Verma V., Lee C., “Konark - A Service Discovery and Delivery Protocol for Ad-Hoc Networks”, IEEE WCNC ‘03, pp.2107-2113
[22] Nguyen T. D., Rouvrais S., “A Socially Inspired Peer-to-Peer Resource Discovery Service for Delay Tolerant Networks”, OTM 2007, pp. 960–969
[23] Le Sommer N., Said R., Maheo Y., “A Proxy-based Model for Service Pro-Vision in Opportunistic Networks”, MPAC workshop, pp. 7–12, 2008
[24] Al Hayat S. A., Aly S., Hares K. A., “Pipe: Impact of Power-Awareness on Social-based Opportunistic Advertising”, Proc. of IEEEWCNC,2014
[25] He Z., Cai Z., Han Q., Tong W., Sun L., Li Y., “An Energy Efficient Privacy-preserving Content Sharing Scheme in Mobile Social Networks”, Personal and Ubiquitous Computing, 20(5):833-846, October 2016
[26] Le Sommer N., Maheo Y., “OLFServ: An Opportunistic and Location-Aware Forwarding Protocol for Service Delivery in Disconnected MANETs”, Ubicomm, 2011, pp. 115–122
[27] Mei A., Morabito G., Santi P., Stefa J., “Social-Aware Stateless Forwarding in Pocket Switched Networks”, IEEE INFOCON 2011, pp. 251–255
[28] Scott J., Gass R., Crowcroft J., Hui P., Diot C., Chaintreau A., \CRAWDAD data set cambridge/haggle (v. 2006-01-31)." Downloaded from http://crawdad.org/cambridge/haggle/, 2006
[29] Kiukkonen N., Blom J., Dousse O., Gatica-Perez D., Laurila J., “Towards Rich Mobile Phone Datasets: Lausanne Data Collection Campaign”, in: Proc. ACM Int. Conf. on Pervasive Services (ICPS), Berlin, 2010.
[30] Laurila J. K., Gatica-Perez D., Aad I., Blom J., Bornet O., Do T.-M.-T., Dousse O., Eberle J., Miettinen M., “The Mobile Data Challenge: Big Data for Mobile Computing Research”, in: Pervasive Computing, 2012
[31] Chessa S., Girolami M., Foschini L., Ianniello R., Corrido A., Bellavista P., “Mobile Crowd Sensing Management with the ParticipAct Living Lab”, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.09.005
[32] Keränen A., Ott J., Kärkkäinen T., “The ONE Simulator for DTN Protocol Evaluation”, in SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, (New York, NY, USA), ICST, 2009.
[33] Lim H., Kim C., “Flooding in Wireless Ad Hoc Networks”, Computer Communications, 4(34):353 – 363, 2001.

Michele Girolami received his MSc and PHD in Computer Science from the University of Pisa respectively in 2007 and 2015. Currently is in the Research Staff at the CNR-ISTI with the Wireless Network Laboratory. His research interests are service-discovery and data diffusion in Mobile Social Networks, context-aware middleware for Smart Environments and interoperable gateways for sensing technologies.

Dimitri Belli received his MSc in Digital Humanities from the University of Pisa in 2016. Currently, he is PhD student in Computer Science at the University of Pisa. His research interests are in Mobile Distributed Systems, Internet of Things and Smart Environment with a special focus on Mobile Crowdsensing and Mobile Edge Computing paradigms.

Stefano Chessa is Associate Professor at the Department of Computer Science of the University of Pisa. He co-authored more than 150 papers published on international journals and conference proceedings. His research interests are in the areas of internet of things, mobile networks, smart environments, ambient assisted living and activity recognition.