OLFServ: an Opportunistic and Location-Aware Forwarding Protocol for Service Delivery in Disconnected MANETs

Nicolas Le Sommer, Yves Mahéo

OLFServ: an Opportunistic and Location-Aware Forwarding Protocol for Service Delivery in Disconnected MANETs

Nicolas Le Sommer and Yves Mahéo
Valoria Laboratory, Université de Bretagne-Sud, France
{Nicolas.Le-Sommer,Yves.Maheo}@univ-ubs.fr

***Abstract*—Handheld devices equipped with Wi-Fi interfaces are widespread nowadays. These devices can form disconnected mobile ad hoc networks (DMANETs) spontaneously. These networks may allow service providers, such as local authorities, to deliver new kinds of services in a wide area (e.g. a city) without resorting to the infrastructure-based networks of mobile phone operators. This paper presents OLFServ, a new opportunistic and location-aware forwarding protocol for service discovery and delivery in DMANETs composed of numerous mobile devices. This protocol implements several self-pruning heuristics allowing mobile nodes to decide whether they efficiently contribute in the message delivery. The protocol has been implemented in a service-oriented middleware platform, and has been validated through simulations, which proved its efficiency.**

***Index Terms*—Opportunistic Service provision, Mobile Ad hoc Networks**

I. Introduction

The increasing interest of people for handheld devices equipped with a Wi-Fi interface and sometimes with a GPS receiver (e.g., smartphones, Internet tablets) offers to service providers, such as local authorities, new opportunities to provide nomadic people with new ubiquitous services without resorting to licensed frequency bands (e.g., UMTS, GPRS). Indeed, these devices can form mobile ad hoc networks spontaneously, and this ability could be exploited in order to artificially extend networks composed of some sparsely distributed infostations with a view to offering a wide service access to end-users. However, designing a routing protocol that allows an efficient and distributed service discovery and invocation in such dynamic networks remains a challenging problem today, because disconnections are prevalent and the lack of knowledge about the network topology changes hinders the selection of best routes for message forwarding. Indeed in disconnected mobile ad hoc networks (DMANETs), devices can communicate directly only when they are in range of one another. Intermediate nodes can be used to relay a message from a source to its destination following the "store, carry and forward" principle. The routes are therefore computed dynamically at each hop while the messages are forwarded towards their destination(s). Each node receiving a message for a given destination is thus expected to exploit its local knowledge to decide which are the best next forwarders among its current neighbors to deliver the message. When no forwarding opportunity exists (e.g., no other nodes are in the transmission range, or the neighbors are evaluated as not suitable for that communication) the node stores the message and waits for future contact opportunities with other devices to forward the message. Thanks to this principle, a message can be delivered even if the client and the destination are not present simultaneously in the network, or if they are not in the same network partition at emission time.

This paper presents OLFServ, a new opportunistic and location-aware forwarding protocol we have designed in order to support both service discovery and service invocation in DMANETs. OLFServ is a key element of a middleware platform we develop to investigate service provisioning in DMANETs [1]. Based on the location data collected by the platform from the wireless interface and/or the GPS receiver of the device, OLFServ makes it possible to perform an efficient and geographically-based broadcast of both service advertisements and service discovery requests, as well as a location-driven service invocation. OLFServ implements several self-pruning heuristics allowing intermediates nodes to decide themselves if they are "good" relays to deliver the messages they receive from their neighbors (i.e., if they contribute to bring a message closer to its destination). These heuristics aim to progressively refine the area where a message can be disseminated until reaching its destination; to perform source routing when it is possible; to support the client mobility by computing the area where the client is expected to be when it receives its response; to avoid message collisions by implementing a backoff mechanism. Thanks to these heuristics, only a small subset of relevant intermediate nodes will forward the messages in given geographical areas or in given directions.

The remainder of the paper is organized as follows. Section II brings to the fore the main issues that must be addressed in order to discover and to deliver some services in DMANETs efficiently. Section III presents the assumptions on which protocol OLFServ is based, the detailed specifications of the self-pruning heuristics it implements, and how it works on an example. Section IV presents some simulations results we obtained for OLFServ. Research works dealing with routing protocols in DMANETs are presented in Section V. Section VI summarizes our contribution.

II. Service-oriented opportunistic computing: main issues

Service provisioning in DMANETs using opportunistic communications is an emerging computing paradigm that has

been recently qualified as opportunistic computing [2]. This paradigm introduces new issues regarding both the opportunistic routing protocols and the middleware platforms: the routing protocols must be suited to the discovery and the delivery of pervasive services, and the platforms must support distributed computing tasks in environments where disconnections and network partitions are the rule. This section presents these new main issues.

1. Broadcast storm issue in the discovery process: No device is stable enough, or accessible permanently, to act as a service registry. Each mobile client should therefore be responsible for maintaining its own perception of the services offered in the network, and for discovering them reactively by processing the unsolicited service advertisements broadcast by service providers, and/or proactively by broadcasting service discovery requests in the network and by processing the advertisements returned in response by providers. In such a distributed discovery process, all mobile nodes receiving an advertisement or a discovery request are not expected to rebroadcast this message systematically, because if they do so, they will generate too much network traffic and could even lead to network congestion. To cope with this problem, some heuristics must be devised in order to reduce the number of broadcasters and to broadcast the messages asynchronously.

2. Forwarding problem in the invocation process: In opportunistic networks, no end-to-end routes are maintained between a client and a provider by an underlying dynamic routing protocol such as AODV or OLSR. A priori, a node does not know which is the best next forwarder among its neighbors for reaching the destination. In order not to forward a message in a blind way, some solutions have been proposed in several related works [3], [4], [5], [6], [7], [8]. These solutions mainly rely on the computation of a delivery probability based on contextual properties [7], on an history of contacts [5], or on both [8], [4]. Nevertheless, these solutions often consider that nodes move following regular mobility patterns, and that their future (direct or indirect) encounters can be predicted. Computing such an history and a prediction is a tricky problem, especially in an environment where people often stroll and move randomly such as in a city, questioning de facto such assumptions. Moreover, during the invocation process, such probabilities must be computed twice: once in order to deliver the invocation request to the service provider, and another time to deliver the response to the client. Indeed, the client and the intermediate nodes are likely to move during this process, the forwarding path followed by the response can therefore be different from that taken by the request.

3. Responsiveness: Opportunistic communications introduce a certain delay in the service discovery and invocation processes. Although client applications must be able to tolerate this delay and to deal with extended disconnection periods, it is suitable to devise solutions that provide end-users with a certain quality of service in term of responsiveness. Consequently, the protocol should not implement a purely periodic and proactive message emission, but instead should adopt a reactive behavior as far as possible. It should be sensitive to events such as the arrival of a new neighbor, the reception of a new message or the location changes.

4. Message redundancy management: In order to increase the message delivery ratio and to reduce the delivery time, several copies of a message are usually generated in the network. In order not to process a request or a response several times, such a redundancy should be hidden from both the client applications and the software services, and be controlled by the routing protocol itself. Moreover, a mobile node should stop forwarding a request for which it has already received a response.

5. Spatial and temporal propagation control of messages: Based on the ”store, carry and forward” principle, messages can disseminate network-wide. However, some services can be relevant only in a given part of the network. In this context, it seems to be suitable to circumscribe the dissemination of the messages geographically, as well as to limit their dissemination in the network by defining a life time and a maximum number of hops.

6. Service selection issues: A selection process may precede the invocation, when the opportunity is given to the client application to choose among several service providers. Thus, it could be interesting to select a provider according to its location, and to transparently select another one among a set of relevant ones when the current provider becomes inaccessible.

The remainder of the paper describes a location-aware forwarding protocol that addresses the first five issues. In previous works, we proposed two different solutions for the last issue: one that relies on a content-based service invocation [9] and another one that relies on a dynamic and transparent update of the service references [1]. These two solutions have been implemented in the service management layer of our middleware platform.

III. The OLFServ protocol

A. Assumptions

The OLFServ protocol relies on 3 main assumptions:

1) Both mobile hosts and fixed infostations are aware of their geographical location and able to compare their location with that of another host. Mobile hosts are expected to indicate their destination/direction if they know them.
2) Mobile hosts are able to perceive their one-hop neighborhood. This neighborhood is obtained using specific messages (beacons) sent by each node periodically.
3) Each mobile host is able to temporarily store the messages it receives, and can associate to each of them some pieces of information, and especially the IDs of the nodes that are known to have received them.

B. Overview of the protocol

a) Heurisitics: OLFServ is an event-driven protocol that implements self-pruning heuristics. The originality of this protocol resides in the adaptation of several well known heuristics to the context of service provisioning in DMANETs, and their combination in a coherent platform. The main implemented heuristics are the following:

Contention resolution in message forwarding: Like DFCN (Delayed Flooding with Cumulative Neighborhood) [10],

which proposes a bandwith-efficient broadcast algorithm for MANETs, OLFServ introduces a backoff mechanism in order to avoid message collisions at message reforwarding time. From this point of view, a node is expected to compute a forwarding delay for each message it receives, and to forward messages when their delay expires. Moreover, a node will abstain from forwarding a message if it perceives that all of its neighbors have already received it (the message was forwarded by at least one of its neighbors before it forwards the message itself, and its one-hop neighborhood is a subset of the set of nodes that are expected to have received the message yet). In addition, in OLFServ, this forwarding delay has two components: one that is inversely proportional to the distance from the last forwarder and another one that is a random value (used in the backoff mechanism). Therefore, only the farther nodes are likely to forward a message, thus improving the geographical propagation of messages while reducing the number of emissions.

Geographically-driven message forwarding: At each step, a message will be forwarded only by the nodes closer to the destination.

Content-based message forwarding: Mobile nodes can establish some correlations between the discovery requests and the advertisements, as well as between the invocation requests and the responses. Thanks to this heuristic, a mobile node receiving an invocation request is expected to send back to the client the response it previously stored for this request instead of forwarding it towards its destination, obviously if this one is still valid.

Source routing forwarding: Nodes can estimate if a message was forwarded quickly (i.e., if a message was relayed following an end-to-end path), and to perform source routing if so. OLFServ is thus able to exploit end-to-end routes when they exist, reducing the propagation time and the number of message copies. If the source routing failed, because an intermediate node becomes unreachable, the selective and controlled broadcast is used. These last two heuristics aims at improving the quality of service offered to end-users in term of responsiveness.

b) Events: In OLFServ, five kinds of events are considered: 1) the reception of a message, 2) the expiration of the forwarding delay associated with a message, 3) the location changes, 4) the arrival of a new neighbor, 5) and the failure in the source routing process.

The first and the last events induce a reactive behavior of the protocol regarding the message forwarding, whereas the other events induce a proactive behavior.

Before giving a detailed specification of the OLFServ protocol, let's see how the above-mentioned heuristics operate in both the service discovery process and the service invocation phase. From this point of view, let us consider the disconnected MANET depicted in Figure 1, which will, for the sake of illustration, be composed of a set of mobile devices carried by pedestrians and a fixed infostation I that offers a service that is relevant only in the geographical area represented by the dotted rectangle. Moreover, let's suppose that one of these mobile hosts, namely node C, is interested in the service proposed by I. The network, which is currently composed of the six distinct communication islands shown in Figure 1, is expected to evolve in an unpredictable manner according to the nodes' mobility. Nevertheless, in order to illustrate our purposes, we will consider subsequently that node C and node N_6 follow the materialized paths so as to reach different destinations at times t_1, t_2, t_3 and t_4.

c) Service discovery: The invocation of a remote service is conditioned by the preliminary discovery of this service. Consequently, in order to call the service offered by I, node C must discover this service. For the sake of illustration, let us consider that infostation I has injected in the network an advertisement A including its location, the geographical area where the service can be accessed, a date of emission, a lifetime, a maximum number of hops this advertisement is allowed to make, and the set of nodes that are expected to receive this advertisement (i.e. I, N_1, N_2, N_3, N_4 and N_5). Nodes N_1, N_2, N_3, N_4 and N_5, which will receive message A first, will store this message locally and will compute a forwarding delay in order not to rebroadcast message A simultaneously.

The coverage radio area of a node is partitioned in several concentric rings. The forwarding delay algorithm (see Algorithm 2) allows mobile nodes located approximately at the same distance (i.e., in the same ring) from the last relay (or from the initial sender) to compute a forwarding delay in a same range of values. In the part of the network depicted in Figure 1, nodes N_1, N_2 and N_3 will thus compute a forwarding delay in a same range of values. This delay will be less than the one computed by N_4, which itself will be less than the one computed by N_5. Moreover, a node perceiving that all of its neighbors have already received the message it plans to forward will cancel its forwarding process, and will trigger it when it is notified of the arrival of a new node in its vicinity. Thus in our scenario, node N_5 will not forward advertisement A, because this advertisement is rebroadcast by node N_4 first. If we consider that all the nodes have the same communication range of radius R, we can deduce, based on geometric properties, that, in favorable conditions, only 3 nodes will forward advertisement A the first time [11]. Consequently at hop n, in favorable conditions the number of forwarders will be $3 \times n$, and in the worst conditions (i.e., when the selected forwarders moved before forwarding their message, and become out of reach of each other), the number of forwarders will be $\sum_{i=0}^{n} 6^n$. This property is thus independent of the density of the network.

By implementing the "store, carry and forward" principle and by exploiting the nodes' mobility and contact opportunities, advertisement A will be propagated in the whole area specified by the infostation, and only in this area. Indeed, the self-pruning heuristics implemented in our protocol prevent mobile devices from forwarding messages outside the area specified in the headers of these ones. For instance, node N_6 that left the island of infostation I at time t_1 and joined that of client C at time t_2 will broadcast advertisement A in this new island. This message will be then broadcast by the other nodes of this island whether it is still valid (i.e. the number of

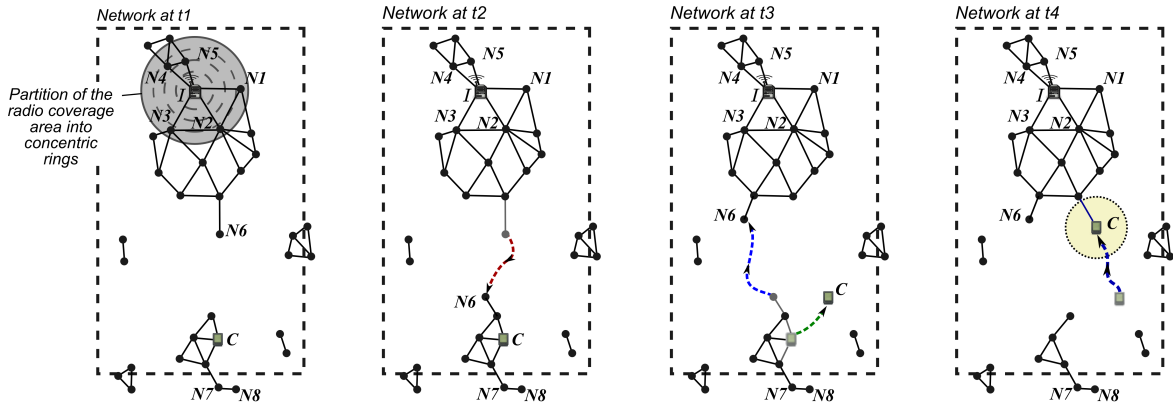

Figure 1. Opportunistic communication in a DMANET with OLFServ.

hops is greater than zero and the lifetime has not expired yet), except by node N_7 because it is outside the area specified by infostation *I*. Thus, node N_8 will not receive message *A*.

d) Service invocation: After discovering the service offered by infostation *I*, client node *C* can invoke this service by sending an invocation request including namely the ID of the infostation, the location of this one, and its own location. Let us also consider that client *C* knows its speed and its direction and that it has also included them in the request it sent, thus allowing to compute with a better accuracy the area where it is expected to be when it will receive the response. Indeed, when the speed and the direction (or the destination) are unknown, the "expected area" is a circle whose center is the current position of the client and whose radius is proportional to a predefined speed (of about 2 m/s for pedestrians) and to the time expected for the response delivery (this time is estimated from the request delivery time). The notion of "expected area" was introduced in [12]. In contrast, when the speed and the direction are known, the "expected area" is a circle centered on the position computed from the speed and the direction indicated by the client, and whose radius is proportional to the inaccuracies of both the speed and the forwarding time (see the dotted circle in Figure 1).

The request sent by *C* will be received by intermediate nodes and broadcast by these ones towards infostation *I* following a forwarding scheme that is quite similar to the discovery forwarding scheme presented previously. The difference between these two schemes resides in the number of nodes that will rebroadcast the messages. Indeed, since the invocation process is usually achieved using a unicast communication scheme, we have introduced additional self-pruning heuristics in comparison to the service discovery process in order that only the nodes closer to the destination than the previous hop can forward the message towards the destination. Thus, the area where the message is forwarded is progressively refined until reaching the destination, and the number of messages that are replicated in the network is reduced while having a good message delivery ratio. A node, receiving a message from a neighbor node closer to the message's recipient than itself, will store the message locally and will forward this message later when it becomes closer to the recipient than this neighbor. For example N_7 and N_8 will not broadcast the request sent by node *C* at time t_2 because they are farther than *C* from infostation *I*. This invocation request will be received by node N_6 at time $t_2+\Delta t$. If N_6 joins the island of infostation *I* at time t_3 as shown in Figure 1, it will broadcast this request in this island because it will discover new neighbors that have not received this message yet. These neighbors will then forward this request towards infostation *I*.

If client *C* has specified its location, its speed and its possible direction of movement, OLFServ can estimate the area where *C* is expected to be when it should receive the response from *I*. So when the response is returned, this area is specified in a header of this message. The response will be then routed towards this "expected area" using a forwarding scheme comparable to that used for the invocation. When the message has reached the "expected area", it will be disseminated in this area following a broadcast scheme comparable to that used for service discovery. This technique is used since the position of the client cannot be computed with a good accuracy due to the delay induced by opportunistic communication. When a mobile device receives a response for an invocation it has previously stored locally, it stops forwarding this request in the network. In our scenario (Figure 1) the response will be routed towards node *C* by nodes N_2, N_3 or N_1 because they are closer to the "expected area" than *I*. Moreover, if an invocation request reaches the provider within a short amount of time (i.e., if a end-to-end route is very likely to exist between the client and the provider), OLFServ tries to follow the same route by applying source routing. If the source routing process failed because an intermediate node has moved, then the node perform a broadcast towards the destination as mentioned before. Finally, if a node stored previously a response for the request sent by client *C*, it will send back this response (if it is still valid) instead of forwarding the request towards infostation *I*. For instance, N_2 can return to client *C* the copy of the response it holds locally, instead of forwarding the request to *I*. Thus, the number of message roaming in the network is reduced and the service invocation responsiveness is improved. The same process is applied when a client is looking for a service: an intermediate node can send back to the client the advertisement it holds locally that "matches" the service discovery request sent by the client.

Algorithm 1 Reaction on message reception.

Data: m: the incoming message ; t: the current time
$\mathcal{C}$, Δ, $\mathcal{K}_m$, $\mathcal{N}$

1: **if** ($m \in \Delta$ & $\mathcal{N} \subseteq \mathcal{K}_m$)
2: **then** $\Delta \leftarrow \Delta - \{p\}$
3: **else if** ($\exists\ p \in \mathcal{C}$ / p is response for m & p[lifetime] > $t - p$[date] & p[hops] > 0)
4: **then** compute forwarding delay for p
5: $\Delta \leftarrow \Delta \cup \{p\}$
6: **else if** ($\exists\ k \in \mathcal{C}$ / m is response for k)
7: **then** $\mathcal{C} \leftarrow \mathcal{C} - \{k\}$
8: **if** ($k \in \Delta$) **then** $\Delta \leftarrow \Delta - \{k\}$
9: **if** ($k \in Q_s$) **then** $Q_s \leftarrow Q_s - \{k\}$
10: **else** $Q_b \leftarrow Q_b - \{k\}$
11: **if** ($t - k$[reception_date] < ε) **then** m[source_routing] ← $k[\mathcal{L}_{relay}]$
12: **if** (m[lifetime] > $t - m$[date] & m[hops] > 0)
13: **then** $\mathcal{C} \leftarrow \mathcal{C} \cup \{m\}$
14: m[reception_date] ← t
15: $\mathcal{K}_m \leftarrow \mathcal{K}_m \cup \{m[\mathcal{K}_m]\}$
16: **if** ($\mathcal{N} \not\subseteq \mathcal{K}_m$) **then** compute forwarding delay for m
17: $\Delta \leftarrow \Delta \cup \{m\}$

Algorithm 2 Computation of the forwarding delay.

Data: m: the incoming message ; rs: the ring size
$\mathcal{R}$: the ring number ; δ: the default forwarding period
$\mathcal{W}$: the wireless communication range
Output: δ_m: the forwarding delay for message m
1: $\mathcal{R} \leftarrow$ f loor(($\mathcal{W}$ − distance($\mathcal{L}$; $m[\mathcal{L}_{relay}]$)) / rs)
2: $\delta_m \leftarrow$ min(δ ; α × random($\mathcal{R} \times rs$; ($\mathcal{R}$+1) × rs))

Algorithm 3 Expiration of the forwarding delay.

Data: m: the message ; t: the current time
$\mathcal{C}$, $\mathcal{N}$, $\mathcal{K}_m$, Q_b , Q_s
1: **if** ($\mathcal{N} - \mathcal{K}_m \neq \emptyset$ & in m[area] & m[lifetime] > $t - m$[date] & m[hops] > 0)
2: **then if** (m[recipient] ≠"*") **then** $d_{this \to recipient} \leftarrow$ distance($\mathcal{L}$; $m[\mathcal{L}_{recipient}]$)
4: $d_{relay \to recipient} \leftarrow$ distance($\mathcal{L}_{relay}$; $m[\mathcal{L}_{recipient}]$)
5: **if** ($d_{this \to recipient} \leq d_{relay \to recipient}$) **then**
6: m[area] ← ($m[\mathcal{L}_{recipient}]$, $d_{this \to recipient}$)
7: $m[\mathcal{K}_m] \leftarrow m[\mathcal{K}_m] \cup \mathcal{K}_m$
8: $m[\mathcal{L}_{relay}] \leftarrow \mathcal{L}$
9: m[nb hops] ← m[nb hops]−1
10: **if** ($t - m$[date] < ε) **then** $Q_s \leftarrow Q_s \cup \{m\}$
11: $\Delta \leftarrow \Delta - \{m\}$
12: **else** $Q_b \leftarrow Q_b \cup \{m\}$
13: $\Delta \leftarrow \Delta - \{m\}$
14: **else** $m[\mathcal{K}_m] \leftarrow m[\mathcal{K}_m] \cup \mathcal{K}_m$
15: $m[\mathcal{L}_{relay}] \leftarrow \mathcal{L}$
16: m[nb hops] ← m[nb hops]−1
17: $Q_b \leftarrow Q_b \cup \{m\}$
18: $\Delta \leftarrow \Delta - \{m\}$

C. Specification of the protocol

The remainder of this section presents how OLFServ reacts when one of the above-mentioned events occurs.

1) Notations: The location of a node is subsequently identified as $\mathcal{L}$, the one of the last relay as $\mathcal{L}_{relay}$ and the one of the destination as $\mathcal{L}_{recipient}$. The one-hop neighborhood of a node is referred to as $\mathcal{N}$. The local cache of a node is identified as $\mathcal{C}$. $\mathcal{Q}_s$ and $\mathcal{Q}_b$ are outgoing queues for the messages that must be sent using source routing techniques and for the messages that must be broadcast respectively. $\mathcal{K}_m$ refers to the set of nodes that are known to have received message m. Δ is the set of messages that must be forwarded and for which a forwarding delay has been computed. Finally, the messages headers can include several properties (the location of the recipient, the location of the sender, a date of emission, a lifetime, a maximum allowed number of hops, the geographical area where the message can be disseminated, etc.). A given property of a message m is identified as $m[property]$.

2) Message reception: When receiving a message m, Algorithm 1 is applied. First, if a node receives from one of its neighbors a message it plans to forward, it checks if all of its neighbors have received this message. If so, it cancels its forwarding process. If the node has in its cache an advertisement p for the service discovery request m (or a response p for the invocation request m) then the node is expected to forward p if this one is still valid. A forwarding delay is computed for message p, and p is put in the set of messages that must be sent. Otherwise, if m is a response for an invocation request k (or if m is an advertisement for a discovery request k), k is removed from the local cache in order not to be forwarded later, as well as from the set of messages that must be forwarded. If message m is still valid and if the number of hops is greater than 0, message m is put in the local cache, and the set $\mathcal{K}_m$ is updated (i.e., the set of nodes that are known to have received message m yet). Message m is put in the set of messages that must be forwarded and a forwarding delay is computed for m. When the forwarding delay δ_m expires, Algorithm 3 will be applied.

3) Computation and expiration of the forwarding delay: Each mobile device computes a forwarding delay for each message it receives. This delay prevents close devices from forwarding messages simultaneously. As mentionned before, in OLFServ the forwarding delay has both a random component and a component that is inversely proportional to the distance from the previous relay. So as to compute this forwarding delay, the wireless communication range of each device has been divided in several rings (see Figure 1), so that the delays computed by hosts in ring i are greater than those computed by hosts in ring $i+1$. The mobile hosts of a given ring are considered as equivalent regarding the spatial propagation of messages. The algorithm used to compute the forwarding delay is described in Algorithm 2. This algorithm has mainly three parameters: the wireless communication range ($\mathcal{W}$), the ring size (rs) and α. This last parameter has been introduced in order to define a relevant delay δ_m: the delay in the largest ring is of the order of a few milliseconds, while in the smallest ring it is of the order of a few seconds typically.

When the forwarding delay of a given message has expired, Algorithm 3 is applied. If there are new nodes in the one-hop neighborhood, if the client is in the area where the message can be disseminated, if the message is still valid and if the message has next hops, the message is then considered as being forwardable. The headers of the message are then updated. If the destination is known, the area where the message can be propagated is updated in order to refine this area progressively until reaching the destination. Moreover if the destination is known, the mobile device checks whether it is closed to the destination than the last forwarder, and if so, it updates the number of hops, the location of the last forwarder with its own location and the set of nodes that have already received the message, and puts the message in the outgoing message queue. If the message has expired or if the number of hops equals to 0, the message is removed from the local cache.

Algorithm 4 Location changes.

Data: m: the message that must be forwarded ; t: the current time
$\mathcal{C}$, $\mathcal{K}_m$

1: **if** (m[lifetime] > $t - m$[date] & m[hops] > 0 & $\mathcal{N} \not\subseteq \mathcal{K}_m$)
2: **then if** (m[type] = response & $\mathcal{L}$ in m[expected area]) **then** m[recipient] ← "*"
3: compute forwarding delay for m

Algorithm 5 Detection of new neighbor nodes.

Data: n: the new neighbor ; t: the current time
$\mathcal{C}$, $\mathcal{N}$

1: $\mathcal{N} \leftarrow \mathcal{N} \cup \{n\}$
2: **for all** $m \in \mathcal{C}$
3: **if** (m[lifetime] > $t - m$[date] & m[hops] > 0)
4: **then if** ($n \notin \mathcal{K}_m$ & in m[area]) **then** compute forwarding delay for m
5: **else** $\mathcal{C} \leftarrow \mathcal{C} - \{m\}$

4) Location changes: When reaching a given location, a mobile host can trigger the forwarding of some messages. For instance, a mobile host that was far from the recipient of a message it received can trigger the emission of this message when it is at a given distance from the recipient. Similarly, when entering the area where a client is likely to be receiving its service response, a mobile host, acting as an intermediate node, can both update the message headers in order that this message can be broadcast in this whole area and trigger its emission. When the mobile host has reached a given location, Algorithm 4 is executed. We change the status of the response in order that it is broadcast by the node in the whole area specified by the provider. And for each message when we become closer to the destination than the previous node (the node from which we have received the message), we trigger a message emission.

5) New neighbor detection: When a new neighbor node is discovered, the mobile host computes a forwarding delay for all the messages that are still valid, that have next hops, if the new neighbor is not in the the list of nodes that have already received the message and if the mobile host is in the area where the message can be propagated. A new forwarding delay is computed in order to prevent the emission of the same messages by different nodes that simultaneously discover the new neighbor node in their one-hop neighborhood.

IV. Experiments and results

In order to evaluate our protocol, we conducted a series of simulations using the Madhoc simulator (http://agamemnon.uni.lu/~lhogie/madhoc), a metropolitan ad hoc network simulator that features the components required for both realistic and large-scale simulations, as well as the tools essential to an effective monitoring of the simulated applications. This simulator, which is written in Java, allows us to run our middleware platform on it. In the current scenarios we focus on, service providers are fixed infostations deployed in a city, while clients are devices carried by humans.

A. Experiments and simulation setup

The simulation environment we consider is an open area of about 1 km^2. Four infostations offering two different services are deployed in this environment. These services can be discovered and invoked in a circular area of a radius of 200 m. The first service delivers the day's weather forecast, while the second provides an access to a "yellow page" service, which can be invoked by nomadic people in order to find restaurants, shops, etc. Mobile clients are thus expected to submit the same request to the first service and different ones to the second service. In our simulations, we have considered successively 50, 100, 500 and 1000 pedestrians carrying a PDA equipped with both a Wi-Fi interface and a GPS receiver. The communication range of both mobile devices and infostations varies from 60 to 80 m. Some of the pedestrians move randomly, while others follow predefined paths. Each pedestrian moves at a speed between 0.5 and 2 m/s. In our simulations, 30 % of the mobile devices act as clients of the above-mentioned services, whereas the others only act as intermediate nodes. The service providers are expected to broadcast service advertisements every 30 seconds when mobile devices are in their vicinity. After discovering the services they are looking for, the clients invoke these services every 3 minutes. In our experiments, we have assigned to all the messages a lifetime of 5 minutes and a maximum number of hops of 8. We present below the results we obtained for OLFServ in these various configurations, and we compare OLFServ with the Epidemic Routing Protocol (EPR) defined by Vahdat and Becker [13]. The objective of these experiments was to measure the ability to satisfy the client service discovery and invocation efficiently with a small number of message copies.

B. Results

Figure 2 and Figure 3 present the simulation results for the two kinds of services considered (the "weather forecast" service S1 and the "yellow pages" service S2). Figure 2 gives the average number of emissions for a service advertisement (for S1 and S2) with OLFServ and with EPR. One can observe that the number of emissions increases drastically with EPR, while it remains relatively constant with OLFServ. Indeed, in EPR when two hosts come into communication range of one another, they exchange their summary vectors to determine which messages stored remotely have not been seen by the local host. In turn, each host then requests copies of messages that it has not seen yet. In contrast in OLFServ, service advertisements are broadcast and not sent using a unicast communication model. Moreover, only a subset of the neighbor nodes are expected to rebroadcast these advertisements in turn. For S2, the number of emissions of a given service invocation request is less than the half of the number of emissions of service advertisements (see Figure 3). These results are consistent with those expected. Indeed, the invocation requests are broadcast only by the nodes closer to the destination at each hop. It must be noticed that the number of emissions of invocation requests for S1 is less than that for S2. Again, the results are consistent with those expected: all the clients interested in the "weather forecast" service submit the same request, and obtain in return the same response during the simulation. The mobile nodes that have stored a request and the associated response are able to establish a correlation between these messages, and are expected to send back to the client the stored response when they receive a new similar request. The number of requests for S1 decreases according to

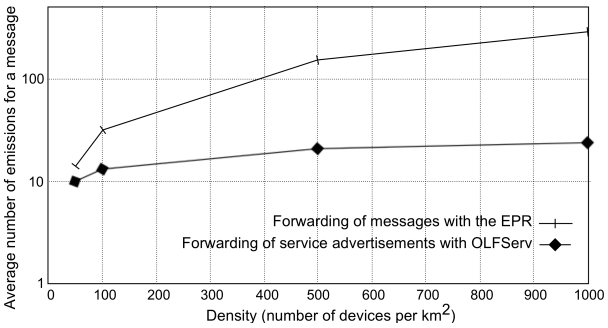

Figure 2. Service advertisement with OLFServ and EPR.

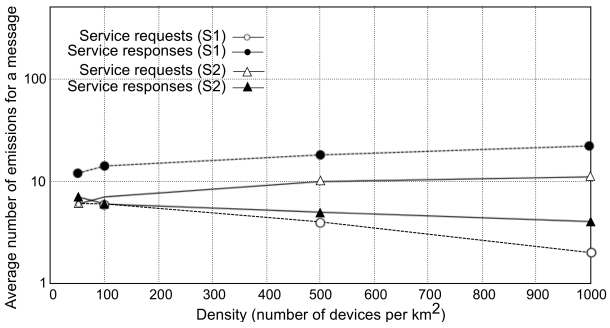

Figure 3. Service invocation with OLFServ.

the number of clients. Such a phenomenon can be explained by the fact that a request is not forwarded by a node towards the destination if this node has already obtained the response associated with this request. This correlation techniques is further detailed in [1]. Finally, it must be noticed that the mobility of nodes between the successive invocations does not allow benefiting from source routing when forwarding a request towards a provider. Nevertheless, source routing has proved its efficiency in the forwarding of the responses, as shown in Figure 2. Thus, the number of messages sent in the network is reduced while offering a better service provision (see Table I).

As shown in Table I, the number of clients that have discovered the service they are looking for is greater with EPR than with OLFServ. Nevertheless, the invocation success ratio with EPR is less than with OLFServ. Indeed, with OLFServ messages are routed only in the areas where the services can be discovered and invoked, whereas with EPR, messages are routed in the whole simulation area. Consequently, with EPR, services can sometimes be discovered by the clients, but not invoked successfully due to the mobility of intermediate nodes, to the periodic exchange of messages (every 20 seconds) and to the fixed number of hops. In contrast, with OLFServ, messages are forwarded few milliseconds after their reception instead of being forwarded periodically. OLFServ thus offers a good responsiveness and delivery ratio while producing a lower network load.

V. Related work

Our work on OLFServ is related to works on broadcast protocols [14], [15]. Indeed, some techniques that aim at reducing the number of message forwarders are adapted or integrated to the specific context of service provision in opportunistic networks.

However, the research works that follow the same objectives as OLFServ are mainly led in the opportunistic networking and/or delay/disrupted tolerant networking domain.

One of the first protocol in this domain is the Epidemic Routing Protocol [13], which can in a way be assimilated to a simple flooding, not suitable for environments with high density regions, since it would generate too much network traffic and could even lead to network congestion. This drawback is addressed by protocols implementing methods aiming to assess the capability of a neighbor node to contribute to the delivery of a given message. These methods usually use a probabilistic metric, often called delivery predictability, that reflects how a neighbor node will be able to deliver a message to its final recipient [16]. Before forwarding (or sending) a message, a mobile host asks its neighbors to infer their own delivery probability for the considered message, and then compares the probabilities returned by its neighbors and chooses the best next carrier(s) among them. In CAR [7] and GeOpps [3], the delivery probabilities are computed using both utility functions and Kalman filter prediction techniques. CAR assumes an underlying MANET routing protocol that connects together nodes in the same MANET cloud. To reach nodes outside the cloud, a sender looks for the node in its current cloud with the highest probability of delivering the message successfully to the destination. GeOpps, which is a geographical delay-tolerant routing algorithm, exploits the pieces of information provided by the vehicles' navigation system in order to route the messages to a specific location. Like CAR, HiBOp [8] also exploits context information in order to compute delivery probabilities. However, HiBOp can be perceived as being more general than CAR since it does not require an underlying routing protocol, and because it is also able to exploit context for those destinations that nodes do not know. HiBOp exploits history information in order to improve the delivery probability accuracy, and does not make predictions as CAR. Propicman [4], as for it, also exploits context information and uses the probability of nodes to meet the destination, and infers from it the delivery probability, but in a different way. When a node wants to send a message to another node, it sends to its neighbor nodes the information it knows about the destination. Based on this information, the neighbor nodes compute their delivery probability and return it. In Prophet [5], the selection of the best neighbor node is based on how frequently a node encounters another. When two nodes meet, they exchange their summary vectors, which contain their delivery predictability information. If two nodes do not meet for a while, the delivery predictability decreaces. When a node wants to send a message to another node, it will look for the neighbor node that has the highest amount of time encountering the destination, meaning that has the highest delivery predictability to the destination. Furthermore, this property is transitive. Unlike OLFServ, most of the above-mentioned protocols rely on an history of contacts and a prediction of encounters in order to select the best next carrier(s). Computing such an history and a prediction is a tricky problem, especially in environments composed of numerous

	EPR(50)	*EPR(100)*	*EPR(500)*	*EPR(1000)*	*OLFServ(50)*	*OLFServ(100)*	*OLFServ(500)*	*OLFServ(1000)*
Number of clients that have discovered a provider	12	25	147	294	10	24	142	290
Avg delay of successful invocations to service S1	120 s	100 s	60 s	40 s	1.02 s	0.58 s	0.43 s	0.42 s
Avg delay of successful invocations to service S2	120 s	100 s	60 s	40 s	3.32 s	2.84 s	2.43 s	2.42 s
Average ratio of sucessful invocations	0.78	0.84	0.92	0.96	1	1	1	1

Table I
SIMULATION RESULTS FOR SERVICE DISCOVERY AND INVOCATION.

mobile devices that move following irregular patterns, such as those hold by pedestrians in a city. Although they implement various strategies aiming to select the next best carriers(s) to deliver a given message, the above-mentioned protocols are not suited to service discovery. Indeed, they implement neither self-pruning heuristics making it possible for mobile nodes to decide if they should rebroadcast a message according to their neighborhood perception, nor methods allowing to designate which subset of neighbor nodes must rebroadcast a message. If used to broadcast service advertisements or service discovery requests network-wide, they will probably induce a storm of messages and perhaps a network congestion.

Geographic routing protocols, such as GeRaf [17], LAR [12] and Dream [18], propose forwarding techniques similar to those implemented in OLFServ. Once a node has a message to send, it broadcasts it while specifying its own location and the location of the destination. All the nodes in the coverage area will receive this message and will assess their own capability to act as a relay, based on how close they are to the destination. Dream and LAR also propose some solutions in order to improve the message delivery in MANETs. For instance, based on location information, they can compute the area where the mobile clients are expected to be when they receive their messages. Nevertheless, on contrary to OLFServ, these protocols do not implement the "store, carry and forward" principle and therefore are not suitable for disconnected MANETs.

VI. CONCLUSION AND FUTURE WORKS

Opportunistic networking is a promising but challenging solution to provide nomadic people with a wide access to pervasive services without resorting to licensed frequency bands. In this paper, we have proposed a new opportunistic and location-based forwarding protocol, called OLFServ, suited for service provision in disconnected, partially connected or intermittently connected MANETs. This protocol implements several self-pruning heuristics to efficiently control the dissemination of service advertisements and service discovery requests, as well as to perform a geographic and source-based routing. OLFServ allows a cost effective delivery of pervasive services in networks composed of numerous mobile devices moving either following predefined path or randomly with respect to delivery delay, delivery ratio and number of emissions (reflecting the network throughput).

In the future, we would like to evaluate our middleware platform and our protocol in real conditions by porting them on mobile devices such as Android smartphones and by leading experimental field tests.

REFERENCES

[1] S. Ben Sassi and N. Le Sommer, "Towards an Opportunistic and Location-Aware Service Provision in Disconnected Mobile Ad Hoc Networks," in *Proc. of the Int. Conference on Mobile Wireless Middleware, Operating Systems, and Applications*, vol. 7 of *LNICST*, pp. 396–406, Springer-Verlag, 2009.

[2] M. Conti, S. Giordano, M. May, and A. Passarella, "From Opportunistic Networks to Opportunistic Computing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 126–139, 2010.

[3] I. Leontiadis and C. Mascolo, "GeOpps: Geographical Opportunistic Routing for Vehicular Networks," in *Proc. of the Int. Symposium on a World of Wireless, Mobile and Multimedia Networks, AOC Workshop*, IEEE CS, 2007.

[4] H. A. Nguyen, S. Giordano, and A. Puiatti, "Probabilistic Routing Protocol for Intermittently Connected Mobile Ad Hoc Network (PROPICMAN)," in *Proc. of the Int. Symposium on a World of Wireless, Mobile and Multimedia Networks, AOC Workshop*, IEEE CS, 2007.

[5] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic Routing in Intermittently Connected Networks," in *Proc. of the Int. Workshop on Service Assurance with Partial and Intermittent Resources*, vol. 3126 of *LNCS*, pp. 239–254, Springer Verlag, 2004.

[6] F. Guidec and Y. Mahéo, "Opportunistic Content-Based Dissemination in Disconnected Mobile Ad Hoc Networks," in *Proc. of the Int. Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pp. 49–54, IEEE CS, 2007.

[7] M. Musolesi and C. Mascolo, "CAR: Context-Aware Adaptive Routing for Delay Tolerant Mobile Networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 2, pp. 246–260, 2009.

[8] C. Boldrini, M. Conti, I. Iacopini, and A. Passarella, "HiBOp: a History-Based Routing Protocol for Opportunistic Networks," in *Proc. of the Int. Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1–12, IEEE CS, 2007.

[9] Y. Mahéo and R. Said, "Service Invocation over Content-Based Communication in Disconnected Mobile Ad Hoc Networks," in *Proc. of the Int. Conference on Advanced Information Networking and Applications*, pp. 503–510, IEEE CS, 2010.

[10] L. Hogie, P. Bouvry, F. Guinand, G. Danoy, and E. Alba, "A Bandwidth-Efficient Broadcasting Protocol for Mobile Multi-hop Ad hoc Networks," in *Proc. of the Int. Conference on Networking*, IEEE CS, 2006.

[11] X. Liu, X. Jia, H. Liu, and L. Feng, "A Location-Aided Flooding Protocol for Wireless Ad Hoc Networks," in *Proc. of the Int. Conference on Mobile Ad-Hoc and Sensor Networks*, vol. 4864 of *LNCS*, pp. 302–313, Springer-Verlag, 2007.

[12] Y.-B. Ko and N. H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," *Wireless Networks*, vol. 6, pp. 307–321, 2000.

[13] A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks," tech. rep., Duke University, 2000.

[14] B. Williams and T. Camp, "Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks," in *Proc. of the Int. Symposium on Mobile Ad Hoc Networking and Computing*, pp. 194–205, ACM, 2002.

[15] I. Stojmenovic and J. Wu, *Mobile Ad Hoc Networking*, ch. 7: Broadcasting and Activity-Scheduling in Ad Hoc Networks, pp. 205–229. Wiley, 2004.

[16] J. Wu and F. Dai, "Broadcasting in Ad Hoc Networks Based on Self-Pruning," in *Proc. of the Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 2240–2250, IEEE CS, 2003.

[17] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Multihop Performance," *IEEE Transactions on Mobile Computing*, vol. 2, pp. 337–348, 2003.

[18] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," in *Proc. of the Int. Conference on Mobile Computing and Networking*, pp. 76–84, ACM/IEEE CS, 1998.