Service Invocation over Content-Based Communication in Disconnected Mobile Ad Hoc Networks

Yves Mahéo and Romeo Said
Valoria, Université de Bretagne-Sud
Université Européenne de Bretagne, France

***Abstract*—This paper presents a middleware platform for the provision of services in disconnected MANETs, focusing on service invocation. This middleware exploits content-based communications (through a publish/subscribe paradigm) and employs a store-carry-and-forward approach for the network-wide opportunistic dissemination of messages. Service invocation is implemented on top of these communication features. The paper first describes the main aspects of the middleware and then details the service invocation mechanism. We namely show that the potential multiplicity of service providers can be exploited in order to increase the client satisfaction and that several network healing techniques allow the reduction of the network load. The paper ends by a description of some simulation experiments in a realistic scenario, whose results reflects the performance of the approach in terms of client satisfaction and network load.**

***Keywords*—disconnected MANETS; services; opportunistic networking; content-based networking.**

I. Introduction

The service oriented computing model has proved successful for the development and the exploitation of distributed applications. In this approach the interactions between service clients and service providers take place in two main phases: the first one is a discovery phase during which the client learns what services are available in its environment and identifies the providers that implement these services. In a second phase, the client establishes a binding with a chosen provider and remotely invokes the methods it proposes. When possible, a directory is used to gather the descriptions of the provided services. In this case, providers register their offer with the directory and clients can retrieve from it information about the services they want to use. Unlike traditional client/server, the service oriented approach introduces a late binding between the client and the provider, thus easing the support of the dynamicity of the environment. When a change occurs in the environment, for example a degradation of the connectivity or the appearance of other interesting service providers, the binding between a client and a provider can be undone and re-established with a new provider without completely stopping the whole application, or at least without rebuilding it.

The decoupling between interacting entities that characterizes the service-oriented approach makes it well suited for mobile ad hoc networks (MANETs) that are formed out of a number of mobile devices that communicate thanks to short-range wireless transmissions. The main advantage of a MANET that it can be used without deploying a specific –and sometimes costly– infrastructure. Several application domains have already been identified that mainly or even specifically target MANETs. They include information sharing during disaster relief interventions, traffic information in town centers, or tourist services in infrastructure-less sites.

MANETs actually cover a wide variety of situations depending on the density of nodes in the network, their volatility – that is the fact that they may temporarily be switched off– or their mobility scheme. The large majority of research work done so far on MANETs concerns what we call *connected* MANETs. In a connected MANET, some routing mechanism is available, implemented by each node of the network that plays the role of a router. Hence, although the topology of the network may vary, the assumption is made that, at any time, any two nodes in the network can communicate temporaneoulsy through one-hop or multi-hop transmissions. Several implementations of well-mastered routing algorithms such as OLSR [1] or AODV [2] are now available. In this context, implementing a middleware support for service-oriented applications poses problems essentially in the design of the discovery phase: no centralized directory can be implemented easily because there is no fully stable node to host this directory, and given the fact that global message diffusion is prohibitive in terms of network load, network-wide discovery is an issue. Nevertheless, the invocation phase does not generally raise particular difficulties as routing provides the ability to perform multi-hop transmissions between the client and the provider.

In this paper we address a particular class of MANETs we will refer to as *disconnected* MANETs. Disconnected MANETs show a low density and/or a high mobility of nodes and do not guarantee end-to-end connectivity. Because of their mobility, their limited radio-range and their volatility, the devices in such networks form so-called communication “islands” whose topology evolves continuously. Figure 1 illustrates a snapshot of such a disconnected MANETs: nodes represent individuals that can move inside buildings and pass from one building to another via fixed paths. Temporaneous communication is impossible between distinct islands. Moreover, an inactive node does not communicate. As a consequence, routing techniques designed for fully connected MANETs cannot be used. When it comes to apply the service-oriented approach, issues concerning service discovery are similar to those found when targeting connected MANETs. On the other hand, service invocation is not as straightforward: new means must be found to transfer the invocation request from the client to the provider and ensure

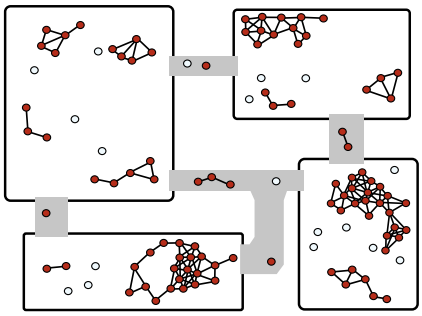


Figure 1. Illustration of a disconnected MANET

that the response arrives back to the client, even if the provider does not reside in the same communication island as the client.

Our overall objective is to build a service platform that will support the execution of service-oriented applications in disconnected MANETs. To our knowledge, this specific issue has not bean studied as such in other previous research work. The key idea of our proposal is to base the network-wide service provision on opportunistic content-based communication. Communication is said *opportunistic* because message forwarding is performed when a node gets the chance to encounter –in its radio-range– another node ready to relay the message. As in the disruption-tolerant networking approach proned by the DTNRG[1], a message can be temporarily stored in a node before being forwarded to other nodes. This store-carry-and-forward form of communication allows a message to disseminate in the network despite its fragmentation, thanks to the mobility of the nodes, as devices can carry messages when moving from an island to another. Communication is *content-based* because a node is willing to relay a message according to the content of this message, making the flow of information interest-driven rather than destination-driven [3]. The paradigm associated with this communication scheme is that of content-based publish/subscribe [4]. A node publishes a message network-wide, and this message is eventually consumed by the nodes that have subscribed for some form of content that turns out to be present in the message.

We chose to exploit this kind of content-based publish-subscribe for the two phases of service provision. Indeed content-based publish/subscribe is well adapted to service discovery insofar as a fully distributed service directory is prefered to a centralized one. Providers publish the description of the services they propose (In our case this publication is performed network-wide). Service clients subscribes for descriptors that match their needs to finally learn the existence of some services they are interested in. As far as service invocation is concerned, it must cope with the fact that the reachability of the devices is unpredictable due to the fragmentation of the disconnected MANETs. So replacing destination-based communication (in which senders specify the address of the receiver in messages) by content-based publish/subscribe is likely to be beneficial. Moreover, this eases the capacity of exploiting the fact that some services may be proposed by more than one provider in the network, thus enhancing the probability for a client to see its invocation request rapidly answered.

[1]Delay Tolerant Networking Research Group, http://www.dtnrg.org

The remaining of this paper is organized as follows. An outline of the main features of our service middleware for disconnected MANETs is detailed in Section II. We namely describe in this section the lower layer of the platform that serves as a communication support and the part of the upper layer that implements service discovery. In Section III are detailed the different mechanisms we devised for service invocation. Section IV presents simulation results of the middleware's performance. The paper ends with related work in Section V and a conclusion in Section VI.

II. Overview of the Platform

The service platform we designed is structured in two main layers. The first layer is a high-level service layer that is in charge of all service-oriented processing, enabling discovery and invocation interactions between clients and providers. The second layer is a communication layer that provides means to disseminate messages in the entire network, through a publish/subscribe interface. We sketch out in the remainder of this section the communication layer and the discovery mechanisms implemented in the service layer. Little detail will be presented on service discovery as it can be viewed as a "natural" extension of the publish/subscribe functionality. On the other hand, the invocation implementation will be more precisely described in Section III.

A. Opportunistic Content-based Communication

The lower part of our middleware consists in a communication support adapted to disconnected MANETs. It ensures a network-wide content-driven dissemination of information, despite the fragmentation of the network into isolated communication islands. It implements a model that manipulates structured pieces of information we referred to as "documents". A document is composed of two parts: its header, and its content. The header can be perceived as a collection of attributes, which can provide any kind of information about the corresponding document, such as its origin, its topic, a list of keywords, the type of its content, etc. Some storage capacity in each node is dedicated to a local cache of documents, so this node can serve as a mobile carrier for these documents while moving in the network. The dissemination model is not mere flooding. Indeed, each node in the network is associated an "interest profile", that determines the kind of information it is interested in, and thus implicitly the kinds of documents for which it is willing to serve as a mobile carrier. Therefore, uninterested nodes do not participate in the dissemination. A gossip-like communication protocol orchestrates interactions between neighboring nodes, allowing them to exchange documents according to their respective interest profiles. Interaction between mobile nodes relies on a simple scheme, whereby each node periodically broadcasts its own interest profile and a catalogue of the document headers that are currently available in its local cache. When a node discovers that one of its neighbors can provide a document it is interested in (that is, a document header that matches its own interest profile and that is not

already available in its own cache), it can request a copy of this document from this neighbor. Upon receiving one or several requests for a particular document from its neighbors, the owner of this document broadcasts it on the wireless medium, so it can be received by all requesters simultaneously. Transient contacts between mobile nodes are thus exploited opportunistically for exchanging documents between these nodes, based on their respective interest profiles, and based on the documents they can provide each other on demand. The protocol is designed so as to maximize the document delivery ratio while remaining very frugal as far as the number and the volume of messages are concerned. Further information about the protocol can be found in [5].

B. Service Discovery

The service discovery implemented in our platform directly leverages on the content-based communication facilities described above. No global service directory is maintained. Instead, by exploiting the content-based publish/subscribe paradigm, we resort to a peer to peer approach in which each provider proactively advertises its services by way of publication and clients are informed of the existence of the services they need through subscriptions.

The service discovery process begins with the description of services by the provider, using all the information needed by potential clients to select the appropriate service. A service descriptor is a document that starts with a header built from a number of freely chosen couples attribute/value that describe the non-functional characteristics of the service (including QoS or semantic information such as a category or a required security level for example). The functional interface of the service is then given in WSDL so that the client can formulate its invocation requests.

A directory module gathers all the descriptors of the services proposed by the node (if it plays the role of a provider). These descriptors are published in the network, so they disseminate until they eventually reach client nodes. The directory module maintains also a list of learned distant descriptors. Indeed, on the client side, another document called service pattern must be created to convey the wishes of a client hoping to discover a suitable service. The service pattern contains components similar to those of the service descriptor, with the possibility to include wildcards and expressions on attribute values. All the service patterns built by a client serve as subscriptions in the publish/subscribe module, and therefore form the interest profile managed by the communication layer. The subscriptions enable a content-based matching process between the client's preferences expressed in the patterns and the descriptors that are disseminated in the network. If a match is made by the publish/subscribe module, the received descriptor is passed up to the directory for service-level caching. The client can now select the descriptor cached in the directory and formulate its requests accordingly.

III. Service Invocation

A. Principle

Once a client has discovered a service, it should be able to invoke this service. More precisely, the objective for the client is to send a request to a provider of the service, and to eventually receive a response computed by this provider. In the case of stateless services –which do not require memorization of the state of the conversation between the client and the provider– the client is most of the time only interested in the response to its request without concern on which provider has processed its request.

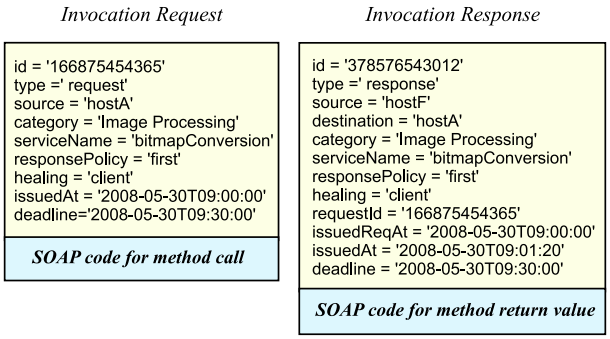


Figure 2. Examples of invocation request and response messages

Content-based Invocation: In our platform, an invocation request is therefore not addressed to a particular provider but contains only the description of the wanted service. We use the publish/subscribe facility for implementing these content-based invocations. The client formulates a request that includes a reduced version of the service description and publishes it in the network. Providers subscribe for any request matching the reduced version of the descriptors of the service they provide. Therefore a request for a service will eventually reach all the "compatible" providers. The main objective of this approach is to increase the probability for an invocation request to be (rapidly) answered, despite of the lack of connectivity inherent in disconnected MANETs. The provider's response is transferred back to the client also thanks to the publish/subscribe mechanism. The published response message contains a destination attribute, and clients subscribe for messages with a destination attribute matching their own identity. The request and the response messages are composed of a header containing a list of valued attributes and a payload consisting in a SOAP document describing respectively the method call and its return value (cf. Figure 2).

Invocation Deadline: Invocations are supposed to be to some extent tolerant to the communication delays induced by the very nature of disconnected network environments. The lifetime of an invocation is set by the client through a deadline property included in the client invocation request and also in the corresponding responses computed by the providers. The communication layer interprets this deadline as an order to no longer propagate the concerned messages: out of date requests and responses are deleted from communication caches and, consequently, are no longer relayed. The deadline property should be set to the minimum of the two following deadlines: (1) an application-related deadline*:* the application needs to receive the response before a certain date in order to perform other processing ; (2) a mobility-related deadline: the mobile node

knows that it will be leaving its current network environment at a certain time, hence it can fix the deadline accordingly.

Response Management Policy: Addressing a service instead of a unique provider increases the reactivity of the system but also possibly generates multiple responses for a single invocation request. We consider two policies for managing this multiplicity. In a first policy (called "multiple") all the responses to a client request are delivered to this client. Indeed, there are situations in which a client could see it as an advantage. Consider for example a client using a service designed for translating an English sentence into French. Multiple responses can be useful to an end-user who can choose the best answer. Similarly, it could be useful that a call to a temperature service in a sensor network let multiple responses reach the client which would average the received values. On the contrary, in the case only one response is useful, we apply a "first" policy, in which only the first received response is delivered to the client whereas the following are discarded. A train schedule information service for example–allowing requests such as 'at what time are the departures to Paris between 12:00 and 18:00'– is normally invoked with a "first" policy for any extra response received after the first one is obviously redundant. The response management policy is chosen by the client and specified in each invocation request.

B. Network healing: Redundant Response and Request Reduction

Benefiting from the fact that several providers are allowed to answer a single invocation request incurs an network extra cost in the case only one response is expected by the client (i.e. when the response management policy is set to "first"): the redundant responses will continue to propagate until their deadline is reached, consuming network bandwidth and storage capacity in the relay nodes. We devised mechanisms for reducing the number of redundant responses (and useless relayed requests) in order to minimize this extra cost. These mechanisms will be referred to as Redundant Response and Request Reduction (R4) in the remainder of this paper. It must be noticed that traditional techniques that consist in performing some form of distributed election among all the providers to ensure a unique response is not applicable because the set of providers is not known a priori and the absence of full connectivity of the network precludes learning its composition in a reasonable time. Our objective is more realistic and so less ambitious: it will not guarantee that only one response reaches the client node but rather tries to cancel the propagation of redundant messages.

Reactive Cancellation: The first R4 technique, that we call "Reactive Cancellation", is always applied. A provider subscribes for messages holding responses to requests it could himself respond to. When a provider P1 receives such a response issued by another provider P2, and if P1 has not yet himself answered the request, it gives up attempting to answer the request and rather starts to relay the response of P2. If P1 had already answered the request, it compares the issued date of the response of P2 and the one of its own response and cancels the more recent one (i.e. it stops participating in the dissemination of the more recent response by removing it from its communication cache). Moreover, any provider answering a request also cancels this request. It is indeed not useful to continue propagating the request as it would anyway travel with the corresponding response to the other providers, and the treatment of the response described above makes the request obviously redundant. Reactive Cancellation is clearly always profitable in terms of reduction of the number of transmitted messages (no extra messages are added) and it does not delay the arrival of the first response at the client.

CureAll control message

```
id = '587409889764'
type =' cureAll'
issuedAt = '2008-05-30T09:02:20'
category = 'Image Processing'
serviceName = 'bitmapConversion'
requestId = '166875454365'
deadline='2008-05-30T09:30:00'
```

CureOthers control message

```
id = '598676454654'
type =' cureOthers'
issuedAt = '2008-05-30T09:01:10'
source = 'host-F'
category = 'Image Processing'
serviceName = 'bitmapConversion'
requestId = '166875454365'
responseId = '378576543012'
deadline = '2008-05-30T09:30:00'
```

Figure 3. Examples of healing control messages

Client-initiated Cancellation: The second R4 technique, called "Client-initiated Cancellation", is more intuitive. When a client has received its first answer, it creates and starts to disseminate a control message (of type *CureAll*) that includes the request id and its deadline. The role of this message is to inform all the other nodes that they should cancel all the requests and the corresponding responses until the deadline is reached. A large number of nodes, if not all, are supposed to subscribe for control messages of this type in order to ensure their rapid dissemination. As extra messages are transmitted, this tends to increase the network load but it is worth noticing that control messages have a very small size so the elimination of pending requests and responses–whose size can be orders of magnitude larger– compensate in most cases for this augmentation. As in some particular network conditions–that can be hardly modeled but almost never encountered in practice–this healing technique may be not profitable, it is not systematically applied but only when a healing attribute containing 'client' is present in the request header.

Provider-initiated Cancellation: The third R4 technique, the "Provider-initiated Cancellation", is a form of combination of the two previous ones. It consists in starting the dissemination of cancellation messages by the providers before a response is known to have reached the client. A provider answering a request creates and starts disseminating a control message of type *CureOthers*, similar to a CurAll message but that also includes the identity *P* of the provider. This message is a command to cancel all the instances of the specified request as well as all the corresponding responses but the one issued by *P*. A network node reacts only to the first CureOther message it receives for a given request, the subsequent ones are simply discarded. A CureOther message is useful only if it is relayed by more nodes than the providers of the concerned service (again this type of control message should be relayed by almost all the nodes). The objective is that a CureOther message issued by a provider *P* reaches the other providers quicker

that the response computed by *P*, and consequently eliminates more redundant messages. However, in unfavorable network conditions, it may happen that a CureOther message reaches a provider which would have otherwise issued a response that would have been the first one to arrive at the client. Thus, this healing technique may theoretically delay the arrival of the response to the client. Experiments we conducted tend to show that this delay is negligible whereas the gain in redundant message cancellation is significant (see section IV). The provider-initiated Cancellation is applied when the healing attribute of the request header contains 'provider'.

C. Destination-based Invocation

If we consider that, in general, a client wants to use a service independently of its provider, we reckon with situations in which a specific provider is expected to be addressed when transferring the invocation request. This can occur because a provider is known to be the only one to ensure a certain quality of service or because of the very nature of the service (For example a service providing refreshed images of a given geographical spot is likely to be implemented on a single provider located in this spot.). In our platform, this is viewed as a particular case of the content-based approach described above, as the request can contain a destination attribute specified by the client to identify the wanted provider. Providers do not respond to requests that contain such an attribute if its value does not match their own identity.

In the case of destination-based invocations, the Provider-initiated Cancellation described above is (optionally) performed: as soon as the provider receives the request, it disseminates a control message that will cancel this request, for this request will be of no use elsewhere.

D. Stateful Invocations

Stateful services require that the state of the conversation between a client and a single provider be maintained during a period of time. A sequence of invocations takes place in a session, whose state is generally handled by the provider. In the kind of environment we consider, a session cannot be considered as reliable. A session loss occurs in case the provider goes down or remains unreachable for too long a period of time. We have adapted solutions for session recovery sometimes implemented in connected MANETs. The general idea is to let the session be managed by the client itself. The session state information is returned back to the client with every response, and is carried out by this client so that it can find another provider in case the original one is unreachable. The content-based invocation scheme described above allowed us to perform such a session management simply but efficiently. When a client wants to use a session, it issues an initial invocation request holding a 'session' attribute. This request is disseminated over the network, and can be answered potentially by several providers. When the first response (issued by a provider *P*) reaches the client, the session is considered opened with *P* from the client point of view, and subsequents requests in the session are addressed to *P* (with a destination attribute set to *P*). The state information maintained by *P* is appended to each response. Any provider is supposed to be able to exploit this information to re-establish on demand a consistent conversational state.

The session is considered broken when the last request deadline has passed. The client recovers the session by re-issuing this request but this time without any destination attribute, as it did with the initial session request. This request contains the state information possessed by the client so the session can be resumed with a new compatible provider. Because any compatible provider in the network can be used for session recovery (and given the fact that the first to respond will be chosen), this mechanism is efficient in terms of the time taken to re-establish the session.

IV. Evaluation

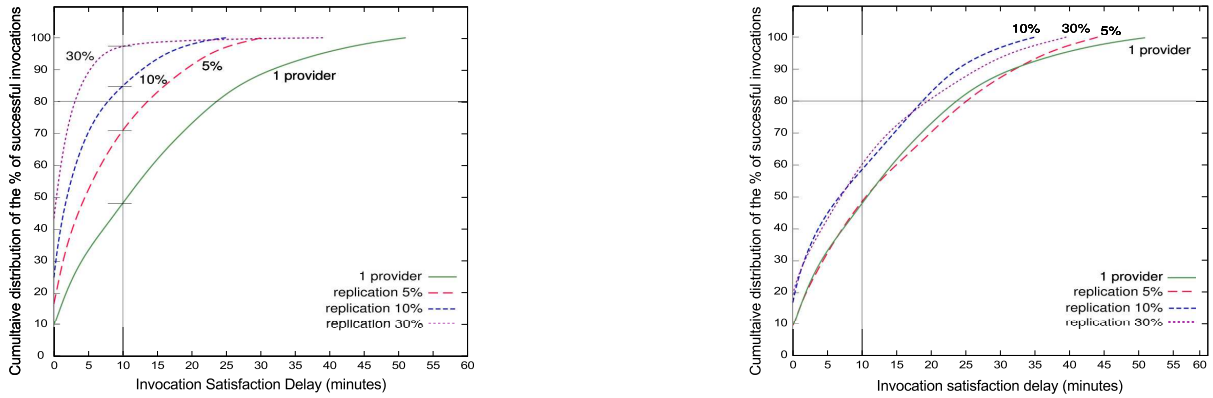
The service platform described in this paper has been fully implemented in Java and a number of experiments have been conducted in networks composed of a few devices. In order to assess the performance of our approach in larger configurations, we interfaced the service platform with the MADHOC simulator [6]. Contrary to more popular wireless network simulator, Madhoc allows us to run the actual code of our platform, hence taking into account not only algorithmic issues but all the implementation choices we made.

A. Simulation experiments

Network environments present different topologies and mobility behaviors that directly influence the quality of the service provision. We strove to adopt an as realistic as possible an environment that is intended to model an urban area with buildings and road mobility restrictions. We considered an area of 500×900 m^2 in which 100 individuals can stay temporarily in one of the four buildings, and take roads to go from one building to another at variable human walking speeds. At any time an individual may stop for a random period of time (between 1 and 6 minutes). Nodes in the network represent users equipped with wireless handheld devices that are switched on and off in a random manner. Nodes have different wireless ranges whether they are indoor (20 m) or outdoor (50 m). A node has affinities with some buildings and is not capable of accessing all the environment areas during a simulation run.

On each node of the network is deployed our service platform. One of the nodes plays the role of a service client that tries to use only one type of service. The service is proposed by a variable number of provider nodes. The remaining nodes are neither providers nor clients but are used as simple message relays. In an environment of *N* nodes, containing *p* provider nodes, we call *r = (100 * p / N)* the percentage of service replication in the environment. In the scenario we studied in the simulation, the client discovers the service once. Then, it performs a succession of invocation requests all along the simulation period. We focus here on the evaluation of the invocation phase.

Two types of measurement are performed. Our first objective is to estimate the client satisfaction in terms of delay (the delay between the time at which the client publishes its invocation request and the time at which it receives its response). Our



(a) Content-based invocations

(b) Destination-based invocations

Figure 4. Impact of service replication on invocation satisfaction

second objective is to estimate the cost of our approach in terms of the load on the radio medium. For this purpose, we measure the number of messages sent and received as well as the global amount of data transmitted. Figures are obtained by averaging measurements over 50 simulation runs.

B. Exploitation of service replication

The content-based approach we followed is intended to accelerate the invocation process when several providers propose the service. Figure 4 shows, for several service replication rates, a comparison of the invocation satisfaction delay obtained when (a) content-based invocation requests can reach any compatible provider and the client exploits the first received response and (b) when the invocation requests are sent with destination-based messages, to a pre-discovered provider. Each curve plots the cumulative distribution of the percentage of successful invocations against the measured average delay. As an example, in figure 4-(a), we see that around 70% of the invocations are completed in less than 10 minutes when the service replication is of 5% (that is, when the service is provided by 5 of the 100 nodes).

The results show that the use of content-based invocation effectively takes advantage of service replication: the more the percentage of replication the quicker the curves reaches high values, whereas replication has little effect when destination-based invocations are used. The curves in figure 4-(b) show for example that, when considering a percentage of 80% of the invocations completed, the satisfaction delay passes from 24 minutes with no replication to 20 minutes with 30% replication (compared to a change from 24 to 3 minutes when using content-based invocations in the same conditions). Moreover, it is worth noticing that the gain brought about by content-based invocations is significant even for a low percentage of service replication.

In the remaining of the presented results, we will consider only one value for the service replication, fixing it at a 10%. A low value has been chosen, knowing that the actual replication rate could vary a lot according to the nature of the application services.

C. Cost of redundant response reduction

Provider replication increases the network load by adding extra full load response messages. That is why we implemented in our platform healing techniques in order to eliminate as much as possible unneeded messages. Here we show the effectiveness of the three R4 techniques described in Section III-B, through their impact on the number of messages sent in the network as well as on the amount of data transferred. We also show that the healing does not influence the satisfaction of clients.

Figure 5 shows the cumulative distribution of the number of responses sent by the providers against time when applying the R4 techniques, namely the Reactive Cancellation, Client-initiated Cancellation and Provider-initiated Cancellation. A reference curve has been added that gives the results obtained without any healing (by removing the code of the Reactive Cancellation that is normally systematically active). For the scenario considered, the number of sent responses is drastically reduced by all the three techniques (by a factor of up to 4.7), even by those which are not always profitable in theory. Interestingly, although the Provider-initiated Cancellation may

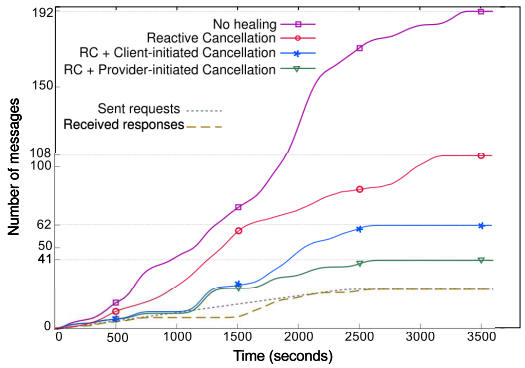


Figure 5. Effect of the three R4 techniques on the suppression of redundant responses

theoretically delay useful responses and even prevent them from arriving at the client, the results for the considered scenario do not reveal such a loss. This is illustrated by the two

dotted lines in the lower part of Figure 5, which represent the cumulative number of requests sent by the client and the cumulative number of responses effectively received by this client (with Reactive Cancellation and Provider-initiated Cancellation activated): responses are, as expected, slightly delayed, but the client eventually receives them all.

When considering the network load, the impact of network healing is even more important. Extra messages are added but they have a small footprint and they allow saving requests and responses (which may be of relatively large size). In practice, the volume of the request and responses saved can largely compensate the volume of extra control messages. This is confirmed by the results given in Figure 6. This figure plots the cumulative network load sent by all the nodes into the network medium when no healing is applied as well as when the three R4 techniques we presented are used together. The network load is divided in two values: the cumulative useful payload (i.e. the requests and responses), and the cumulative size of control messages (at the service and the communication level, including extra control messages due to healing). We considered an average size of 26 kB for requests and responses. As expected, the amount of control data slightly increases when the healing is activated, but this extra cost is very small compared to the amount of data that is required for redundant requests and responses when no healing is applied: for the entire experiment, 27 MB of control messages are added but 142 MB of useful payload are saved.

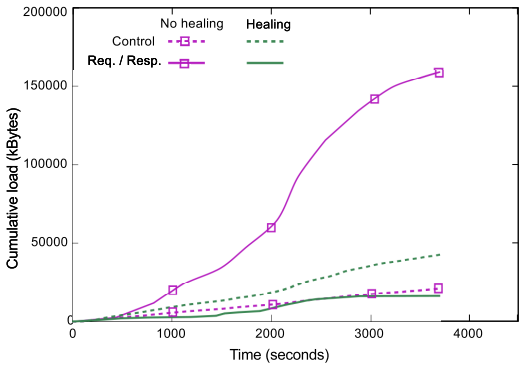


Figure 6. Cumulative load sent into the network medium

D. Session recovery

We compared the efficiency of our session discovery mechanism based on content-based invocation to a more classic approach used in MANETS. In this latter approach, communication can only take place inside an island, using multi-hop transmissions that exploit a routing algorithm. We measured the time taken for session recovery in a simulated scenario where a client tries to maintain a session with a service provider, within which it periodically sends invocation requests with a 5-minute timeout (The session is considered broken if the response has not arrived 5 minutes after the request is sent.). Each of the providers are randomly turned off and back on 5 times during a simulation run.

In the routed scheme, the client iterates on the following steps: (1) it discovers a list of providers that propose the service it wants by broadcasting a discovery request in its entire communication island and by waiting for a discovery response ; (2) it opens a session with the first provider that responds ; (3) it performs a sequence of invocations to the chosen provider until the session is broken. The scenario with our session recovery scheme is simpler as the discovery phase has not to be repeated: after having discovered the existence of the wanted service, the client iterates on the following steps: (1) it performs a first content-based invocation, opening a session with the first provider that responds ; (2) it performs a sequence of destination-based invocations to this provider until the session is broken.

Figure 7 shows the cumulative distribution of the percentage of session recovery in the two schemes. Our session recovery scheme clearly outperforms the routed scheme. For example, with our approach, 60% of the sessions are recovered in less than 20 minutes whereas only 25% of the sessions are recovered in less that this same duration with the routed scheme. Two main reasons explains it: the discovery phase is performed only once, and our communication method is not limited to the communication island so it allows the client to reach any provider in the network when trying to re-establish a session.

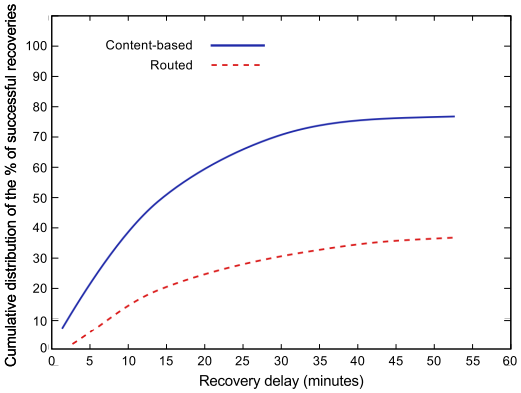


Figure 7. Session management.

V. Related Work

A number of research works are concerned with communication in disconnected MANETs, mainly with the objective of providing destination-based communication [7], [8] and sometimes proposing content-based approaches [9]. A noticeable contribution in this area has been made in the European Haggle project [10]. All of these works however focus on communication issues and do not address the problems of the design and the implementation of a service platform targeting disconnected MANETs. On the other hand service-oriented programming in pervasive environments has been the object of many research activities. Most of them primarily focus on service discovery protocols using traditional centralized service directories over infrastructure-based (LAN) or single-hop wireless networks ([11] presents a survey of the main protocols).

Recently, the characteristics of MANETs (dynamicity, low resources, unreliable transmissions) have been taken into account, With the development of specific discovery protocols, as part of the routing layer [12], [13] or as part of the service layer with no dependency on routing [14]. In these protocols, communications are still considered achievable and to some extent sustainable (routes between hosts can always be established). Konark [14] uses a distributed multi-hop advertisement/discovery scheme but relies on a local multicast and implements connected invocations that are not suitable for disconnected environments. Other efforts try to create a virtual higher-level network in MANETs where some selected nodes are assigned special functionalities. Some of them implement semi-centralized directories for discovery [15] where each interconnected network (small connected network island) has one directory, while others assign network service brokers [16] or create network backbones [17]. Our prototype does not assume any viable interconnections so each host is responsible for its own service repository. In addition, we do not use overlay network structures nor maintain discovery routes. Other solutions propose session management over routed MANETs. GSR-S [12] enables group-based service discovery using discovery requests and advertisements, and reuses the paths of discovery to also handle data transmission for routing invocations with session failure management. "Follow-me" sessions [18] are built on the same concept of interchangeably invoking compatible providers in a MANET. A form of content-based addressing is used for maintaining opened sessions but the focus is put on server code migration rather than on the support of a possible fragmentation of the network.

VI. Conclusion

We presented in this paper the outline of a service middleware platform that specifically targets disconnected MANETs. The originality of our approach resides in the fact that we build service-level facilities on top of a communication layer that provides network-wide dissemination of messages based on content-based and opportunistic communication. This communication layer is used through a publish/subscribe interface not only for service discovery but also for service invocation.

We described the implementation of a form of content-based invocation that can benefit from the potential multiplicity of providers of a given service in the network. Several compatible providers can reply to an invocation request issued by a client, enhancing the probability for this request to be rapidly answered, which can prove crucial in a context where frequent contacts between clients and providers are not guaranteed. As exploiting multiples providers generates redundant responses, we proposed several healing techniques to compensate the extra cost in terms of network load. The approach is applied for the invocation of stateless services and is also used for session recovery when using stateful services.

Simulations experiments involving the real code of the platform have been conducted. The simulation conditions were as realistic as possible, modeling pedestrians in an urban environment. Of course the delays observed for service invocation are directly correlated to the mobility of the network nodes. As we rely on a store-carry-and-forward approach for message dissemination, some messages need to be physically transported, at human speed. However, simulation results namely show that the multiplicity of compatible providers is effectively exploited as the satisfaction delays perceived by the clients can be drastically reduced even with a relatively low percentage of replication of the service providers. Moreover, as far as network load is concerned, the implemented healing techniques proved efficient: the number of redundant messages can be significantly decreased while keeping the satisfaction ratio at a high value.

References

[1] T. Clausen and P. Jacquet, “Optimized Link-State Routing Protocol (OLSR),” IETF, RFC 3626, oct 2003.

[2] C. Perkins, E. Royer, and S. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing,” IETF, RFC 3561, 2003.

[3] A. Carzaniga and A. L. Wolf, “Content-based Networking: A New Communication Infrastructure,” in *Workshop on an Infrastructure for Mobile and Wireless Systems*, ser. LNCS, no. 2538. Scottsdale, USA: Springer, Oct. 2001.

[4] E. Patrick, P. Felber, R. Guerraoui, and A.-M. Kermarrec, “The Many Faces of Publish/Subscribe,” *ACM Computing Surveys*, vol. 35, pp. 114–131, 2003.

[5] J. Haillot and F. Guidec, “A Protocol for Content-Based Communication in Disconnected Mobile Ad Hoc Networks,” in *Int. Conf. on Advanced Information Networking and Applications (AINA’08)*, Okinawa, Japan, Mar. 2008.

[6] L. Hogie, P. Bouvry, and F. Guinand, “The MADHOC simulator,” http://www-lih.univ-lehavre.fr/~hogie/madhoc.

[7] Z. Zhang, “Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges,” *IEEE Communications Surveys and Tutorials*, vol. 8, no. 1, pp. 24–37, Jan. 2006.

[8] L. Pelusi, A. Passarella, and M. Conti, “Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks,” *IEEE Communications Magazine*, Nov. 2006.

[9] P. Costa, M. Musolesi, C. Mascolo, and G. P. Picco, “Adaptive Content-based Routing for Delay-tolerant Mobile Ad Hoc Networks,” UCL, Tech. Rep., Aug. 2006.

[10] “Haggle Project,” http://www.haggleproject.org.

[11] F. Zhu, M. Mutka, and L. Ni, “Service Discovery in Pervasive Computing Environments,” *IEEE Pervasive Computing*, vol. 4, no. 4, pp. 81–90, Dec. 2005.

[12] D. Chakraborty, A. Joshi, and Y. Yesha, “Integrating Service Discovery with Routing and Session Management for Ad-hoc Networks,” *Ad Hoc Networks*, vol. 4, no. 2, pp. 204–224, Mar. 2006.

[13] P. E. Engelstad, Y. Zheng, R. Koodli, and C. E. Perkins, “Service Discovery Architectures for On-Demand Ad Hoc Networks,” *Ad Hoc and Sensor Wireless Networks*, vol. 1, 2006.

[14] S. Helal, N. Desai, V. Verma, and C. Lee, “Konark : Service Discovery and Delivery Protocol for Ad-hoc Networks,” in *Int. Conf. on Wireless Communication Networks (WCNC)*, New Orleans, USA, Mar. 2003.

[15] F. Sailhan and V. Issarny, “Scalable Service Discovery for MANET,” in *Int. Conf. on Pervasive Computing and Communications (PerCom’2005)*, Hawai, USA, Mar. 2005.

[16] A. Nedos, K. Singh, and S. Clarke, “Service*: Distributed Service Advertisement for Multi-Service, Multi-Hop MANET Environments,” in *Int. Conf. on Mobile and Wireless Communication Networks (MWCN’05)*, Marrakech, Morocco, Sep. 2005.

[17] U. C. Kozat and L. Tassiulas, “Network Layer Support for Service Discovery in Mobile Ad Hoc Networks,” in *Joint Conf. of the IEEE Computer and Communications Societies (IEEE/INFOCOM-2003)*, San Francisco, USA, Apr. 2003.

[18] R. Handorean, R. Sen, G. Hackmann, and G.-C. Roman, “Context Aware Session Management for Services in Ad Hoc Networks,” in *Int. Conf. on Services Computing (SCC’05)*, Orlando, USA, Jul. 2005.