Earth Sci Inform (2015) 8:439–452
DOI 10.1007/s12145-014-0164-4

REVIEW ARTICLE

Service discovery with time constraints in mobile ad hoc networks

Janine Kniess · Orlando Loques · Célio V. N. Albuquerque

Received: 11 June 2013 / Accepted: 22 May 2014 / Published online: 10 June 2014

Abstract In service discovery protocols for MANETs, a key problem is to reduce the energy consumption and keep the system operating despite the occurrence of faults. In this paper, we propose a location-aware service discovery protocol that consists of a service selection, a service discovery and service invocation mechanisms. A service selection mechanism applies a data aggregation scheme in intermediate nodes to reduce the message replies in these networks. A service discovery mechanism adjusts the search area for each specific request and considers the requesting node's location, the request response time and the maximum node speed to provide fault tolerance in order to deliver packets in adversarial environments. A service invocation mechanism sets up how the services could be accessed and used. Results show that these mechanisms make it possible to maintain the trade off between the discovery success rate and the message reply reduction, hence minimizing the network traffic.

Communicated by: H. A. Babaie

J. Kniess (✉)
Computer Science Department, Santa Catarina State University, Joinville, Brazil
e-mail: janine@joinville.udesc.br

J. Kniess
e-mail: jkniess@ic.uff.br

O. Loques · C. V. N. Albuquerque
Institute of Computing, Fluminense Federal University, Niterói, Brazil

O. Loques
e-mail: loques@ic.uff.br

C. V. N. Albuquerque
e-mail: celio@ic.uff.br

Keywords Service discovery and selection · MANETs · Fault-tolerance

Introduction

A mobile ad hoc network (MANET) can be defined a self-organizing and dynamically reconfigurable wireless network which operates in the absence of a fixed infrastructure (Balaji et al. 2008). The extremely dynamic nature of MANETs has motivated the development of their application in adversarial environments like scenarios that have been hit by natural disasters, military and rescue sites. Some of the issues identified in MANETs are service discovery, mobility management, and energy consumption. Service discovery is an essential component for the usability of such self-organizing networks on the grounds that service discovery enables devices to use their functions to automatically locate network services and to announce their own functions to the network.

Since MANET provide a resource constrained dynamic environment, they introduce new aspects to time constraints and fault-tolerance thereby affecting reliability of the services offered by providers.

Although considerable previous research has been done into service discovery for distributed systems (Marin-Perianu et al. 2005), few protocols have considered the development of time-constrained service discovery mechanism combined with data aggregation techniques to offer fault-tolerance and to reduce the energy consumption in the network. The response time constraints for the target scenario means the maximum time to attend a request (by the provider node).

In service discovery for MANETs, a common approach to disseminate requests is to use broadcast mechanisms (Mian et al. 2009). This technique is commonly used because the service provider's location is not previously known. This is often the case where multiple providers can offer the specified service. In this case, a data aggregation approach can be used to minimize the reply transmission cost in network.

This paper focuses on the challenges involved in supporting service discovery in MANETs. Thus, we propose a time-constrained location-aware service discovery protocol for MANETs composed of a distributed and autonomic location-aware service selection mechanism, named Location Aware Service Selection (LASS) that uses a data aggregation scheme in intermediate nodes to combine the responses coming from different providers. The goal is to minimize the number of transmissions, thus improving the network performance and saving energy. Also the protocol provides a fault-tolerant service discovery mechanism, named Location Aware Service Discovery (LADS) that operates over MANETs in adversarial environments such as search and rescue operations following the occurrence of natural disasters. There is also a service invocation mechanism that specifies how the providers selected in the service selection phase will be accessed and used by the requesting node.

In MANETs distributed in adversarial environments such as emergency missions, the maximum response time to attend one request is essential to guarantee that service discovery is successful. Thus, we identified a number of aspects as being critical to a successful outcome: the geographic location where the service provider is being requested; the maximum response time (we suppose that the provider needs to arrive at the place where the service is required within a maximum time); the speed at which the service provider moves; and the number of providers that must be requested. The proposed protocol takes into account the above mentioned aspects in the service discovery process.

Briefly, our contributions are the proposals of: (1) an autonomic and distributed service selection mechanism that applies a data aggregation scheme in intermediate nodes to filter the responses before arriving at the requesting node, aiming to reduce the exceeding replies and save energy; (2) a fault-tolerant service discovery mechanism, that exploits redundancy to keep the system operating despite the existence of faults, specifically, component and message failures. Moreover, the discovery mechanism adjusts a search area for each individual request. This approach determines that only nodes capable of responding to a request within a maximum time limit be considered in the discovery process; and (3) a service invocation mechanism that specifies how the service providers will be accessed and used by the requesting node. Also, all mechanisms take into account time constraints in managing the requests or responses.

This paper is organized as follows. Section II presents the main related proposals regarding service discovery for MANETs and the motivation scenario; Section III introduces the time-constrained Location Aware Service Discovery Protocol (LADP); Section IV describes the evaluation and results; and finally, Section V presents the final conclusions and suggests further research directions.

Application context and requirements

Service discovery in MANETs has emerged as a useful application in scenarios where the service providers are mobile and must be efficient and capable to attend a request within a maximum time, for instance, MANETs in assisting emergency missions (Serhani et al. 2010).

For example, in Fig. 1, we consider an area that has been struck by a natural disaster. In this context, we believe that the distribution of a specialized rescue team (formed by vehicles, robots and humans beings, for example), interconnected by a wireless mobile ad hoc network could contribute greatly to the rescue of survivors.

In the proposed scenario, some of the mobile elements of the rescue team provide a resource i.e., a resource could be an ambulance, a robot with the ability to access places that are not reachable by man, such as areas with a contamination risk, or even a human being transporting medication. The nodes that provide services are the providers and those that request the service are the requesters. We assume that each node in the network is aware of its geographic position by means of a localization system, such as a GPS, and the time in the devices is synchronized. In such case nodes would be synchronized by GPS.

For instance, is possible that a node (equipped with WiFi 802.11) identifies the existence of a leak of a given type of gas and, from specialized algorithms, the node can derive the need for a provider, such as a fire engine, which could be at the location in a matter of minutes. After identifying which resource is needed, the node will send a request message to the network searching for the appropriate resource.

The service discovery in this scenario involves a number of challenges:

– Node mobility: Since the mobile devices may move very rapidly, the service discovery protocol must be able to find the service providers and ensure that the providers

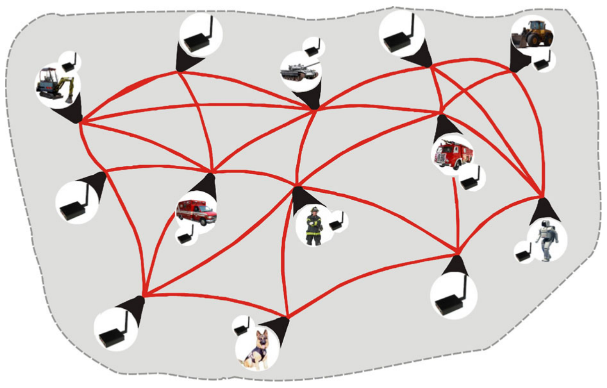

Fig. 1 Scenario application

selected in the service selection phase will be correctly accessed and used by the requesting node.

– Energy awareness: energy is an important resource that needs to be preserved in order to extend the lifetime of the network. In service discovery protocols for MANETs distributed in scenarios like the one shown in Fig. 1 changing the battery of the mobile devices can be difficult or impossible. Thus, mechanisms to reduce the message request and reply transmissions from the network can be used to save energy.
– Fault tolerance: this deals with the behavior and robustness of the discovery process under erroneous conditions. Since the discovery system is a support function to service infrastructures, its failure may hinder the entire service provisioning process (Chowdhury et al. 2012). In the Fig. 1, context messages should be delivered within a maximum time. If a message cannot be delivered within this time, it is considered that a failure occurred.
– Location-aware: the nodes are not aware of the location of the others. The service discovery protocol should consider the development of mechanisms capable of dealing with the localization of the nodes.
– Response quality: the discovery process is only interested in nodes which are capable of arriving in time where the resource is needed and which should send a response to the requester.

Related works

Service discovery is the action of finding and locating a service in the network (24 S. Marin-Perianu et al. 2008). Given a description of a requested service, the result of service discovery is the address or geographic coordinates of one or more service providers that are able to offer the specified service. When the address or geographical location is retrieved, the user may further access and use the service offered by the provider (van der Aalst 2010).

Service discovery protocols can be generally classified into three categories (Mian et al. 2009). In chronological order we mention: fixed network protocols such as Jini (Arnold 1999), Salutation (24 Salutation, "Salutation architecture specification," 1999) and SLP (Guttman 1999); wireless single-hop networks such as DEAPspace (Nidd 2001) and Bluetooth (Miller et al. 2001); and wireless multi-hop networks. Neither the first category (fixed network protocols), nor the second one (wireless single-hop networks) are feasible for decentralized environments such as the one presented in the scenario above. In these environments, it is possible to create multi-hop networks. The following protocols stand out in this last category: GSD (Group-based Service Discovery) (Chakraborty et al. 2006), Konark Gossip (Helal et al. 2003), FTA (Field Theoretic Approach) protocol (Lenders et al. 2005), the P2PDP protocol (Gomes et al. 2008), the cross-layer approach of Varshavsky et al. (Varshavsky et al. 2005) and the protocol proposed in (Gadallah et al. 2011).

Notwithstanding considerable previous research has been done into service discovery for distributed systems, few protocols as P2PDP protocol (Gomes et al. 2008) have considered to add a distributed aggregation response in the selection phase to save energy. Service selection is the phase that comes after service replies are gathered by the service requester. Besides aggregation response in the selection phase, another aspect not properly explored is fault tolerance issues.

The approaches that have been proposed by (Gomes et al. 2008), (Lenders et al. 2005) and (Varshavsky et al. 2005) provide a mechanism for autonomic service selection. However, aggregation and fault tolerance issues are not considered in these works.

The FTA approach proposed by (Lenders et al. 2005) is based on the theory of electrostatic fields. Requests to an instance of a given service type are routed selectively in the direction of the provider that generated the highest field gradient. However, this approach does not scale well when different types of services are available. The scalability problem in the FTA results from the fact that providers need to periodically exchange update messages about the server capacity value through a flooding mechanism. Each service type has a specific capacity. Another limitation of this proposal is that only one response for the requested service is sent to the requesting node. In (Varshavsky et al. 2005) the selection service function is integrated into the routing mechanism in a cross-layer approach. Moreover, the selection occurs only when the replies arrive at the requesting node. In (Gomes et al. 2008) it shows a suppression vicinity service selection mechanism that can discard exceeding replies during the reply transmission. However, this mechanism filters response messages through the reverse path, i.e., messages are routed by a reverse path traversed by the request. This concept can easily fail in MANETs due to highly dynamic topologies. The service discovery protocol proposed in (Gadallah et al. 2011) can be applied in disaster situations. However, no strategy for selecting responses is considered. Service providers only send the responses to a central node that performs the selection.

Unlike the mentioned approaches, our work also considers that the providers must move as far as the place where the resource is needed. The other approaches consider that the services can be remotely accessed and used.

Location aware service discovery protocol

In this section, we will first describe the architecture of the Location Aware Service Discovery Protocol (LADP) (Kniess et al. 2009). Secondly, we will describe its mechanisms. Specifically, how the service discovery mechanism named Location Aware Service Discovery (LADS) can provide fault tolerance in MANETs and how the service selection mechanism named, Location Aware Service Selection (LASS) scheme aggregates replies in intermediate nodes, and selects the best providers during the reply transmission. Third place describes a service invocation mechanism and the strategies adopted by it to set up how the service providers could be accessed and used.

The architecture of the LADP protocol was specified in order to support the development of applications for service, discovery, selection and invocation in multi hop MANETs even in the presence of obstacles in the path and failures of nodes. The LADP architecture is presented in Fig. 2.

Suppose in Fig. 2 that the requesting node needs to find a service provider and sends a service discovery message (*msgDiscovery*) (1) to the network. This message is sent through a broadcast mechanism in order to find providers that meet the requests profile. The message will travel within a radius R_i defined in the service discovery phase. Only providers able to meet the request in a timely manner send a response message (*msgReply*) (2) to the requesting node. Each service provider stores information about the resource that it provides. The nodes outside the radius receiving the discovery message will discard it. The service selection mechanism uses a distributed and autonomic selective filter in the intermediate nodes and sends the responses only from providers that fit in the profile of the request. After receiving responses from suitable providers, the requesting node sends a service invocation (*msgInvocation*) (3) to the providers selected.

When the provider receives a (*msgInvocation*), it sends a confirmation message (*msgConfirmation*) (4) to the requester. The confirmation message informs that the service provider will physically move to where the resource is needed.

Table 1 summarizes the notation used in this paper.

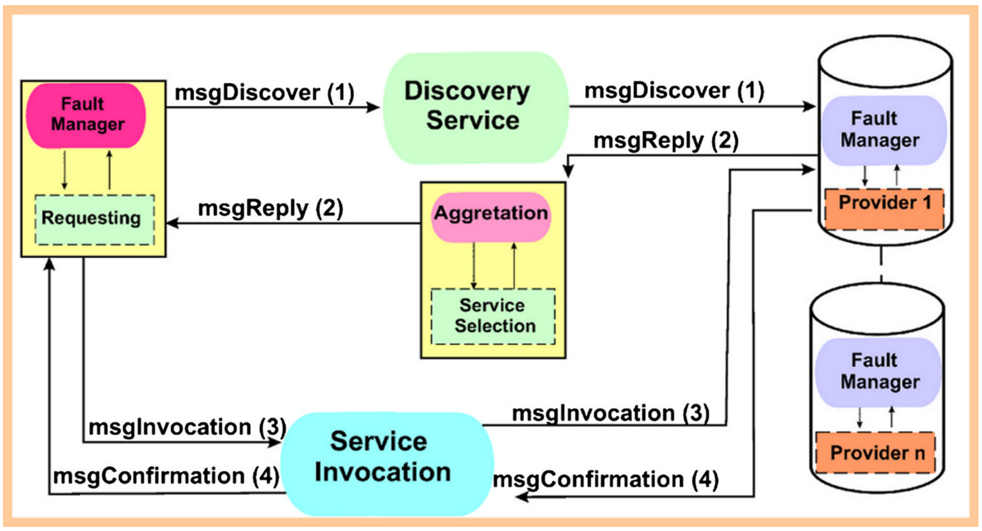

Fig. 2 LADP architecture

Table 1 Summary of notation

Symbol	Definition
v_{max}	maximum node speed
Δ_{tmax}	max response time for one request
R_i	search diameter for requesting node
$n_{provider}$	reply counter
$max_{provider}$	number of responses specified
$\Delta_{trequester}$	client node timer
$\Delta_{tintermediate}$	intermediate node timer
β	max time for the closest node to store responses before forwarding them
α	represents network delays
Service time provider	max time for the provider to arrive where the resource is needed
TTL_provider	provider time to live

Location aware service discovery (LADS)

This section describes the strategy adopted by the service discovery mechanism, LADS, to provide fault tolerance and to reduce messages in service discovery for MANETs. In this context, we assumed that messages should be delivered within a maximum time. If a message cannot be delivered within this time, it is considered that a failure occurred. Furthermore, the LADS mechanism considers two types of failures: components (by a crash) and messages. Component failures are those related to physical devices, such as electromagnetic interference. Message failures are those related to congestion, signal fading and involuntary disconnections.

LADS works as follows. Suppose a node in the network needs information about service providers and sends discovery messages. The LADS mechanism limits the search diameter R_i, on the basis of the maximum speed that a node can reach, v_{max} (each type of $_{tmax}$resource knows this value), and the maximum response time for one request, Δ. Using R_i, this mechanism prevents unnecessary request and reply transmissions in the network. The diameter R_i is given by the equation:

$$R_i \leftarrow v_{max} \times \Delta_{tmax} \tag{1}$$

The mechanism defines the diameter using v_{max} so that the search area includes the greatest number of suitable providers. Given the pair (i,j), i being the requester and j the provider, it is assumed that the speed (v_j) of the latter is known.

After sending a message, the requesting node starts the timer, $\Delta_{trequester}$. This timer defines the maximum amount of time during which the requester will wait for responses, and this time is proportional to the diameter of the request. If a requester does not receive responses within this time, it will resend the discovery message. $\Delta_{trequester}$ is given by,

$$\Delta_{trequester} \leftarrow K \times (R_i/R_t) \tag{2}$$

The value of K is defined as $\alpha+\beta$. The value of α specifies the forward and backward delay for one hop on the network. The value of β specifies the maximum amount of time that the intermediate node closest to the requester can store one response before forwarding it. The range of the antenna is given by R_t, and R_i/R_t represents an estimate of the number of hops. The closest intermediate nodes maintain the replies stored for longer time compared with more distant nodes.

Algorithm 1 describes this mechanism, where, $number_{provider}$ is a variable responsible to count the number of responses received by the intermediate node, and $max_{provider}$ means the minimum number of providers that must be delivered.

Algorithm 1: Service discovery algorithm

```
PROCRECVMSGDISCOVERY()
1 node $j$ receives a $msgDiscovery()$;
2 if $d_{ij} > Ri$
3     node $j$ discards $msgDiscovery()$;
4 else if $d_{ij} / v_j \leq \Delta_{tmax}$
5     if findResource(s) = TRUE
6         if resourceFree = TRUE
7             node $j$ sends a $msgReply$ to node $i$;
8
2  PROCENDTIMERREQUESTER()
2   if $\Delta_{trequester}$expires and $number_{provider} < max_{provider}$
3         node $i$ resends $msgDicovery()$;
4         node $i$ starts a new $\Delta_{trequester}$;
5   else
6         if
7         $\Delta_{trequester}$expires and $number_{provider} = max_{provider}$
8             node $i$ cancels $\Delta_{trequester}$;
```

In the discovery message, the requesting node sends the following information: a node identification, its coordinates, $coord_X$, $coord_Y$, the maximum response to attend the request Δ_{tmax}, the service, s, and the number of desired providers. If node j receives a request from node i, the algorithm verifies the distance (d_{ij}) between both nodes. If $d_{ij} > R_i$, the request is discarded by j because this node is out of the search area. Conversely, if $d_{ij} \leq R_i$, the algorithm verifies the speed of the service provider j. Moreover, the algorithm verifies if this provider offers the searched resource (s) and if the service provider j is available at the moment. If the restriction given by Eq. (3) is satisfied, node j sends a response to node i.

$$d_{ij}/v_j \leq \Delta_{tmax} \tag{3}$$

Assuming that the provider node j has the resource, but v_j is insufficient, j does not send a reply to i, and it only resends the request message. It is assumed that the nodes have maximum speeds defined. The distance in the proposed protocol is calculated based on straight line (Euclidian distance). The nodes attend the requests on demand. Each node maintains information about the resource that is offered by it.

Location aware service selection (LASS)

As a result of the discovery process, multiple providers can respond to a service request. The LASS mechanism takes into account such aspects as the requesting node's geographic location, the maximum response time to attend one request, the speed that the service provider moves, and the number of service providers desired to select and discard answers. The aim is to reduce the number of reply transmissions from the network.

LASS works as follows. Suppose that node *k* (intermediate node) receives the reply message from one of its neighbors, node *m*. Then, node *k* starts a timer, named $\Delta_{tintermediate}$, with the function of storing replies. This timer is given by Eq. (4).

$$\Delta_{tintermediate} \leftarrow \gamma \times (1/d_{ik}) \leq \beta \qquad (4)$$

where γ specifies a proportionality factor with the function to maintain the timer $\Delta_{tintermediate}$ less than β. $1/d_{ik}$ denotes the inverse distance between the requesting and the provider node. Since the storage period is inversely proportional to the distance between the intermediate node and the requester, the closest nodes store replies for a longer time.

For example, in Fig. 3b, $\Delta_{tintermediate}$ of node I2 is greater than $\Delta_{tintermediate}$ of node I1. This occurs due to the fact that I2 is closest to the requesting node. The LASS data fusion scheme is shown below.

LASS data aggregation scheme

The LASS data aggregation scheme can assume two distinct schemes depends on the quality of responses expected by the requesting node:

– ***wait***$_{maxprovider}$: in this scheme, a requesting node wants to receive responses from suitable providers, not necessarily the fastest among the ones that fit. In the *wait*$_{maxprovider}$ scheme, if the maximum number of responses (*max*$_{provider}$) that meet the requests profile is reached before $\Delta_{intermediate}$ expires, the intermediate node aggregates these responses and sends only one response to the requesting node. After this step, the timer $\Delta_{intermediate}$ is canceled by the intermediate node. Other responses received for the same request are discarded.

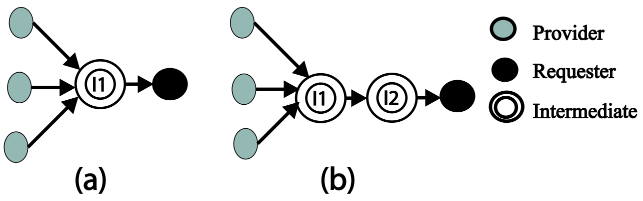

Fig. 3 Lass data aggregation scheme

This scheme is shown in Algorithm 2, where *number*$_{agregate}$ means the number of responses aggregated and *number*$_{totalresponses}$ the sum of the aggregate responses with number of replies already received for the intermediate node.

– ***wait***$_{fastest}$: in this scheme, a requesting node wants to receive responses from faster service providers. Therefore, the intermediate node will wait until the $\Delta_{intermediate}$ timer has finished. When this timer expires, this node chooses the responses from fast providers under the condition, $d_{im}/v_m \leq d_{ij}/v_j$. After these steps, the intermediate node aggregates these responses in one response message and sends this message to the requesting node. The number of responses aggregated in one response message must be less than or equal to *max*$_{provider}$. If *max*$_{provider}$ is not reached within this time, the intermediate node aggregates the responses received and sends them to the network. The *wait*$_{fastest}$ scheme is presented in Algorithm 2.

In both schemes, an intermediate node collects responses from different providers, and extracts three individual parameters, which are: provider identification (*IdProvider*), maximum time that the provider needs to arrive at the place where the service is required (*ServiceTimeProvider*), and the type of data aggregation scheme (*max*$_{provider}$ or *wait*$_{fastest}$) specified by the requesting node. Then, the intermediate node concatenates these parameters in one string. The other parameters are common for all replies. In the response message (*msgReply* ()), a service provider sends the client identification, the resource identification, the request identification, the sequence number identification and the maximum number of replies desired by the requesting node.

As we can see, in Fig. 3, an intermediate node can receive a response directly from a provider or from another intermediate node:

– intermediate node receives a response directly from a provider (Fig. 3a): this response is not aggregate yet. Then, the intermediate node checks in the message response for the data aggregation scheme informed by the requesting node. Considering that the *max*$_{provider}$ scheme was chosen, the intermediate node verifies the number of responses stored by it. If the maximum number of responses is not reached, the intermediate node stores the new response. On the other hand, if the *wait*$_{fastest}$ scheme was chosen, the intermediate node just stores the response and does not verify the number of responses received.

– intermediate node receives a response from another intermediate node (Fig. 3b): this message can have aggregated response messages. Suppose that in Fig. 3b that the intermediate node *I2* receives one response from the intermediate node *I1* and assuming that the *wait*$_{maxprovider}$ scheme was adopted. First, *I2* will verify the number of aggregate responses in this message. After this step, *I2* will confirm the number of responses stored by it. If the $max_{provider}$ number is not achieved, *I2* will store the aggregate response.

In the $wait_{maxprovider}$ scheme, the number of stored responses plus the number of aggregate responses may not exceed $max_{provider}$. In order to solve this problem, the LASS algorithm will update the stored responses related to the aggregated responses on the condition that the aggregate responses have better quality than the stored responses, that is, if there are faster providers in the aggregate responses. If the $wait_{fastest}$ scheme is adopted, *I2* only stores the received responses and does not verify the number of aggregated responses.

The LASS mechanism is presented in Algorithm 2. It is noteworthy that the repeated answers (already treated by a particular provider) are discarded.

The LASS mechanism can be used independently from the LADS mechanism in MANETs. The discovery mechanism could act upon a type of diffusion protocol. In LASS, no additional information would be necessary. The only restriction is that the nodes must send their geographic location, as in the LADS mechanism.

Algorithm 2: Service Selection Algorithm

```
ProcWaitMaxprovider()
1 intermediate nodes aggregate replies
2 node $k$ verifies the number of aggregated replies ($number_{agregate}$) in $msgReply()$;
3 node $k$ verifies the number of replies already received by it ($number_{provider}$);
4 $number_{totalresposes} \leftarrow number_{agregate} + number_{provider}$;
5 if $number_{totalresposes} < max_{provider}$
6       $number_{provider} \leftarrow number_{totalresposes}$;
7       node $k$ stores the new responses;
8       if $\Delta_{intermediate}$ expires
9           node $k$ aggregates received replies;
10          node $k$ sends $msgReply()$;
11 else if $number_{totalresposes} = max_{provider}$
12      node $k$ aggregates received replies;
13      node $k$ sends $msgReply()$;
14      node $k$ cancels $\Delta_{intermediate}$;
15 else
16
17        node $k$ discards $msgReply()$;

ProcWaitFatest()
1 intermediate node $k$ received a $msgReply()$ message;
2 node $k$ verifies the number of aggregated replies ($number_{agregate}$) in $msgReply()$;
3 node $k$ stores the new responses;
4 $number_{provider} \leftarrow number_{provider} + 1$;
5 if ($\Delta_{intermediate}$ to expire) and ($max_{provider}$) is not reached
6      node $k$ aggregates received replies;
7      node $k$ sends $msgReply()$;
8 else if ($\Delta_{intermediate}$ to expire) and ($max_{provider}$) is already reached
9      intermediate node $k$ chooses the best provider responses;
10     node $k$ aggregates replies;
11     node $k$ sends $msgReply()$;
```

Service invocation mechanism

The service invocation mechanism operates after the selection service phase and establishes rules for access and use of the providers already selected. In the service invocation phase, requesters and providers verify the viability of the attendance and the providers physically move to the place where the service is required.

As we can see in Fig. 4a, after the selection service phase represented by (2), the requesting node sends a service invocation message (3) to the selected provider (Fig. 4b). Then, the selected provider sends a service confirmation message (4) in which the provider informs that it is available go to the place where the service is required.

In the service confirmation message, the provider sends the following information: request identification ($id_{Request}$), provider identification ($id_{Provider}$), and requesting node identification (id_{Client}).

While answering a request, the provider maintains a table with information about this request. That is: id_{Client}, $id_{Request}$, $seq_{Request}$, and *TTL* (*Time To Live*), where *TTL* is the period in which the information will be kept in the table. The provider's TTL is calculated according to Eq. (5).

$$TTL_{provider} \leftarrow \Delta_{tmax} - service_{TimeProvider} \tag{5}$$

Where, $service_{TimeProvider}$ is the maximum time that the provider needs to arrive at the place where the service is required.

During the discovery process that involves discovery, selection and invocation, failures can occur. Failures can have different origins and can interfere or even derail functioning of the service discovery protocol. LADP protocol uses time-out to resend the messages. In this context, we defined that if the provider does not receive the service invocation message within a time limit (defined in Eq. (5)), the provider will exclude the request information. Otherwise, the provider will maintain the request information stored until it finishes the attendance.

Experimental evaluation

In order to evaluate the proposed service discovery protocol, we used the Network Simulator ($NS-2$) (Issariyakul et al. 2009) and the Gauss-Markov mobility model by (Waal & Gerharz 2003). We generate our movement patterns via the well known BonnMotion tool. Several mobility models are supported, by BonnMotion. We chose this model because it eliminates sudden changes in direction and abrupt stops of random models and enables a closer approximation of the actual movement of humans and vehicles. Our aim is to evaluate the LADP protocol performance in reducing the number of response messages without compromising the discovery process.

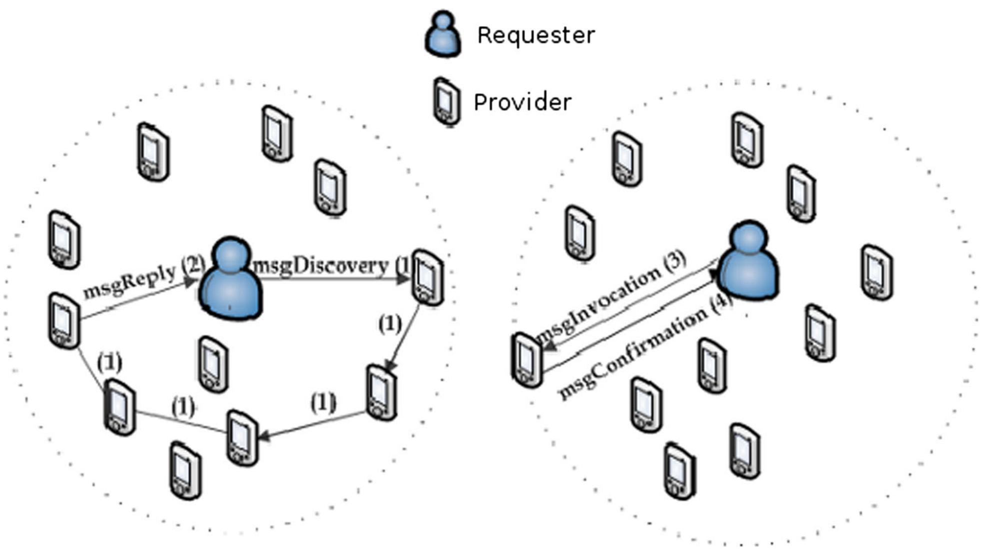

Fig. 4 Transition diagram of the messages in the discovery process

In the simulation, we used the OLSR (Optimized Link State Routing) protocol, (Clausen & Jacquet 2003) to send the service discovery messages to the neighbor nodes and to sent packets back to the requesting node. The key concept of the protocol is the use of multi-point relays (Clausen & Jacquet 2003). This characteristic makes such a protocol particularly suited to the network represented by the scenario proposed herein. However, the mechanisms proposed in this paper may be used in addition to other routing protocols for MANETs, as long as the discovery messages are sent through a diffusion mechanism.

We conducted experiments with four mechanisms: LADS (discovery without selection), LASS with aggregation (named at this point LASS + Fusion), Flooding-1 (traditional flooding), Flooding-2 and Exact Approach. In Flooding-2, we extracted the responses of suitable nodes from the responses obtained with Flooding-1 when these responses reach the requester node, that is, the providers that can reach in time where the resource is needed.

The method called Exact Approach performs outside NS2 and is applying in the mobility model. This method determines at each moment in time the most appropriate providers that fit the request. The input data of this method is compared with the results obtained with the network simulator NS-2.

The simulation environment

The scenario presented in Section II was mapped for different area size in order to evaluate the scalability of the protocol, specifically, 4 (2,000 m×2,000 m), 121 (11,000 m×11,000 m) and 25 km^2 (5,000 m×5,000 m). The number of nodes in the network varies between 50, 100, 150, 200, 250, 300, 400 and 500, and the node speed varies between 0.5 and 15.0 m/s. The parameter α defined=100 ms.

Experiments were carried out where resources were attributed to 1, 5, 10, 15, 20, 25 and 30 % of the nodes in the network. Each provider offers one type of resource. A random distribution of resources was used.

All scenarios started with a 3,600 s initialization phase. The minimum number of providers that must be delivered could be 1, 2 and 3, and the maximum response time was set at 1.5, 5.0 and 10.0 min. All nodes in the network are mobile. However, the requesting node remains static while waiting for the reply. The confidence interval presented in the results is 95 %.

Each node is equipped with the default wireless network energy module that NS-2 provides, which has a 250 m transmission range. We adjusted the transmission power at 42 mW and the receiving power at 55 mW. The initial energy of all the nodes is set at 1,000 J. Each node makes requests during the 28,800 s duration of the simulation. Table 2 represents the parameters used in the experiments.

Table 2 System parameters

Parameter	Value
Field dimension	4, 25, 121 km^2
Number of nodes	50 up to 500
Simulation Time	28,800 s
Transmission range	250 m
Service discovery package size	120 bytes
Response package size	136 bytes
Node speed	1.8 km/h −>0.5 m/s
	2.0 km/h −>0.56 m/s
	3.6 km/h −>1.0 m/s
	5.4 km/h −>1.5 m/s
	7.2 km/h −>2.0 m/s
	18 km/h −>5.0 m/s
	36 km/h −>10.0 m/s
	54 km/h −>15.0 m/s

Metrics

The metrics that were analyzed are: (1) discovery success rate (SD) = number of responses received/number of requests sent; (2) energy consumption (EC) = total energy consumption consumed/initial energy of nodes; (3) quality of response (TQoS) = average time to attend the request/total number of responses received; (4) overhead LADP protocol (OP) = total messages sent by LADP protocol/total messages sent; and (5) packet loss (LK) = number of lost packets/total number of messages generated by LADP.

Results

In the experiments presented below, we evaluate the behavior of the protocol in a small area 4 km^2 (2,000 m×2,000 m) and where the nodes have low travel speed (0.5 m/s and 1.0 m/s). The maximum response time was set at 10.0 min.

–Analysis of LADP data aggregation scheme and fault tolerance mechanism

Figure 5 shows the discovery success rate (SD) according to the node's mobility. The discovery success is the most important metric as it determines whether or not a client discovers a service. The discovery succes rate was obtained per node. We measured the discovery success for the following mechanisms: LADS, LASS + Fusion, Flooding-1, and Flooding-2. In this experiment, 10 % of the participating nodes have the resource searched for, the nodes presented limited energy supply, and the number of providers that must be delivered is one.

For some scenarios we observed that the SD rate obtained with Flooding-2 was lower than the one from the LADS and LASS + Fusion mechanisms. This result is because the Flooding-1 mechanism generates a larger number of responses and as a consequence, increases the number of collisions and dropped response messages.

The LASS + Fusion has enabled a greater response suppression without causing a negative impact on the discovery process, i.e., it has reached the goal of 1 (one) answer for all evaluated speeds.

Figure 6 depicts the packet loss rate (LK). As we can see, Flooding-1 lost a significant number of messages compared with LADS and LASS+Fusion. For example, when 30 % of the participating nodes have the resource, the packet loss rate was 29.9 % for Flooding-1, and 13.5 % for LASS + Fusion. This result shows that aggregating replies can contribute to decreasing collisions and avoiding packet loss.

In Table 3, we present the percentage of success (PE). Percentage of success is based on the replies obtained from the Exact Approach. We implemented the Exact Approach to verify the mobility scenarios, and to show the best provider (s) in each time instant for the requesting node. For this experiment, we calculated the PE among the responses obtained with LADS, LASS + Fusion and Flooding-2, with the responses presented by the Exact Approach. In this experiment, 10 % and 30 % of the participating nodes have the resource. The minimum number of providers that must be delivered is one. Table 3 summarizes the results obtained, where CF (%) means the confidence interval.

Results show that the PE with Flooding-2 was 69 %, and 81.5 % for LASS+Fusion when 30 % of the participating nodes have the resource. The PE of Flooding-2 was lower for scenarios where nodes have more resources or higher speed. This occurred due to collisions that increased dropped response messages. This exemplifies that reply aggregation can greatly benefit the percentage of success by reducing the number of reply messages.

Figure 7 shows the reply message overhead (OP) of LASS + Fusion as compared with LADS and Flooding-1. As we can

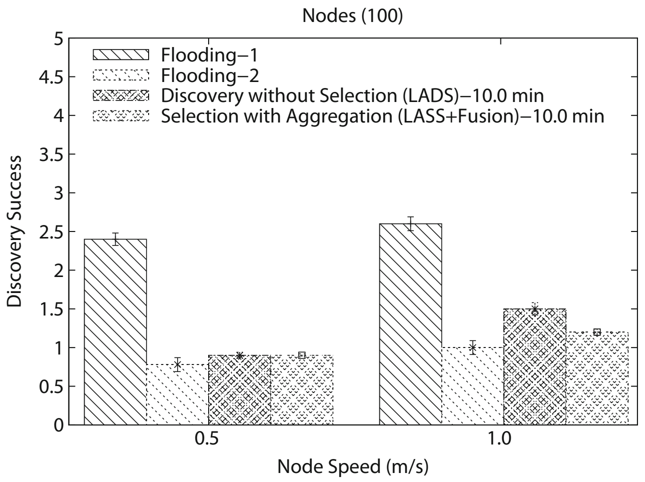

Fig. 5 Discovery success in relation to node speed

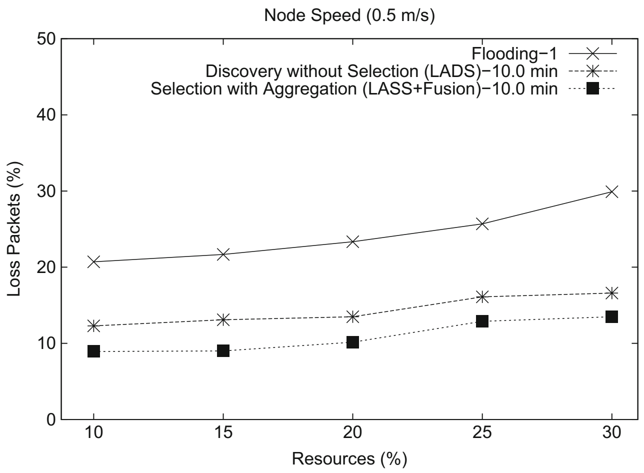

Fig. 6 Lost packets in relation to resource percentage

Table 3 Percentage of success (%) × node speed

Resource	Flooding-2	LADS	LASS+Fusion
10 %	75.0 (CF:1.5)	70.0 (CF:3.0)	71.0 (CF:3.1)
30 %	69.0 (CF:2.5)	77.5 (CF:3.2)	81.5 (CF:3.1)
10 %	77.0 (CF:3.0)	78.5 (CF:3.0)	81.0 (CF:3.3)

see, the OP increases for higher speeds. When the node speed is 1.0 m/s, the reply message overhead is 12.5 % for LASS + Fusion, 14.6 % for LADS, and about 16.8 % for Flooding-1. This result indicates that reply aggregation can be used for reducing the overhead while keeping the discovery success rate.

Figure 8 gives the SD rate according to the n*ij*umber of nodes. The minimum number of providers that must be delivered is 2, 10 % of the participating nodes have the resource, and nodes have limited energy supply.

Results show that LASS + Fusion performs well as the number of network nodes increases. As can be seen in a scenario with 400 nodes, the goal of 2 (two) responses was achieved. Moreover, the reply aggregation scheme can save 40 % of the replies compared with Flooding-1. When the number of network nodes is 200, the SD rate obtained with LASS + Fusion was 1.8 and 1.95 with LADS. LADS generated more responses due to the fact that some nodes within the radius R_i also sent responses. But these nodes don't have enough speed to arrive in time where the resource is needed. We observed that the timer $\Delta_{trequester}$ can provide fault tolerance, since it maintained the desired SD rate, and did not increase the overhead (as we can see in Fig. 6).

Figure 9 shows the response quality received by the requesting node, i.e., the average attendance time (TQoS), which is given by d/vj. The total number of nodes in the network remains 200 and the number of providers that must be delivered is one.

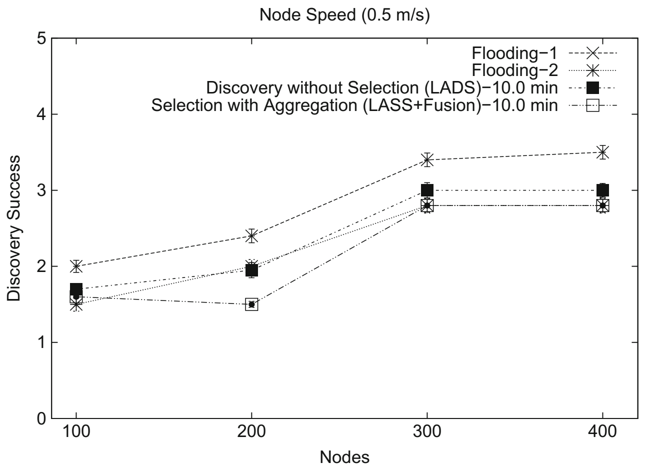

Fig. 8 Discovery success in relation to nodes

The results show that using aggregation scheme, the TQoS was lower compared to LADS and Flooding-1. This happens because during the reply transmissions, LASS + Fusion selects the responses of the best providers and discards those with a longer time limit. We also present the TQoS obtained with the Exact Approach. As we can see in Fig. 9, the TQoS was the same for LASS + Fusion and for the Exact Approach, except when 10 % and 15 % of the participating nodes have the resource. In these cases, the TQoS of LASS + Fusion exceeded the Exact Approach. The results suggest that LASS + Fusion performs better in scenarios with more resource percentage.

In the LASS + Fusion data aggregation scheme, the intermediate node starts the timer, $\Delta_{tintermediate}$, when it receives one response. Until ($max_{provider}$) is not reached and the $\Delta_{tintermediate}$ does not expire, the intermediate node aggregates the received responses. After $\Delta_{tintermediate}$ finishes, the intermediate node aggregates these replies and sends only one response to the requesting node. In the scenarios with low resource percentage, the intermediate node does not receive

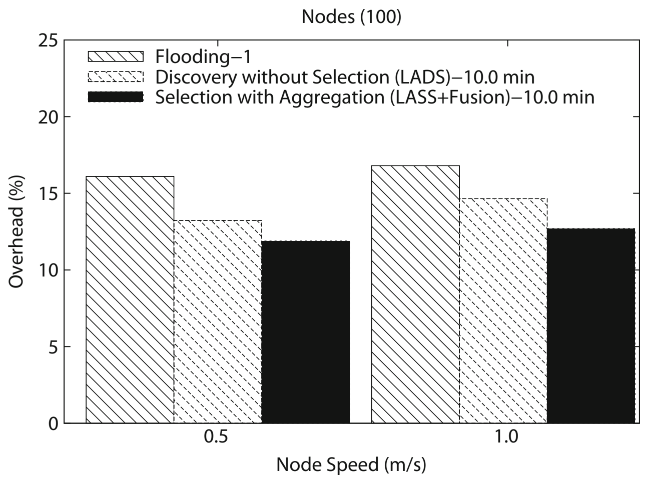

Fig. 7 Overhead in relation to node speed

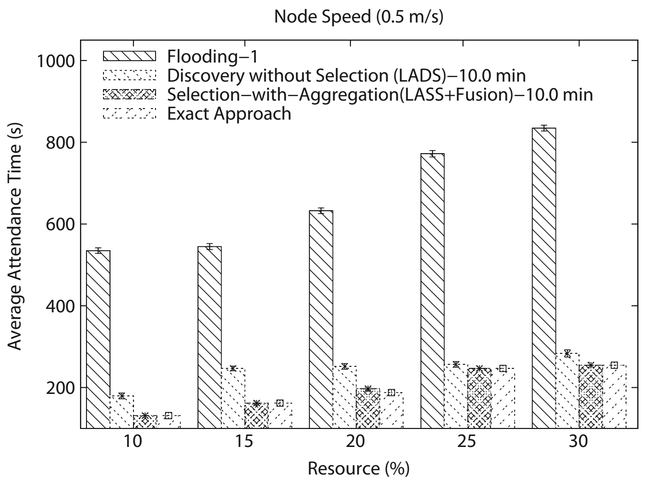

Fig. 9 Time response in relation to resource percentage

enough replies during $\Delta_{tintermediate}$. One solution to this problem would be to increase the value of K. However, the time response of the reply message can increase too.

Energy consumption experiments were performed. In terms of energy consumption, we observed that the LASS + Fusion mechanism outperforms the other evaluated mechanisms without compromising the discovery process.

In Fig. 10, we analyzed the node energy consumption (EC). For this experiment, we compared the LASS Aggregation (LASS + Fusion) mechanism with the LADS mechanism and Flooding-1. The number of nodes in the network is 100. The maximum time to attend the request is 10.0 min, the resource percentage is 10 %, and the node speed varies from 0.5 to 1.0 m/s.

From these results, we observed that the LASS Aggregation mechanism saves considerable energy by discarding responses with a longer time limit. For instance, in a scenario where the node speed is 0.5 m/s, the energy consumption was 41.5 % with Flooding-1, 34.9 % with LADS and 33.7 % with LASS + Fusion.

Through this experiment, we concluded that the use of data aggregation for response suppression in service discovery protocols can reduce the energy consumption of network nodes and increase the network lifetime.

In order to evaluate the fault tolerance mechanism proposed in this paper, an experiment was conducted with the same configuration in Fig. 8, however, the timer $\Delta_{trequester}$ has been disabled. This timer is used by the requester node to set the timeout for a request.

We also compared the discovery success rate for the LASS + Fusion mechanism without the addition of the timer $\Delta_{trequester}$, with the results presented in Fig. 8 using the timer. The number of providers that must be delivered is two. Table 4 summarizes the results.

Results show that with 100 nodes, the discovery success rate was the same using the timer or not. This occurred because new providers were not found to send responses during the period determined by the timer. In denser scenarios (200, 300, and 400 nodes), it was observed that the discovery success rate increased with the use of the timer. One solution to increase the effectiveness of the timer in scenarios with few providers is to reduce the time period of the timer in the requesting node. In this case, the requester will send requests in shorter times.

Table 4 Discovery success rate × nodes

Nodes	LASS+Fusion (timer)	LASS+Fusion
100	1.6	1.6
200	1.7	1.8
300	2.4	2.7
400	2.4	2.7

Also, experiments with larger areas were performed in order to evaluate the scalability of the protocol. These results are presented below.

–Analysis of LADP scalability

In the experiments presented below, the scenario presented in Section II was mapped for an area of 25 km^2. The number of nodes in the network varies between 50, 100, 150, 200 and 250 and the node speed varies from 2.0 to 15.0 m/s. The maximum time limit to service the request is 1.5 min.

Figure 11 shows the discovery success rate (SD) according to the node's mobility. In this experiment, 10 % of the participating nodes have the resource desired and the network has 200 nodes.

The LASS + Fusion mechanism has enabled a greater response suppression without causing a negative impact on the discovery process, ie, it has reached the goal of one (one)

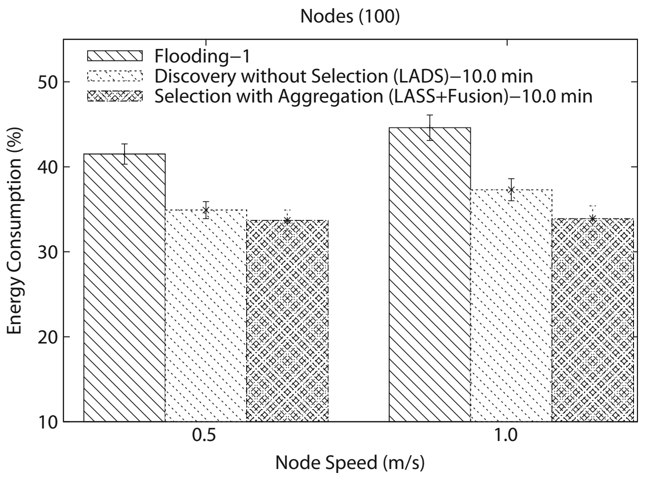

Fig. 10 Energy consumption in relation to node speed

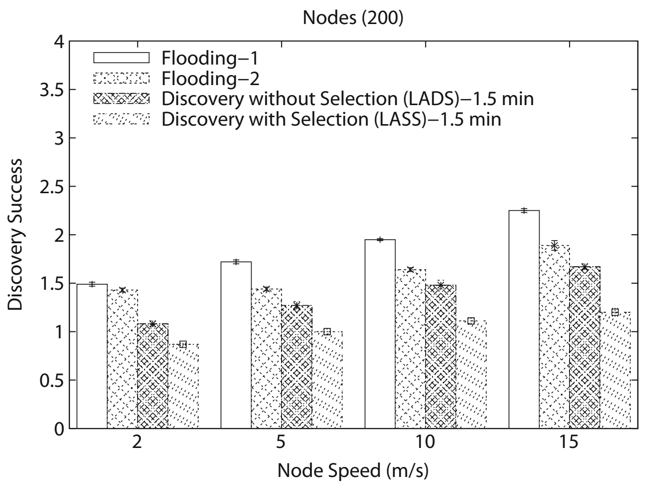

Fig. 11 Discovery success in relation to node speed

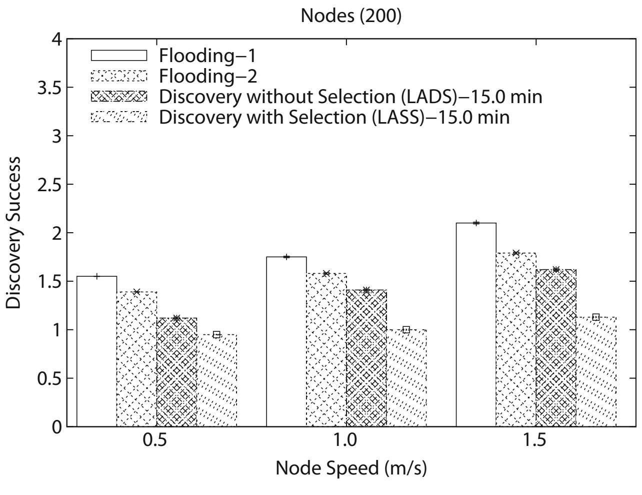

Fig. 12 Discovery success in relation to node speed (15 min)

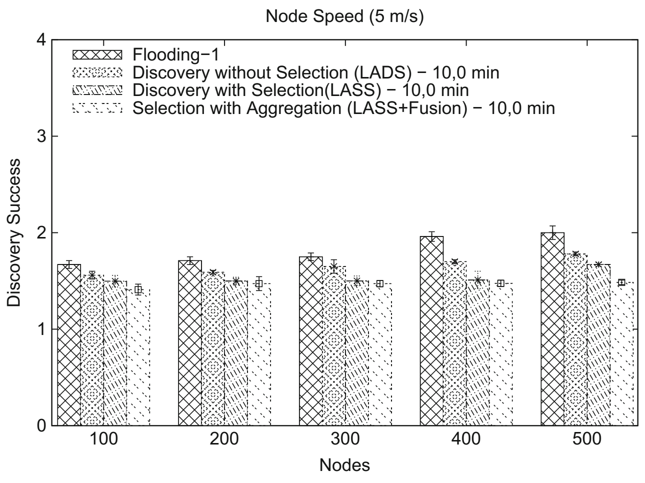

Fig. 14 Discovery success in relation to nodes

response at all speeds, except for the scenario where nodes move at 2.0 m/s. In this case, the discovery success rate was 0.87. This behavior is due to the low number of providers within the radius. Besides that, there was not a substantial change in the topology in order for additional providers to be found.

In Fig. 12, we measured the discovery success rate (SD) related to the node speed. However, unlike previously presented scenarios, the maximum time limit to attend the request is increased to 15 min and the node speed varies between 0.5, 1.0 and 1.5 m/s. Therefore, in this scenario, the radius R_i remains the same as in Fig. 11. In this experiment, 10 % of the participating nodes have the resource desired and the network has 200 nodes. The results obtained were similar to those presented in Fig. 11. These results indicate that the performance is mainly a function of the radius R_i and that a longer attending time limit can be reached if nodes move at proportionally lower speeds.

Simulations have been carried out to evaluate performance of the LADP protocol in sparse areas. In particular, the scenario presented in Section II was mapped for an area of 121 km^2. The number of nodes in the network varies between 100, 200, 300, 400 and 500. The node speed is 5.0 m/s and the maximum time limit to attend the request is 10 min. In these experiments, 10 % of the participating nodes have the resource desired. The number of providers that must be delivered is two. The goal of the experiment is twofold: to verify the discovery success rate, and the energy consumption in sparse areas.

Figure 13 evaluates the energy consumption related to the number of nodes. Checking Fig. 13, the economy of energy was greater with LASS + Fusion. In a scenario with 500 nodes, it was possible to save 33.0 % of the network energy with LASS + Fusion in comparing with the Flooding mechanism and 31.0 % with LADS and 32.5 % with LASS.

Figure 14 shows discovery success rate for the same scenario presented in Fig. 13. We can observe that the goal of two responses was not achieved by the evaluated mechanisms. This behavior is attributed to the fact that the network is sparse

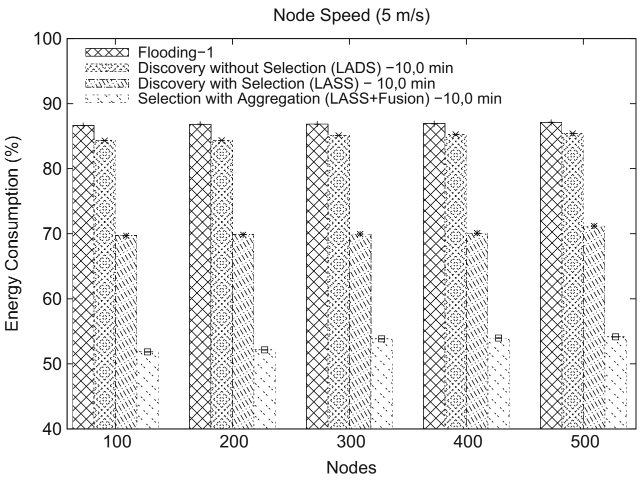

Fig. 13 Energy consumption in relation to nodes

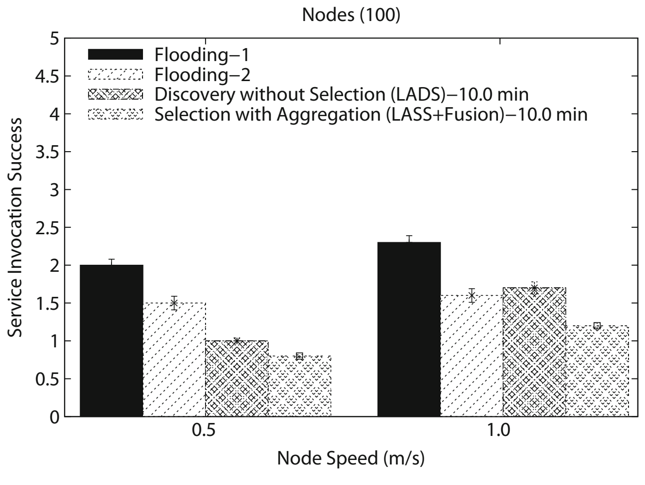

Fig. 15 Service invocation success × node speed

which makes the communication among the nodes difficult. In this case, the effect of LASS + Fusion in sparse scenarios did not add any benefit because the network topology is unfavorable to the response delivery.

Figure 15 shows the service invocation success rate (SI) according to the node's mobility. The service invocation success determines whether or not a client discovers a service after the service invocation phase. SI is defined as number of responses received after the service invocation phase divided by number of requests sent. We measured the service invocation success rate for mechanisms: LADS, LASS + Fusion, Flooding-1, and Flooding-2. In this experiment, 10 % of the participating nodes have the resource and the number of providers that must be delivered is 1. The area size is 4 km^2 (2,000 m$\times$2,000 m).

The results show that LASS + Fusion performs well for all scenarios. However, we observed that the SI rate obtained with LASS + Fusion when the node speed is 0.5 m/s was less than 1. This result is due to loss of invocation or confirmation messages.

Conclusion

This paper presents the Location Aware Service Discovery Protocol (LADP) for MANETs and its mechanisms for service discovery, LADS, and a mechanism for automatic service selection, LASS. The service selection mechanism presents a data aggregation scheme, namely, reply aggregation in intermediate nodes.

The protocol is implemented and analyzed in NS2 and results are shown in Section IV. Results show that the use of LASS + Fusion enabled a reduction in the total number of messages in the network when compared with the other evaluated mechanisms. Furthermore, the LASS + Fusion mechanism outperforms the other mechanisms with respect to node energy consumption through the discards of additional responses without compromising the discovery process. However, in sparse networks with frequent partitions the results show LASS + Fusion did not add benefit. In future works we will intend to investigate the use of Delay Tolerant Networks (DTNs) in sparse scenarios.

We observed that the use of the fault-tolerant service discovery mechanism by means of the $\Delta_{trequester}$ timer contributed keeping the system operating despite the existence of faults. In addition, the LASS mechanism also contributed to improving the response quality received by the user's application.

Another contribution is a service invocation mechanism that specifies how the service providers, chosen by the selection mechanism, are accessed and used. This phase is generally not considered in service discovery protocols, however, it is crucial to ensure the attendance success.

Finally, both mechanisms, LADS and LASS, can be used independently. The only restriction is that the nodes must supply their geographic location, as in the LADS mechanism.

Some aspects not addressed within the scope of the LADP protocol can be interesting future research, including the development of algorithms that collect and analyze information obtained from mobile devices and to identify, from this information, what type of service provider is necessary for a particular type of event.

Acknowledgements This work was primarily supported by Santa Catarina State University (UDESC) and in part by CNPq, CAPES, FAPERJ, TBE/ANEEL and CELESC and ANEEL.

References

Arnold K (1999) *The Jini specification*, ser. Jini technology series. Addison Wesley

Balaji R, Jarmo H, Bilhanan S (2008) "Service discovery framework for manets using cross-layered design,". In: Proc. 6th international conference on Wired/wireless internet communications (WWIC'08). Tampere, Finland, pp 152–163

Chakraborty D, Joshi A, Yesha Y, Finin T (2006) "Toward Distributed Service Discovery in Pervasive Computing Environments," IEEE Trans Mob Comput, pp. 97–112

Chowdhury C, Neogy S (2012) "Reliability of mobile agent system in qos mobile network," in *COMSNETS*, pp. 1–2.

Clausen T, Jacquet P (2003) "Rfc3626: Optimized link state routing protocol (olsr)," *RFC Editor United States*

Gadallah Y, Serhani MA (2011) "A wsn-driven service discovery technique for disaster recovery using mobile ad hoc networks," in Wirel Days, pp. 1–5.

Gomes ATA, Ziviani A, dos L, Lima S, Endler M, Chelius G (2008) "Mitigating reply implosions in query-based service discovery protocols for mobile wireless ad hoc networks," in *The 11th International Conference on* Ad Hoc Netw Wirel (ADHOC-NOW), vol. 5198. Springer pp. 29–42

Guttman E (1999) Service location protocol: automatic discovery of ip network services. IEEE Internet Comput 3(4):71–80

Helal S, Desai N, Verma V, Lee C (2003) "Konark - a service discovery and delivery protocol for ad-hoc networks," in IEEE Wirel Commun Netw Conf (WCNC'03), pp. 2107–2113

Issariyakul, Hossain E(2009) *An Introduction to Network Simulator NS2*. Springer [Online]. Available: http://books.google.com.br/books?id=cD69He_oU60C

Kniess J, Loques O, Albuquerque CVN (2009) Location aware discovery service and selection protocol in cooperative mobile wireless ad hoc networks. In: Proceedings of the 28th IEEE international conference on computer communications workshops (INFOCOM'09). IEEE Press, Piscataway, pp 367–368

Lenders V, May M, Plattner B (2005) Service discovery in mobile ad hoc networks: a field theoretic approach. Pervasive Mob Comput 1(3): 343–370

Marin-Perianu S (2008) "Wireless sensor networks in motion: clustering algorithms for service discovery and provisioning," Ph.D. dissertation, University of Twente, Enschede, [Online]. Available: http://doc.utwente.nl/60028

Marin-Perianu R, Hartel P, Scholten H (2005) *A Classification of Service Discovery Protocols*. Centre for Telematics and Information Technology, University of Twente

Mian AN, Baldoni R, Beraldi R (2009) A survey of service discovery protocols in multihop mobile ad hoc networks. IEEE Pervasive Comput 8(1):66–74

Miller B, Bisdikian C (2001) *Bluetooth revealed*. Prentice Hall PTR

Nidd M (2001) Service discovery in deapspace. IEEE Pers Commun 8(4):39–45

Salutation (1999) “Salutation architecture specification,” [Online]. Available: ftp://ftp.salutation.org/salutesa20e1a21.ps

Serhani M A, Gadallah Y (2010) “A service discovery protocol for emergency response operations using mobile ad hoc networks,” in *Telecommunications (AICT), 2010 Sixth Advanced International Conference on*, pp. 280–285

van der Aalst WMP (2010) Process discovery: capturing the invisible. IEEE Comp Int Mag 5(1):28–41

Varshavsky A,Reid B,de Lara E (2005) “A cross-layer approach to service discovery and selection in manets,” in The 2nd IEEE Int Confon Mob Adhoc Sens. Syst Conf, pp. 459–466.

Waal C, Gerharz M (2003) “Bonnmotion: a mobility scenario generation and analysis tool,”. In: Communication systems group, institute of computer science IV. University of Bonn, Germany