Semantic enabled Resource Discovery, Composition and Substitution in Pervasive Environments

Michele Ruta
Politecnico di Bari, Italy
Email: m.ruta@poliba.it

Tommaso Di Noia
Politecnico di Bari, Italy
Email: t.dinoia@poliba.it

Eugenio Di Sciascio
Politecnico di Bari, Italy
Email: disciascio@poliba.it

Francesco Maria Donini
Università della Tuscia, Italy
Email: donini@unitus.it

***Abstract*— This paper provides a general framework for service discovery, composition and substitution in a mobile ad-hoc network scenario. Knowledge representation techniques and approaches are exploited and adapted to the highly flexible and variable ubiquitous computing environment. A polynomial algorithm for composing and orchestrating mobile resources is outlined. Also is presented a simple approach to determine the compatibility among mobile resources and implement a dynamic substitution of services failed or no longer available.**

I. Introduction

Service oriented Computing is a rapidly emerging paradigm also for Mobile Ad hoc Networks (MANETs). Obviously, peculiarities due to the absence of an infrastructure and the continuously changing of MANETs topology have to be taken into account when it comes to adequately support service discovery and invocation, possibly exploiting semantic descriptions of services in the discovery and composition. Nevertheless, while various routing protocols have been specifically devised for MANETs, *e.g.*, AODV [6], service discovery for MANETs has been basically oriented to adapting protocols originally designed for wired contexts (Jini, Salutation, Service Location Protocol, UPnP or UDDI).

Such service discovery architectures use a centralized and registration-oriented mechanism. Each service advertises and registers itself to a service lookup that keeps track of resources in the network. Nevertheless in ad-hoc infrastructures a decentralized approach is required. A node should not be depending on some other node to advertise/register services. Each resource should be autonomously exposed and at the same time other applications should be able to discover it.

Currently, resource discovery in ad-hoc networks is obtained by broadcasting a service discovery request throughout the network. If a node contains the service responds with a service reply [2]. This is obviously inefficient in terms of both bandwidth and power consuming: the request has to be processed by all the nodes, which have limited processing capabilities and battery duration.

Another approach is the advertising of services to all the nodes in the network [4]. Each node interested in discovery maintains a cache containing the advertisements which will be matched, during the discovery phase, against service requests. Since many nodes have limited memory, they can be unable to store all the advertisements and soon their cache gets filled up. On the other hand also this method is inefficient in terms of bandwidth usage because of the advertisements flooding and does not take into account modifications in resources availability.

In addition, existing service comparison techniques use a simple unique-identifier based matching. Service discovery is performed in a purely string matching fashion [11].

Finally routing protocols for MANETs assume the destination address of data packets is already known to the source device before starting an interaction [6]. Hence these standard routing protocols cannot be conveniently adapted within the application layer for service discovery.

In a mobile environment it is reasonable that resources more frequently and efficiently used are in the device neighborhood and it is important for applications to quickly discover other remote services/resources present on nearby mobile devices. Obviously, in this case standard ad hoc routing protocols are unsuitable for service discovery since the destination address of the service is still unknown.

In this paper we outline framework and algorithms for semantic-based service discovery and composition, which adapt, taking into account MANETs peculiarities, a Semantic Web service composition approach, concentrating on substitutability issues which is of particular interest in a scenario where the structure is very volatile as the MANET one.

In this approach, given a semantic based specification (request) described using a subset of OWL-DL [10] of user requirements, we are able to carry out discovery and composition of mobile resources covering as much as possible the needs of requester and orchestrating them according to specified prerequisites. The proposed approach also copes with non exact matches. That is, those cases where the user request is partially satisfied or, in other words, it is not completely covered. Furthermore when discovered resources are unsuitable to fulfill the request, an explanation of what is missing to fully cover the demand is provided.

In order to increase the flexibility and the fault tolerance of the system, our composition protocol integrates the substitutability among mobile resources. Considering the service discovery and composition as a collaborative process, the analysis of the compatibility between two services allows to support the dynamic substitution in case of failure or unavailability. As soon as a set of substitutes for a resource is identified, the orchestrator can get a candidate for the substitution automatically taking into account preconditions it wants and effects it produces.

II. Service Discovery Framework

In the rest of the paper we will assume the reader be familiar with basics of Semantic Web languages –in particular OWL– and of Description Logics (DLs) [1], the formal language we adopt here.

The Service Discovery Protocol we consider is based on service advertisement mechanism. An advertisement is composed by an OWL annotated semantic description of the service offered by a mobile device in the MANET [1]. Obviously the service advertising implies the existence of a cache memory for each mobile device. Each entry of this cache stores, among other things, the semantic description of a service, the identifier of its reference ontology and a *timestamp* updated when a service is stored or is used. In what follows both service advertising mechanism and caching one are outlined.

Each node hosting one or more services "advertises" them to nodes in its direct proximity. All the offered resources are semantically annotated using OWL-based ontologies.

The advertisements are sent every fixed time interval and also a node could decide to send them across multiple hops. In this case it has to specify the maximum propagation distance for the advertisement. The greater the number of hops, the greater the number of nodes in the vicinity receiving an advertisement. However, care has to be paid in avoiding an increase of advertisement packet flooding with subsequent cost of increased number of messages exchanged.

In what follows we briefly outline fields of an advertisement frame. The information about refresh time occurring between two subsequent sending of a service descriptions is a parameter of the advertisement packet. It depends on the mobility of the node and on the general variability of the MANET. For example in highly dynamic ad-hoc networks, it is desirable to have a small refresh interval of advertisement sending since the network topology could rapidly change. Another parameter in the advertisement packet is its propagation distance. It helps us in regulating the extension of the advertisement travel and could be used selectively in different mobility situations. Figure1 shows an example of a MANET where some resource providers advertise their services to nearby hosts.

Each semantic service description is expressed with respect to a specific ontology identified by a numeric identifier. This information will be used in the matching phase. In Figure2 the structure of an advertisement frame is shown.

Notice that the triple *(ontology identifier, advertisement identifier, source address)* identifies in an unambiguous way a specific service description. This identifier labels in a different way different service descriptions referred to the same ontology. Furthermore, when an advertisement is resent by a service provider (after its refresh time is expired), its associated advertisement ID is reused.

A simple example will make clear this feature. Let us suppose we have three nodes providing services in the MANET:

[1]Currently we are working to OWL coding in order to reduce the payload of each packet due to the XML syntax.

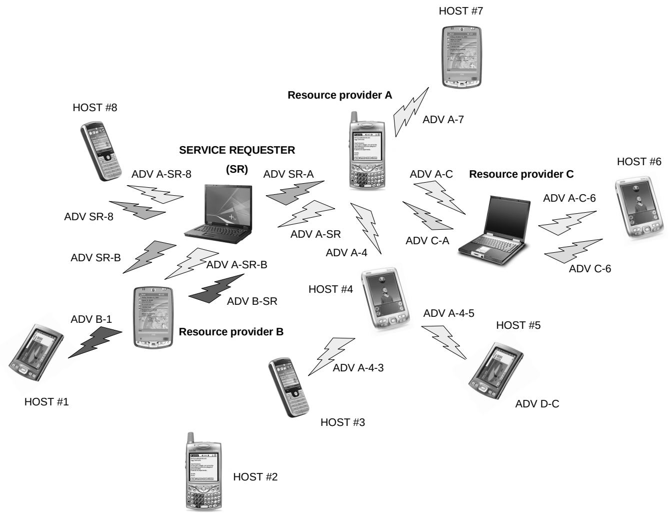

Fig. 1. Advertisement propagation mechanism. *Resource provider A* sets a two hops travel distance

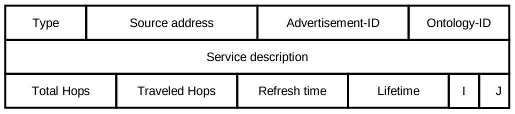

Fig. 2. The advertisement PDU elementary framework

host#1 manages the service *X* referred to the ontology *i*, *host#2* manages both the service *Y* and the service *U* both referred to the same ontology *j* in addition to the service *W* referred to the ontology *k*, and the *host#3* manages the service *Z* also referred to the ontology *i*. Corresponding service descriptions will be classified respectively as:

- (i_ontology-ID, adv-ID=0, host#1_address)
- (j_ontology-ID, adv-ID=0, host#2_address)
- (j_ontology-ID, adv-ID=1, host#2_address)
- (k_ontology-ID, adv-ID=0, host#2_address)
- (i_ontology-ID, adv-ID=0, host#3_address)

Since in the proposed approach we adopt different kinds of PDU (Protocol Data Unit) for different purposes, the *Type* field allows to rapidly distinguish among them.

TYPE	BIT SET	KIND OF PACKET
A	00	Advertisement packet
B	01	Cache entry packet
C	10	Solicit packet
D	11	available

TABLE I
VARIOUS KIND OF PDUS USED IN THE PROPOSED FRAMEWORK

Packet of B and C type will be analyzed hereafter.

The pair *(Total Hops, Traveled Hops)* in the advertisement PDU allows to establish if an advertisement has to be forwarded or if it has accomplished its travel. Notice that each node receiving an advertisement, increases the value in the field *Traveled Hops* and forwards the frame only if the increased value is lower than that contained in *Total Hops* field. The following algorithm summarizes the advertisement forward management by a node.

Algorithm $advForward(Trav\_Hops, Tot\_Hops)$

input the traveled hops value $Trav\_Hops$ and the number of total hops to be travel Tot_Hops
output the forwarding flag f_fwd

```
begin algorithm
  reset $f\_fwd$;
  $Trav\_Hops = Trav\_Hops + 1$;
  if $Trav\_Hops < Tot\_Hops$ then
    set $f\_fwd$
  return $f\_fwd$;
end algorithm
```

If the f_fwd flag is set, the advertisement PDU will be broadcast to nodes in radio range.

The *Refresh time* field indicates the time interval which has to elapse for receiving a further sample of the same advertisement, whereas the *Lifetime* field represents the duration of an advertisement. Generally a service provider should impose a *Refresh time* smaller than *Lifetime*.

Last fields in the advertisement PDU are two bits *I* and *J*. They allow to establish if a packet has to be sent in uni, multi or broadcast:

I BIT	J BIT	KIND OF TRANSMISSION
0	0	Unicast
0	1	Multicast
1	0	Broadcast
1	1	available

TABLE II

PDU TRANSMISSION METHODS IN THE PROPOSED APPROACH

Each node in the MANET maintains a cache to store its service descriptions (if it manages a resource) as well as those of advertised remote hosts. The cache we implement realizes a simple *Least Recently Used* (LRU) mechanism for data updating. The *timestamp* marks the last reference to the service (advertisement arrived or service used). That is when a new service is stored in the cache or when an existing one is invoked, the *timestamp* field is updated.

A lifetime mechanism completes the LRU one. Each entry of the cache has a specified field (lt) which is updated when an advertisement packet arrives and it is progressively decreased. An entry could be removed if its lifetime has expired.

To store local resources in the cache memory we use a further field $local$, which is set for services resident on the node. Recall that the *Service Cache* of a resource provider contains a list of local as well as remote services that the node is aware of.

The $lifetime$ of a service description is fixed and it is initially established by a resource provider according to its mobility conditions (typically fast moving nodes should impose a small lifetime for their advertisements). Nevertheless it could be partially reduced by the receiver according to the remaining capacity of its cache.

In particular we suppose a receiver could reduce the original $lifetime$ value only if the remaining capacity of the cache is lesser than 10% of the total. The value of reduction is set according to the availability of entries in the cache. In particular reduction mantains a direct proportion w.r.t. the cache occupancy. The greater the availability of cache positions the lower is the imposed reduction. Lifetime value is calculated by a node by means of the following formula: $lifetime = lifetime \cdot [1 - s \cdot H(s)]$ where H is the Heaviside step function and s is:

$$s = \frac{total - available}{total} - 0.9$$

where $total$ is the total number of entries in the cache memory whereas $available$ is the number of available ones.

The generic structure of a cache entry is outlined below.

srvc descr	source address	ontology ID	advert ID	local	lt	timestamp	traveled hops

Fig. 3. A generic entry of the *Service Cache* in each mobile device

A fundamental role is covered by the $ontologyID$ field. In fact before starting any service composition, the selection of component services is made on the basis of the compliance of the ontology $\mathcal{T}$ among demand and supplies.

III. Semantic Based Service Composition

Here we show how to compute a semantic based automated composition of mobile services and how to orchestrate them for building personalized resources.

Let us suppose a generic client submits a request for a resource in an ad-hoc network and sets together with its request also both a minimum covering threshold and the maximum discovery distance (in terms of number of hops).

We basically hypothesize a requester-centric composition system. The requester searches for a complex service and collects various component resources coming from nodes at one hop. Hence, it attempts to cover the request starting the composition algorithm. If it fails to overcome the minimum covering threshold, it will request to nearby nodes all the service descriptions in their respective cache. This step has to be repeated until the threshold is surpassed or the maximum hops number is reached.

Note that, in the proposed approach, the requester is exactly the service orchestrator and other nodes in the ad-hoc environment assume only a passive role.

We are not necessarily interested in a full satisfaction of the request; we want to satisfy it as much as possible. If the system is not able to find a set of adequate resources from the repository it forwards a "solicit packet" to hosts in its direct vicinity (at distance of one hop, that is in its radio range) and looks for a response. Nearby hosts answer with service descriptions contained in their cache. Hence the service requester again attempts to cover the request.

Finally, if retrieved resources do not allow to completely fill the request, an approximate solution has to be taken into account, possibly explaining the approximation.

Observe that the composition is totally decentralized. In fact various resource providers take part to it. Each host concurs to cover the whole request or part of it by means of resource descriptions in its cache. If the resulting composite mobile service does not allow to surpass the minimum covering threshold, the requester could probably give up some part of the demand.

To explain and motivate the approach and the rationale behind it, we present a simple semantic based service discovery model and we add to it resources composition features, enriching the model. In the initial architecture we define both the request D and the description of each available resource on a mobile device, as DL concept descriptions w.r.t. an ontology $\mathcal{T}$shared among some users in the network.

We perform a preliminary selection procedure based on ontology numeric identifiers to extract devices that manage the same ontology $\mathcal{T}$. Hence, given a request D modeled w.r.t. $\mathcal{T}$, in what follows we briefly outline the service discovery algorithm applied by the service requester to retrieve resources possibly satisfying her/his demand:

1) Reset *current hop* counter.
2) Extract service descriptions in the requester cache.
3) Select all the resource descriptions which refer to the same $\mathcal{T}$.
4) Put all the retrieved descriptions in a set $\mathcal{R}$.
5) Call resource composition algorithm with input $\mathcal{R}$, D and $\mathcal{T}$.
6) There is an exact solution?
   a) If yes, continue.
   b) If not, go to step 8.
7) The algorithm outputs the set of resources representing the exact solution to the retrieval problem. Exit.
8) The covering level between a demand and the set of retrieved resources is under a specified limit (minimum threshold)?
   a) If yes, continue.
   b) If not, go to step 13.
9) Increase *current hop* counter.
10) Have been reached maximum distance range (MAX_{hop})?
    a) If yes, go to step 13.
    b) If not, continue.
11) Broadcast a *solicit packet* up to nodes at *current hop* distance. Reached nodes response with their service cache content.
12) Collect arrived service descriptions and steps from 2 to 5 are repeated w.r.t. these semantically annotated resources.
13) The algorithm outputs the set of resources representing an approximate solution to the retrieval problem and an explanation on why the solution is not an exact one. Exit.

To better explain the above composition framework we will refer to the simple MANET pictured in the previous Figure1. It is modeled in the Figure4 below. Firstly the requester use service descriptions in its cache to cover the request. Hence no traffic is induced in the ad-hoc network.

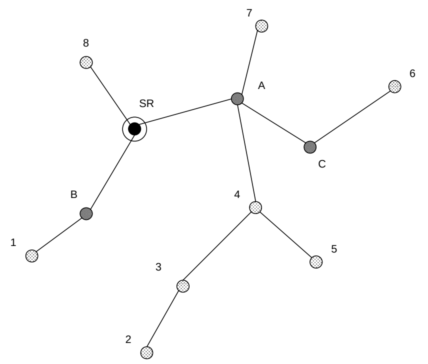

Fig. 4. A simple scheme of the MANET pictured in Figure1

Let us suppose the requester imposes a specific minimum covering threshold and a maximum search range up to three hops. If the obtained covering is under the threshold, the requester will ask for other services potentially covering the request. Figures 5–7 show the traffic in the MANET generated respectively by a two and three hops range search.

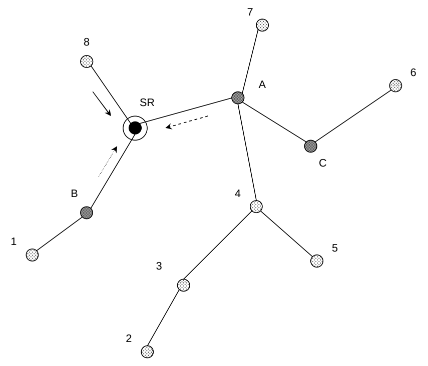

Fig. 5. The service requester searches for further services from nearby hosts

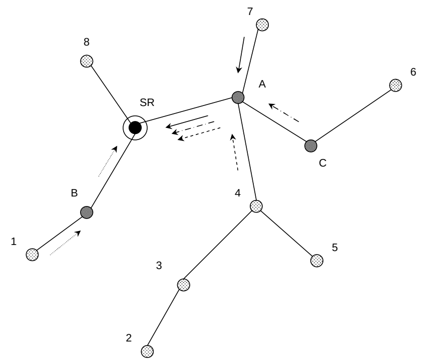

Fig. 6. A two hops solicit request has been submitted by the requester

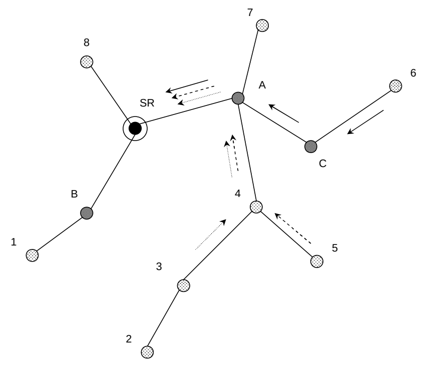

Fig. 7. A three hops solicit request has been submitted by the requester

Finally in Figure8 we report a qualitative Gantt diagram of operations performed by each host in the MANET. Note that the extension of service search range allows to involve a greater number of hosts with a subsequent improvement of covering level, but also with an increase of the network flooding.

The composition of discovered services in the MANET requires to take into account also their execution information,

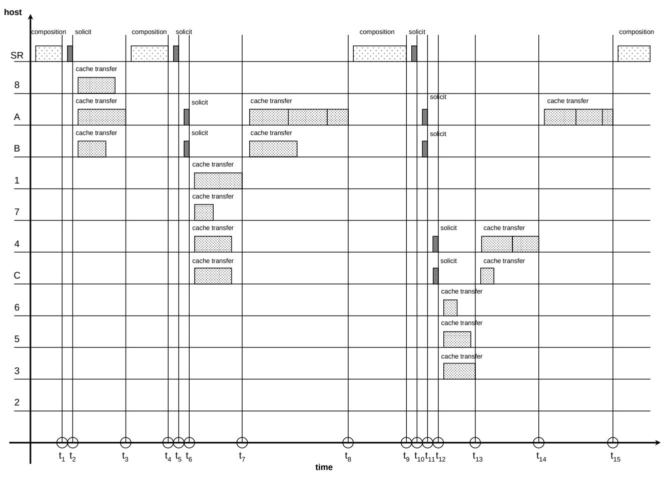
host
SR
8
A
B
1
7
4
C
6
5
3
2
composition
solicit
cache transfer
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15}
time

Fig. 8. The temporal sequence of operations in the reference MANET

i.e., inputs, outputs, precondition and effect specifications. In fact some services may set specific execution requisites to be carried out. Here if a device does not manage them, it cannot use the service. Without loss of generality, in what follows we will consider only preconditions and effects, as it is straightforward to extend the approach also considering inputs and outputs. Now we briefly explain precondition and effects function in the orchestration of mobile services and we show how to compose resources.

In order to use a resource on a mobile device, its preconditions must be satisfied and, if the resource is a component of a service set, this has to do possibly using information provided by other component services. Moreover when composing services the duplication of effects can happen.

Here we extend the service composition model in [8] to deal with a pervasive scenario. For the sake of clarity we only recall main terms and definitions. In particular we define:

Mobile Service: a triple $\langle MS_D, P, E\rangle$ where MS_D is the description of provided service, P its preconditions and E the effects.

Furthermore, indicating with AI_i the available information for the i-th mobile service ms_i and with E_j the effects produced by ms_j, with $j < i$, the following relation ensues: $AI_i = P_0 \sqcap E_1 \sqcap E_2 \sqcap ... \sqcap E_{i-1}$.

Based on the definition of mobile service flow with respect to some initial preconditions P_0 –from now on $\mathcal{MSF}(P_0)$ [8]– here we define a **composite mobile service** with respect to a request D.

A ***composite mobile service*** for $\langle D, P_0\rangle$ w.r.t. the set of discovered mobile services $\mathcal{R}$, from now on $\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle)$, is a mobile service flow such that for each ms_j in the execution flow: $D_{\mathcal{CMS}(\langle D,P_0,\mathcal{R}\rangle)} = \{MS_D(j)|ms_j \in \mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle)\}$ covers D.

An ***executable mobile service*** ms^{ex} for $\mathcal{MSF}(P_0)$ is a mobile service which can be invoked after the execution of $\mathcal{MSF}(P_0)$, *i.e.*, its preconditions are satisfied after the execution of $\mathcal{MSF}(P_0)$, and such that its effects are not already provided by $\mathcal{MSF}(P_0)$.

The *serviceComposer* algorithm is the core of the system. The entire ad-hoc network has to be imagined as a black box rapidly in evolution where mobile resources are internal components invisible to the requester. They can be composed to give in output a complex service to the user. In input, among other things, we submit preconditions that must be satisfied as well as the maximum discovery distance (search diameter) and the minimum threshold covering level (see Figure11).

In order to allow a primary composition with services in the cache of the requester device, the algorithm will be run in here. It outputs the $\mathcal{CMS}$ as well as $D_{uncovered}$. If the covering level is under a specified threshold, the uncovered demand as well as the temporary resulting $\mathcal{CMS}$ are stored and the requester broadcasts a solicit packet to nodes at one hop for requiring other service descriptions.

Before starting the solicit packet, the requester increases the *hop counter* and verifies if the maximum discovery distance has been reached. If not, the frame starts. When a node receives the PDU (shown in Figure9), it firstly increases the content of the field *Traveled hops* and then verifies if the increased value is equal to the content of the field *Total hops*. If this condition holds, the node sends back its cache content (by extracting only service descriptions referred to the same *Ontology-ID* contained in the solicit packet). Otherwise the solicit packet is forwarded to nodes at one hop of distance.

Type	Source address	Total Hops	Traveled Hops	Ontology-ID
Request-ID	I	J		

Fig. 9. Structure of the solicit frame

The algorithm below represents the steps followed by a node in the MANET after receiving a solicit packet. It has to establish if the PDU must be forwarded or if it must send back its cache content.

Algorithm $advForward(Trav\_Hops, Tot\_Hops)$

input the traveled hops value $Trav\_Hops$ and the number of total hops to be travel Tot_Hops
output the forwarding flag f_fwd

```
1  begin algorithm
2    reset $f\_fwd$;
3    $Trav\_Hops = Trav\_Hops + 1$;
4    if $Trav\_Hops < Tot\_Hops$ then
5     set $f\_fwd$
6    else
7     select(srvcdescr) from(cache) where(ontology ID=Ontology-ID)
8     for each $srvc\ descr$
9      create $cacheTransfer$ packet;
10    end for each
11   end else
12   return $f\_fwd$;
13 end algorithm
```

A cache transfer PDU is sketched in Figure10. In it we have the previously described *Type* field, the source and destination address fields, the ontology identifier as well as the semantic service description. Furthermore the *Request-ID* field, also present in the solicit PDU, allows to distinguish within different solicits coming from the same requester and referred to the same ontology identifier. Finally the *I*, *J* bits allow to establish the kind of transmission for the packet (uni, multi or broadcast), whereas the *L* one is set if the PDU represents the last entry of a cache and then the requester should not wait for any following packet (that is the composition of discovered resources can start).

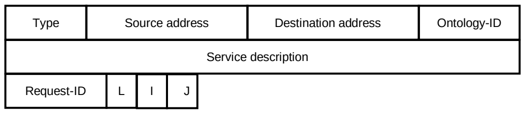

Fig. 10. The PDU prototype of the cache content of a node in the MANET

When the service requester, after submitting a solicit packet, receives descriptions of resources belonging to other far service providers in the MANET, it runs again the composition algorithm and tries to better cover the request. If the covering level is still under the threshold, it can newly broadcast a solicit packet to other nodes in radio range increasing by one hop its destination radius. Obviously is desirable to discover and compose services close to the requester.

Notice that in such a scenario the discovery and covering of services is performed by an association of nodes which take part to the final result. In order to make that, the composing algorithm has to be run in subsequent and coordinate sessions. This is a significant feature w.r.t. classical service discovery mechanisms. In fact the discovery and composition of resources in the MANET is really **not centralized**.

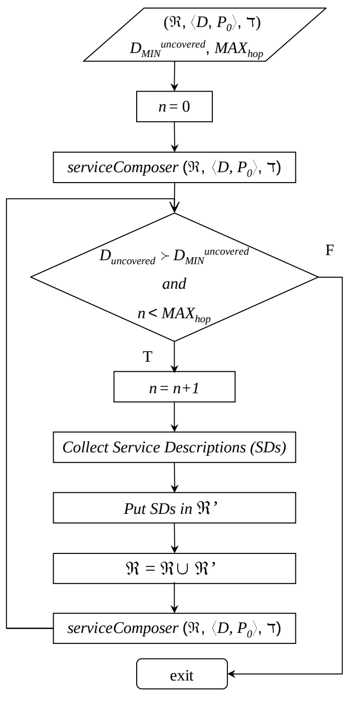
$(\Re, \langle D, P_0\rangle, \neg)$
$D_{MIN}^{uncovered}$, MAX_{hop}
$n = 0$
$serviceComposer\ (\Re, \langle D, P_0\rangle, \neg)$
$D_{uncovered} \succ D_{MIN}^{uncovered}$
and
$n < MAX_{hop}$
F
T
$n = n+1$
Collect Service Descriptions (SDs)
Put SDs in $\Re'$
$\Re = \Re \cup \Re'$
$serviceComposer\ (\Re, \langle D, P_0\rangle, \neg)$
exit

Fig. 11. The flow chart of the composition framework in the MANET

In the composition algorithm we take into account the influence of distance between offered and demanded service, the structure of the communication among hosts (service providers) in a MANET and finally the dynamic structure of the composition. In fact the set of components services is not assigned a priori, but it changes according to network characteristics. W.r.t. web services composition on the wired systems [5], in this case we have more subsequent composition processes. The orchestration goes through several steps producing a progressive refinement of results.

Algorithm $serviceComposer(\mathcal{R}, \langle D, P_0\rangle, \mathcal{T})$

input a set of services $\mathcal{R} = \{ms_i = \langle MS_D(i), P_i, E_i\rangle\}$, a request $\langle D, P_0\rangle$ - where D and $MS_D(i)$ are satisfiable in $\mathcal{T}$-
output $\langle \mathcal{CMS}, H\rangle$

```
1  begin algorithm
2    $\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle) = \emptyset$;
3    $D_{uncovered} = D$;
4    $H_{min} = D$;
5    do
6      compute $\mathcal{EX}_{\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle)}$;
7      $MS_{Dmin} = \top$;
8      for each $ms_i \in \mathcal{EX}_{\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle)}$
9        if $\mathcal{D}_{\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle)} \cup \{MS_D(i)\}$ covers $D_{uncovered}$ then
10         $H = solveCAP(\langle \mathcal{L}, MS_D(i), D_{uncovered}, \mathcal{T}\rangle)$;
11         if $H \prec H_{min}$ then
12           $MS_{Dmin} = MS_D(i)$;
13           $H_{min} = H$;
14         end if
15       end if
16     end for each
17     if $MS_{Dmin} \not\equiv \top$ then
18       $\mathcal{R} = \mathcal{R} \backslash \{ms_i\}$;
19       $\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle) = (\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle), ms_i)$;
20       $D_{uncovered} = H_{min}$;
21     end if
22   while($MS_{Dmin} \not\equiv \top$);
23   return $\langle \mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle), D_{uncovered}\rangle$;
24 end algorithm
```

In spite of increasing of covering possibility, the involvement of nodes more and more far in the MANET implies a greater risk in persistence of links among requester and set of providers. Hence it is useful to define a metric which takes into account distance (in terms of number of hops) from service requester to service providers for weighting the matching degree. Services which are "located" on mobile devices in proximity of requester have the maximum weight and this weight progressively reduces with distance. A logarithmic function is adopted. In fact it presents a growth more or less proportional with the distance for a short number of hops, but assumes values almost constant over a specified limit.

In line 11 of the algorithm we correct the "semantic distance" from offered service to request by computing the influence of "physical distance" from the requester. A $(1 + log_{10}n)$ factor –where n is the number of hops from requester to provider– is inserted.

Services at one hop of distance, *i.e.*, in the radio range of requester, have a *semantic distance* determined by their effective semantic similarity w.r.t. the demand. Instead, for services at two or more hops of distance, the *semantic distance* is increased of a logarithmic factor. Notice that distances over 10 hops substantially double the *semantic distance*. Nevertheless, if there is an exact match the influence of physical distance is absent. This means that, in the presence of a possible exactly matching service, we can tolerate the overload and the risk in using far resources.

IV. Service Substitutability

Within discovery phase, all the services composing a set should be identified. Nevertheless, especially in a pervasive environment, rarely all of them are simultaneously available. In fact, during the execution, a service could fail and second, because the host is mobile, it could not be reached. Furthermore, since a MANET is a complex system continuously in evolution, while the composition step is in progress, a better resource could being detected as well as newer releases of an already discovered service could be now available. In these cases, in a dynamic fashion, the system should substitute mobile services no more suitable with better ones.

To implement the substitution feature we define a *Similarity Group* as a collection of component services which can be substituted with each other [7]. Now the classification of an object to obtain its belonging to the group is the central question. A set of rules for substitution of a mobile resource with another one has to be defined [3].

Notice that the *Similarity Group* (from now on $\mathcal{SG}$) is a simple set of services without any order relation. This is strictly correlated to the variable structure of a MANET. So it is unnecessary to order the class of substitutable services because they are too instable. The only reasonable order relation should be based on the vicinity among resources, but due to the host mobility it is inapplicable to the $\mathcal{SG}$.

Also notice that a $\mathcal{SG}$ is created w.r.t. each resource in the $\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle)$ so each constituent service can have a set of substitutes.

To build the $\mathcal{SG}$, information about candidate substitutes are required prior to admit them in the substitutability class [9]. Furthermore we need some data about the interface of a resource, *i.e.*, required preconditions and provided effects; this is essential for evaluating its correct inserting in the $\mathcal{MSF}(P_0)$. In order to decide if a generic service can belong to a *Similarity Group* two conditions about preconditions and effects have to be verified. Let us suppose $(ms_1, ms_2, ..., ms_N)$ are services in a $\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle)$ and let us imagine we must constitute the ms_i similarity group $\mathcal{SG}$(i). We say $ms_j^{sub}(j = 1...L) \in \mathcal{SG}$(i) iff the following conditions hold:

1) ms_j^{sub} is an *executable mobile service* for $(ms_1, ms_2, ..., ms_{i-1})$
2) for $h = i + 1...N$, ms_h is an *executable mobile service* for $(ms_1, ms_2, ..., ms_{h-1})$

This is the theoretical framework but, in practice, the substitutability is implemented in a more compact way. In fact, to verify if a substitute service ms_j^{sub} is suitable, we take into account the already available AI_i (if ms_i is the service to substitute) and we recalculate next AI_k (with $k = i + 1...N$) checking the satisfiability of the respective ms_k. In other words if AI_i is the available information for ms_i and if we want to substitute the same ms_i, the new $AI'_{i+1} = AI_i \sqcap E_j^{sub}$ has to be determined. Hence it will be used for checking the satisfiability of the next ms_{i+1} and so on. If one of these checks fails, we can conclude that the ms_j^{sub} is effectively unsuitable. In the progressive substitution of services in a $\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle)$ it is desirable to start with first services in the flow to reuse the already determined AI in the next substitution steps and consequently reduce the overhead deriving from this processing.

Hence if the generic ms_i belonging to $\mathcal{CMS}(\langle D, P_0, \mathcal{R}\rangle)$ suddenly turns unavailable, we can take a generic ms_j^{sub} from the $\mathcal{SG}$(i) with $j = 1...L$ and substitute it.

Notice that the *Similarity Group* of the service ms_i is created at discovery phase, but it is more and more enriched while the discovery progresses, that is new services are discovered extending to the next hop the search. Hence, if $\mathcal{SG}^k(i)$ is the *Similarity Group* of the service ms_i at hop k, we can say finally $\mathcal{SG}(i) = \mathcal{SG}^1(i) \cup \mathcal{SG}^2(i) \cup ... \cup \mathcal{SG}^{MAX_{hop}}(i)$.

In addition to the fault tolerance feature, the substitutability can also be used as load balancing of resources in case of high concentration of them on a single device or group of devices.

V. Conclusion

We have proposed a framework to enable the service discovery, composition and substitution in ubiquitous environments. Functionality tests have been carried out in a reduced simulation environment. We are currently working on a full evaluation of the approach by adapting *ns-2* simulation environment to our framework.

Acknowledgments

We wish to acknowledge partial support by projects FAR-PITAGORA, CNOSSO, EU-STREP TOWL and ACAB-C2.

References

[1] F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.

[2] M. Barbeau. Service discovery protocols for ad hoc networking. In *Workshop on Ad-hoc Communications (CASCON '00)*, 2000.

[3] L. Bordeaux, G. Salaün, D. Berardi, and M. Mecella. When are two web services compatible? In *TES*, Lecture Notes in Computer Science, pages 15–28. Springer, 2004.

[4] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha. Gsd: A novel group-based service discovery protocol for manets. In *4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN)*. IEEE, September 2002.

[5] S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, and A. Ragone. Fully automated web services orchestration in a resource retrieval scenario. In *ICWS '05*. IEEE, 2005.

[6] S. Das, R. Castaneda, J. Yan, and R. Sengupta. Comparative performance evaluation of routing protocols for mobile ad-hoc networks. In *7th Int. Conf. on Computer Communications and Networks (IC3N '98)*, pages 153–161, October 1998.

[7] V. De Antonellis, M. Melchiori, B. Pernici, and P. Plebani. A methodology for e-service substitutability in a virtual district environment. In *CAiSE '03*, LNCS, pages 552–567. Springer, June 2003.

[8] T. Di Noia, E. Di Sciascio, F. Donini, A. Ragone, and S. Colucci. Automated semantic web services orchestration via concept covering. In *WWW '05*, pages 1160–1161. ACM, 2005.

[9] M. Mecella, B. Pernici, and P. Craca. Compatibility of e-services in a cooperative multi-platform environment. In *TES*, Lecture Notes in Computer Science, pages 44–57. Springer, 2001.

[10] OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.1/overview/.

[11] M. Ruta, T. Di Noia, E. Di Sciascio, F. Donini, and G. Piscitelli. Semantic based collaborative p2p in ubiquitous computing. In *WI '05 Web Intelligence*. IEEE/WIC/ACM, 2005.