GSD: A Novel Group-based Service Discovery Protocol for MANETS

Dipanjan Chakraborty, Anupam Joshi, Yelena Yesha, Tim Finin
Computer Science And Electrical Engineering.
University of Maryland, Baltimore County.
Email: {dchakr1,joshi,yeyesha,finin}@cs.umbc.edu

***Abstract*— This paper proposes a novel distributed service discovery protocol for Mobile Ad hoc Networks. The protocol is based on the concept of peer-to-peer caching of service advertisements and group-based intelligent forwarding of service requests. It does not require a service to register to a registry or lookup server. Services are described using an ontology based on the DARPA Agent Markup Language (DAML+OIL). We exploit the semantic class/subClass hierarchy of DAML to describe service groups and use this semantic information to selectively forward service requests to respective nodes. DAML-based service description helps us in achieving increased flexibility in service matching. We also present simulation results of our protocol and show that our protocol achieves increased efficiency in discovering services by efficiently utilizing bandwidth by controlling forwarding of service requests.**

***Index Terms*—MANET, DAML+OIL, Service Discovery, Group-based Selective Forwarding, Semantics, Broadcast**

I. Introduction

Resource discovery and management is a challenge for Mobile Ad hoc Networks (MANETS). Research in the field of MANETS has primarily focused on routing and several routing protocols like DSR [7], AODV [12], TORA [10], DSDV [11] have come up in the past few years. They primarily deal with the problem of managing the flow of data from a mobile node in the MANET to another node in the same MANET with the help of intermediate mobile nodes. However, the primary assumption in such routing protocols is that the destination address of the data packet is known to the source beforehand.

Applications of top of MANETS often times need to utilize resources or services that are present on other mobile nodes in its neighborhood. This leads to greater utilization of resources that are available in our service-rich vicinity. Hence, it is important for applications to be able to seamlessly discover other remote services/resources present on nearby mobile devices and to carry out transactions with other services. Obviously standard ad hoc routing protocols are unsuitable for service discovery since in the later case, the destination address of the service is still unknown.

There has been considerable research and industry effort in service discovery in the context of wired as well as partly wired/ wireless networked services. Two important aspects of service discovery are the *discovery architecture* and the *service matching mechanism*. Protocols like Jini [2], Salutation and Salutation-lite [1], UPnP [13], UDDI [5], Service Location Protocol [6] have been developed to facilitate applications to discover remote services residing on stable networked machines in the wired network. Some of these protocols can also be used by mobile devices to discover networked services using wireless networking technologies like 802.11 (a or b). The general architecture followed by all these protocols is like this: A service advertises and registers itself to a service register that keeps track of the networked services. Services can de-register at any point of time. Most of the communication happens over IP-type networks. Thus in summary, these architectures are primarily centralized/semi-centralized, registration-oriented and have an implicit assumption that the underlying network is stable and is capable of providing reliable communication. Clearly, service discovery in MANETS require a decentralized design approach where a node should not be depending on some other node to advertise/register services. Each service should be autonomous and at the same time other applications should be able to discover them.

Currently existing service matching techniques use simple interface-based or attribute-based or unique-identifier based matching. Service discovery is effectively done at a syntactic level. However, syntactic level matching and discovery is inefficient in MANET-type environments. This is due to the heterogeneity of services and their interfaces in such a domain. For example, we can have the same service implement different interfaces, which could result in the failure of a syntactic match if the service query does not match with any interface. We need more expressiveness to describe services/resources and there is a need to discover services in a semantic manner.

A standard solution to resource discovery in MANETS is to broadcast a service discovery request through out the network. If a node contains the service, it responds with a service reply. The protocol is guaranteed to discover a service if it is present in the MANET. Apart from the fact that it is inefficient in terms of bandwidth and resource usage (since the request has to be processed by all the nodes which have limited processing capability and battery power), it gives rise to the broadcast storm problem [14]. The other solution is for the services to advertise themselves to all the nodes in the MANET. Each node interested in discovering services caches the advertisements. The advertisements are matched with service requests and a result is returned. In this solution, the cache size increases with the number of services. Many of the nodes have limited memory and are unable to store all the advertisements and soon the cache gets filled up. This is also inefficient in terms of bandwidth usage, since the whole network has to be flooded with advertisements every subsequent time interval.

We introduce a novel group-based distributed service discovery protocol (GSD) for MANETS, which does not have broadcast storm problem and efficiently uses network bandwidth and is guaranteed to discover a service (if present) in the MANET. Our solution exploits the semantic capabilities offered by the DARPA Agent Markup Language (DAML) to effectively de-

scribe services/resources present on nodes in the MANET. Service requests are expressed in DAML and are matched with service descriptions using a service-matching module. This increases the flexibility in discovering services and is ideally suited to address the heterogeneity of services in a MANET. The services present on the nodes are classified into several groups based on the class-subclass hierarchy present in DAML. A service thus belongs to a hierarchy of groups starting from the parent group called "Service". Each node advertises its services to its neighbors within n hops (n ranges from 1 to MAX_ADVERTISEMENT_DIAMETER). Thus each node does not have to maintain a cache of all services in the MANET. An advertisement also includes a list of the several groups that the sender node has seen in its vicinity. This group information is used to intelligently selectively forward a service request to other nodes in the MANET where there are chances of service availability. Thus, we use the semantic features present in DAML to reduce network flooding and at the same time guarantee that the request is forwarded towards the node(s) that has seen or has the requested service.

We introduce a new metric called *Service Discoverability* that quantitively expresses the chances of a protocol discovering a service in the MANET. We show that Service Discoverability remains the same (and sometimes increases) in GSD when compared to a simple advertisement-based or broadcast-based protocol. Furthermore, GSD achieves the same efficiency in discovering services with lesser number of messages and thus achieves efficient bandwidth utilization.

II. Group-based Semantic Service Description

We have chosen DAML+OIL [8] to define an ontology to describe services/resources in a MANET. There are couple of reasons for choosing DAML+OIL. Firstly, we use the semantics of DAML+OIL to describe services in different nodes and also to enable semantic matching support with those service descriptions. Secondly, DAML+OIL which is based on eXtensible Markup Language (XML) [3] and Resource Description Framework [9] is also being used to describe any information/service on the wired infrastructure and the Web. This would make our description interoperable with the current standard used to describe resources in the Web.

We have extended the DReggie Ontology [4] that contains a comprehensive ontology to describe services in terms of its capabilities, inputs, outputs, platform constraints, device capabilities of the device on which it is residing etc. Using the class/subClassOf axiom of DAML, we have incorporated a preliminary grouping of different possible services in a MANET primarily based on service functionality. One great advantage of our discovery architecture is that this ontology is extensible and one can add more groups to it without altering the process in which the discovery mechanism works. The discovery mechanism would take into account the incorporation of the additional groups. Due to space restrictions, we are unable to provide the ontology. However, it can be downloaded from http://daml.umbc.edu/ontologies/dreggie-ont.daml. The generic class *Service* is functionally classified into two main sub-groups: Hardware and Software Service. Each sub-group is further classified in this manner till we rich a very specific service. For example, a *color printer* service may be classified under *Service/ Hardware/ Input-output-type-Service/ Printer-Service*.

In section III, we describe how the information about groups is disseminated to peer nodes and how it helps in selectively forwarding a service request to the node that might contain the service requested. Service descriptions developed using this ontology are also used to carry out semantic service matching when a service request is matched with a service description. This eliminates strict syntactic matching based on unique identifiers or interfaces and provides us with the flexibility to do loosely coupled matching amongst heterogeneous services.

III. GSD Discovery Protocol

Our protocol is based on the concepts of peer-to-peer caching of advertisements of services and group-based selective forwarding of requests based on the DAML-based semantic information present in the service requests and service descriptions. This helps us achieve efficient network bandwidth usage as well as gives us increased flexibility in the service matching process. Peer-to-peer caching of services gives our protocol the distributed approach that makes it different from other registry-based centralized/semi-centralized service discovery protocols.

A. Service Advertisements

Service Advertisements are initiated by each individual node that is hosting one or more services. Each service has a corresponding service description using or DAML-based ontology. Every node after every ADV_TIME_INTERVAL sends a list of the services it has to all the nodes in its radio range. All the peer nodes that are in its radio range are considered to be within 1 hop of the sending node. The source can send an advertisement across multiple hops too. In that case, the field ADV_DIAMETER in the advertisement packet is set to the number of hops the advertisement is expected to travel. The greater the number of hops, the more number of nodes in the vicinity receives the advertisement. However, this comes with a cost of increased number of messages. The incorporation of the ADV_DIAMETER helps us in regulating the extent to which an advertisement can reach and this can be used intelligently in different mobility situations.

The ADV_TIME_INTERVAL is a parameter that depends on the mobility of the node and the general mobility of the MANET. In highly dynamic situations, it is desirable to have a small ADV_TIME_INTERVAL since the vicinity changes rapidly and vice versa. An advertisement packet consists of the following fields:

<Packet-type, Source-Address, Service-Description, Service-Groups, Other-Groups, Lifetime, ADV_DIAMETER>

Each node contains a *broadcast-id*. The *broadcast-id* increases monotonically with each broadcast from a node. A *source-address* and a *broadcast-id* is used to disambiguate duplicate advertisements. The *Service-description* and *Service-groups* contain information about the service(s) and their groups present *locally* in the sender node. Each advertisement packet also contains a field called *Other-groups*. This field contains an enumerated list of the groups of all the *non-local* services

that the sender node has seen in its vicinity. This information is obtained from the service cache (to be explained in section III-B). The *lifetime* field determines the length of time the advertisement would be cached in the receiver node. The entry is expunged from the node after the advertisement has expired.

B. Peer-to-peer Caching

Each node in the MANET maintains a finite cache to store service descriptions of remote services. Whenever a node receives a service advertisement, it performs the following steps:

```
if (Duplicate(Advertisement))
  then discard packet;
else {
  Extract Service Description;
  Extract group Information;
  Store Entry in Service Cache;
  Set the lifetime on the Entry;
}
```

Each entry in the Service Cache contains the following fields:

<Source-Address, local, Service-Description, Service-Groups, Other-Groups, Lifetime>

The Service Cache in each node contains a list of the local as well as remote services that this node has seen through advertisements. The field *Other-Groups* contain a list of the groups that the corresponding *Source-Address* has seen in its vicinity. In case of a local service, this field is empty. The *lifetime* of a cache entry is determined by the corresponding advertisement packet. Fast moving nodes in the MANET have a small lifetime for its advertisements. If a Service Cache is full, then we use a lowest-lifetime policy to remove an entry from the Cache. Thus the entry whose lifetime is the least is expunged from the Cache. A Cache is also expunged of the entry once its lifetime expires.

C. Request Routing

A Service Request originates from a source whose application layer requests a certain service. A Service Request is described using our DAML-based ontology. Firstly, the request description is matched with the services present in the local cache of the source node. By virtue of peer-to-peer caching, the local cache (in an ideal scenario) contains the descriptions of services present in the nodes within ADV_DIAMETER hops. Thus, by comparing the request with the local cache, the protocol automatically is able to check for the availability of the service within ADV_DIAMETER hops. A simple broadcast based protocol (without any caching) would broadcast the request to all the nodes in the vicinity. Thus our protocol achieves increase in efficiency in discovering a service (if available) in the vicinity. When a match is not found in the local cache, a service request is constructed. The service request contains the following fields:

< Packet-type, broadcastId, Service-Description, Request-Groups, Source-Address, Last-Address, Hop-Count >

The field *Request-Groups* contain the Service Group(s) to which this request belongs. Hop-Count specifies the number of hops that this request can travel. This is specified by the source of the request. This determines the search radius for discovering a service in the vicinity. Standard broadcast-based solutions calls for broadcasting the request to other peer nodes in the vicinity. This is not efficient as far as bandwidth usage is concerned. We use the Other-Groups information present in each entry of the Service Cache to determine a set of neighboring nodes to whom to selectively forward this service request. The Other-Groups field in each Service Cache Entry specifies the groups of services that the sender of that advertisement has seen in its vicinity. Thus, if the request belongs to one of those groups, then there is a chance that the requested service might be available near the node sending that advertisement. Thus, instead of broadcasting the request, the protocol selectively forwards the request to those nodes that have seen the same group of services in its vicinity.

The pseudo code of the algorithm is as follows:

```
if (Hop-Count of Request Packet >0) then {
   for (each entry in Service Cache} do {
     If any one of the request groups G1..GN
     belongs to Other-Groups then {
     Decrease the Hop-Count of the packet by 1.
     Forward the service request to the node;
     }
   }
if (the request was never forwarded) then {
  Decrease the Hop-Count of the packet by 1.
  Broadcast the request to the neighboring nodes.
  }
}
```

When a Service Advertisement is sent from a node to the nodes in its vicinity, the field *Other-Groups* is constructed in the following manner:

```
For each Entry in the Service Cache do {
  If (not local){
   for
   (each group G1..GM belonging
          to Service-Groups) do {
     if (Gi is not in Other-Groups) then
Add Gi to Other-Groups
   }
 }
}
```

Thus a list of the groups of services seen by the sender of the advertisement is also sent with the advertisement. This way, by sending some limited group information about services along with the advertisements, we can intelligently regulate the direction in which a service request travels. This method of sending only group-related information does not impose memory problems since we are not sending the whole service descriptions of the all the services seen by a node (in which case it would have lead to cache overloading).

1) Reverse Path Setup: Each Request Packet contains a *Last-Address* field which contains the address of the node from which a request is coming. Each node in addition to maintaining the Service Cache also maintains a Reverse-Route Table. The Reverse-Route Table contains the following entries:

<Source-Address, BroadcastId, Previous-Address>

The Reverse-Route table is updated whenever a request is forwarded to other nodes in the vicinity. The entry in the table is kept for REV_ROUTE_TIMEOUT time units. The value of the REV_ROUTE_TIMEOUT depends on the Hop-Count of the Request Packet. If the Hop-Count of the Request Packet is more, then the value of the REV_ROUTE_TIMEOUT increases accordingly. This entry is used to send a Service Reply back to the source of the request. The *Source-Address* and *BroadcastId* uniquely identifies a reply packet that corresponds to a request packet.

2) Service Reply Generation: When a Request matches a service description in a node, irrespective of whether the service is local or not, the node generates a service reply. The node first updates its Reverse-Route table entry. A service reply contains the following fields:
< packet-type, Service-Description, Source-Address, Destination-Address, Last-Address, Source-broadcastId, Hop-Count >
The *< Destination-Address, Source-broadcastId >* pair is used to look up the Reverse-Route table to find the previous address to send the reply to. The Reply thus follows the path through which the request had traveled. Since, there can be multiple replies generated, each reply may potentially follow a separate path to the source of the request. The Hop-Count is incremented in every intermediate node and the Last-Address is changed in every intermediate node to reflect the actual previous node address from where the reply is coming. This field has been kept for future work purposes.

3) Service Matching Module: Each node contains a lightweight Service Matching module that performs several activities. It matches a service request to a service description present in the service cache. Services are matched based on Service groups, input/outputs, functional similarity, service capabilities etc. This matching module may potentially use the semantic features (class/ subClass, unionOf etc) of DAML to increase the richness of service matching. This module also provides functionalities to extract Group related information from the service description using our DAML-based ontology. This is used to fill up the respective entries in the service advertisements and service requests.

IV. Simulation Results

We simulated the GSD protocol on the well-known simulator Glomosim [15] under different mobility scenarios. Several interesting experiments were carried out to test the scalability and the efficiency of GSD in discovering services in a MANET. We carried out simulations ranging from a topology consisting of 25 nodes to topologies consisting of 100 nodes. In this section, we report a few of the experiments carried out. Due to space limitations, we are unable to provide all the results. We used a random-waypoint [7] movement pattern of the nodes.

We used an application layer packet generation function to generate service requests at regular time intervals.

For the purposes of the simulation, we used representative services S0 to S24 to represent actual services and groups G1 to G10 to represent service groups with G10 being equivalent to the actual service group called “Service” in our DAML-based ontology. This helped us in carrying out various types of simulation by varying the *Request Groups* and *Service Groups*. We used a Acknowledge-noise model [15] to simulate the radio model. Radio range of each node was assumed to be 31 meters.

We carried out experiments to determine the effect on the number of hops a request has to travel (referred as Request Diameter) with increasing ADV_DIAMETER. Intuitively, with increasing ADV_DIAMETER, the same service would be cached in more nodes and hence the Request Diameter should decrease. We carried out this experiment with a relatively stable mobile environment with node movement (under random-waypoint pattern) being defined by P(Sm,SMax) where :
P= Pause (in seconds) after the node has moved to a new position
Sm= Minimum speed (meters/sec) of movement of the node
SMax= Maximum speed (meters/sec) of movement of the node
The node moves with a speed within the range (Sm, SMax). The actual values of P, Sm and SMax were 30, 1 and 4 respectively (in Figure 1). Figure 1 contains the results. The results validate our intuitive claim.

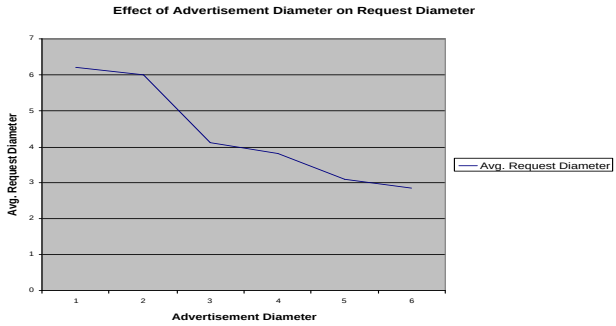

Fig. 1. Effect on Request Diameter by Advertisement Diameter

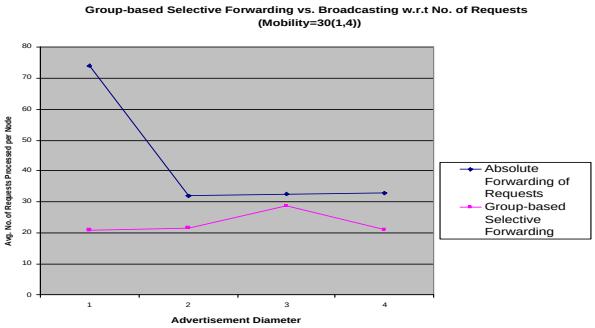

Fig. 2. Effect on the average number of requests in GSD when compared to broadcast

A. Service Discoverability

We define a new metric, *Service Discoverability* that quantitively expresses the chances of a protocol discovering and using a service in a MANET (provided it is available).

Service Discoverability = Expectation of a service being discovered and used given that the service is there somewhere on the network.
Thus the value of Service Discoverability should predict the capacity of a protocol to discover a service in the MANET. In MANETS, assuming that the request is for a single service, we can predict the factors that affect the value of Service Discoverability. We make the following observations:

- If ADV_DIAMETER is increased, the same service is cached in more nodes, thus increasing the chances of a request discovering the service. Hence,*Service Discoverability is directly proportional to ADV_DIAMETER*.
- The further the requested service is away from the source, the lesser is the chance of the service being available or discoverable. Due to mobility, the service has a higher chance of moving out or the route breaking between the two nodes for the service reply to propagate back. Hence,*Service Discoverability is inversely proportional to the distance between the source of the request and the requested service(Hop Count of the Request)*.

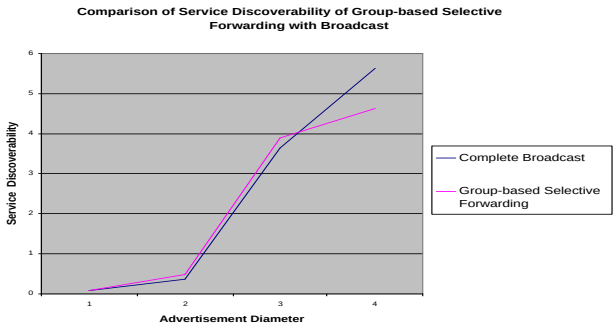

Fig. 3. Effect on the metric Service Discoverability of GSD as compared to Broadcast

- Response Time (difference of time when a request is issued and the service reply is received)also affects Service Discoverability in the same way as Request Diameter (Hop-Count of Request). This is because, the more the Response Time, the lesser is the chance that the service is nearby. Hence:
  Service Discoverability is inversely proportional to Response Time.

Taking these factors into consideration, we can define *Service Discoverability* as:

$$ServiceDiscoverability = \frac{ADV\_DIAMETER}{\prod(ResponseTime, RequestHops)}$$

If n = Total number of Service Responses obtained in the course of a simulation, we define *Average Service Discoverability* as:
Average Service Discoverability =

$$\frac{\sum_{i=1}^{n} \frac{ADV\_DIAMETER}{\prod(ResponseTime_i, RequestHops_i)}}{n}$$

where:
ResponseTime$_i$= Response Time for ith response.
RequestHops$_i$= Hop-count of Request for ith response.

We carried out experiments to test GSD with respect to standard broadcast based protocol to study the effect on service discoverability and number of messages. Our protocol improves significantly in terms of reducing the number of messages to discover the same service under identical mobility scenarios. The service discoverability remains same (and in some cases improves) in GSD. Service Discoverability may increase due to the increased number of responses in GSD. This is because, many requests are dropped due to broadcast collisions in a simple broadcast-based solution. Figure 2 shows the decrease in number of messages in GSD. We observe that the average number of requests processed per node decreases almost 50% when ADV_DIAMETER=1. For higher values of ADV_DIAMETER, the decrease is not that much, but still its approximately 30 to 40%. Figure 3 shows the effect on service discoverability.

In conclusions, we have introduced a novel service discovery protocol for MANETs which uses peer-to-peer caching and DAML+OIL based service description and groups to selectively forward requests, thus achieving the same efficiency as other broadcast-based protocols with a great improvement in the network load.

REFERENCES

[1] The Salutation Consortium Inc 1999. Salutation architecture specification (part 1), version 2.1 edition. World Wide Web, `http://www.salutation.org`.

[2] Ken Arnold, Ann Wollrath, Bryan O'Sullivan, Robert Scheifler, and Jim Waldo. *The Jini specification*. Addison-Wesley, Reading, MA, USA, 1999.

[3] T. Bray, J. Paoli, and C. Sperberg-MacQueen. Extensible Markup Language. `http://www.w3.org/TR/1998/REC-xml19980210`, 1998.

[4] Dipanjan Chakraborty, Filip Perich, Sasikanth Avancha, and Anupam Joshi. Dreggie: A smart service discovery technique for e-commerce applications. In *20th Symposium on Reliable Distributed Systems*, october 2001.

[5] Universal Description Discovery and Integration platform. World Wide Web, `http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.PDF`.

[6] E. Guttman, C. Perkins, and J. Veizades. RFC 2165: Service location protocol, 1997.

[7] D.B. Johnson and D.A Maltz. The dynamic source routing protocol for mobile ad-hoc networks. *Mobile Computing. Imielinksi and H. Korth. Eds. Kluwer.*, pages 153–181, 1996.

[8] DARPA Agent Markup Language and Ontology Inference Layer. `http://www.daml.org/2001/03/daml+oil.daml`.

[9] O. Lassila and R. Swick. Resource Description Framework. `http://www.w3.org/TR/1999/REC/rdf-syntax-19990222`, 1999.

[10] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. IEEE INFOCOM.*, april 1997.

[11] C.E Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing(dsdv) for monbile computers. *Comp. Commun. Rev.*, pages 234–44, October 1994.

[12] C.E. Perkins and E.M Royer. Ad-hoc on-demand distance vector routing. In *Proc. 2nd IEEE Workshop. Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[13] Rekesh John. UPnP, Jini and Salutaion - A look at some popular coordination framework for future network devices. Technical report, California Software Labs, 1999. Available online from.

[14] Y. Chen S. Ni, Y. Tseng and J. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proc. MOBICOM. Seattle. Washington*, 1999.

[15] Mario Gerla Xiang Zeng, Rajive Bagrodia. Glomosim: A library for parallel simulation of large-scale wireless networks. *Proc. 12th Workshop on Parallel and Distributed Simulations*, 1998.