DOI: 10.21917/ijct.2012.0082

SERVICE PROVISIONING IN MANETS USING SERVICE PROVIDER'S METRICS

K. Ponmozhi[1] and R.S. Rajesh[2]
[1]*Department of Information Technology, Hajee Karutha Rowther Howdia College, India*
E-mail: arulchezhiyan96@gmail.com
[2]*Department of Computer Science and Engineering, Manonmanium Sundarnar University, India*
E-mail: rs_rajesh1@yahoo.co.in

Abstract
Service discovery technologies are exploited to enable services to advertise their existence in a dynamic way and can be discovered, configured and used by other devices with minimum of manual efforts. Automatic service discovery plays an important role in future network scenarios. Service discovery in distributed environment is difficult that too if the availability information of the services cannot be in a centralized node. The complexity is increased even further in the case of MANETs in which there will not be central intelligence also, the nodes involved may be on the move. The mobility issue leads to the situation of uncertainty about the service availability of the service provider. In this paper we propose a decentralized discovery mechanism. The basic idea is, distributing service information along with the availability metrics to the nodes. The metrics will give us the information to evaluate the goodness of the service provider. Every node will form multi-layered overlays of service providers sorted based on the metrics. When we send a query, each node will identify the service provider from the overlay with the good metric among the available providers (i.e.) the one in the first position in the overlay. We define the message structures and methods needed for this proposal. The simulation result shows that in the high mobile environment too we could have a better convergence. We believe that the architecture presented here is a necessary component of any service provision framework.

Keywords:
Mobile Ad Hoc Networks, Service Discovery

1. INTRODUCTION

Mobile Ad hoc Networks (MANETs) are networks comprised of mobile nodes equipped with wireless interfaces and communicating with each other without relying on any infrastructure. In these networks each mobile node may act as a client, a server and a router which catches the services provided in the vicinity. Often mobile nodes inside of MANET need to utilize resources or services that are present on other mobile nodes in their neighborhood.

We refer service as work or resource contributed by one or more entities that can help accomplishing the task of other entities. To make greater utilization of resources in vicinity, it is important for nodes in MANET to be able to discover remote services seamlessly and carry out transactions with service providers. However all these processes are complicated by the fact that there is no fixed infrastructure and established administration. Therefore a decentralized approach is required for maintaining service and information about the service object.

Service oriented applications involve distributed components that can either play the role of service providers or service clients. Service provision is generally performed in two main steps: *service discovery*, during which services are advertised by providers, which can be discovered by potential clients, and *service invocation*, during which a given client actually interacts with the provider of a previously discovered service. If there is more than one provider in the network for the same service the clients can be given a chance to select among the several providers.

Though the service oriented approach is relevant for MANETs, the changes of the unpredictability of the providers' availability and communication delay makes it difficult to implement as that of the wired network.

Thus, Service discovery, which allows devices to advertise their own services to the rest of the network and to automatically locate network services with requested attributes, is a major component of MANETs. In the context of service discovery, service is any hardware or software feature that can be utilized or benefited by any node; service description is the information that describes a service's characteristics such as its types and attributes, access method etc. In this paper we focus on service discovery, which is the major issue of service provisioning. We concentrate on the structure of the messages related to this issue. The remaining paper is organized as follows: In section 2 we discuss the issues to be solved for service discovery, followed by existing architectures. In section 3 we discuss our frame work and section 4 gives the simulation results, finally we conclude our work followed by future work direction.

2. RELATED WORK

A number of research work concentrated on service discovery available in the literature. Traditional service discovery protocols like UDDI [1] and Service Location Protocol rely on centralized nodes as look up servers. In this architecture the service providers register their services to service directories and service requestors are informed about the available services in the network only through these directory nodes. In the case of MANETs we cannot be sure of any node to be always available and reachable by all the nodes in the network. One of the alternatives to this approach is *distributed directory architecture*, in which virtual back bone of hierarchical directory nodes are formed [16]. The service providers have to advertise their service to one of these directories. The clients can access to the services by sending queries to one of these directories. In Jini [2], where a few nodes, named Lookup servers act as directories. In [16] the author uses adaptive mechanism to select directory nodes. The other alternative approach is *Directory-less* architecture. In this type of architecture [12] there are no service directories to mediate communication between the service providers and service requestors. Service providers broadcast service advertisements and service requestors broadcast service requests. Some alternatives are: In [3] service providers periodically advertise their services along with service that they became aware of. In [12], the author decides to use multicast the advertisements, Instead of broadcasting. But selection based on the providers' goodness is not considered in these works.

All these service discovery architecture should have some mechanisms to describe the services and the matching. The descriptions formed by the servers to give the information related to the services, the location, the accessing methods etc. A simple but powerful service description facility and matching mechanism can help service discovery protocol achieve high efficiency. In many of the well known protocols they use simple matching schemas such as interface descriptions [2] or attributes [7][22] or even unique-identifiers [8]. Service matching is done at a syntactic level. However syntactic level matching and discovery is inefficient in MANET environments due to the autonomy of service providers and the resulting heterogeneity of their implementation and interfaces. To alleviate this problem, there has been work to develop languages [9] to express service requirements and facilitate flexible semantic-level service discovery. The most widely used description format is attribute-value structure. Some other users XML based descriptions which enables a better classification and make use of schema for attribute definition. GolServ[17] uses RDF for service description, where as GSD [18] uses DAMIL+OIL.

Independent of the chosen service discovery architecture, service information can be gathered either of these Discovery modes: reactive, proactive and hybrid way.

In Reactive mode [12] no node sends any advertisement to announce their service information. Therefore, the nodes in the network do not know where services are available. A service requestor node creates a query on-demand whenever a certain service is desired. The query is then sent to the network either unicast, broadcast or multicast depending on the service discovery architecture.

In Proactive mode, service providers proactively distribute their available services [16]. The distribution is performed either directly to potential service clients or to service directories depending upon the architecture used. Though this method causes more traffic at the initial stage, the service discovery delay will be reduced.

Hybrid discovery mode [18],[22] supports both reactive and proactive service advertisements. Generally advertisements are sent to a subset of nodes. This approach supports the service information may be distributed in several ways depending upon the topology. Some nodes may know all service information while some nodes have no information at all and must rely on flooding service request.

Searching request for services described in user requests or queries. The queries comprised of the attribute value pairs. If one attribute is not specified it is generally considered it can take any value. Wild-card matching is supported in INS [19]. Some protocols support filtered query flooding to enable multi-criteria selection as in SSDS [20]. Issues on the service description and matching are the study reserved for our future work.

3. OVERVIEW OF OUR SCHEME

Services that are available in the network can be accessed by others in the same network. In order to enable this kind of service reuse and access by other nodes, the nodes which are ready to share their service should make available the descriptions of the service to others in the network. We call those nodes which can provide service as provider and those which use them as clients. The set tasks to make service availability and enabling the service access by other nodes are called service provisioning. The basic operations to be performed are service advertisement, service request matching or forwarding, service reply routing, service invocation. We used pee-to-peer caching of service advertisements. When a provider decides to share services it should form a service advertisement with required fields. The advertisements will be broadcasted in the network with a hoping limit specified. We use multi-layered overlay, where each layer represents the layer for a service provided in the network. We assume that the client and the provider use the same rule to form the service type. We assume that the client and providers use the same rule to form the service type. Each node in the network when receives an advertisement will store the advertisement in its respective overlay layer in the sorted order of the providers' metric. In the advertisement the provider specifies the time of expiration. When the stored advertisements are timed out they will be marked to be removed from the overlay meaning that that service is not provided by the provider now. When we receive a service update message or hello packet for that service again the marking will be removed. We limit the number of entries so in an overlay so whenever it overflows the marked items to be removed will be replaced. When the client needs a service it will form a service request and sent that to the SDP in its local machine. The SDP will check for the availability of provider(s) from its local cache of the corresponding overlay of providers of same services to which the request is received. If the service provider is available it will form a separate request and forwarded to that provider in a unicast manner instead of broadcasting. Thus instead of broadcasting service request to all the neighbors, our scheme forwards the request towards those node who are having the possibility to provide the service. The provider will send response sending the descriptor of the service. If the service is not available in the local cache it will flood the query into the network with the TTL value specified. The node which receives the query will check into its local cache, and if it finds one in its cache it will send that information to the requester. If the node itself is a provider then it will send the address of the provider along with the descriptor. The discovery process ends with the reception of the reply from an intermediate node or from the provider itself. Once the response is received the client will formulate SOAP request for the service invocation based on the service descriptor received.

3.1 SERVICE DESCRIPTION

The provider creates and sends its service descriptor when it receives a request from the client. The descriptor is an XML document it has the following parts,

Table.1. Service descriptor

Service-type	Functional properties	Non-functional properties	Context properties

A service descriptor is a document that starts with a header built from a number of freely chosen attribute/value that describes the non-functional characteristics of the service (including QoS or Semantic information such as a category or a required security level etc.) The functional interface of the service is given in XML which is similar to the parts of WSDL [10]; the client can formulate its invocation requests based on this information. Service descriptions concern not only the functional characteristics of a service. They can also be used to provide context awareness, scope awareness along with QoS awareness regarding a service.

3.2 SERVICE ADVERTISEMENT AND OVERLAY FORMATION

Each server will generate service advertisement packets periodically and broadcast. We use the hoping limit to 3. When a node shares a service, the node stores its service in its Sharable Service Table, and then advertises the service. The service advertisement is in the form of XML file. Table.1 defines an advertisement. Each intermediate node has two tables one to store its own sharable services, the other one is to cache the service advertisements sent by other service providers in the network.

```
<?xml ?>
<message>
<message_type> Advertisement</message_type>
<sender_id> forwading peer_id </sender_id>
<provider_id> service providers id </provider_id>
<hopcount> 0</hopcount>
<Adv_num>number which will be incremented each time it
readvertise</Adv_num>
<TTL> advertisement scope value</TTL>
<Adv_life> value in terms of seconds after which the
advertisement is invalid</Adv_life>
<providers_metric>
<attribute>
<parameter>routing metric </paramenter>
<value> …</value></attribute>
<attribute> … </attribute>
</providers_metric>
<service_type> printer </service_type>
<attribute>
<parameter>… </paramenter>
<value> …</value></attribute>
…
</service_type> </message>
```

Fig.1. Advertisement message

Message type - is used to differentiate different types of messages used in the protocol.

Service type - specifies the type of service provided by this provider. Since a provider may provide more than one service if it wants it can create a message to advertise all its services.

Hopcount – Initially it is set to 0. When it reaches the next hop it will be incremented. This is used to know the distance between the provider and the receiver node.

Adv_life – specifies the time when the advertisement expires.

Adv_num – This is the number used to identify whether the advertisement is already received. If the new advertisement number is less or equal to the advertisement already stored then this advertisement packet will be discarded.

TTL – This is the value set by the provider to limit the advertisement scope. This value will be reduced by 1 at each hop. When this value is 0, the packet will not be forwarded further.

Parameters of providers metric are the battery power at the time of sending (BatP), number of services provided (NS) to other nodes. Moving speed (Speed). Based on this we can calculate the provider metric as follows: we give less weightage for fast moving providers,

$$MPi = W1 * \text{Speed} + W2 * \text{BatP} + W3 * \text{NS}$$

where, Wi are the weightage assigned and this can be changed by the designed if needed.

Other Service attributes specified are divided as mandatory attributes and secondary attributes. These attributes of the service has to be matched against the service request parameters. These may be defined by the application designer during development. Application designer can decide the weight age for the parameters.

Whenever a node receives an advertisement,

1. Access the service table with service_ type
2. Calculate the metrics and add the provider in the correct location in the list. (see [21])
3. Add the description to the overlay for this service, along with the context parameters and with provider_id

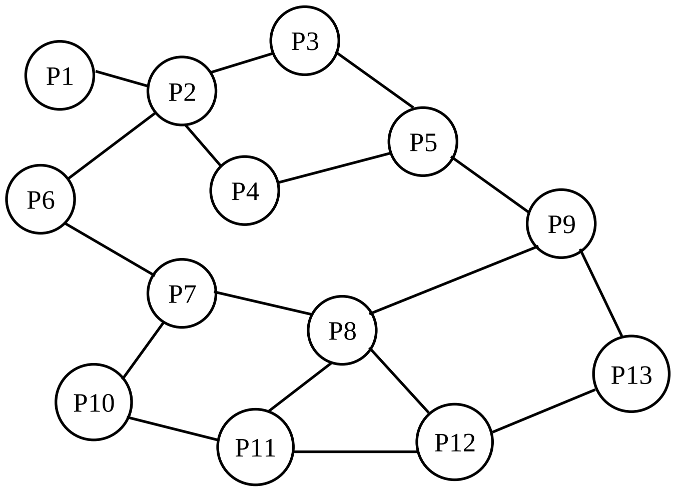

Fig.2. Physical Network

Let Fig.2 specifies the physical network. Let P1 to P13 be the peers in the network. Assume that Peers P1, P9, P4 and P11 are providing service_i ad P1, P5, P4 and P12 are providing service_J. Fig.3 specifies the overlay two services. P9 – Peer with the service of that service (printer) with highest rank. P11 – is the peer with next highest metric value and so on. Whenever a new provider with printer is advertising we will calculate it's metric and put it in the correct order.

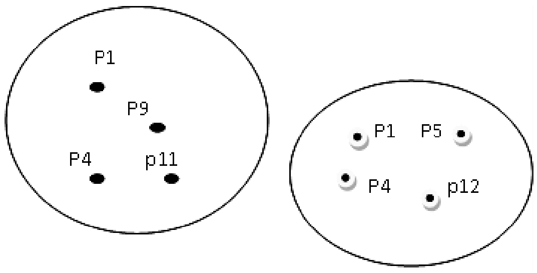

Fig.3. Overlay for a service 2 services

Service-i	→	P11	P4	P1	P9
: :					
Service-j	→	P1	P5	P4	P12

A provider may belong to more than one overlay if it provides more than one services. In our example provider 4 and 1provides both service_i and service n, they will be two overlays. Every node maintains a table to store all the advertisements received, its local services, services provided to other nodes, overlays. The above diagram shows the local table with two services stored in the sorted order of their metrics.

3.3 SERVICE AVAILABILITY MAINTENANCE

3.3.1 Sending Hello Messages:

It is challenging to maintain accurate and valid service information and service state especially in MANETs where the inherent dynamism leads to frequent changes in service availability. One approach is to maintain a hard state of services where a provider must de-register its services before leaving the ad hoc network. However, in MANETs where unpredictable disconnections occur assuming that a provider will be able to de-register its service before disconnecting is not realistic. The opposite approach is to maintain a soft state of services. In this case each service entry is associated with a time to live counter; upon expiry the service entry is automatically deleted. It is the job of the service provider to periodically refresh that counter by re advertising the services in the form of hello packet, so that the nodes will have up to date knowledge of the service availability. We use both soft state and hard state to maintain consistency in the SST. Fig.4 defines the structure of "hello" message.

```
<message>
<message_type> Hello</message_type>
<senderid> forwading peer_id </sender_id>
<provider_id> service providers id </provider_id>
<TTL>  value </TTL>
<service_types>
<service_type> printer/service_type>
</service_type>
<service_type>
…
</service_type>
<piggyback> <service_type> … </service_type>
</service_types> </message>
```

Fig.4. Structure of "hello" message

In "hello" messages the service providers send the list of available services for sharing, TTL value which is used to limit the scope. Since the parameters are sent already in the advertisement itself they need not be sent again. We have provided a facility to send new service advertisement if it is not sent already as a piggyback type service description. This new service-type will be added to the list already available, the processing is similar to the processing of the services in the advertisement. This will make the local directory up to date, also the nodes may know whether the providers are available or not. Those providers' entry, for which the hello packet has not been received, will be removed from the table.

Upon receiving the Hello packet:

```
For (all the entries for this provider_id)
{
   If (Message_type = "hello")
   For (all the services in this packet)
   {
     If (the service_type exists in the table)
        Update the time_of _expiary
     If (piggyback)
     Call the method for advertisement to add these
     services to the table
   }
  For (all the service without matching entry in the
packet)
  {Delete the entry from the table}
}
 For (all the out-of-date entries regardless of the server)
{Delete the entries}
```

3.3.2 Service Updation:

It is a way of maintaining services consistently, if there is need to change any of the parameters of the services already sent that can be done by the use of these packets. Service update packet will be generated by the server when the already published service has been modified, or the service descriptions have been changed.

```
<message_type> Updation</message_type>
<senderid> forwading peer_id </sender_id>
<provider_id>  providers id </provider_id>
<TTL>  value </TTL>
<service_types>
<service_type> printer/service_type>
<parameter> color </parameter>
…
</service_type>
<service_type>
…
</service_type>
</service_types>
```

Fig.5. Structure of Service Update

When the service Update packet is received:

```
For(all the entries for this provider_id)

{   If (Message_type = "updation")

    For (all the services in this packet)
      {
   if (the service_type exists in the table)
       Replace the service descriptions
        Update the time_of _expiary
    Else
       Add this service to the table
        }
  }
 For (all the out-of-date entries regardless of the server)
 {Delete the entries}
```

3.3.3 Service De-registering:

This message is sent by the service providers so that the clients of the services of this server may find new alternative services for their continuous service usage. We send this message only to the clients of this server. As the other nodes will delete the entries of advertisement is the entries are timed out. Whenever the server decides to deregister it generates deregister message and send to the clients of this server.

```
For (all the clients in the Service status table)
Send the deregistering message
```

When the client receives the de-register message for a particular service:

```
Go to the overlay of this service; take point to the entry in the advertisement of this service from the service advertisement table

Delete the entry from the list

Prepare the request to the provider in the overlay list for re-selecting a new service provider for this particular service for the application.

Notify this to the (broker) module which takes care of service rediscovery
```

On the client side, document called service pattern must be created to convey the wishes of a client hoping to discover a suitable service. The service pattern contains components similar to those of the service descriptor, with the possibility to include wildcards and expressions on the attribute value. The context and some non functional properties which are mandatory will also be mentioned in the request.

```
<message_type> request</message_type>
<requester_id> forwading peer_id </requester_id>
<req_id>…</req_id>
<TTL>  value </TTL>
<service_type>
<attribute>
<parameter>…</parameter>
<value>…</value></attribute>
            // list of attributes

 </service_type>
```

Fig.6. Structure of request message

The service request may be received by the node which may be,

- Provider
- Intermediate node with the cached description for this service
- Intermediate node without service information

This service request pattern is matched with the descriptors. If a match is made from the cached descriptor actual service request will be formulated and sent to the service provider in the overlay formed.

Whenever the node receives a request (intermediate nodes):

```
If  service_type esists go to its corresponding overlay
For (all the providers in the list)
       If (the functional properties are matching)
            {
            If (the non-functional properties are matching)
            {
                    Send the provider details to the client
                    exit the loop
            }
            }
If no match create service_request broadcast in the network
```

Whenever the node receives a request (application client nodes):

```
If  service_type esists go to its corresponding overlay
For (all the providers in the list)
       If (the functional properties are matching)
            {
            If (the non-functional properties are matching)
            {
                    Create service_request message and send
                    to the provider,
                    exit the loop
            }
            }
If no match create service_request broadcast in the network.
```

Whenever the node (provider) receives a request:

```
If (the functional properties are matching)
    {
    If (the non-functional properties are matching)
        {
          Create service_resrponse message with
          description send it to the requester as unicast
          communication,
        }
    }
    Reply with neg_response with the req_id
```

If the request is received by the provider, it will send the descriptor of the service which will specify all the details related to invocation of the service, its input and output parameters if any, etc. Once the client has discovered a service, service invocation request has to be sent to the provider. SOAP is a protocol based on HTTP for communication and on XML for communication between services, which allows not only data to be passed from object to object around the network but also full objects, including code. In our approach the invocation request will be sent to the service provider as SOAP message along with header, which is not discussed here.

4. SIMULATION RESULTS

4.1 SIMULATION OVERVIEW

The proposed scheme was implemented using NS2. Initial energy is set to 0.5 Joules, transmission power t x Power as 0.6 watt and r x Power as 0.3 watt. Simulation area is 500 x 900 meters. Nodes can move within the speed range 0 to 20 meters per second. The nodes were randomly distributed using Random Waypoint model. The random waypoint model defines the mobility pattern of nodes by pause time and maximum node speed. Each node began the simulation by remaining stationary for a specified period of pause time. It then selected a random destination and moved to that destination at a speed distributed uniformly between 0 to some maximum node speed. Upon reaching the destination the node paused again for the pause time, selected another destination and move towards that. The simulation time is 1000ms. We use different movement pattern generated for 0, to 1000 in steps of 100. A pause time of 0 means continuous motion and 1000 is no motion since the simulation time is set to 1000. We use 100 wireless nodes forming ad hoc network. Among them 15 play the role of service providers and 35 as clients. The remaining nodes act as intermediate nodes. To start with the providers publish their services in the form of service descriptors and wait for the client's service request. Every client's objective is to invoke a desired service once. We make ten clients to send service request constantly at the rate of 4 packets per second. We set like this to capture the performance of various moments of the moving nodes. The clients for service request make a matching process in its local cache and send as request, if not found, to its overlay nodes. When it receives the reply, it makes the actual invocation to the specified provider using the descriptors.

4.2 EVALUATION

We evaluate two measures the rate of success and the time we need for the response on which the overall service discovery performance is based. Query success rate and Query response time.

The *query success rate* – this is the ratio of queries that are replied by one or more providers over total initiated queries. The result for various pause times is shown in Fig.7. When the maximum node speed is 1 m/sec, the system gave about 21% higher than the system without providers metric. When the maximum node speed was 20 m/s, the proposed system provided 16% better query success ratio the system without providers metric. Such an improvement is possible only because the clients' nodes make connections to the nodes with higher battery power and low mobility.

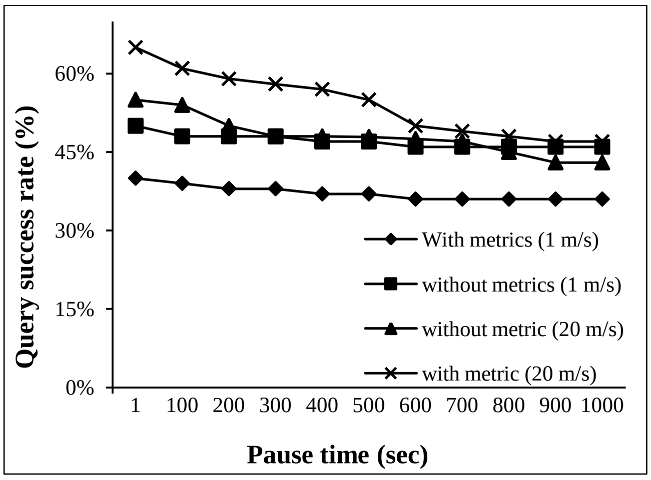

Fig.7. Query success ratio

Next we evaluate the *average response time*. This is the time taken from the time when the query is initiated and the time when the requestor receives the response. Fig.8 shows for both speeds. The system with metrics resulted 26% faster than the one without using metrics at the maximum speed of 0 m/s, and 33% higher than at the maximum speed of 20 m/s. This clearly shows the advantage of using the providers' metrics for selecting the services.

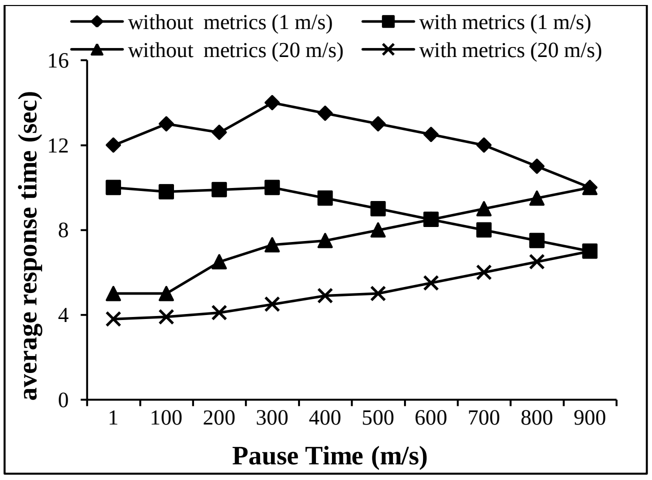

Fig.8. Average response time

5. CONCLUSION AND FUTURE WORK

We presented in this paper the design and the main implementation features of a service platform that specifically targets MANETs. We have chosen fully distributed service architecture which is the best suitable one for the MANETs. The ranking of the providers based on the current execution environment context makes the accessing of the services easier and faster which helps to acquire better Quality of Service. We use separate message formats for service request and reply, the API created by us provides facilitation for service advertisement, discovery messages, and also invocation messages. The service descriptors we used are defined in XML to facilitate interoperability. As the message size has direct influence in the transmission, which has its impact on the power used and bandwidth ultimately to the performance, our future work direction may be on how depth the detailed information has to be advertised.

REFERENCES

[1] Bowman M, Debray S. K and Peterson L L, "Reasoning about naming systems", *ACM Transactions on Programming Languages and Systems*, Vol. 15, No. 5, pp. 795-825, 1993.

[2] Sun Microsystems, Jini Network Technology, <http://www.sun.com/software/jini/>.

[3] M. Nidd, "Service Discovery in DEAPspace", *IEEE Personal Communications*, Vol. 8, No. 4, pp. 39-45, 2001.

[4] Koodli R and Perkins C E, "Service Discovery in on-demand Ad Hoc Networks", IETF Internet Draft, 2002.

[5] Engelstad P, Egeland G and Thanh D V, "Name Resolution in on demand MANETs and over External IP Networks", *IEEE International Conference on Communications*, Vol. 2, pp. 1024-1032, 2003.

[6] Salutation Consortium, "Salutation architecture Specification Version 2.0c – Part 1, The Salutation Consortium, 1999, http://www.salutation.org

[7] Bluetooth SIG, Specification http://bluetooth.com/.

[8] R. Chinnici, J J Moreau, A Ryman and S Weerawarana, "Web Service Description Language (WSDL) Version 2.0 – Part 1: Core Language", 2006, http://www.w3.org/TR/2006/CR-wsdl20-20060327.

[9] M Gudgin, M Hadley, N Mendelsohn, M Jean-Jacques and H Frystyk Nielsen, "SOAP version 1.2 Part 1: Adjuncts", 2003, http://www.w3.org/TR/2003/REC-soap12-part2-20030624/

[10] S Helal, N Desai, V Verma and C Lee, "Konrark – a Service Discovery and Delivery Protocol for Ad-Hoc Networks", *IEEE Conference on Wireless Communications and Networking*, Vol. 3, pp. 2107-2113, 2003.

[11] D Chakraborty, A Joshi and Y Yesha, "Integrating Service Discovery with Routing and Session Management for Ad-Hoc Networks", *Ad Hoc Networks*, Vol. 4, No. 2, pp. 204-224, 2006.

[12] A Varshavsky, B Reid and E De Lara, "A Cross-layer approach to service discovery and selection in MANETs", *IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, 2005.

[13] F Zhu, M Mutka and L Ni, "Service discovery in pervasive computing environments", *IEEE Pervasive Computing*, Vol. 4, No. 4, pp. 81-90, 2005.

[14] Cynthia Jayapal and Sumathi Vembu, "Adaptive service discovery protocol for Mobile ad hoc networks", *European Journal of Scientific Research*, Vol. 49, No. 1, pp. 6-17, 2011.

[15] Knarig Arabshian and Henning Schulzrinne, "Gloserv: Global service discovery architecture", *The first Annual International Conference Mobile and Ubiquitous Systems: Networking and Services*, pp. 319-325, 2004.

[16] Dipanjan Chakraborty, Anupam Joshi, Tim Finin and Yelena Yesha, "GSD: A novel group-based service discovery protocol for MANETs", *Proceedings of 4th IEEE Conference on Mobile and Wireless Communications Networks*, pp. 140-144, 2002.

[17] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley, "The design and implementation of an intentional naming system", *ACM SIGOPS Operating Systems Review*, Vol. 34, No. 2, pp. 186-201, 2000.

[18] Todd D Hodes, Steven E Czerwinski, Ben Y Zhao, Anthony D Joseph, and Randy H Katz, "An architecture for secure wide-area service discovery", *Wireless Networks*, Vol. 8, No. 2/3, pp. 213-230, 2002.

[19] K Ponmozhi, and R S Rajesh, "Applying P2P in MANETs for resource sharing", *Proceedings of IEEE International Conference on Control, Automation, Communication and Energy Conservation*, pp. 1-5, 2009.