A Model For Dynamic Service Discovery In Wireless Adhoc Networks

Şebnem Baydere Mesut Ali Ergin

Yeditepe University
Department of Computer Engineering
Kayısdagi Campus, 81120, Istanbul, Turkey
{sbaydere,ergin}@ics.yeditepe.edu.tr

ABSTRACT

Dynamic service discovery and late binding to the most appropriate network service will be the key characteristics of future adhoc networks. In adhoc networks, the nodes are mobile and the multihop environment does not rely on an existing communication infrastructure. Mobile nodes dynamically establish routes among themselves to form a network "on the fly". "Appropriateness" of a service is a system-specific definition associated with the current state; such as "the nearest accesable" or "the least loaded". These environments require a resource discovery model based on expressive and descriptive service definitions for binding. In this paper, we present a service discovery protocol based on a node mobility and service definition model formally specified in eXtensible Markup Language(XML). The protocol employs a session-based approach to service access whereas the discovery algorithm returns both the service location and the route information for late binding. Initial thoughts on the simulation model that will be used for the validation and performance analysis are also given.

I. INTRODUCTION

The discovery and use of network resources such as printers, web servers, file systems, video cameras, directories and a variety of services are typically client-server applications which establish a connection between two networked nodes. Accesing those services in traditional client-server model requires static binding and as the focus of interest is mostly on internet, manual configuration of DNSs by the network administrators. Future networks will be characterized by the mobility of nodes. Therefore, there will be an inherent dynamism in users and services provided to them. Although the packet routing problem in adhoc networks has been extensively researched [1,2,3,4], few works have been done on resource discovery. Support for describing, locating and gaining access to services for mobile entities is recently emerging [5,6]. Sending a job to the closest printer or retrieving a file from the least loaded replicated server in an infrastructureless network requires a descriptive service definition based on an appropriate node mobility model.

In a wireless network, mobile nodes can move in many different ways. Mobility models are used to analyze new systems or protocols in cellular or adhoc networks. In cellular networks, mobility models not only describe indvidual motion behaviors such as changes in the direction or speed, but also collective motion of all the nodes relative to a geographic area (cells) over time. Adhoc network mobility models either reflect the behavior of individual nodes or a group of mobiles moving towards a target such as fire, flood, earthquake, search rescue or a battlefield. Some group mobility models are proposed in the literature[7,8].

In our target environment, we use random walk mobility [9]. In this model, mobile nodes move randomly with a given speed and direction of motion. The speed and direction of the motion in a new time unit is not related to the previous epoch. This model may generate sudden stopping or sharp returns which may not be suitable for some applications. Some variations have also been used in simulations, such as; random direction but constant speed. Other mobility models used in adhoc networks are discussed in [7,8].

The rest of the paper is organized as follows; Section II discusses the related work and motivation. Section III introduces the mobility model and its formal description in XML. Section IV explains the service discovery protocol in a pseudo code based on predicate calculus and set theory. Section V contains the future work, initial thoughts on the simulation model and some concluding remarks.

II. MOTIVATION AND AIMS

Most research activities concentrate on automatic service discovery and configuration when a mobile internet user visits a foreign infrastructured network. The service discovery protocols; Sun's *Jini*[10], Service Location Protocol(*SLP*)[6], Universal Plug and Play(*UPnP*), Bluetooth Service Discovery Protocol(*SDP*) and *Salutation*[11] provide mechanisms to search for and choose the most appropriate service using an attribute based centralized directory service, complementing DNS. Directory services are maintained by one or more directory agents(DAs) periodically advertising themselves to the network. A mobile device dynamically obtains the directory agent address in the domain and gets the service access information from it. The Berkeley Service Discovery Service (SDS)[5] extends this concept with security and a fixed hierarchical structure for wide-area operations. Some research activities concentrate on naming schemes used to describe services; Intentional Naming Scheme (INS)[12] is a simple language based on attributes and values for its names. It achieves expresiveness in the service definition and defines name resolution process and late binding in the INR(Intentional Name Resolvers Network). All above mentioned protocols are useful to form an application-level overlay network supporting discovery and late binding. They are intended for intra-domain deployment with generally no security consideration. The network and internet layer functions in the underlying infrastructured network are left to the existing protocols; DHCP, Mobile IP.

On the other hand, future adhoc networks, namely; "networks on the fly" may neither rely on an existing infrastructure underneath nor an overlay lookup network. For example; a group of nodes in a battlefield intending to form a temporary network to share some resources. Discovery protocols are needed which operate directly between clients and servers in a dynamic environment where no static configuration at any level exist. In our proposed model, mobility in adhoc mode is supported. Each client node establishes its local directory service dynamically by collecting information from its neighbors. Intermediary centralized lookup services are not needed. The model consists of a set of mobiles acting either as client or server, and a set of valid services on which clients can make requests. Each node has its own view of the multihop network composed of a number of reachable nodes. The service discovery protocol provides network level functions including routing as well as light-weight sessions over the established routes.

III. THE PROPOSED ARCHITECTURE

The proposed model is an enhanced client-server architecture in which clients are supported to adapt themselves to the new network resources they discover "on the fly". Each node instance is identified by a number of characteristics given in the host definition. There is a set of valid services that a client node can ask for binding. The nodes get service information either by catching the periodically advertised services or by generating a service request. Applications establish light-weight sessions with the resources using the address returned to them by their local agent. Binding is done when the service is needed.

The initial route establishment for a new session is a part of the discovery algorithm. The algorithm collects information from the network and the decision is made using the local service information available on the client. In the current model, every node is capable of resolving the descriptive names to network addresses, namely; specific name resolvers or directory services are not needed. The environment can be characterized as follows:

- network connectivity is continuous but varying,
- accessable services vary as the environment changes around the client,
- routes for active sessions may change,
- sessions are not premptive, once a session is established with a resource, service handoff with a closer replica is not supported,
- the underlying discovery architecture is agent-based .

These characteristics are discussed further in the following sections.

1. Definitions with XML

Formal object definition and data representation language for the proposed model is chosen to be eXtensible Markup Language that has been produced as the part of *XML Activity* by *W3C* [13].

XML provides an effective representation of structured data. As a result, the data can be parsed in a more flexible way compared to any other text-only format , while preserving a good level of human readability. Since XML is vendor neutral, parsers are readily available for many different development environments. In addition to that, data models and query languages have been proposed for accessing data efficiently (i.e.*Xquery*) [14,15].

XML Schema language has been selected to define the structure, content and semantics of XML documents. XML Schema has currently been advanced to 'Proposed Recommendation' status by W3C and seems to replace *DTD (Document Type Definition)* that is the original XML specification language. By using XML Schema, productivity in creating *XML* definitions has been increased by far[16]. All XML definitions for the service discovery model make up a particular namespace called *http://www.tnl.yeditepe.edu.tr/2001/03/MANET* (MANET) which is both default and target namespace for all *xsd* (*XML Schema Definition*) documents.,

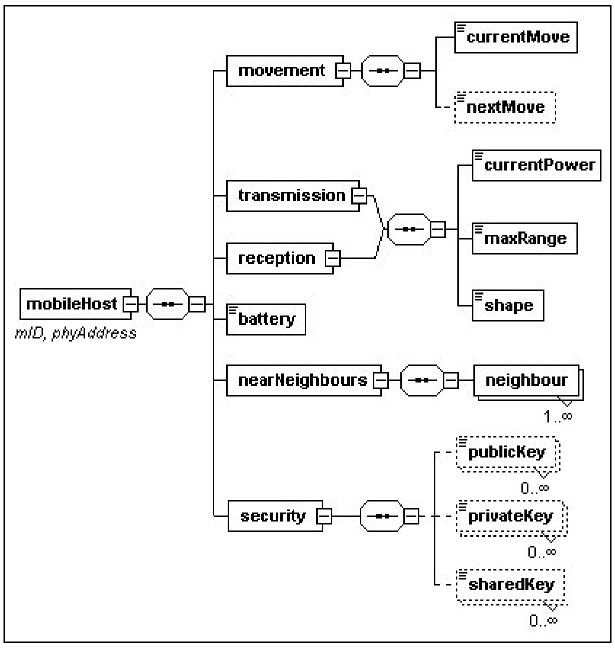

Figure 1. Element hierarchy for a mobile host

Two important XML Schema documents and their conforming instances will be overviewed in the following sections to summarize XML usage in the model.

2. Host Definition Schema

The hosts that are subject to the proposed service discovery protocol conform to this definition schema, namely *HostSchema.* This schema allows a mobile host or fixed host object to be defined with all its characteristic properties.

A well-formed and valid XML instance of HostSchema is the representation of a mobile host with its current state at any time. Therefore a simulation for the mentioned host will give life to this host by keeping its all related elements and attributes up-to-date.

```
<?xml version="1.0" encoding="UTF-8"?>
<mobileHost
    xmlns="http://www.tnl.yeditepe.edu.tr/2001/03/MANET"
    mID="192.168.0.1" phyAddress="A3:55:4F:C3:64:B4"
    lastUpdate="1999-05-31T13:20:00.000-05:00">
  <movement>
        <currentMove x="13" y="44" speedunit="m/s"
                speed="244"> The fourth floor room number 443.
        </currentMove>
        <nextMove speedunit="m/s" speed="200">45</nextMove>
  </movement>
  <transmission>
        <currentPower>200</currentPower>
        <maxRange>500</maxRange>
        <shape>circle</shape>
  </transmission>
  <reception>
        <currentPower>120</currentPower>
        <maxRange>500</maxRange>
        <shape>circle</shape>
  </reception>
  <battery type="NiMh" maxCapacity="450">200</battery>
  <nearNeighbours>
    <neighbour mID="192.168.0.2"
        phyAddress="33:4F:C5:67:75:BD"/>
    <neighbour mID="192.168.0.5"
        phyAddress="42:CC:56:FE:55:A1"/>
  </nearNeighbours>
  <security level="high">
        <sharedKey id="03" algorithmName="DES"
length="64">110 … 00</sharedKey>
  </security>
</mobileHost>
```

Figure 2. A mobile host instance conforming to the *HostSchema.*

```
<?xml version="1.0" encoding="UTF-8"?>
<service name="printer"
    xmlns="http://www.tnl.yeditepe.edu.tr/2001/03/MANET">
  <keyword attribute="location">Engineering B.443</keyword>
  <keyword attribute="color">no</keyword>
  <keyword attribute="papersize">A4</keyword>
  <keyword attribute="papercount">81</keyword>
  <keyword attribute="postscript">yes</keyword>
  <keyword attribute="maxResolution">600*600</keyword>
</service>
```

Figure 3. A sample printer service definition

Figure-1 shows fundamental element hierarchy that the HostSchema defines for mobile host instances. Topmost element '*mobileHost*' is characterized by a mobile host ID (*mID*), a physical address (such as an IEEE 802.11 address) and homes all other properties of the host under definition. The '*currentMove*' child of the '*movement'* element represents the host on move by giving its current coordinates (for a given space), its current speed, and current location in a human-readable form, if classification of the field exists. Continuous updates must be made for this element in order to correctly simulate the moving host, given a mobility model. Depending on the use of HostSchema conforming mobile host object, properties of the hosts' next probable movement may well reside in *'nextMove'* element if a mobility prediction scheme is applicable. The '*transmission'* and '*reception'* elements represent mobile hosts' radio resources in the sense that a transceiving range, power and shape characterizes the RF communication properties. '*battery*' element is for the mobile hosts' finite resource, consumed on every transaction realized. The '*nearNeighbours*' element is an important element, keeping the list of hosts' MAC layer neighbours, such that the host may communicate with those hosts without the need of a routing scheme. These neighbourhoods are easily extracted given a simulation with hosts' transceiving ranges and current locations. In order to maintain any routing scheme, host will be in the need of these near neighbours for relaying. To implement a secure service discovery protocol, hosts need cryptographic keys that are essential for many well-applicable security protocols. The '*security'* element is the place where related keys of the host are stored. A mobile host instance with the above elements and attributes are given in Figure-2.

A proper subset of the mobile host elements are used to define and maintain a fixed host object, still conforming the same HostSchema.

3. Service Definition Schema

Upon starting to offer a service, a host creates (via its service agent) a service object instance conforming to the *ServiceSchema*, in which a service is defined via its attribute-value pairs as suggested in SLP by IETF[6]. A well-formed and valid instance of ServiceSchema holds all the necessary information for a host to correctly determine whether its need for a service can be satisfied by this instance. This service definition instance travels the network encapsulated in an appropriate broadcast message together with other information added by hosts on route.

An example printer service definition is given in Figure-3. Attribute-value pairs of this printer service in the definition help the user agent to gracefully select the appropriate service currently available to the host needing the printer service (i.e. user agent may try to find another printer for the user if application tries to print a colored document of a hundred pages).

IV. THE SERVICE DISCOVERY PROTOCOL (DIP)

The main objective of DiP is to maintain dynamic address binding to named services in an infrastructurless network. We assume that each node has a unique network identifier (*mID*) besides its MAC address (*phyaddress*) defined in its object instance. A s*ervice object* instance may be bound to more than one *host object* instance indicating the existence of replicated services in the network. A client who wishes to use the service, binds itself to an instance of it. Server nodes advertise their service instances periodically and client nodes keep the anouncements in their local service table with an attached anouncement timestamp. A routing decision is made if multiple services are advertised in the network. Once the address of the 'best' service is determined, a session with the returned handle is established. The route for an active session needs to be maintained transparently without changing the end-to-end binding. Route maintanence will be discussed in another paper.

Two types of agents are defined in the protocol. The ***user agent*** acts on behalf of the client node. Tasks executing on the client describe the services they need in an expressive way and the agent returns a handle for binding to the "best available service". The user agent also keeps track of the current routes for each service instance it has discovered and uses this information for the route maintanence. The ***service agent*** acts on behalf of a service application and announces the availability of a service (*service_enable*) or its disconnection (*service_disable*) from the network in a predefined time interval.

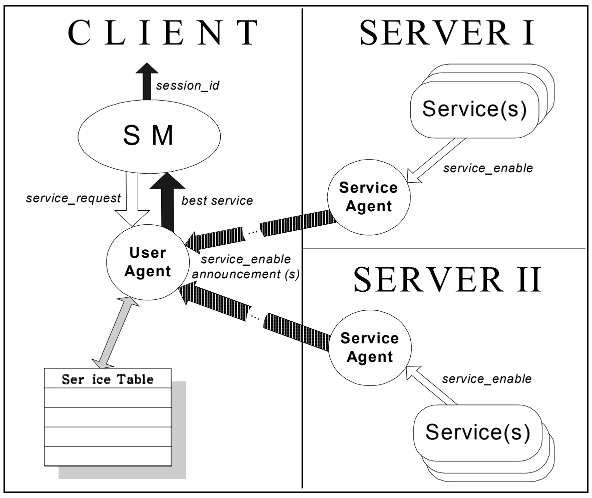

Figure 4. Message flow during service anouncement

1. The Discovery Algorithm

The nodes that are advertising services, periodically broadcast an anouncement to the network. This anouncement is caught by all reachable nodes in the same transmission range and kept in the local service table. The nodes increment the hop count and forward the anouncement by appending their own address to the message. Each node receiving the same anouncement more than once identifies it by the unique anouncement–id. The hop-count is used to prevent circular flooding. The same anouncement arriving with a hop_count greater than the previously arrived anouncement is discarded. Eventually all reachable nodes in the network get the anouncement with the full route record. Since flooding is used, the shortest path will have been arrived among the multiple anouncements. As the performance of the existing adhoc routing algorithms is extensively published [17,18,19], they will not be discussed further. The proposed discovery algorithm inherently returns the shortest path available. Multiple routes may be kept in the service table and used if the established route fails during the session. Figure-4 depicts message flow between the session manager (SM), user agent and service agent during discovery.

The algorithms for ***service-enable*** *and* ***service_disable*** operations originating from

```
service_enable( s,m ): $S \times M \rightarrow \{true\}$
begin
$m \in M$ , $s \in S$
$M' \supset M \leftarrow \forall m' \in M' : m <> m'$
$\forall m' \in M'$: {
    $e \in (SxExRxI) \rightarrow m'$: $e = s \cup$ 'enable' $\cup\ m \cup$ all-id's
    $T_{m'} = T_{m'} \cup s$
    if anouncement-id not seen before {
        m=m'
        service-enable(s,m)
    }
return()
}
end
```

Figure 5a. The *service_enable* algorithm

```
service_disable(s,m): $S \times M \rightarrow \{true\}$
begin
$m \in M$ , $s \in S$
$M' \supset M \leftarrow \forall m' \in M' : m <> m'$
$\forall m' \in M'$: {
    $T_{m'} = - T_{m'} - s$
    $e \in SxExI \rightarrow M'$: $e = s \cup$ 'disable' $\cup$ anouncement-id
    if anouncement-id not seen before {
        m=m'
        service-disable(s,m)
    }
return()
}
end
```

Figure 5b. The *service_disable* algorithm

the server are illustrated in Figure-5a & 5b. User originated message broadcast; ***service-lookup*** algorithm is given in Figure-6. All algorithms are presented in a pseudo-code like notation based on predicate calculus and set theory. General set definitions used in the algorithms are given below:

M	← set of all mobile nodes
S	← set of all available services
T_m	← service table entries on node m
R	← set of ordered lists of nodes indicating the route
E	← set of messages defined in the protocol {enable, create, accept,disable}
C	← set of active sessions in the network
I	← set of integers (all unique id's).

<> : nodes in the same transmission range.

When a task wants to access a network service, it passes the descriptive service name to the ***session manager (SM).*** The route selection is made by the user agent and session establishment over the route is done by the SM. The user agent gets the list of

```
service-lookup (m, s)
begin
$m \in M$ , $s \in S$, $t \in T_m$
$T' \supset T_m \leftarrow \forall t_1, t_2 \in T' : t_1.name = t_2.name$
$\exists t' \in T' \leftarrow \forall t \in T'$: t'.timestamp < t.timestamp
return(t')
end
```

Figure 6. The *service_lookup* algorithm

available services and the full route records from its local service-table. If the required service exists with an acceptable timestamp *(current_time – service_timestamp < threshold)* binding can be done without generating any additional network traffic. All nodes are either active or passive name resolver in the environment. If there is no local entry for the service, user agent broadcasts a look-up message to the network. In the first stage, nodes in the same data link get the message. Each node checks its own table to see if there is a valid entry (recently anounced) for the requested service. If so, returns a direct reply to the requester and stop flooding. Otherwise, the node waits for a random amount of time to see if any node will reply to the request. If not, then broadcasts the request in its own transmission range. Each service entry in the table has an attached timestamp indicating the time when the service anouncement was originated. The user agent may choose the most recently advertised service as the "best" instead of the shortest path with the assumption that the probability of the recently advertised route being up-to-date is greater than an older advertisement. If the look-up message arrives at the service provider, the server may either decide to generate an immediate anouncement or wait for a random period of time for the session request thinking that at least one node in the network would have a service entry in its table and would have replied to the request. The random delay introduced may prevent unnecessary message overflow.

2. Session Establishment

The session manager(SM) obtains a binding for the best service available from its user agent. The binding info includes the full route record besides the address of the service. SM forms a *create-session* request and sends it to

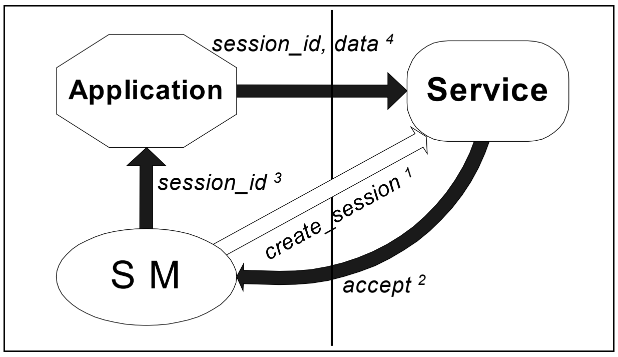

Figure 7. Message flow during session establishment

```
create-session (c,s): MxM → Cx{success,fail}
begin
c,s ∈ M
e∈ ExI : M→ M : e= ‘create’ ∪ session-id
if ∃e∈ ExI : e=‘accept’ {
  (∀j∈ C : j ≠ i)  Λ (C ≡ C ∪ j)
  return(j, success)
}
else if ∃e∈ ExI : e=‘reject’
    return(fail)
end
```

Figure 8. Session establishment algorithm

the first node on the route. This process is repeated by the intermediate nodes until the message reaches the service. The server returns an *accept* message and the session is established. SM returns a session-id to the user, over which dataflow can begin. Message flow during session establishment is illustrated in Figure-7 and the algorithm is given in Figure-8.

V. FUTURE WORK AND CONCLUSION

The proposed model will be augmented for completion by defining the detailed structure of the service announcement messages. Depending on the needs of the application, user agent would make use of the information that has been gathered from the hosts on the current route to the service. (i.e. A service announcement that is delivered via a mobile host with low battery indication may effect the selection of the service phase.) Additionally, routing maintenance, multiple session management and service migration are to be considered for a complete discussion of the topic in the sense that the creation, maintenance and termination of an adhoc network should be clearly defined.

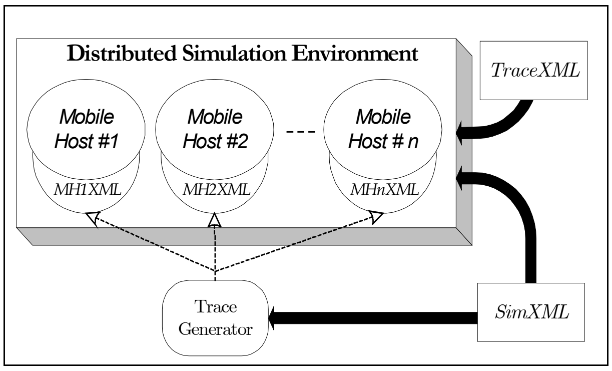

Figure 9. Simulation environment

Different scenarios of mobility must be used in order to evaluate the performance of the proposed model. (i.e mobile hosts acting for common goals or moving based on daily needs). For this purpose, the simulation environment is being designed carefully to apply different mobility models without affecting the service model itself.

Mobile hosts intending to form adhoc networks just based on their service needs will be simulated in a distributed simulation environment, where the movements of host instances on nodes of the simulation environment are driven from a trace generator. The trace generator and mobile host instances are synchronized using barriers as illustrated in Figure-9. Also pre-defined or generated mobility patterns can be replayed for mobile hosts by using *TraceXML* format (i.e. for deterministic testing or to use real world mobility data). Simulation parameters are read from *SimXML* document and XML format is used in all intermediate processing to preserve human readability.

In this paper, we have addressed that applications on hosts of mobile adhoc networks will need a facility to describe *what* they are looking for, instead of *where* to find it. There are some important issues to be covered before introducing the proposed model for wide area usage. First is the need for a domain restriction (i.e. a domain service resolver) in order to have a well scaling protocol. The second obvious need is the integration of a security model to define hierarchical access structure for services.

VI. ACKNOWLEDGEMENTS

The authors would like to thank Nurcan Tezcan who has been implementing the *trace-generator* as her senior engineering project.

REFERENCES

[1] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, *Proceedings of ACM/IEEE MOBICO*M, pages 85–97, Oct.1998.

[2] D. B. Johnson, D. A. Maltz, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, IETF Draft, *http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-03.txt*, Oct.1999.

[3] V.D. Park and M.S. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, *Proceedings of INFOCOM'97*, Kobe, Japan, pp. 1405-1413, Apr.1997.

[4] C. E. Perkins and E. M. Royer, Ad-hoc On-Demand Distance Vector Routing (AODV), *Proceedings of WMCSA99, 2nd IEEE Workshop on Mobile Computing Systems and Application*s, pp. 90–100, Feb.1999.

[5] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph, and R. Katz, An Architecture for a Secure Service Discovery Service, *Proceedings of ACM/IEEE MOBICO*M, pages 24–35, Aug.1999.

[6] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan, Service Location Protocol, *RFC 2165,* Jun.1997.

[7] X. Hong, M. Gerla, G. Pei, and C.C. Chiang, A Group Mobility Model for Ad Hoc Wireless Networks, *Proceedings of ACM/IEEE, MSWiM'9*9, Seattle, WA, pp.53-60, Aug.1999.

[8] G. Pei, M. Gerla and X. Hong, LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility, *Proceedings of IEEE/ACM MobiHOC 200*0, Boston, MA, pp. 11-18, Aug. 2000.

[9] M.Zonoozi, P.Dassanayake, User Mobility Modeling and Characterization of Mobility Patterns, *IEEE Journal on Selected Areas in Communications,* Vol. 15, No. 7, pp. 1239-1252, Sep. 1997.

[10] Jini ™ , *http://www.sun.com/jini/.*

[11] Christian Bettstetter, Christoph Renner, A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol, *Proceedings EUNICE 2000, Sixth EUNICE Open European Summer School*, Twente, Netherlands, Sep. 2000.

[12] W.Winoto, E.Schwartz, H. Balakrishan, J.Lilley, The Design and Implementation of an Intentional Naming System, *ACM Operating Systems Review,* 34(5) pages 186-201, Dec 1999.

[13] T. Bray, J. Paoli, C.M. Sperberg-McQueen, Extensible Markup Language (XML), W3C Recommendation, *http://www.w3c.org/TR/REC-xml-20001006,* Oct. 2000.

[14] D. Chamberlin, P. Fankhauser, M. Marchiori, J. Robie, XML Query Requirements, W3C Working Draft, *http://w3c.org/TR/2001/WD-xmlquery-req-20010215,* Feb. 2001.

[15] D. Chamberlin, D. Florescu, J. Robie, J. Siméon, M. Stefanescu, Xquery: A Query Language for XML, W3C Working Draft, *http://www.w3c.org/TR/2001/WD-xquery-20010215,* Feb. 2001.

[16] D.C. Fallside, XML Schema Part 0: Primer, W3C Proposed Recommendation, *http://www.w3c.org/TR/2001/PR-xmlschema-0-20010330,* Mar. 2001.

[17] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva, A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, *Proceedings of ACM/IEEE MOBICOM* Dallas, TX, pp. 85-97, 1998.

[18] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek and M. Degermark, Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks, *Proceedings of ACM/IEEE MOBICOM'9*9, pp. 195-206, Aug.1999.

[19] S.R. Das, C.E. Perkins and E. M. Royer, Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks, *Proceedings of IEEE INFOCOM 200*0, Tel Aviv, Israel, pp.3-12, Mar.2000.