Chapter 3

LITERATURE SURVEY

Introduction

The term *service* is used for multiple meanings and can be defined according to various perspectives. However, a common sense definition for a service is a set of goods or valuable functions offered by a service provider to a consumer. Examples of services include conferencing, online gaming, printing, files and so on. Service providing facilities for application development in mobile ad hoc networks is a difficult task due to the absence of any fixed infrastructure. Service discovery in mobile ad hoc network is challenging because of the issue of mobility. Many of the existing work are suitable for fixed infrastructure; they have given least concentration on mobile ad hoc network’s issues. This chapter focuses on the basic requirement for service provisioning in mobile ad hoc networks and analyse the various existing work in the literature. After the identification of the limitations of the existing work, this chapter provides the statement of the problem that this thesis addresses.

3.1 Requirement Analysis

3.1.1 Overall Requirements for Service Provisioning Framework in Mobile ad hoc networks

The service provisioning framework in mobile ad hoc network has to provide a complete life cycle of any service while sharing. (i) The framework should support service advertisement of its availability, deployment, consumption by the other nodes/consumers and termination of the
services. (ii) The second requirement is that the provisioning should not depend on any fixed infrastructure or any central entity. The architecture should provide mechanisms to advertise and discover the services. It should facilitate effective decision making for service selection. Therefore, it is needed to gather and store information related to the services that exist in the network. Because of the node mobility the node availability and thus the service availability is unpredictable. (iii) To compensate for mobility there need a facility to monitor the changes in the topology of the network and capability of the participating nodes.

(iv) Unannounced and frequent disconnections mandate alternate service re-selection. (v) The placement of the framework in the protocol layer is another factor to be considered. The service provisioning is the problem of application layer. Because of the nature of routing protocols of mobile ad hoc networks there may be dilemma to place this framework at the routing layer namely cross-layer. So, the advantages and disadvantages of these two techniques were studied. (vi) As the context of the user, the network and the provider influences the selection there should be mechanism to match the user context with the non-functional properties provision of the service providers is needed. (vii) Context accessing mechanism has to be there in order to access both the static and dynamic context information. The following sections are dedicated to analyse these requirements in details along with the existing work.

3.1.2 Requirement for Discovery Architectures and Related Publication

Most of the existing service discovery protocols are primarily designed for fixed networks and are not directly applicable for mobile ad hoc networks without adaptations. Since different mobile ad hoc networks of varying size, equipment, applications are there, a variety of proposed service discovery architectures for mobile ad hoc network exists to solve specific purpose. Some of the solutions focus mainly at scalability in order to support hundreds of nodes, where as some other focus on minimizing the latency in the discovery process etc. Whatever may the service discovery architecture (directory-based, directory-less, or hybrid) or the discovery mode (reactive, proactive, hybrid) there are two possible approaches when designing a mobile ad hoc network service discovery protocol based on the placement of the discovery mechanism.

Placement of the Architecture
Application-layer service discovery – Refers to protocols independent of the underlying routing protocol

Cross-layer service discovery protocol – refers to protocols integrated with the Network layer routing protocol, be it either reactive or proactive.

Application layer service discovery

Most mobile ad hoc network service discovery proposals belong to the first category and place service discovery at a layer above routing. Such mechanisms create an overlay on top of the network layer to disseminate service advertisements, requests and replies in the network. Advantages of using this method are: (i) as no assumption is made about underlying network, it is possible to create pervasive service discovery across different network domains (ii) architecture can be based on existing standards, since the size of the service descriptors is not limited by routing protocol (iii) modular and layered approach is maintained making it possible to replace protocols at any layer.

Most application layer service discovery protocols rely on the network-layer support for multicast or broadcast the service discovery messages. They disseminate service discovery through intermediate nodes provided the mobile ad hoc network supports multicasting.

Cross layer service discovery

Cross layer refers to the protocol design done by actively exploiting the dependence between protocol layers to obtain performance gains (Srivastava and Motani (2005)), though they violate the modular layered approach.

Cross layer techniques in service discovery are developed in order to benefit from information available at lower layers of the protocol stack such as link lifetime estimation using node location and the movement estimated from signal strength (Zhao cheng (2008)). Thus, most of the approaches are based on integrating the routing process with service discovery process. The idea of providing routing layer support for service discovery was first introduced by Koodli and Perkins (2002). Fan and Ho (2007) have provided additional extension to the integrated AODV protocol to also support quality of service aware service selection.
Service discovery protocol usually relies on a number of lower layers, such as network or link layer. They benefit from using those services provided by the lower layers. The specification of how the protocols make use of the lower layers and what operations are performed for achieving service discovery is called service discovery architecture.

Service availability and the information related to the services have to be disseminated through the network so that they can be availed by the participating nodes. As far as the research of service discovery is concerned there are three types of networks exist. First are the wired, second one single hop wireless networks and third are the wireless multi-hop mobile ad hoc networks. Information retrieval of available service relies on the storage system type. As the storage system type directly influences the performance of the service discovery protocols in terms of scalability, mobility support and resource awareness.

Depending upon the network type, various storage systems can be designed. In the case of wired networks the availability of intermediate storage may be of sure, whereas in the case of mobile ad hoc networks there may not be substantial amount of intermediate storage. Thus, the protocols suggested for one type of network will not be suitable for other networks. In the first type devices do not move, where as in the second type very restricted mobility nodes forms the network, so there may be one or more stable nodes in the network, but in the third case of networks the nodes can join/leave without any restriction leads to no fixed node. The architecture mainly depends on having or not having a directory. With respect to directory, the protocols could be directory based (centralized), directory-less and hybrid.

The Service discovery protocol architecture for mobile ad hoc networks can also be categorized by having overlay network support or not. In an overlay network a node which is a part of an ad hoc network, knows the address of another node of the same network can communicate with it, then it can be said that there is an overlay link between the two nodes (Michael Klein et al (2003)). Overlay link may or may not be a direct physical link with the other node. They will be connected by one or more intermediate nodes. An overlay network is a collection of such overlay links and the nodes they connect.
Overlay can be: *Structured* – when there is some organization between the nodes forming the network. This pattern is used in the works of Campo et al (2006), Varshavsky et al (2005) and Michael Klein (2003). The structured overlay has the advantage of controlled multicast of the advertisement and queries which leads to reduction in network traffic. *Unstructured* when there is no organization of forming the overlay Allia (Olga Vladi Ratsimor et al (2002)), (George Oikonomou (2011)).

*Directory-based centralized architecture*

A node in the mobile ad hoc network can be a server, a service requestor or the directory node or a mediator node. Service providers register their services to service directories. The directories will store all services available in the network. The requestors know about the availability only through these directories. One or more nodes are chosen as service coordinators, which are similar to Directory Agents (DA) in SLP or look up service in JINI. These coordinators announce their network periodically by flooding. Directory based protocols include the one described by Tyan and Mahmoud (2004).

*Directory-less or decentralized directory or distributed directory based architecture*

This can be further sub divide into structured and unstructured

*Unstructured*

In this type of storage mechanism the communication is based on broadcast or multicast. Every node has a local service directory maintained as a limited-time cache. The service providers floods the advertisements and clients broadcasts the query requests. The clients and the intermediate nodes match the queries with the stored advertisements as in Konark (Helal et al (2005)), and other defined by Campo et al (2006) and George Oikonomou (2011) Allia (Olga Vladi Ratsimor (2002)). Jing Zhou (2011) developed a protocol based on peer-to-peer.

*Structured*

These methods are commonly found in the context of large networks. Directory nodes organize themselves in a structure that allows them to route the discovery messages in a limited
number of hops. The structured distributed system can be further classified as *hierarchical, flat and hybrid.*

**Hierarchical:** In this type the protocol maintains directory nodes in a structure like tree, ring etc. The information, advertisements and queries are propagated up and down through the hierarchy. CSP (Lee and Helal (2003)) uses a hierarchical structure of directories and aggregation of service descriptions, based on centroid aggregation method. Search is directed through the route and follows the hierarchy for information flow. The root may be a bottleneck in these cases.

**Flat:** The protocols construct overlays by means of distributed hash tables as in Pastry (Rowstron and Druschel (2001)). DHTs are used to store key-value pairs and the addresses of the service provider nodes. Based on the hash function on the value the resulting key is used for routing the requests to the designated nodes. Each node maintains a routing table with the addresses of the other nodes in the overlay. These protocols mostly rely on peer-to-peer mechanism.

**Hybrid:** Architecture combines the above two architectures which contains both advertisements and queries. Basically they use both of these approaches to avoid large amounts of advertisements and queries. In adaptive service discovery model the advertisement and query frequency is regulated. Service providers within the announcement scope of one or more Service Coordinators will register with them their available services and access information. Lanes developed by Michael Klein (2003) builds up a two dimensional CAN structure overlay. Service announcements are propagated throughout a lane, whereas service requests are sent to other lanes too. Overlays can also be built based on virtual backbone nodes (Ranwa Al Mallah and Alejandro Quintero (2009)).

**Discovery mode**

Independent of the chosen service discovery architecture, service information can be gathered either in a reactive, proactive or hybrid way.

**Reactive mode:** In reactive mode, No node send any advertisement to announce their service information. Therefore, the node in the network does not know where services are
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.

**Proactive mode:** A proactive mode implies that, service providers proactively distribute their available services. 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:** A hybrid discovery mode 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 creating service request.

**Service description**

Service description format is directly related to the size of the advertisement packets and to the storage space used for caching. Services often described in many ways such as textual form, attribute-value pairs etc. which can contain lot of information. But all the information is not required for discovery, though they may be required at the time of the utilization of the service. If they are supplied at the time of advertisement it consumes space and bandwidth unnecessarily. But at the time of service consumption they may be used. But requesting service invocation is unicast to the provider, compared to the broadcast or multicast nature of advertisement this will consume less bandwidth. So the level of information to be sent in the advertisement has its impact on the bandwidth. This leads to the decision of using methods of space-efficient and enough detailed service advertisement in order to reduce the amount of queries during the discovery phase.

Service description has to be comprehensive and unambiguous. The description should clearly specify what the service will do, how and where it can be accessed with a clear syntax. The description should be machine interpretable and human readable to facilitate both automation and ease of formation for the user.
A service may have many attributes, which are the characteristics of the service. As an example a printer service can have the following attributes associated with it like position, resolution, colour, type etc. Each attribute can be associated with a value. Service information and its attributes can be categorised (Marin-Perianu (2008)) into the following types:

- **Textual**: A service can be described using a textual description. The service description chooses a set of keywords and associates them with key values, the searching algorithms will search or these keywords (Castro (2002)).
- **Attribute-value pairs**: the most widely used description format is attribute-value pairs. An attribute is a category in which an service can be classified, for example printing color of the printer, the value can be either “black-weight” or “color” (George Oikonomou (2011)).
- **Hierarchy of attribute-value pairs**: some protocols use a hierarchical arrangement of attribute-value pairs, such that one pair depends on the other (Joakim Flathagen (2008)).
- **Markup languages**: like UpnP (UpnP Forum (2008)), many service discovery protocols uses XML based technologies like RDF, or DAML etc.
- **Object Oriented Interface**: service can be described using an object oriented programming interface. In JINI given by Sun Micro system, service descriptions are expected to be as java interfaces.

Using a complex form may provide the possibility of describing in detail. But it will consume lot of memory resource. The choice of description depends on the requirements of the applications. If detailed description is the demand then use of compression may be the best choice (Marin-Perianu (2005)).

**Service Maintenance**

Service maintenance is a phase needed to have up to date information related to the advertised services in the cache or the directories. Maintenance has to done to cater the need of (i) changes in the service description - by either advertise the change to all the nodes in the network or notify only those nodes which have subscribed to the service will be notified if there is any change in the advertised services. Upon receiving the notification the nodes will change their local cache   (ii) changes in the case of service availability – the addition and deletion of the service information in order to maintain a consistent view of available services the protocols use
two techniques (i) Soft state - Soft state maintenance is done by the use of Time To Live concept. If the time expired the entry will be deleted from the table (Cynthia Jayapal and Sumathi Vembu (2011)). (ii) Hard state by sending explicit de registration message from the server. (iii) Hybrid some protocols uses both the soft and hard state maintenance as in George Oikonomou (2011) and Hosseini-seno (2009). Polling: by which server periodically checks for the availability of the services (Joakim Flathagen (2008)).

Soft state techniques and polling induce a high maintenance cost in terms of traffic. Though hard state maintenance may lead to delays in achieving consistency, hard state can be used with networking information, which gathers rapid identification of server unavailability. Similarly hybrid state methods can be used to minimize overhead.

Service selection

After submitting a query or a service, there may be a chance that, the same service may be provided by more than one server. The selection may be done by (i) Selected by the client: An optimization algorithm implemented in the client side can choose the server automatically. (ii) Selected by the directory: The server can be selected by the directory nodes. The selection may be based on using service metrics like smallest response time, hop count etc.

Search methods

It is important to mention that the extension of the search is conditioned by the dispersion degree of the information in the network. The threshold between the initial dissemination of service description and the extension of the following queries needs to be considered. On the one hand more organized and distributed information leads less effort during searching, on the other hand complex storage mechanisms make the information consistency difficult to maintain. Depending upon the directory structure, the search flow may be

Flood-based: Unstructured distributed architecture uses this method. In order to get or send the service description nodes use this type. The flooding may be limited to a certain number of hops or may cover the entire network (Helal et al (2005)).

Directed flow: Distributed structured architecture uses this method of discovery. Search queries flows to a specific node depending upon the protocol. As an example, it the case of
hierarchical the search directed to the local directory and to the global ones (Cynthia Jayapal and Sumathi vembu (2011)), Information flow follows the hierarchy.

**Hybrid:** They using both the above mentioned flow. The Service discovery protocol based on clustering like the work of S.A. Hosseini-seno (2009), directed flow in the inter-cluster and flooding based in the case of inter-cluster discovery.

**Discovery scope**

Advertisement policy service discovery protocols are different in replying methods to the queries or making announcements to the network. In replying the queries, the servers or their agents can reply to any query they receive, regardless of being necessary or not. In making announcements, they also send periodic advertisements to the network, not caring how clients are interested in receiving the advertisement. This is called blind advertisement. The benefit of blind discovery is simplicity at the cost of redundancy.

**Retransmission policy**

Service discovery protocols use retransmissions of advertisements or queries in different situations. Retransmission of an advertisement can be made for emphasising the advertisement and assurance on receipt of advertisement by the clients, or for refreshing and updating service information. Refreshing advertisements are usually less frequent than retrying ones. Retransmissions of advertisement can stop after a certain time and interval between the advertisements can vary. Retransmission of queries can be made if there are no or few replies from servers. For improving performance and smart advertisements each retransmission may carry the status of already discovered services. This prohibits those severs to re-advertise their services unnecessarily.

**Cache management**

Cache containing service information can expire with time, based on lifetime of the services and change of context. As an example, possibility of unavailability of services in a network can cause cache contents to be expired. Cache management system can deal with
invalidity of data in different ways; it can either remove the data from the cache, or mark it as expired asking for an update from the servers. In addition, servers should refresh service specification prior to expiring lifetime of the services. Choosing proper timeout intervals for refreshing and expiring the cache content is very important, since improper setting values can cause resonance and instability in the system. Proper on time sending an advertisement can prohibit unnecessary cache entry deletion or updating requests.

**Mobility Support**

In networks where dynamics is the primary characteristics, regular maintenance has to be done to have up-to-date information. Mobility has to be provided in the systems which (i) have directory nodes that keep all the information about other services in the network. (ii) Have nodes that keep partial information about the services (iii) form structured overlay networks. The information can be updated in one of the two ways namely Reactive: The information changes according to the events in the network (e.g. link failures, server might have been moved) and Proactive: Nodes maintain a consistent view by periodically exchanging updated messages (Helal et al (2005)).

Implementation of these strategies is depending upon the Service discovery protocol and the storage method used. Methods proposed for handling mobility are:

- Adjustment of service advertisement rate and coverage area: The time interval for service advertisement has to be adjusted depending upon the dynamic environment. The number hops to which the advertisement is sent may be regulated or changed based on the dynamicity of the network (Dipanjan Chakraborty (2006)), (Olga Vladi Ratsimor (2002)).
- Aggregation and filtering: For preventing directories to keep the information up to date and the status of the service availability this technique may be used. A service will register only if there is no other server within the database that offers the same service (Zhu et al (2003)).
- Late binding: Piggybacking the service discovery messages in the routing messages which will be directed to the server. Even if the service changes during service usage
there will be communication between the client and the server by the use of directory nodes (Prezerakos et al (2007)).

- Overlay network restructuring: Structured distributed systems construct overlays based on some structural restrictions, in the case of nodes mobility some restructuring algorithms have to be used to maintain the structure (Michael Klein (2003)).
- Using routing Information: The routing information is useful to determine whether certain node has been moved or added (Varshavsky et al (2005)).

**Protocol capabilities**

The protocols are different from each other in terms of capabilities and functionalities they offer.

**Discovery and provision:** the following phases are needed in operation of service discovery and provision:

- Service discovery: in this phase the client detects the service availability and becomes aware of the types of the available services and their relevant attribute values
- Service Initialisation: in this phase, the client sets up a session with the server for use of the discovered service
- Servicer session: the session is established and the client actively uses the service
- Service termination: either client or server can terminate the service provision session by purpose. Mechanism for managing unintended service termination should be considered here, such as node or link failure.

Support for all the above functionalities in a protocol results in improved performance, because of consistency and completeness of the discovery/provision procedure. The implementation of discovery and provision phase is dissimilar between the investigated discovery protocols. Some of the protocols only cover the actual service discovery, while others also provide the mechanism to actually use and benefit from the services.

**Control and event notification** the service provision phase mainly consists of control and notification signalling. During the service provision phase, there are occasions where clients need to control the server to influence service execution parameters. In addition, there are circumstances under which servers must notify clients about the status of the service being
offered. The control and notification signalling provides a bidirectional reliable link between the client and server, which improves the performance of the service provision.

*Service selection:* a response to a service discovery query may contain several matching services, the user usually needs to select only one service among a number of matched services, one option is manual selection, in which the user is returned in response to a query a list of matched services and he/she selects one of the services. Alternatively, in automatic selection, the most suitable service is selected by the service discovery protocol.

*Service Invocation:* a service is invoked after selection. Existing service discovery protocols, support service invocation at three levels, firstly, they provide service location then at the second level by means of a communication mechanism, the application establishes a session with the service. This can be carried out by using remote procedure calls e.g. remote method invocation in java or using simple object access protocol. At the third level, some of the protocols support application operations

*Service usage:* in most of the service discovery protocols, a client must explicitly release the service for use. The service session can be terminated by one of the parties. However, in few protocols, the service can be leased. After expiration of the lease, the service session automatically terminates unless the lease is renewed.

*Service status inquiry:* there are different approaches for awareness from the status of a service. The service status can be obtained by periodically polling, in which a client regularly asks the service status. Another approach is notification, in which the client subscribes to a service status event and any changes on the subscribed event are notified to the subscribers of that particular event.

### 3.2 Research in Service Provisioning Systems in Dynamic Networks

Dynamic networks are characterised by network nodes that can join or leave the network due to their mobility. Typical examples are local networks in home and enterprise environments, where computers can be connected using wired LAN or Wireless Wi_Fi hotspots or using Bluetooth ad hoc communications. In these environments, connections and disconnections
happen at a slow rate. Therefore service provisioning systems focus on discovering newly connected nodes and the services they offer. Once discovered, invoking these services is not considered as a challenge because a network connection is available between the service and the consumer.

*Java Intelligence Network Interface JINI* is a protocol designed by Sun Microsystems (2011) has an environment for creating dynamically the networked components, applications and service based on Java. It is a service-oriented architecture for building secure distributed systems. It defines a programming model that extends Java technology. The JINI architecture consists of three types of entities: service provider, clients and lookup services. A service can be a software component, a hardware device, or a combination of the two. A service provider uses multicast to find a lookup service and then registers a proxy object of its service. The node lookup service registers devices and services available. A client also uses multicast to find a lookup service, then it specifies an interface that the desired service might implement. If a matching service is found, the client copies the proxy object of the service. This proxy object is then used to directly invoke service provider. JINI uses serialization to send objects over the network. Multiple lookup services can coexist to prevent failure points.

*The Service Location Protocol SLP* was developed by SRVLOC work group of IETF, as a service-oriented computing standard. There are three main agents in the SLP framework: User agent- which issues service request on behalf of the client application to the directory agent using attributes in order to discover matching service, Service Agent advertises the location and attributes on behalf of the service, directory agent, which stores the information about the services announced in the network. If there is no directory agent, client and service agents can still perform peer discovery using multicast when a directory agent becomes available, it multicasts its presence for the client and service agents to use. The SLP does not specify an invocation protocol.

*Universal plug and Play* (2008) was developed by Microsoft and is now maintained by the UPnP forum. When a device joins the network, it advertises its presence, so it could present its description of capabilities on demand (XML descriptions). A control point acts as a directory that collects device descriptions. The control point discovers devices using SSDP, then it asks for a detailed description of the device’s services, then it uses SOAP to discover and invoke specific
actions. The GENA notification architecture reports events to the control point whenever the state of a service changes. UPnP is deployed over TCP.

_Bluetooth Service Discovery Protocol_ designed by Bluetooth Special Interest Group (2007) is a service discovery protocol for Bluetooth devices in proximity (only one hop communications). Service characteristics are described using attribute-value pairs. A client device has the option of searching for specific services by querying a provider device using desired service attributes, for the provider to match and return relevant services. Otherwise using a generic query, the client can ask the provider for a list of all available services. The protocol does not specify advertising, service directories or service invocations.

### 3.3 Service Provisioning Systems for Mobile Ad Hoc Networks

The design of service provision systems for mobile ad hoc networks is inspired by some of the previously described systems for stable and dynamic networks, yet it takes into account the special characteristics of mobile environments.

_Konark_ (Helal et al (2005)) is a directory-less service discovery architecture based on a peer-to-peer model using lightweight HTTP servers. It is a discovery and invocation protocol for multi-hop ad hoc networks. The protocol defines its own description language loosely based on WSDL which has been designed by _W3C CONSORTIUM_ (2007). HTTP and SOAP which is designed by _W3C CONSORTIUM_ (2009) are utilized to handle service delivery. The konark maintains tree-based architecture to cope with service classification. The format support search for either all, generic or specific service in category and the request can be done either using simple keywords or a more fine-grained service description if a specific service is desired. It supports semantic search which though increases the energy consumption, responses are more accurate. It supports both reactive and proactive service advertisements and both servers and clients can actively discover and advertise services on a need basis. A service request is send to a fixed multicast group and all the nodes with a matching service will respond. A time-to-live field is specified in the service advertisement process and enables local caching of service descriptors on each node.
*Pervasive Discovery Protocol PDP* is a directory-less protocol aimed for ad-hoc network defined by Campo et al (2006). Each node has a user agent and a service agent. The user agent processes search information in the network and the service agent advertise service offered by the device. For each advertisement, availability time is included. Entries are removed from cache of each node when the availability timer goes out without being updated. It operates in reactive mode and assumes that the underlying network is either a one-hop network, a multi-hop ad-hoc network with multicast support.

*Adder* is a directory-less architecture targeting ad-hoc networks designed by George Oikonomou et al (2011). It runs at the application layer. It works in both proactive and reactive mode of service discovery. In reactive mode a will form a service request message in order to discover services matching specific criteria. This message is broadcasted to the nodes single hop neighbours. In proactive mode, nodes will periodically send service advertisement messages to their single-hop neighbours. Nodes receiving them will store the information in the local cache and include it in their own advertisements. It does not depend on underlying routing protocol. It tags a metric distance between the advertising node and the provider with all the advertisements. It uses sequence numbering for loop-free message passing also for the state maintenance. Each service is uniquely identified across a deployment by a UUID. Service descriptions has two sections generic contains information about all attributes that are common to all services, such as address and port over which is service is available. It also contains the distance between the provider and the advertising node, TTL value etc. The second section is service specific details. It is a list of key value pair to describe all the services’ details in the ASCII key value.

*SPIZ*, the protocol designed by Donggeon Noh and Heonshik Shin (2007) is a hybrid integrated protocol, where an autonomous and adaptive zone radius determination mechanism is provided. They use call rate, mobility, service popularity for radium determination. Ranwa Al Mallah and Alejandro Quintero (2009) described a light weight strategy for service discovery in ad hoc networks in order to minimize network load and other performance gain. They elect the nodes based on the residual battery power, average node velocity, available processing capabilities and decide these nodes to store the services available to the network. The proposal by S.A. Hosseini-Seno et al (2009) placed the service provisioning module in routing layer and
distributed the services among clustered nodes. They use cross layering in order to reduce control messages sent for service provisioning.

*Mercury* (Joakim Flathagen (2008)) is a cross-layer service discovery protocol. It is a distributed architecture. It acts as a common framework that connects services distributed in the ad-hoc network to users and applications (i.e.) an interface between the application and the network. It uses Bloom filters to describe services. The filters provide optimized description in terms of number of bits. Services are not distributed in human readable form, so only hidden resources and services can be advertised in the network. The service dissemination is done by piggybacking service information on OLSR routing messages. It uses local caching with service handover support.

*K-Directory Community* (Vaskar Raychoudhury et al (2010)) is a distributed directory based service discovery protocol. It selects top K nodes as directories. These K directory nodes are called as directory community. The directory nodes then form quorums among themselves. The services are replicated in the quorum members. The network is divided in to one or more tree-structured domains. Service providers register only with functional information. The detailed service context is forwarded to the user on request. It maintains the domain quorum on lease based.

*Cynthiya Jayapal et al* (2011) designed distributed directory based service discovery architecture. In which the entire geographic zone into several region into zones. The Local Coordinator is selected for every zone and a Global coordinator or the entire region that is common to the entire multicast group. LC is responsible for maintaining the position information and service details for all the requestors and providers within a zone. The service is register by a provider with its LC and when the requestor needs a service it forwards the request to its LC. The LC finds a suitable provider with the available service information or through the GC.
### Table 3.1 Comparison of Service Discovery Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Service Descriptor</th>
<th>Dissemination</th>
<th>Architecture</th>
<th>Maintenance</th>
<th>Discovery Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Konark</td>
<td>WSDL</td>
<td>Multicasting</td>
<td>Directory-less, P2P</td>
<td>Passive service adv. Event notification</td>
<td>Hybrid</td>
</tr>
<tr>
<td>Mercury</td>
<td>Bloom Filter</td>
<td>Cross-layer</td>
<td>Directory-less</td>
<td>Polling</td>
<td>Hybrid</td>
</tr>
<tr>
<td>ADDER</td>
<td>UUID + Text (ASCII)</td>
<td>Multicasting</td>
<td>Directory-less</td>
<td>Hybrid</td>
<td>Hybrid</td>
</tr>
<tr>
<td>Cynthia</td>
<td>Not defined</td>
<td>Broadcasting</td>
<td>Directory-based</td>
<td>Soft state</td>
<td>Proactive</td>
</tr>
<tr>
<td>K-Directory Community</td>
<td>Not defined</td>
<td>Multicasting</td>
<td>Directory-based</td>
<td>Polling by clients, hard state by server</td>
<td>Proactive</td>
</tr>
<tr>
<td>S.A. Hosseini-seno</td>
<td>Not defined</td>
<td>Cross-layer</td>
<td>Directory-based</td>
<td>Hard state</td>
<td>Proactive</td>
</tr>
<tr>
<td>Jing Zhou et. al.</td>
<td>WSDL-S</td>
<td>Multicasting</td>
<td>Unstructured P2P</td>
<td>Hard state</td>
<td>Hybrid</td>
</tr>
</tbody>
</table>

#### 3.4 Requirement for Adaptation - Context Awareness

Context awareness is one of the advanced features of some of the service discovery protocols. Context is defined as the user, network and environment conditions that are variable from time to time. It can influence the demands for a specific service at certain time. It can influence the demands for a specific service at a certain time. Therefore context aware service discovery can improve performance and usability of the services as suited as possible. There are many researches carried out on context aware service discovery, advertisement and selection. Lee and Helal (2003) have proposed a multi-tire service discovery and advertisement by proposing context-aware, ubiquitous service discovery architecture. And categorising service
into three domains: proximity, domain and global. A new attribute presenting context information has been added to service descriptions, containing e.g. distance to the server, server load and service channel.

Services discovered are initially advertised in the proximity and then extended to domain and global. For discovery of potentially wide range of services in global tire, a class based service propagation model has been proposed and evaluated by simulations. Kuck and Reichartz (2007) have proposed another context sensitive service discovery. Their approach is built on a feature-based service context model and the user context. On the server side documents are expanded based on the context and results due to various static and dynamic service features are filtered. On the client side, query expansion with user context is proposed. Another aspect of context aware service announcement is adapting the advertisement rate to the context. If the network context indicated highly changing network, service advertisement rate can be increased, whilst in low dynamic network, the rate can be kept low.

One of the important issues to be discussed is the distinction between context-aware and context-based services. There is a conceptual difference between these two types of context-sensitive services. In case of context-based services context inputs are in fact normal inputs i.e. context is treated as an input parameter before starting service execution. Then the context is as necessary for the service operation as the other input parameters, without which service cannot basically operate. While context-aware services utilize contexts in an indirect fashion, using them as a sort of control facility that help them change the structure of their execution flow when needed. The issue concerned here is context–aware services.

This thesis uses context awareness in service discovery which can improve performance and usability of the services as suited as possible. Context information is dependent on individual systems; in one system they may consider a particular type of information as context information whereas the same may not be considered as so in another system. Context-awareness ensures the adaptation of the application or the middleware/framework which facilitates the services. Context information is any additional information that can be used to improve the behaviour or the performance of the system. When applications are built from different services provided by
or hosted in heterogeneous environment it is challenging to make the applications context-aware, as this requires services to be aware of each other and aware of the context of customers and applications. This challenge is due to the distributed environment. Being context aware allows software not only to be able to deal with changes in the environment the software operates in, but also being able to improve the response to the use of the software. The context-awareness technique aims at supporting both functional and non-functional software requirements.

**Context information and their representation:** The common and popular type of context information is location information. Other types of common information are device profiles, user preferences, devices and network status. To describe context information, different languages and models can be used. A language can be used to model or represent the context information, while in the implementation time the instance context might be described by the same or another representation. In existing context aware system, XML is already used widely for modelling and implementing context information. The PPDL (Pervasive Profile Description Language) is an XML based language that can be used to describe preferences of peer (Ferscha et al (2006)) In the CoSAr prototype; a context sharing architecture is defined. RDF and OWL are also used to describe context information. CC/PP (Composite Capability/Preference Profile) can be used to describe contextual information of capabilities and user preferences. CC\PP is based on RDF, which is mostly used in context-aware middleware and applications. SOCAM (Gua (2008)) used OWL (Web Ontology Language) to describe context information. Various works support the modelling of the context together with the modelling of services, thus context and services can be specified and coupled at the design time. This approach is different from other approaches where the web services are defined in WSDL and supporting components are used to access the context. Prezerakos et al (2007) defined contextUML which can be used to specify context and context-awareness for web services. They defined classes to define context types, sources, context services as well as model context-aware objects with in a service and the binding of context aware mechanism. In CoWSAMI (Athanasopoulos et al (2008)) context information can be integrated along with service modelling phase.

**Storage technique used:** Relational database are widely used to store context information. The context storage can be provided as a separate service or can be a part of context management as in Akogrimo (D’andria et al (2011)). In case of XML and Ontology based systems, context
information is retrieved in XML or ontology, the back-end database in many of those systems are actually relational databases. It is because there are many libraries supporting the serialization of XML to relational table.

*Distribution techniques:* Context information can be distributed in three ways

(i) **Using direct transport protocol** – context information will be transferred between two parties based on SOAP messages. The basic idea is to define context in XML and use the SOAP message header to transfer context information (Little et al (2007))

(ii) **Using overlay network protocols:** Service based context aware systems also depends on the distribution technique build on top of an overlay. They either depend on WS-Notifications or specific protocols. The distribution can also be centralized or P2P (Truong et al (2008), (2007)).

(iii) **Supporting access mechanisms:** The context information can be accessed context store or through web interfaces. Context information can also be retrieved from the context management service by query (Truong et al (2007)) or subscription.

*Adaptation*

Many context-aware middleware allow the developer to specify actions that should be performed, given particular contexts. In most cases, the middleware just support the management and exchange of contextual information. Adaptation can be specified at design time or at run time. The adaptation in a service can also be supported by the middleware and framework or has to be implemented by the services itself using supporting components for context-awareness. The reasons for adaptations may be

*Service selection:* context information is mostly used to select the most suitable service to perform actions, at a given situation.

*Communication Optimization:* Context information is used to select communication protocols and optimize the communication. One example of this purpose is to utilize context for optimizing communication in SiWS.

*Content adaptation:* Context information can be used to adapt content resulting from a request and to return the content in a suitable form to the requester (Han, B et. al. (2004)).
3.5 Research in Context Awareness

ESCAPE (Truong et al (2007)) framework is a web services-based context management system for teamwork and disaster management. ESCAPE services are designed for a front-end of mobile devices and the back-end of high-end systems. The front-end part includes components support for context sensing and sharing that are based on web services and are executed in an ad hoc network of mobile devices. The back-end includes a web service for storing and sharing context information among different front-ends. In ESCAPE framework context distribution is based on a specific protocol as well as web services. Context information can be retrieved from the context management service by query or subscription. Context distribution is based on specific protocol as well as web services. Context information can be retrieved from context management service by query or subscription.

The inContext project by Truong et al (2008) provides various techniques for supporting context-awareness in emerging collaboration. It is designed for web services based collaborative environments. Reference to the context source is transferred using SOAP message headers and context information can be retrieved from the source specified by the reference. The distribution technique supports both centralized and distributed models for context distribution in which context information can be accessed from a context store or different services which provide context information.

The project Akogrimo developed by D’andria et al (2011) aims at supporting mobile users to access data, knowledge and computational services on the grid. It concentrates on context that is related to situations of mobile users, such as presence, location and environmental information. The core component in this project is context manager which collects contextual information and delivers it to the application. Akogrimo does not provide context storage as a separate service. Instead, context storage is part of a context management.

Handorean et al (2006) uses context-aware information to support predictable service provision in mobile ad hoc networks. Discovery is implemented using tuple spaces. A node places a tuple in a tuple space to make it available to all other nodes that are sharing the same tuple space. A node needing to read something from the tuple space creates a template describing
the interesting tuple. The tuple system enables a coordination model. It enables a client node to
discover the available provider nodes in its proximity. The system implements a session
management protocol called “follow-me”. A client starts by invoking a neighbour provider by
opening a session. When the client moved, the session migrated to another close by provider that
can be accessed by the client. In case of a disconnection, the session can be reopened at another
provider when one comes into the client neighbourhood. The system uses strong process
migration, everything needed for the execution of code on another node. Sometimes volunteer
nodes can play the role of a service provider. The system uses one hop communications.

<table>
<thead>
<tr>
<th>System</th>
<th>Context information type</th>
<th>Context representation</th>
<th>Storage technique used</th>
<th>Context distribution</th>
<th>Adaptation technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>inContext</td>
<td>Location, individual preference, machine,</td>
<td>UML+OWL at modelling, XML,RDF at implementation</td>
<td>Both centralized, distributed. Stored as RDF</td>
<td>Centralized, accessed through query</td>
<td>Service and task selection at middleware</td>
</tr>
</tbody>
</table>
### Table 3.2 Comparison of Context Aware Frameworks

<table>
<thead>
<tr>
<th>Framework</th>
<th>Location, individual preference, machine, network</th>
<th>XML at modelling, tool specific at implementation, Centralized, relational database</th>
<th>XML at modelling, tool specific at implementation, Centralized, accessed through subscription</th>
<th>Content, task selection at application layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akogrimo</td>
<td>Location, individual preference, machine, network</td>
<td>XML at modelling, tool specific at implementation, Centralized, relational database</td>
<td>Centralized, accessed through subscription</td>
<td>Content, task selection at application layer</td>
</tr>
<tr>
<td>ESCAPE</td>
<td>Location, individual preference, machine, network</td>
<td>UML at modelling, XML at implementation</td>
<td>Distributed, Stored as XML</td>
<td>P2P, accessed through query as well subscription</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Service and task selection at application</td>
</tr>
</tbody>
</table>

### 3.6 General Issues in Mobile Ad Hoc Network service Provisioning

Centralized storage solutions are typically found in small to medium size local area networks, where service registries are usually available. Unstructured distributed storage is commonly used within infrastructure-less environments. However unstructured distributed storage may lead to high discovery cost. Thus the hybrid method of storage will be the better choice. As the unstructured and structured distributed storage leads to use flooding or directed flow of searching they may cause higher cost either at the time of construction or search. Hybrid search can be better choice. Provision systems are primarily intended to support network-wide service discovery. The goal is to enable every client node to discover service providers with a high success rate. In addition, systems try to minimize the network traffic induced by the discovery protocol using different combinations of broadcast, multicast or unicast transmissions. The duration of the discovery is also taken into account, which is the time between the moment when the client issues its discovery request and the moment when this client receives a discovery reply. This duration depends on the topology and the density of the network. It can be of the order of seconds in some connected environments.

Compared to the placement of the framework integrated with routing placing it as a separate layer below the application layer will be better choice. These protocols can be called as
application-layer protocols. The mentioned Service discovery protocols, even if they are designed for mobile ad hoc networks and meet several requirements (lightweight, low overhead, distributed) they fail to meet the requirement that independence from routing protocol. Without this independence, users and providers in different mobile ad hoc networks using dissimilar routing protocol cannot talk to each other. Application layer protocols offer better portability than the network layer protocols. Some provision systems are designed so that the service discovery is directly dependent on an underlying routing protocol. These protocols are referred to as network layer protocols or cross layer protocols. Cross layer proposals also require changes to the routing protocol to ensure packet forwarding of service discovery messages. These protocols usually make use of limited service descriptions, but significantly reduce the network overhead due to their use of the routing information.

Among application layer discovery protocols, service discovery is designed using various architectures. Centralized directory architectures are excluded, because a directory hosted by a single node cannot always be accessible to all nodes of the network, it also introduces a single point of failure and does not necessarily have enough resources to server all nodes in the network. Therefore all discovery protocols use distributed directory architectures with two main variations: Peer-to-Peer, or using overlay networks. In peer-to-peer only architectures, each node has a cache containing a list of its services and descriptions of others found in the network. The main disadvantage is that the service descriptions must be periodically advertised from each node (broadcasts or multicasts). Descriptions travel from node to node in order to properly disseminate to all network nodes, which produces large amount of network traffic examples of such systems are Konark and GSD. In systems using overlay networks, the directory is distributed on only some nodes that form the virtual overlay network. Distribution can be done using fully duplicated directories or partially duplicated but coherent directories. The overlay nodes are dynamically and uniformly deployed in the network, so that any client can have access to nearby directory nodes. Directory nodes collect service descriptions and communicate with each other to maintain the known service list, which restricts heavy discovery communications to the overlay network only.

It is observed that resource awareness is often neglected in most of the service discovery protocols. Service discovery protocols which achieve efficient service look up in large-scale ad
hoc networks exploit either flat or hierarchical overlay structures. But the most suitable one are unstructured distributed storage. Hard state maintenance may lead to delays. Networking information is used in this thesis for a rapid identification of server unavailability also soft state maintenance.

The minimal method to describe a service is using unique identifiers UUID (Universal Unique Identifier) that should be known to all network nodes. This minimal description is mostly used in network layer discovery systems. More attributes can be added to the UUID to enhance the description. A small size description helps lower energy consumption and network bandwidth. Meanwhile other systems enable enriched service descriptions in order to help clients with service selection. These descriptions mostly use attribute-value pairs written in XML-based languages, describing functional (e.g. konark) and appending non-functional service properties (e.g. Sailhan et al). Non functional properties are mostly quality of service properties to help client select the closest provider node or the most available one in an effort to limit long-distance and failure-prone invocations. Descriptions can also help classify services according to predefined trees (e.g. Konark, PDP). Service selection is usually done manually by the client, except for some systems where selection is automatically incorporated in to discovery protocols.

There exist two main modes of discovery interactions between clients and providers: push and pull modes. In push mode also called reactive discovery providers push their service descriptions using unsolicited advertisements and clients listen and select the services they need. Provider advertisements are usually broadcast to the one hop nodes or multicast with multi-hop support. In pull mode or proactive discovery, a client asks for needed services by sending a discovery request directly to provider nodes or to directory nodes. The discovery request is either sent using broadcast, multicast or unicast. A provider or directory answers back with matching service descriptions.

Once a matching service found, binding process is completed by selecting it. Selection consists of informing the task of the chosen service and is accomplished by saving the provider’s network address in the task’s state. Once saved, the task is able to access the service by invoking it. Note that this address is hidden from the application developer, who is able to invoke the service based on its name. The above mentioned provision systems offer varying support to service invocation. A large majority of systems designed for connected mobile ad hoc networks
rely on dynamic routing protocols, using one-to-one or one-to-many invocation. But few provision systems present characteristics viable in disconnected in mobile ad hoc networks, that is without routing or overlay directory networks, but still they do not enable network-wide invocations. Some only enable proximity-only invocations, where a client and a provider are neighbours. And others enable proximity-only invocations with service migrations to maintain service sessions (Radu Handorean (2006)). Other solutions propose to handle service disruptions using dynamic service-oriented computing. In dynamic SOC, services register and unregister themselves at the directory according to their contextual availability and clients therefore can dynamically use the available provider. Lionel Touseau (2008) proposed service level agreements for dynamic SOC, where the client and the provider agree on service disruption concerns.

The dynamic nature of the mobile ad hoc networks should be reflected in the bindings between applications and services since the set of services that are available and the wireless link quality between them the application and the services is continuously changing environments. Adaptive service provisioning is critical since selecting the correct service provider may result in significant energy savings and improvement in quality of service. The service discovery and binding has to be done by the framework/middleware so that the complexity of application development will not be increased. In order to facilitate this adaptiveness this thesis concentrates on the issue of context-awareness.

The context information for distributed application with mobile nodes invalidates the use of centralized context management. Currently most work relies on centralized context management. In some cases they use P2P model of context storages, in fact based on the consumer-registry-publisher model rather than adaptive P2P context registry system with distributed query and subscription support. A fully distributed P2P based context management and sharing system would be beneficial.

Context information modelling: Currently in most of the system use XML, UML and Ontologies. The use of Ontologies allows the inclusion of existing common Ontologies. However the issues related to context update has to have some deeper study. Modelling context information with service description: Many works have developed context descriptions along
with service descriptions at design time. It will be useful to have dynamic context information access along with design time context for better performance.

### 3.7 Problem Statement

Using appropriate service provisioning architecture in the mobile ad hoc environment is crucial to help spreading of mobile ad hoc networks in everyday life. This architecture must be able to cope with the inherent properties of mobile ad hoc networks. In general, a mobile ad hoc network consists of resource constrained heterogeneous devices which can move freely and which organize themselves in an ad hoc way to form a network. The communication links between the nodes are unreliable and error-prone wireless connections and there is no pre-established central infrastructure to administer and manage such networks. The architectures of traditional service provisioning solutions used in communication and data networks (e.g., Jini, UPnP) are not well suited for mobile ad hoc networks and thus it is difficult to adapt them for the mobile ad hoc environment. Usually they are based on centralized service management model using the client/server architecture. The shortcomings of the client/server architecture are the low fault tolerance due to reliance on a single, central server node and limited scalability of the server. So, fully distributed model using the peer-to-peer architecture is the only possibility to meet these shortcomings of having centralised server.

In traditional service discovery protocols focus on capturing in their service advertisements and queries only the static characterisation of services and clients. They do not take into account the user’s current needs and situation or control policies. Service selection in mobile ad hoc networks not only based on the functional properties of the services they also depend on the non functional properties like cost, quality etc. These properties may also vary depending upon the user too. Therefore, service user should be provided with a mechanism to express their wish and the preferences among the properties. Then only the selected service will fulfil the need of the user correctly. Guaranteeing more accuracy in the selection of available services is especially beneficial to mobile users, who are often on the move with constrained resources.
To increase the stability of the selected service provider node, taking node mobility and the battery power remaining into account is indispensable. High mobility of the node can result in frequent changes of service provider node and thus the binding changes. Thus selecting the service providers based on their metrics which takes into consideration of moving speed and battery power remaining as metrics will lead to a stable provider selection. A mechanism to calculate the metrics of the providers before assigning them to the client node have to be developed.

The mobility of users, terminals and service components require novel middleware solutions to handle a set of bindings to needed services. The bindings to the services must be properly arranged to maintain service accessibilities. The choice of the proper binding depends on several factors, from runtime conditions and access-device properties to security requirements and user preferences. Mobile applications require greater binding flexibility than is provided by conventional approaches, in which developers hard coded strategies into service logic. The flexibility in binding relieves from the requirement of generating multiple statically specified versions of an application to accommodate different scenarios.

Adaptation is the process of making adjustments to suit the environment and to adjust different conditions. Adaptation is very important in dynamic and ever changing environments. The adaptation problem is one of the biggest challenges for software engineering (Jacqueline Floch (2006)). This is particularly the case since mobile and pervasive computing have tuned adaptation from the slow process of software evolution into a highly dynamic run-time procedure that needs to occur as devices and applications are on the move. There are different layer where adaptation can be applied: the service execution and deployment adaptation, the resource adaptation and so on. In author of this thesis is interested in adaptation as a technique of service reselection and binding choice of services.

Next major step in service provisioning is maintenance. This step is not needed in the fixed network where the service provider will not move. Whereas this step essential in the case of mobile ad hoc networks in which node mobility is a norm. When something new happens in the environment it should have a means to identify them and adapt itself according to that. When some new services available which make the selection perfectly fit and provide best of what it
can do then that service has to be selected and provided to the user. When some services disappear from the environment the action to do is to adapt the environment to this loss. Good service selection localizes communication, which in turn reduces inter-node interferences and allows for multiple concurrent retransmissions in different parts of the network. Less optimal service selection spreads traffic over the network, increases interferences and reduces overall network throughput.

Over time, changes in network topology degrade the optimality of service selection requiring clients to continuously re-evaluate their choice of a service provider; a process referred to as reselection. And even actively probe the network for availability of new service providers, a process referred to as rediscovery.

Many research fields are interested in one of these development trends, but few have tried to combine all these functionalities into one framework. Current works in the literature offers mechanisms to access context information, defining non functional properties and requirements, identifying best providers. But no unified work is there to propose mechanism to access the current context, use context for best service selection, binding and rebinding decision in the field of service provisioning.

The main objectives of this thesis is therefore; to find the required components and to propose a framework which enables the application developed for the Mobile ad hoc networks by taking advantage service oriented architecture for provisioning of services. To create APIs which are used by the client to access the services in an efficient and satisfactory manner based on his/her current context.
Summary

Based on the analysis of the state-of-the-art in service discovery and provisioning protocols a set of essential requirements to be fulfilled by the proposed framework have been derived. Several sets of requirements at different levels: service architectures, service publication/discovery mechanisms, service provisioning in mobile ad hoc networks, service execution architectures were analysed. Different aspects of the protocols which differentiate various protocols from each other, such as structure, architecture, protocol capabilities, context awareness and service description etc. have been discussed and they have been compared based on which requirements for the proposal has been finalised.

Service discovery and provision protocols discussed were belonging to different categories: from generic protocols, some particularly proposed for mobile ad hoc networks. Although several protocols have been proposed in this area, they suffer from disadvantages which each of them exposes when they are applied to mobile ad hoc networks. Some of the short comings are: (i) some protocols depend on specific routing protocols for example Konark, DEAPspace, depends on a specific ad hoc routing protocol such as M_ZRP, ODMRP-based, OLSR-based etc. (ii) some depends on high-cost and heavyweight components/infrastructure like salutation, imposing heavy data traffic load.

Finally, it has been identified that none of the existing solutions meet the requirements of the proposed work. Therefore, novel architectures and enhancement to existing frameworks need to be proposed. The chapter that follows discusses the design challenges for the above mentioned problem statement and discusses solution methodology for each of the issues.