Chapter 2

BACKGROUND

Introduction

In chapter 1, the research motivations and the research issues that have been taken to address in this thesis were discussed. This work fits into a wider context set such as service oriented architecture, context-aware computing and policy management. It addresses the important aspect of service provision in mobile ad hoc environment which has been considered only marginally. This chapter will discuss the research background. To start with a brief description about mobile ad hoc networks and its characteristics, routing protocol used in mobile ad hoc networks were given, followed by an elaborate study on service oriented architecture. Next follows an elaborate discussion about cross layering the technique used to access the user context, context-awareness and finally the policy management and their requirement.

2.1 Overview of Mobile Ad Hoc Networks

Mobile ad hoc networks (MANETs) can be defined as a collection of autonomous and self-configuring nodes or terminals that communicate with each other by forming a multihop radio network and maintaining connectivity in a decentralised manner (Xiang-Yang Li (2008)). The term “ad hoc” means that the network is established arbitrarily for a limited period of time and for a specific purpose (Mohapatra and Krishnamurthy (2005)).

Mobile networks are often thought of as mobile computers connected to the internet through cellular networks (e.g. EDGE, UMTS and HSPA (GSM Alliance (2008))) or through Wi-Fi access points (IEEE 802.11 -IEEE standards specifications, 2003). These networks are called infrastructure-based networks. Mobile Ad hoc networks are a different type of networks, where mobile computers can communicate with each other via short-range wireless standards (e.g. Wi-
Fi in ad hoc mode, Bluetooth, 802.15 (IEEE Standards (2002)). The nodes can communicate with each other relatively bandwidth constrained wireless links by wireless medium without any infrastructure. If the nodes are not having direct link, they can communicate by the use of intermediate nodes of the mobile ad hoc network. So, each node has to act both as source and also a router for other nodes. Since there is no centralized node to control on routing and also the network topology may change rapidly and unpredictably over time due to node’s mobility, each node is responsible for finding its neighbours by themselves and adapt to the changes in the network due to any node’s mobility. An example ad hoc network is shown in Figure 2.1.

![Figure 2.1 Mobile Ad Hoc Networks](image)

Mobile ad hoc devices and nodes can range from laptops to small handheld gadgets: Palmtop, Personal Digital Assistants, mobile smart phones and the like. Mobile ad hoc network nodes are equipped with wireless transmitters and receivers using antennas which may be Omni directional (broadcast) and highly directional (point-to-point). Due to the nature of mobility of nodes mobile ad hoc networks are suffering from both the challenges of wireless medium for transmission and also the challenges of mobility. Generally the nodes of mobile ad hoc networks need not exchange any information such as configuration files to join the network and so they are called as zero configuration networks (Steiberg and Cheshire (2005)). The scenarios in which the mobile ad hoc networks used (Schiller (2000)) are like office and conference centres, Disaster relief areas, in some specialized forms like Vehicular Ad hoc Networks, Wireless Sensor
Networks. Mobile ad hoc networks have opened up multiple commercial applications such as entertainment, education and collaborative work.

Classification of Mobile ad hoc networks

Mobile ad hoc networks are generally classified according to the communication coverage area. In fact they include four types: Body Area Networks (BAN), Personal Area Networks (PAN), Wireless Local Area Networks (WLAN) and Wireless Wide Area Networks (WWAN).

The infrastructure-less WLAN targets medium-size areas such as campus or an enterprise or an airport. PAN allows users to establish connections with other entities in the surrounding area using personal devices. A BAN is linked to wearable devices and provides connectivity through these gadgets. A BAN can be either interconnected with other BANs to communicate with other people or connected to a PAB for internet access.

Furthermore, they can be classified based on connectivity too. Three kinds of mobile ad hoc networks are defined by Merrill Douglas (2005). They are (i) A Fully connected mobile ad hoc network otherwise called standalone Mobile ad hoc networks. In this type all the nodes are mobile nodes; they communicate each other when they come into its proximity or by using intermediate mobile nodes. These types are very useful since they are easy and cost nothing to set up. (ii) Mobile & Fixed mobile ad hoc network. This kind uses a mixture of fixed and mobile routes. This may even connect to internet world over the fixed wired world. This model is standalone ad hoc networks with an access point to a large network. (iii) Fixed mobile ad hoc network this kind of mobile ad hoc network nodes uses only fixed routers. This model is the result of the integration and coexistence of standalone mobile ad hoc networks with an infrastructure. Usually this integration is achieved with wireless cellular networks. Mobile devices transmit radio signals to those routers, not directly to one another. The routers pass data wirelessly among themselves. One or more of the routers may connect to the wired backbone. Fully connected mobile ad hoc networks are very useful since they are easy and the can be
established even without infrastructure. This pure mobile ad hoc network is the targeted network of this thesis.

*Mobile ad hoc network’s Characteristics*

Mobile ad hoc networks bring new challenges and add new constraints. Several characteristics distinguish them from classical networks.

*Infrastructure-less*: Basically, ad hoc networks do not rely on any infrastructure support. The network must operate independently of pre-established or centralized entities. Network management and routing should be done in a cooperative way. Each node acts as a client, a server and a router in a distributed peer-to-peer mode.

*Dynamic topology*: because of node mobility and membership changes, the network topology varies continually and frequently. While moving, nodes alter their relation to their neighbours. Furthermore, new nodes can join at any moment; whereas connected nodes may leave in arbitrary fashion. Thus, routes break down and unannounced disconnections are to be expected frequently.

*Heterogeneous nodes*: ad hoc networks very often consist of a mix of different devices. In fact, the network is open to any user holding any wireless gadget. As a result, nodes may have dissimilar features, may be of diverse size or may be configured with different software/hardware capabilities. These differences must be taken into account when designing algorithms or protocols for ad hoc networks.

*Resource constraints*: devices have become smaller and smaller, with less resources (e.g. memory, processor, battery power). When these limited devices come into an ad-hoc networks they bring these issues related to these constraints.

*Support for Multicasting and Broadcasting*
Discovery of an entity in a network is mainly based on multicast queries to all entities and waiting for a reply from them. In a mobile ad hoc network, multicast is usually performed by different techniques; however there are some difficulties with multi and broadcasting in mobile ad hoc networks because of the routing issues.

Mobile ad hoc networks implicate difficulties for multicasting and flooding (Obraczka and Tsudik (1998)): multicast routing generally relies on the routers’ state. In mobile ad hoc networks all hosts act as a router, thus every node holds its state. In mobile ad hoc networks 1) behaviour of a host is completely independent from other hosts 2) there is no limit on the speed of the nodes, 3) there are no constraints on direction of the movement of the nodes and 4) the network is highly potential to frequent and temporary network partitions. Because of the aforementioned characteristics of MANTEs, maintaining the states for multicast routing is cost-inefficient. The problem exposes severity while both storage and power capacity of the nodes in a mobile ad hoc network are very limited.

Broadcasting in particular cases, is also shown in (Feeney and Nilsson (2001)) as bring energy consuming than multiple unicasts, because of the energy consumed for processing the broadcast messages on the receiver side, because the nodes do not require to process unicast messages that are not destined to them.

Network Architecture

Different technologies are used at different layers of the protocol stack, which are considered as another aspect of heterogeneity.

Physical layer: At this layer, depending on the type of interfaces (wired/wireless), different technologies exist. In wired mode, Ethernet and USB are the most popular technologies and in the wireless case, Bluetooth, Wi-Fi, Multi-Carrier Spread Spectrum (MC-SS) (Fazel and Kaiser (2003)), OFDM and the like are exists.
**MAC and Link Layer:** A variety of MAC protocols is proposed for wireless ad hoc networks, such as Bluetooth, 802.11a-b, but particularly 802.15.3 and 802.15.4 are standardised for WPAN in high and low data rate modes. Link layer generally is based on 802.2 Logical Link Control (LLC) standards. The structure of the MAC layer protocol is one aspect of MAC and Link layer, interesting for mapping structure of both routing and service discovery protocols.

Bluetooth platform had been defined to provide wireless connectivity within a small coverage area (Bhagwat ((2001)). Research is carried out for enabling coexistence between Bluetooth and Wi-Fi (Landsford et al (2001)). Formation of a Bluetooth network is based on setting up small Mobile ad hoc networks named Piconets. In a typical Piconet there is one master and up to seven slave devices. Bluetooth is based on a completely centralised structure, all the messages is sent through the master and slaves cannot directly communicate with each other.

**Network layer:** IPv4 is mainly used protocol at network layer; however, IPv6 for enterprise networks is also growingly used. Non-IP network layer in small piconets and body area networks are also applicable, but they eventually require a gateway to IP network for connectivity to the enterprise networks. Different routing protocols for mobile ad hoc networks are preset.

*Routing Protocols in Mobile Ad hoc Networks*

As mobile ad hoc networks are having nodes communicating through ‘slow” wireless links and the devices are mobile in nature, the network topology may change rapidly and unpredictably over time. All network activity including discovering the topology and delivering messages must be executed by nodes themselves. Hence, routing functionality will have to be incorporated into the mobile nodes. Each node in a mobile ad hoc network acts both as a host and a router and the control of the network is distributed among the nodes. The network topology is in general dynamic, because the connectivity among nodes may vary with time due to node departures and new node arrivals.
Routing in mobile ad hoc network involves finding a path from a source to the destination and delivering packets to the destination, while the nodes in the network are moving. There are two broad approaches for routing viz., topology based and position based routing. The topology based protocols establish and maintain routes to the destination either pro-actively or reactivity.

**Proactive Protocols**: These protocols maintain routing information even before it is needed. These protocols are also called as table driven. Each and every node in the network maintains routing information to every other node in the network as routing tables. The table will be updated whenever it receives a periodic topology broadcasts from other nodes. When the packet arrives the node checks its routing table and forwards the packet accordingly. Every node monitors its neighbouring links and every change in its neighbours results in topology broadcast packet. That is flooded over the entire network. Other nodes update their routing tables accordingly upon receiving the update packet. Proactive protocols attempt to maintain up-to-date routing information between each couple of nodes by using a combination of the following techniques:

- **Periodic updates**: routing information are exchanged periodically by the nodes of the network
- **Event-trigger updates**: external events such as link addition or removal may cause the transmission of routing updates.

Optimized Link state Routing (OLSR) (Clausen et al (2008)) is a proactive, link-state routing protocol where each node maintains topology information by periodically exchanging link-state messages. OLSR employs multipoint relays (MPRs) to minimize the number of control messages flooding in the network. Each node chooses a subset of its one-hop neighbours in such a way that these MPRs will over all two-hop away neighbours. Hence messages are flooded only through MPRs.

**Reactive Protocols**: These protocols don’t maintain routing information or routing activity at the network nodes if there is no communication. If a node wants to send a packet to another node then this protocol searches for the route in an on-demand manner and establishes the connection in order to transmit and receive the packets. Using such an approach lowers the total cost of the initial delay finding the optimal route. The reactive protocols are also called as
on-demand or source initiated routing protocols for example AODV designed by Perkins et al (2003) (Ad-hoc On-demand Distance Vector Protocol). It aims to obtain routes on-demand. AODV uses three control messages to obtain and maintain routes: Route Request (RREQ), Route Reply (RREP) and Route Error (RERR).

In addition to this AODV performs route maintenance on active routes. Route maintenance is a mechanism by which each node is notified that a link along an active path has broken to the other node. If one node in an active path discovers a link breakage, a route error message will be transmitted upstream. Upon reception of the route failure the nodes will initiate a route discovery again to find a new route for remaining packets destined for that node.

**Geographic Position based Routing:** These protocols use minimum of topology information. But they use the location of the nodes; they are based on the structure of the network and on the geographic position of the network.

### 2.2 Overview of Service Oriented Architecture

The economy today is becoming more and more service based rather than manufacturing or even product based (Juneja (2007)). An evolution can be observed in the way functionalities have been specified, provided and consumed. Indeed level of abstraction has continued to rise. We have moved from modules, to objects to components and now to services. A service is a task or a function that a provider can provide to a consumer.

Service Orientation is a design paradigm that helps to build distributed computing systems around the service as a fundamental unit (Thomas Erl (2007)). The service-oriented paradigm gets its inspiration from object-oriented computing, but offers further enhancements and additions.

#### 2.2.1 From Object Oriented to Component Oriented Distributed Systems

A distributed system consists of multiple autonomous computers that are communicating through a computer network. Remote procedure Call was first introduced in procedural
languages (Judith Myerson (2002)). Since then, the main distributed systems are object-oriented and component-oriented distributed systems.

In distributed object systems, a software agent exposes the semantics of object initialization and method invocation to remote agents. Objects maintain complex internal states to support their methods. Proprietary or standardized mechanisms broker communications across system boundaries. Communication interactions between remote agents are fine grained and these agents must share the same implementation technology. One object sends a request to another object in a remote machine to perform a task, the result is sent back to the calling object. Examples of technologies that implement distributed objects are: Java RMI (ORACLE), DCOM (Microsoft corporation).

In distributed component systems, a software component is an encapsulated entity that communicates with other component via interfaces. A component offers its services through a provided interface; another component needing these services adopts a user interface. This is how components communicate without being concerned with the inner workings of one another. A computer running several components is called an application server, the combination of application servers and software components invoking them forms a distributed system. Examples of distributed software component technologies are: CORBA component model, Enterprise Java Beans, .NET Remoting and FractalRMI.

2.2.2 Service Oriented Paradigm

Service oriented computing is a programming model that consists of service consumers, providers and a service registry. Its primary advantage stems from the ‘decoupling’ of the consumers and providers, enabling them to be developed by different organizations and specifically from Service Oriented Architecture to automatically match and bind consumers to providers. SOC enables loose coupling between service consumers and providers through service descriptions that can be automatically compared and matched. This decoupling enables independently developed applications to work together. For example, it enables an internet-based application running on a web server to access data produces by another application.

SOA is an architectural paradigm and discipline that may be used to build infrastructures enabling those with needs and those with capabilities to interact via services. As with any other
architecture, this architecture can be expressed in a manner that is decoupled from implementation as stated by Duane Nickul (2007). The dynamic binding between service consumers and the providers is the key enabler that one of the concepts we exploited in this thesis. The dynamic binding between the service consumer and provider is one of the key enabler of our framework.

2.2.3 Phases of Service Provisioning in SOA

A service-oriented system follows the principles of the Service oriented design paradigm shown in Figure 2.2. A service-oriented system is built using an architecture called SOA (Service Oriented Architecture). Entities in service-oriented architecture are of three types: a service provided by a provider, a registry helping services get discovered and a consumer wanting to use the services.

![Figure 2.2 Interactions among the Entities of Service Oriented Architecture](image)

The basic actions taken by these entities and the interactions between them – in order to create a viable service-oriented system are:

**Description:** The functional capabilities and non-functional properties of the service are described in syntax understandable by all the entities of the system. It is also human readable to enable easy creation. The description has well defined semantics to enable proper discovery. The description is created by the service programmer.

**Advertisement:** The description is meant to be advertised by the service provider on a registry entity. The advertisement may add more information to the service description (e.g.
provider availability times, providers’ capabilities in terms of memory space, speed of movement). This information is supposed to assist the consumer on whether or not he would like to use the service. The implementation of a service registry varies depending upon the architecture of the system. Wired and stable networks use centralised registries where a consumer searches for registered providers. In distributed architectures, the registry can also be distributed using total or partial duplication over network nodes. In highly dynamic distributed environments, the registries can completely disappear in favour of a peer-to-peer type of discovery.

**Discovery:** Discovery is a core principle of the service-oriented design. The consumer prepares a description of his needs, put together in a format resembling that of the service description. With these needs, the consumer seeks matching services. Discovery enables the matching between the consumer’s needs and services of the registry having similar descriptions. The matching can result in many appropriate services; the consumer finally selects one service out of all these matching services.

**Invocation:** Invocation enables a consumer to use a service. Once the consumer discovers an appropriate service, invocation consists of transmitting request from the consumer to the service and then receiving responses back from the service. The communication link should be available during the interaction. Invocation requests and responses should be implementation-neutral in order to comply with the principles of the service-oriented paradigm. In other words, consumers and services built using different technologies should still be able to interact.

2.2.4 **Client Provider Interactions in Distributed Computing**

In this section we enumerate the main techniques that are used in distributed systems to enable interactions between clients and providers. The ability of the techniques to provide asynchronous interactions between communicating entities is the focus now. Asymmetry is good because it minimizes the overhead of the coordination. For example suppose device $A$ needs to communicate with device $B$ but not vice-versa. In this case, an asymmetric framework can facilitate communication from $A$ to $B$ without having to enable communication from $B$ to $A$. 
Message passing

Message passing is considered as the oldest form of interaction between distributed agents. An agent communicates by sending and receiving messages. It is a low-level interaction mode.

Message passing systems can be synchronous, where the sender and the receiver have to wait for each to transfer the message. In other words, the sender is blocked until the receiver has received the message and the receiver is also blocked until the reception is over. Synchronous communications has the advantage of simplifying communications between senders and receivers. Message passing system can also be asynchronous. To deliver a message from sender to receiver, the sender is not blocked, but still the receiver is locked since it is listening synchronously on the communication channel. This communication channel is setup before any message exchange, therefore the receiver must be known to the sender and both of them are active at the same time.

Remote Procedure Call

Remote procedure call RPC was first introduced in procedural languages. Since then, it has been implemented in object-oriented systems as remote method invocations (e.g. Java RMI, DCOM and CORBA (Object Management Group)). The invoking object holds references to the invoked object and remote interactions appear as local interactions. The client calls a client stub as a local method call, the stub puts the call parameters into a message, the message is sent over the network to the server, the server stub calls the server method and the server reply is sent back to the client using a similar behaviour. The SOAP (W3C Consortium (2007)) invocation specification is considered as message-passing, since it is a protocol for exchanging structured information in Web Services (W3C Consortium (2004)), but the message contents are usually RPC between the sender and the receiver.

Usually in RPC, the client sends a synchronous call (and is blocked until the reception of a response), the server can process the call asynchronously. To remove the synchronism from these remote invocations, some systems allow the caller to send one way calls and not expect a response, other system (e.g. Future interface of Java Concurrent ) allow the caller process to keep working and access the reply when it becomes available (using a call handle). In RPC, the two
interacting entities must be active at the same time and the caller holds remote references to the invoked entity.

*Event Notification*

Notification systems use events for client-provider interactions. A normal RPC invocation is broken into two one-way invocations; the client calls the provider and puts a call back reference to it, the provider responds by calling the client back using the reference. The observer pattern (Erich Gamma (1994)) uses event notifications, where an object notifies its observers by calling their notify methods. Event notifications remove the synchronism that blocks the invoking client in RPC; therefore the client provider interaction becomes completely asynchronous. But still, an event notification does not work unless both interacting entities are active at the same time and they always hold references to each other.

*Tuple spaces*

In distributed tuple space systems, the tuple space is either a shared memory or a data container, where a network node inserts tuples for other nodes to access and read concurrently. The Linda (David Gelernter (1985)) Coordination language introduced this notion of tuple space. A tuple space is a collection of ordered tuples that any network node can manipulate using three main operations: *out* puts a tuple in the tuple space, *read* reads a tuple from the tuple space and *in* reads and removes the tuples from the tuple space. Lime (Amy L. Murphy et al (2006)) is an implementation of Linda for mobile environments.

In tuple spaces, the interacting entities do not hold references to each other in order to communicate. The node that inserts a tuple in the tuple space does not know who is accessing this tuple. A node reading a tuple does not know who put it in the space. Furthermore, the read operation lets many nodes read a tuple, which enables a one-to-many communication mode. Addition, the interacting entities do not have to be active at the same time for communication to succeed. But the synchronization problem remains, since a node is blocked when reading a tuple from the tuple space. Therefore, some implementations of tuple spaces like JavaSpaces (Sun Microsystems (1999)) provide asynchronous notifications of new tuples.
Another close example to shared data is flow-based programming (Paul Morrison (1994)) where applications are considered as “black boxes” that exchange messages through predefined connections. This flow-based programming is mainly used in component-oriented systems. Like in tuple spaces, interacting entities do not hold references to each other.

**Message queues**

Message-oriented middleware (MOM) (Barb Gomolski (1997)) emerged from the need to glue together remote applications without re-engineering individual modules. A MOM depends on message queuing (e.g. Apache Qpid, JMS) for interactions between clients and providers. A queue is a storage space where messages can be placed by producers and then concurrently pulled by consumers. The queue stores messages in a first in first out manner and provides transactional and ordering guarantees. To some extent, message queuing systems relate to the publish/subscribe interaction model.

Message Producers and consumers do not share references to each other and they do not have to be active at the same time in order to communicate. A producer is not blocked when storing messages, but a consumer is blocked when pulling these messages.

**Publish/Subscribe**

A Publish/subscribe system (Patrick Eugster (2003)) lets producers send messages to a message broker via a publish method, the broker also has a subscribe method to let consumers subscribe to message of interest. The broker filters published messages according to the interest of each of the subscribers and notifies these subscribers of matching messages. There are two main techniques of filtering: topic-based and content-based. In topic based pub/sub systems, a topic is a keyword representing a class of messages; each published message belongs to a topic. Subscribers subscribe for the kind of topics they are interested in. When a producer publishes a message, the broker notifies the subscribers of the message’s topic. The broker is often centralized, it may also be distributed and it may sometimes disappear in favour of a peer-to-peer interaction model. In content-based pub/sub systems, the subscription is based on the actual contents of a published message. The broker filters messages according to their internal attributes or their meta-data (e.g. Siena (Antonio Carzaniga et al (2000)), Jedi (Gianpaolo Cugola et al (2001)), JMS).
With pub/sub systems, the interacting entities do not need to know or hold references to each other. A publisher does not know who and how many are receiving its message and the subscriber does not have to know who published the message. Furthermore, a communication succeeds even if the entities are not active at the same time. In addition, the publisher is not blocked when publishing a message and a subscriber is not blocked when subscribing and gets notified at message reception, therefore the communication is completely asynchronous.

**Web Services**

The service-oriented paradigm is an abstract and implementation-neutral paradigm. Still, its association with web services has become very common. A reason for this association is that the majority of SOA vendors have modelled their service-oriented platform as web services.

Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols.

The basic architecture of web service technology is capable of:

- Messaging feature
- Describe web services
- Publishing and discovering web service descriptions

The basic Web service architecture models the interactions between three roles: the service provider, service discovery agency and service requestor. The interactions involve publish, find and bind operations. These roles and operations act upon the web service software module and its description. A service provider hosts a network accessible software module (an implementation of a web service). The service provider defines a service description for the web service and publishes it to a requestor or service discovery agency. The service requestor uses a find operation to retrieve the service description locally or from the discovery agency (i.e. a registry or repository) and uses the service description to bind with the service provider and
invoke or interact with the web service implementation. Service provider and service requestor roles are logical constructs and a service may exhibit characteristics of both.

Requesters and providers interact using one or more messages. A service description is hosted by a discovery service, to which a provider publishes the description and from which the requester discovers the description.

A message is defined as a construct that can include zero or more headers in addition to data. The header part of a message can include information pertinent to extended Web services functionality, such as security, transaction context, orchestration information, or message routing information. The data part of a message contains the message content or data.

A web service is described using a standard, formal XML notion, called its service description, that provides all of the details necessary to interact with the service, including message formats, transport protocols and location, which are used to bind the service. Usually by the format called Web Service Description Language (WSDL). It defines the name and description of the service and for all operations that can be invoked their names, input and outputs. The nature of the interface hides the implementation details of the service so that it can be used independently of the hardware or software platform on which it is implemented and independently of the programming language in which it is written. WSDL interface descriptions can be translated into interfaces of many languages, such as C# and Java.

In order for an application to take advantage of Web services, three behaviours must take place: publication of service descriptions, finding and retrieval of service descriptions and binding or invoking of services based on the service description.

Before invocations, the request and the provider entities become known to each other and they agree on the service description and semantics that determine the integrations between the requester and the provider. This is the discovery phase that can be implemented using three approaches: (1) using a registry service that is centrally controlled by an authority where providers actively register their services. (2) Using the index approach where an index is automatically maintained by Web Crawler for example (3) using peer-to-peer discovery where a requester directly queries the providers to search for services.
Web Service Description Language WSDL (W3C CONSORTIUM (2007)) was originally developed by IBM, Microsoft, and Ariba. The current version WSDL 2.0 is a W3C recommendation. A WSDL document describes a Web service using XML format. It contains an abstract section and a binding section. The abstract section describes the interface composed of operations, each operation defines its input and output messages. The abstract section also contains the data types of these inputs and outputs written in XML schema. The binding section binds the operations and messages to a concrete network invocation protocol (e.g., SOAP binding, HTTP binding, etc.). In order to enhance the discovery, the WSDL document can include semantic annotations in SAWSDL (W3C CONSORTIUM (2007)). Semantics help categorizing the interface when publishing to a registry and therefore it helps requesters with discovery. Semantic annotations can also map data types from ontology during invocation.

Simple Object Access Protocol SOAP (W3C CONSORTIUM (2009)) was originally developed by Microsoft; the current version SOAP 1.2 is a W3C recommendation. It is a protocol specification for exchanging invocation messages between requesters and providers. The message format is XML. SOAP relies on RPC for the invocation model. Messages are usually transmitted over HTTP, but there are other transmission modes like JMS and SMTP. A SOAP message is an XML envelope containing an optical header part and a body part. SOAP helps create stateless or stateful Web Services, where a service may expose RESTful Web services. REST enables service invocation but only using a uniform set of stateless operations over HTTP (create, retrieve, update, delete).

Universal Description Discovery and Integration UDDI OASIS (2004)) was originally developed by IBM, Ariba, and Microsoft. The current version 3 is sponsored by the OASIS organization. UDDI specifies the creation of a central registry called UBR (UDDI Business Registry) and specifies a framework to describe service metadata and another framework to discover them. An XML document describes the capabilities and properties of a service. Description can include information about the business providing the service like its name and contact information of administrators, it also includes identifiers for the service and descriptions. Registration at the UBR is made using SOAP over HTTP and service requesters query the UBR for services. UDDI is mostly implemented for the enterprise world.
In comparison to traditional distributed systems, Web Services are appropriate for applications operating over the Internet where reliability cannot be guaranteed. In addition, components of the system run on different technologies, platforms and vendor products. Web services help wrap and expose existing applications over the network. Web services do not require to synchronously managing deployment at all consumers and providers. And finally, web services can be composed and orchestrated.

The quest for enabling those open XML web services interfaces and standardized protocols also on the radio link with the latest developments in cellular domain, lead to a new domain of applications called mobile web services.

2.3 Overview of Cross Layering

Cross layer design is an emerging proposal to support flexible layer approaches in mobile ad hoc networks (Srivastava and Motani (2005)) & (Bisnik (2005)). The cross layer design introduces the advantages of explicit layer dependencies in the protocol stack, to cope with wireless links and mobile terminals, high error rates and Quality of service. Many cross layer design solutions have proposed in the literature (Bahador Amiri et al (2011)) and (Saymon Jakubczak and Dina Katabi (2011)).

Protocols can be designed by violating the reference architecture, by allowing interactions and state information flowing among non-adjacent levels of protocol stack. Generally speaking, cross layer design refers to protocol design done by allowing layers to exchange state information in order to obtain performance gains (Mangesh Nikose and Anni Gupta (2011)). The cross layer architecture is shown in Figure 2.3.

There are three main motivations supporting the adaptation of cross layer design in protocol design for mobile ad hoc networks: the need by protocols to be adaptive to network dynamics, to support the requirements specified by the applications and to tackle the energy and
security concerns. It is observed that several design challenges in mobile ad hoc networks (security, energy issue, topology control) cut the layers and requires joint solutions involving multiple layers. Cross layer interactions are shown in Figure 2.3. Through cross layering information can flow from any to any other (follow the dotted line). It need not follow the strict layer interactions of the OSI model (as that of double headed arrow between layers). Information can also be passed to any layer from the layer (cross layer adaptivity layer) which spans across all the layers.

2.3 Cross Layer Interactions
Adaptivity and Self-Organization  Network protocols for Mobile ad hoc networks must be adaptive to many factors to effectively support fair sharing of devices and resources and to hide the system dynamics to the upper layers. The system dynamics include a wide range of communication conditions a wireless node can experience inside a mobile ad hoc network, including changing topology, shared medium contention, varying traffic patterns and distributions. The adaptive behavior can be implemented if the following requirements are met:

- **Context awareness**, i.e. the knowledge of the parameters affecting the network state (channel condition, congestion, traffic demands, etc);
- **Protocol tuning**, i.e. the possibility for each protocol to adjust its behavior according to the current network state.

Context awareness sometimes requires re-designing the way protocols are organized and interaction among each other. Cross-layer architectures have been proposed to guarantee protocols cooperation with sharing of network-status information while still maintaining separation among the layers (Conti et Al (2004))

QoS and Applications Requirements: QoS is a guarantee by the network to provide certain performance for a flow in terms of bandwidth, delay, jitter, packet loss probability, etc. At the MAC layer, QoS is related to the fraction of time a node is able to successfully access and transmit a packet. Actually, the 802.11e protocol extension provides mechanisms to support different priorities in WLAN networks: the 802.11e EDCF protocol supports 8 different service priorities, mapped on 4 different access categories. Each category defines a set of parameters governing the access to the shared medium.

In multi-hop environments, QoS must be addressed by considering the QoS requirements on the end-to-end path as well as on each hop. Wireless channel fluctuations, self-contention, limited bandwidth and dynamic topology make the QoS a strong issue for mobile ad hoc networks. What appears clear is that the QoS requirements cannot be met in mobile ad hoc networks unless they are supported across all the layers of the network. For these reasons, many
recent works investigate the joint optimization of physical layer power allocation, MAC layer link scheduling and network layer flow assignment (Lijun Chen(2004)).

Energy Conservation: Energy efficiency is a limiting factor in the successful deployment of mobile ad hoc networks, because nodes are expected to rely on portable, limited power sources. Moreover, energy conservation is extremely challenging in multi-hop environments, where the wireless nodes should also consume energy to route packets for other nodes and to guarantee the connectivity of the network. At the MAC layer, some techniques can be used to reduce the energy consumed during transmission and reception; additionally, a careful policy may turn off the wireless device when the node is idle. At the network layer, the route selection process should be performed by reducing the end-to-end power needed to forward the packet; if the network layer may have access to energy information battery-level metrics can be used in the routing process.

Security: Because nodes in mobile ad hoc networks communicate each other via open and shared broadcast channel, they are more vulnerable to security attacks. Moreover, the support for multi-hop communication implies that the network has to rely on individual solutions from each mobile node, resulting vulnerable to infiltration, eavesdropping, interference, denial of service attacks. Many research efforts have mostly concentrated on secure data forwarding: secure routing protocols face the attacks that intentionally disrupt the routing protocol execution and guarantee the acquisition of correct topological information. On the other hand, data-link security solutions are implemented as parts of wireless standards (WEP/WPA for 802.11) to provide authentication and privacy issue on infrastructure single-hop wireless networks. However, the solutions proposed at MAC, routing and transport layer only cover a subset of all possible threats. A cross layer design of MAC, routing and transport protocols allows to take into account the security issues in all the stages of protocol design.

Though, a cross layer approach to network design can significantly increase the design complexity (Goldsmith and Wicker(2002)), better performance gain can be achieved if we slightly adapt to the current working environment as the need demands.

Relationship between cross layer information & context information
Network status is one of the context in which the mobile node works. This information can be had from the cross layering architecture to the layer which needs it. Thus by the use of cross layering we can have the dynamic context information of network or in other words provides context awareness to the network. Most of the mobile ad hoc network applications mandate them to be of context-aware in order to perform well and correctly. One among the dynamic context is network context which means the network topology, link status of the nodes, neighbour nodes in the network etc. This information may be accessed or derived from the lower layers parameters. Thus context information is derived by the use of cross layering.

2.4 Overview of Context-Awareness

Context information has to be represented using uniform structure and vocabularies. A challenging and critical task that researchers tried to address over the past few years is the development of monitoring and management schemes that suit the characteristics of these networks. Many applications of mobile ad hoc networks have strict quality of service requirements for their communications systems, making mobile ad hoc network quality of service provisioning mechanism very crucial (Marwaha et al (2011)). Quality of service provisioning and awareness in mobile ad hoc networks is quite tough due to the dynamic network topology, unpredictable communication medium, constrained bandwidth, risk of failure of routes (Renesse et al (2008)).

Venkatasubramanian and Gopalan (2010) have proposed a quality of service architecture for bandwidth management and rate control in mobile ad hoc networks. In this architecture, each node will continuously estimate its available bandwidth. The bandwidth value will then be used by the source nodes for quality of service capable routing protocols. Stephen and Yau Dazhi Huang (2011) presented a design of distributed quality of service monitoring and adaptive modules for multiple workflows in adaptive service based systems.
Monitoring is intricate and the problem is complicated when it is to be done for mobile ad hoc networks (Demetrius (2011)), because of the special characteristics of the mobile ad hoc network networking platform and the continuous monitoring of the metrics. One of the challenging task in making context awareness is the designing of monitoring infrastructure for mobile ad hoc networks because constructing abstractions, which are able to capture the properties of ad hoc network Tuduce and Gross (2004)).

In order to achieve context-awareness, there must be some mechanism to manage the context information. Context-aware computing is concerned with enriching computing environments with concepts that can improve interaction and ultimately increase productivity and reduce the burden of users. The key concept in this area is context modelling and context-awareness. Context is a powerful and longstanding concept in human-computer interaction. As human beings, we can more efficiently interact with each other by fully understanding the context in which the interactions take place. It is difficult to enable a machine to understand and use the context of human beings.

Nowadays, the notion of context is much more widely appreciated. The term context-aware is generally defined by those working in ubiquitous/pervasive computing, where it is a key to the effort of dispersing and enmeshing computation into our lives. Consequently, context also refers to the physical and social situation in which computational devices are embedded. One major task in context-aware computing is to acquire and utilise information about the context of a user in order to provide the most adequate services. The service should be appropriate for a particular person, place, time, event etc. where it is required. A user may be a person who uses the service or who is the target object of a service. What makes the context information more useful is to relate it to the tasks and activities, to organize it well and to increase its utility.

This thesis uses context information as one of the important factor for service selection. In order to achieve this we create a context model which is efficiently used for service selection.

Context modelling is demanded for the wide range of heterogeneous context information in context-aware computing. It helps application designers and developers to uncover the possible context and simplify the context manipulation, the conceptual viewpoints of context
models summarised by Razzaque (2005)) include: who, where, what occurs, when, what can be used and what can be obtained. Most of the researchers follow this categorization.

The ultimate purpose of context modelling is not only to categorise the context information but also to introduce a way to use context information efficiently. In our thesis the context model is implemented in XML.

2.5 Service Selection and Selection Requirements

Service selection is a process of allowing a prospective user to choose services which best suit his/her functional and non-functional requirements. With the rapidly growing number of available services users are presented with a choice of functionally similar services. Therefore this choice strongly depends on the non functional properties that provide differences among competitive services. Service selection can be performed by two types of methodologies:

*Static service selection*: the service discovery and selection are manually conducted through the service discovery protocols and human readable representation. Once selected it will not change in future, which is not suitable for Mobile ad hoc networks.

*Automatic service selection*: The services are dynamically discovered and selected based on machine understandable representations and run-time collected service requirement constraints. The selected service is only used one or few times. When new requirements come next time, a different service may be used.

Apart from context-awareness, the following are the major requirements for service selection methods based on non functional properties are proposed by Yu and Reiff-marganiec (2008).

*Model for non functional properties*: service requesters or passive users need to objectively distinguish services based on their non-functional properties to make the most appropriate choice amongst a number of services. In the light of that a model for non-functional
properties is required, which can be used in service descriptions as well as service request. Due to the versatility of non-functional properties it is unlikely that a complete set can be identified. Non-functional properties should be considered differently depending on the specific service domain.

Properties preferences: service requesters usually have various preferences for the non-functional properties depending upon the situation they find themselves in and of course different situation will mostly have different preferences. A good mechanism should not only express values for each property, but preferable also represent the elations among the preferences.

Evaluation of properties: As the type of the properties can be different, it is impossible to define a universal evaluation function for all kinds of the properties. Hence evaluation framework must not only adapt to various numbers of non-functional properties, but also automatically identify the measurement methods that should be used to evaluate each non-functional property.

Dynamic aggregation: when all desired non-functional properties can be evaluated, the next step is to aggregate individual scores to gain a final score for the service. In this step a suitable aggregation method needs to be selected. Intuitively, arithmetic or geometric means based on weighted sums or product might appear to be efficient and understandable choice.

Automatic service selection can be performed by a human to look up suitable services in a registry and make decisions as to which one to choose. However, the ultimate goal of service selection research and especially service selection based on non-functional properties is to provide fully automatic processes. A service designer would still specify data for the service when making it available and a user would still be able to specify requirements, but selection would be performed without human intervention. Such automatic selection methods are essential when for example, considering context-aware service selection. For context awareness automatically using context information to affect the service selection is also one aspect of automation.

2.6 Overview of Adaptation
The runtime adaptation support is in charge of dynamically managing and modifying context information. Paolo Bellavista (2013) stressed that a crucial adaptation support aspect (i.e.) service level can affect the adaptation process by influencing the decision of the run-time adaptation support with different level of control. (i) Unaware (ii) partially-aware and (iii) totally aware. Figure 2.4 shows runtime adaptation support its taxonomy and interaction with other layers.

(i) Un-aware adaptation: the service level neither reaches nor influences run-time adaptation supports strategies

(ii) Partially-aware adaptation, there is more collaboration between the two: the service level supplies profiles that describe the required kind of service while the run-time adaptation support modifies context data distribution facilities to meet those requests.

(iii) Totally-aware: the runtime support does not perform anything on its own and it is the service level that completely drives reconfigurations

2.4 Runtime Adaptation Support: Taxonomy and Interaction with Other Layers

Adaptation to changes might happen in Middleware (Jacqueline Floch (2006)) or in the applications such as Haggle (Jing Su et al (2007)). This thesis use adaptation done at middleware
and just a notification will be given to the application layer when there is a failure in the adaptation.

Adaptation software (middleware) is defined as software that can change its behaviour at runtime, has a number of potential benefits ranging from the ability to respond rapidly to security threats, to the opportunity to optimize performance as characteristics of the underlying execution environment change. The value is especially pronounced in distributed systems, which are arguably more prone to the type of situations that would benefit from an adaptive response.

The issues to be addressed for constructing adaptive software are:

- Change detection: involves detecting changes and determining if they are significant enough to warrant adaptation.
- Adaptation policy: involves deciding which component should make adaptation once a change has been detected.
- Coordination. Involves doing an adaptation in a coordinated manner so that both the component and system functionalities are preserved while the adaptation is in progress.

The framework has to execute a series of actions, in order to ensure service guarantees, despite the occurrence of certain events termed as adaptation trigger events. Some examples of events belonging to this category are mobility of clients, advertisements of events with special requirements like delivery within a short time of publication. In this thesis concept of adaptation triggers are used. Based on these adaptation trigger events can be classified as

- Adapting to client related adaptation trigger events (client mobility)
- Adapting to event model based trigger events (notification of an events)
- Adapting resource based trigger events (channel failure, resource availability variation)

Policy-based approach may greatly benefit from existing research on adaptability in open component based middleware (Coulson et al (2006)) and autonomous behaviour. This feature has been adapted in this thesis.
Summary

Active research has been into the area of mobile ad hoc networks for a quite a long period as this domain has many important applications. Many of the closely related networks of mobile ad hoc networks have been studied in this chapter. The similarities and differences between the different networks were analysed by studying the topology, support for multicast and broadcast facilities, as well as network architecture, mobility characteristics, traffic models and applications.

One of the emerging paradigms in distributed computing is service oriented architecture. SOA provides facilities for organizing and utilising distributed components to offer services discover services and interact with end-users. Mobile ad hoc network may benefit out of SOA to implement service provisioning because SOA has been efficiently used in the wired counterpart. The modifications to be done for this usage have to be studied and the following chapter focus on this too.

Developments in context awareness influenced design of many user centric protocols. Context aware computing enriches computing environments with concepts that can improve interaction and ultimately increase productivity and reduce burden on users. Context aware computing has been used in this proposed framework to provide relevant information about the services and user.

Even at this stage it is not yet clear how the characteristics of each area influence the usability and applicability of proposed service discovery and provisioning protocol. However, the investigation of the specifications aids in selecting a proper architecture and methodology for a good service discovery and provision solution. The proposal should rely on investigating state-of-the-art and existing discovery and provision solutions. This proposed framework attempts to utilise both SOA and context awareness.
In the following chapter, the requirements to be satisfied for any framework which has been designed for service provisioning were studied. The existing work in the service provisioning area has been analysed to formulate the issues to be addressed.