CHAPTER 1

INTRODUCTION

Middleware resides in a layer between the hardware and operating system. The term context refers to the change in behavior of the system through aspects of environment, activity and location of the user. The profile of the user is sensed by the sensor node and depending on the profile, services are offered to the user. The middleware layer is designed between network and transport layer in the protocol stack. Interoperability among the service discovery protocol is attained at the middleware. In all the literature survey reviewed, interfaces or plug-in's are used to build interoperability among different heterogeneous platforms. The newly designed middleware differs from already existing work as seamless interoperability is attained without interfaces or plug-in's.

1.1 Benefits of Middleware

Middleware provides functionality of transparency, scalability and interoperability across dissimilar networks. Middleware is created to reduce complexity of the networks. Middleware provides cross platform infrastructure to increase the efficiency of search. Middleware provides a client server relationship, where there is a consumer-producer relationship or peer to peer relationship to exchange information. In the proposed work, middleware is designed to provide peer to peer relationship. Middleware must be available in many platforms and should provide interfaces allowing the same code to be used in multiple environments. Middleware solution will have

low complexity and transparent interface to provide flexibility, maintainability, reusability and portability. Mobile middleware is light weighted in order to manage resources, and supports context awareness. Proxy middleware send request to proxy to be converted in to soap request to improve performance. Web services are used to provide service irrespective of location no matter where service requestor or provider is located [1].

1.2 Existing Standards

Middleware Standards and Implementation

COM/DCOM: COM provides interface definition through IDL (Interface Definition Language). Protocols are used to communicate between objects. DCOM extends to communicate across networks. The main disadvantage of DCOM and COM architecture is that the core communication are synchronous which are subjected to failures very often and all these fixed middleware systems are heavy weight implementation systems. So these systems could not be used in wireless environments [2].

CORBA: Provides naming, trading and event service through ORB protocol and are used to facilitate communication. ORB establishes communication in a distributed environment. CORBA provides an object model to communicate. Interfaces are defined using interface definition language. CORBA demands continuous connection which cannot be expected in a wireless environment. Moreover CORBA cannot fit in limited resource environment [2].

RAPP: The reactive adaptive proxy placement architecture provides proxy to establish client-server communication. This method uses proxy for communication on behalf of mobile host. RAPP provides solution for variable bandwidth. Since RAPP uses proxy it incurs high overhead [2].

Java RMI Solutions

Simple Object Access Protocol

SOAP is a light weight protocol for distributed environment. It is XML based protocol. It consists of three parts: the envelope, a set of encoding rules and conventions. SOAP traverses through firewalls and provides a solution for serializing. SOAP works on text based protocol which consumes bandwidth. But the solutions of SOAP does not address terminal mobility and variable bandwidth [3] [4].

Analysis of the Existing Middleware Framework

All the middleware discussed above are for fixed applications. None of the solutions are favorable for wireless networks and does not address the problem of battery consumption. Most of the middleware requires proxy or gateway for the mobile devices to rest on. In order to connect the fixed network to wireless environment, the interfacing of proxies or gateways are used.

1.2.1 Asynchronous Middleware

Tuple Spaces

Tuples in tuple spaces are typed data structures and are like objects in object oriented programming. LINDA consists of a single tuple space. In an environment where communication links are highly infeasible, multiple tuple spaces are created and communication is established between tuple spaces by bridging and thereby forms L^2limbo architecture. The replication of tuple spaces makes disconnection accessible. It manages the wireless network problem of low bandwidth and addresses migration and minimizes power usage to a minimum [2].

LIME

LIME is the extension of LINDA in mobile environment. Global context is by means of interface. The tuples are merged in its architecture upon arrival at the new mobile space. LIME allows the ability to work in context. The functionality of the LIME is similar to LINDA in terms of wireless environment [2].

TOTA

TOTA is a context aware middleware that provides distributed feature in adhoc spaces. TOTA consist of peer to peer networks and each of the networks runs middleware and provides reference to neighboring node.

Tuple spaces provide good abstractions but could not be applied to real world problems. In mobile computing, java spaces are not supported and only T-spaces are provided because it has smaller footprint. Generally the tuple space operation does not support context awareness which is a very important feature of mobile computing. There is no provision of reconciliation mechanism in tuple spaces. Reconciliation is an important property of the mobile middleware system, as the data should be available even when the mobile leaves the system [2].

1.2.2 Publish-Subscribe Middleware

Asynchronous publish-subscribe is important for event notification. A component subscribes the event and consumes it when published. Publish-subscribe allow the message to be exchanged based on content. The decoupled nature is well suited for mobility [2].

SIENA

SIENA is a publish subscribe interface. SIENA server acts as an access point in providing service. Publisher performs publishing through the access point and subscriber uses the access point to subscribe the event of interest. SIENA does not help to work in wireless environment. It is designed only for fixed networks. The disconnection will lead to the lost of events [2].

Scalable Events and Real time Mobility (STEAM)

STEAM uses a centralized architecture for small area networks. STEAM is an event based model with no separate middleware to offer services. Group communication is allowed in STEAM. It could be a member publishing to several subscribers [2].

REBECA (Event Based Electronic Commerce Architecture)

REBECA is a content based publish-subscribe architecture. This architecture is unsuited for wireless networks.The broker architecture is used to transfer the subscriber's call from one location to another when the subscriber moves across the broker [2].

ELVIN

ELVIN is the publish-subscribe architecture for mobile environment. The ELVIN can have multiple subscribers communicating with publishers. In this architecture, the subscriber can also act as producer. It is similar to peer to peer networks. ELVIN architecture makes use of proxy so that the client can remain always connected to the server and vice-versa through proxy [2].

Analysis of Asynchronous Middleware

Asynchronous middleware are suited for wireless environment. The problem of week connection is solved in asynchronous middleware by enabling proxy in most networks. Tuple spaces are suited for mobile environments and lime is a tuple space middleware that supports

communication across heterogeneous platform. Event based publish-subscribe provides a good asynchronous middleware model. This solution does not address power consumption to a larger extent and also the problem of variable bandwidth in a wireless environment is not considered. Almost all middleware models address the problem of migration, low bandwidth and disconnection in mobile environment. None of these middleware discussed above has been designed for adhoc environment without fixed infrastructure [2].

1.2.3 Data Sharing Middleware

Data sharing middleware is very essential in mobile environment where the disconnection is high and network bandwidth is minimum. Data sharing middleware designed for fixed networks employs a distinction between server and client where server holds large copies of data files and client holds personal cache [2].

BAYOU

Bayou is a middleware that provides communication. It enables the communication to be established even when client and server are not in the same range. Bayou uses reconciliation technique to ensure that all copies of database resides in the server i.e. the server receives the replica of all databases and client shares the data of the server. It solves the problem of wireless environment i.e. the low bandwidth, disconnection and address migration. However a single client out of the range of the server cannot access the devices of the server. In this architecture the server stores all the data, and client accesses the data of the server [2].

AdHocFS

AdHocFS is a middleware platform for adhoc wireless networks that permits data sharing. The group members share the same data. The group members can leave the group at any time. To maintain data consistency, a unique token is associated with the shared data and the members modify the data using this token. This gets automatically modified in the shared data item and the member gets the modified data when he tries to access next. In the AdHocFS architecture, the data is stored in both the server and client cache and so adaptation policy is used to share the data when server or client moves out of range of each other. A profile is maintained in each host which tells the data that is replicated, the amount of storage and estimated time. The replication of data allows the data to be available always with any of the node and it provides a good solution to wireless environments [5].

XMIDDLE

XMIDDLE is a data sharing middleware that provides both reconciliation and replication over adhoc networks. Each devices store data in the form of a tree. Access point is located in each device that allows the data to be read and modified by peer. The host then checks the data modified and later on it gets reconciled. Using XML, the data and the meaning are stored. It also shares the context items. Sharing of XML trees is very important as it requires extra overhead [2].

Analysis of Data Sharing System

A data sharing system addresses weak connection, mobility and address migration of nodes. Changing data replication solves the problem of poor bandwidth and low memory resource. XMIDDLE is a data sharing middleware designed to support mobile applications that uses replication of data. Replication here refers to duplication of data. The files are shared through the tags associated with the XML data which incurs extra overhead when compared to standards file sharing systems. When replication is involved transmitting large XML files may be expensive over low bandwidth wireless networks and utilizes more resource [2]. This suits well only for collaborative and information sharing system. Therefore this system could only be used for specialized applications.

1.2.4 Adaptive Middleware

Middleware for fixed networks does not support the property of mobile environment and reconfiguration. Mobile environment permits context changes and therefore solutions have to be derived to change the middleware as the context change. Features could be added to the middleware at any time and the middleware gets adapted to those features spontaneously [6].

1.2.5 Reflective Middleware

The reflective middleware provides representation for introspection and adaptation. The key feature of the reflective middleware is meta interfacing and meta-object protocol. The meta interfacing is to study the internal features of the middleware and meta protocol is used to perform operations on the middleware.

OPENORB

OPENORB is the component based middleware and promotes technique of reconfigurability, configurability and reuse of middleware. OPENORB is structured as a set of configurable and reconfigurable component middleware and supports reflection through current structure and behavior. The end system is flexible middleware System [7].

DYNAMICTAO

DYNAMICTAO is reflective CORBAORB, where TAO is portable, flexible and extensible ORB that conforms to different aspect of ORB engine. The reflective mechanism supports inspection and configuration of the ORB. TAO based models are used in static real time applications [2].

FLEXINET

FLEXINET is a component based model that provides reflection within the protocol stack. It is based on binding between components which is established through policies. FLEXINET is a layered protocol stack based on java core language. Complex layer of the FLEXINET is used for adding and removing the sub components and also allows adaptation in changing environment [2].

K-ORB

It is based on CORBA architecture. The K-ORB framework is an extension of mobile IIOP engine developed in ALICE project. The K-ORB framework allows set of resources available at ORB to change at run time. K-ORB uses dynamic reconfiguration to tackle many of the challenges of mobile computing like disconnection, address migration, low bandwidth and small memory [8] [9].

OPEN-COM

The fundamental concept of OPENCOM is interface, receptacle and connections. An interface represents a unit of service provision. Receptacle describes a unit of service requirement. OPENCOM maintains a standard runtime for creating and deleting components. Pre-method and post-method support could be added by dynamically attaching a component interface [10].

1.2.6 Policy based Middleware

These middleware's are completely context based and gets adapted to the application based context aware systems. These middleware generally allows the application to state the rules so that the rules could be interpreted by the middleware [2].

ODESSEY

ODESSEY is a file sharing systems and it supports mobile clients. It is completely resource usage middleware. When the

resource availability gets reduced below the handler value, the handler gives the sign and the application need to get accumulated to the change [2].

PUPPETTER

It is an adaptive middleware to manage the energy consumption of wireless devices. It deals with documents and media format. By utilizing the exported API's of each application, it modifies the behavior without access to source code. It is proxy based architecture and deals with local and remote proxies. The architecture has to bear with the overhead of proxies [2].

LANCASTER Context Architecture

This deals with multiple contexts. So application running on one context will affect the other. LANCASTER university has considered a large space with adaptive application and context attribute. It has the property to be integrated to other middleware platforms like event based, object based and tuple spaced. Lancaster architecture could not be used in mobile application due to heavy overhead incurred by the architecture [2].

CHARISMA

It is a reflective policy based middleware. The Work concentrates on issues of context awareness, power, memory capacity and external context of network connection, bandwidth, and location awareness. Behavior of the application contains the service, the policy

to be applied to service and the context configuration to hold the policy. CHARISMA provides the reflective architecture to change the policy when needed. CHARISMA also manages the end-system resource of the mobile devices. Different policies have different functional requirements and different names of resources. Reflection also deals with non-functional properties of the resources. CHARISMA is a part of reflection and policy based middleware [2].

Analysis of Adaptive Middleware

The fixed middleware does not solves the problem of wireless networks like weak connection, address migration and poor bandwidth and cannot be adapted to wireless environment. Hence to suit this middleware in the wireless environment, adaptive enabled reflective middleware has to be used. The adaptive middleware takes care of variable resources and variable network problems. But except to K-ORB and UIC, none of the adaptive middleware could be used in reflective environment. Policy driven methods are best methods to support adaptiveness and reflection [2].

The above discussed middleware could not provide a standard solution and thereby could not support interoperability and work on heterogeneous platforms. For e.g. SOAP and CORBA are used for remote method invocation, SLP, UPnP are service discovery protocol and SIENA and CEA are for publish-subscribe. All these cannot interoperate. Even interoperability across different service discovery protocol in heterogeneous environment is not possible. GAIA and CENTARUS are the only middleware that supports pervasive environments. Hence a need to design a middleware to support

heterogeneity issues across platforms and to provide interoperability becomes essential.

Tacking Middleware Heterogeneity

The need to solve heterogeneity issues had its origin as different middleware does not interact and other service discovery and routing protocols used within the middleware also does not interoperate. This problem across fixed network is solved with bridges. Universal Interoperable Core reflective middleware is available to solve the problem in mobile domain.

1.2.7 Web Service Architecture

Web service is an open standard that supports interoperability. Web service uses the WSDL language for interfacing. A particular service should be implemented in SOAP based agent at a particular instant of time and on the next instant it may take up RMI or CORBA. The services, or client application using the service continues interacting transparently. A centralized architecture is designed in WSDL for service discovery. So it becomes a part of fixed network interoperable platform [11] [12][2].

Model Driven Architecture

Model driven architecture also supports interoperability in fixed domain. It does not support interoperability in wireless domain as it cannot foresee changes in heterogeneous environment during the life cycle.

Middleware Bridges

A Software bridge is used for communication among different middleware environments. The bridge takes up the message, marshals it and transfers it to the server. Many bridging solutions are available between DCOM/CORBA and CORBA/SOAP. OrbixCoMet are implementations of DCOM-CORBA bridge.SOAP2CORBA is an open source of a fully functional bidirectional SOAP to CORBA bridge [2].

Unified Component Meta Model Framework

The unified component metamodel is similar to model driven approach but leads to generating of bridges for interoperation. Interoperation of components is an expensive operation that must be executed for changes in heterogeneous context environment. Dynamic nature of mobile environments generates bridges very frequently. It makes use of component framework to support interoperability [13].

Analysis of Middleware Bridges

There are two types of bridges - the static bridge and dynamic bridge. The static bridge is used for fixed network and does not support mobile networks. The dynamic bridges could be applied to mobile environments. The insertion of bridges is the concept used for fixed environments. But the insertion is very expensive and requires heavy overhead in mobile environments.

1.2.8 Logical Mobility

Two platforms are used to support interoperability in the mobile code environment. Service discovery and service interactions are combined in this interoperability, as the client receives both the code and service for interaction.

SATIN

SATIN is a low foot print middleware that provides component based interoperability. Services are discovered by obtaining the code. At the heart of SATIN is the ability to discover the code and it uses a higher level language for service discovery. Codes are downloaded in the system dynamically and services are retrieved. The capability of interoperability is also downloaded when needed.

JINI

Jini makes use of proxy for interaction and though mostly used in the RMI environment, it could be used in any service discovery environment. The solution of heterogeneity is to wrap codes and make use of it when needed. Proxies are used for service discovery. In some case complete user interface will be available and in some cases it must be invoked remotely [2].

Analysis of Logical Mobility

Logical mobility is the best method of interoperation as the application need not know anything about the service they are to

interoperate but they make use of code during run time. Both Jini and SATIN insist that the user should know the proxy of Jini or the abstract service discovery of SATIN. Thus both techniques do not address heterogeneity to a larger extent [2].

1.2.9 Universal Interoperable Core

The Universal Interoperable Core is a reflective middleware of Dynamic TAO. The middleware supports multiple middleware platforms in ubiquitous environment. UIC interacts with the service in CORBA and same service in Java RMI and SOAP.

UIC allows interoperability across multiple middleware platforms. The components are loaded and unloaded when configuration and reconfiguration are done. Interoperability is attained by configuration, reconfiguration and loadable libraries at run time.

UIC uses dynamic adaptation and tackles the problem of heterogeneity in mobile environments. However the design only targets to object oriented architecture. It offers no solution to publish-subscribe or data sharing systems [2]. UIC does not solve the problem of service discovery protocol heterogeneity and it cannot be used in adhoc environments. It could be used only in the environment of single service discovery protocol.

Hence in the proposed work, middleware has been developed to solve the problem of interoperability across multiple service discovery platforms. Proposed middleware is a reflective middleware that supports heterogeneous services.

Analysis of existing Middleware Heterogeneous Environments

1) A higher level solution does not exist in web services and MDA. The higher level abstractions will increase the chance of heterogeneity.
2) All the middleware solutions discussed deals with platform composed of single discovery protocol. But in wireless environment to work with single service discovery protocol is difficult. Hence interoperability across multiple service discovery protocols is the urgent requirement of wireless environment.
3) The bridging architecture of MDA and web services works for fixed networks and to bring them in wireless environment is difficult. Therefore platform that changes based on the context becomes the urgent need.
4) Adaptive middleware to transparently detect functioning of both end services becomes the urgent need. For a wireless environment context aware reflective middleware that transparently detects context at both ends becomes the real essential feature.

1.3 Middleware for Smart Environment:

COOL TOWN is a web and content based solution running on underlying framework of network technologies. Content based is done with URL and is achieved by means of cookies. User context is detected by location,identity and browsing capacity. XML description

is used to describe the information and the relationship between identities that is stored in relationship directories. MUSE is the intelligent smart space made intelligent with the help of sensors. Generally smart spaces are designed in java and the migration of entities from one smart space to another is done with the help of sensors. A community can be defined as a set of services. Java space is used for sharing in the community pool. All the smart space comains are used to track the user movements within the domain. Only COOL TOWN helps to sense the user movement in the web. Distributed computing concepts allow the access of elements across smart spaces. The implementation of JXTA is the only smart space comain that allows communications across smart spaces and it is only Jxta that supports peer to peer technology. All other spaces are based on client-server applications [14].

1.3.1 SMART SPACE LAB

Smart space is designed to deal with interoperability challenges. Reconfigurability is the concept which arises when mobile users move and continue to receive the services even in the new environment. Issues of service discovery include which equipments are available, which services are available and availability of computing power and storage capabilities. The smart space laboratory has been designed to address the measurements, standards and interoperability challenges of smart spaces.

Analysis of Smart Environment Middleware

Most of the smart environment middleware's discussed are java enabled middleware. Only the smart space lab project deals with interoperability and reflectiveness which is an important property in pervasive environment. Seamless service discovery protocol interoperability is not dealt with in smart space project. As a result our work designs smart spaces enabled on service discovery protocol to attain seamless interoperability which is a feature of pervasive environment.

1.4 Middleware for Mobile Computing

MICROSOFT .NET FRAMEWORK: .NET is targeted at mobile systems. Middleware logically exposes functionality as component of name space. Middleware addresses heterogeneity and interoperability of system. Design of most of the middleware is targeted to stable systems.

GAIA: Physical spaces in GAIA are active spaces. GAIA includes operating system concepts and support event notification, mobility, location awareness, service discovery and code updates. GAIA converts the physical space with ubiquitous programmable environments into active spaces. Users can interact across active spaces and can move form one active space and get inculcated to the other. It is an event based middleware and each agent performs a task. It performs task like service discovery, context deduction etc. GAIA uses a central file system to monitor context. It is applicable only to small scale networks and is not applicable to large scale wireless networks.

EXORB: It is the distributed and reflective middleware that supports configurability and allow replacement of certain concept to adapt to changes in environment.

WSAMI: Pervasiveness in WSAMI supports automatic retrieval of services. WSDL is the language used in web interface, SOAP defines light weight mechanism for information exchange and UDDI are used for locating and registering web services. WSAMI middleware provides retrieval of matching services in the environment [15].

Analysis of Mobile Middleware

GAIA is the only middleware that supports adhoc environment and work on active physical spaces. Interoperability features has not been added to GAIA. Other middleware's discussed in mobile environment are component, web or proxy based and are not suitable for adhoc environments. GAIA works even in pervasive environments. The strength of adhoc networks resides in diversity of computer networking and growth of wireless over IP. Mobile computers and applications are available at any time and space even when infrastructure is not available. The wireless device performs their own routing topology keeping track of connection between nodes. These networks are self established and all the nodes have to carry out their function. Nodes must be able to automatically integrate with the device and configure them as part of adhoc networks.

Reflectiveness enables application to lesson and perform changes in behavior. Specifically, reflection collects details of internal

structure and later provides means to dynamically alter the system changing the current state and adding new features to the system. It is difficult to get scalability in adhoc networks especially in peer to peer networks due to the limited capacity of memory and temporary storage of data.

REMMOC is a publish/subscribe architecture that provides support to context, content and subject based filtering. REMMOC works on framework of OPENCOM and supports adhoc architecture of primitive components. The overall architecture of REMMOC is a reflective framework which is divided in to component framework that could be configured and automatically reconfigured. REMMOC provides two architectural bindings - the service discovery architecture and binding architecture. The binding framework includes plug-ins for different binding components. Fundamentally synchronous or asynchronous middleware can be plugged in the framework. Interoperability in REMMOC is brought across different heterogeneous middleware [10].

ROVER was designed for solving heterogeneity issue in mobile environment. ROVER uses asynchronous call to discover service, when devices move in the distributed environment. JEDI allows the construction of dynamic tree of network in which members can connect and disconnect frequently. Client-server application cannot work on ubiquitous environment. Reflective middleware reconfigures, inspects and configures at runtime.Reflective middleware is made of components of desired profile. RCSM is a context sensitive middleware and adapts to object discovery. CORTEX creates entities called sentinent object for processing and providing context related

information. Sentinent objects are defined as autonomous object that senses the environment. Context tool kit framework separates the acquisition and presentation of context information from application that uses widgets [16]. KONARK is a middleware for discovery and delivery of service in multihop adhoc network for service discovery [17].

In context aware middleware for adhoc networks, asynchronous method of communication is adapted. EMMA is message parsing method by means of which message is propagated to each host in the network. Using context aware routing protocol, the message is not transmitted to all nodes but to single node based on context. Use of CAR promises better use of resources and thereby better efficiency [18].

NOMAD provides seamless transparent service discovery by means of intelligent services. Main aim of NOMAD is to bring connectivity with the physical space of user and virtual internet space. Personalized user query is accepted by NOMAD and corresponding service is obtained in the web by Service Location Protocol, Dynamic Host Configuration Protocol and Light Weight Directory Access Protocol [19].

Open service discovery architecture enlightens service discovery and provides a model for cross technology. Distributed information storage and querying are provided as a unified representation using inter domain models. Service discovery protocol differs in architecture, message pattern, expected operating environment, service representation and description.These differences

make interoperation difficult. Existing work of internetworking include Bluetooth-UPnP, Bluetooth-salutation, Bluetooth-Jini, Jini-UPnP, Jini-SLP, Jini-Twine, Salutation-SLP [20].

Already existing service interoperable platform provides service interoperability with service ontology. Ontology refers to the formal explicit description of the concept which is often conceived as a set of entities, relations, instance, functions and axioms. Open service gateway initiative based service delivery and management platform is used in global health care services. OSGI is an open standard that provides horizontal platform for hosting service components in service bundles but takes place with the access of API's [21].

1.5 Review of Work Done and Drawbacks

Most existing middleware is designed for fixed infrastructure networks typically operating in a stable environment and use a client-server semantic which is not suitable for MANET (mobile adhoc networks). Further these middleware are not reflective and are not suitable for heterogeneous networks. The middleware design in this research adapts to mobile environments, is context aware and reflective, and allows interoperability between heterogeneous nodes and networks. Another major disadvantage of existing work is that lack of interoperability of service discovery protocols, i.e., service advertised by a particular discovery protocol is not identified by other discovery protocols. For example the service discovered by UPnP (Universal plug and play Protocols) cannot be used by the devices that supports SLP (Service Location Protocol). Mobile computing works on a dynamic environment with devices of variable size and

having diverse resources. Other limitations observed in the existing works, is that all existing middleware involves plug-in's or bindings which incurs heavy overhead. Designing of middleware across higher layers is a challenging task. The major problem is adaptation of services on the device of the user.

1.6 Objectives of the Proposed work

In our proposed work services gets automatically deployed in the device of the user. The main aim of our work is to design a context aware middleware for MANET's that provides communication across heterogeneous platform based on protocol interoperability. Thus the main objective of this work is the design of: (i) A novel middleware architecture for service delivery in smart spaces (ii) To provide a cross layer approach at the middleware between routing and transport layer, thereby improving the functionality of middleware (iii) To create peer to peer clusters, offering services across smart spaces (iv) To provide seamless service discovery across heterogeneous smart space by peer to peer optimization (v) To provide security for entire architecture.

1.7 Research work Done in This Thesis

The objective of the research is to design a mobile dynamic reflective context aware middleware for protocol interoperability in heterogeneous smart spaces and to add features of cross layering, clustering and security.

1.7.1 Context Aware Middleware

The novel idea of the proposed middleware is the seamless interoperability of protocols. The second goal is to bring together

large number of heterogeneous smart spaces. The third important property is that the middleware is context dependent.

1.7.2 Smart Spaces

Smart spaces are adhoc networks composed of sensors, actuators and other smart and dummy nodes. These smart spaces consist of smart nodes that are capable of offering services and dummy nodes which are capable of consuming the service offered by the smart ones. Each of these smart spaces is built over a service discovery protocol. Service discovery protocol has its own means of advertising, publishing, and consuming services. In spite of the heterogeneity, these spaces are brought together by protocol interoperability of the service discovery protocol at the middleware.

1.7.2 Interoperability

Interoperability is a feature where in which heterogeneous platforms having different components and functionalities are brought together. Spaces designed are mobile, adhoc and sensors enabled spaces. Mobile spaces have fixed infrastructure, adhoc spaces does not have fixed infrastructure and pervasive spaces are sensor enabled. The interoperability feature of service discovery protocol gathers all smart spaces together. As route passes through the middleware to reach the transport layer, the routing protocol becomes interoperable and thereby increases the speed of search and delivery of services. The integrity and reductancy are checked by sensors.

1.7.4 Cross Layer Approach

Cross layering is an important phenomenon by means of which data is transmitted between layers by skipping the intermediate layer. The functionality of the intermediate layer is automatically taken during transmission even when the layers are skipped. The proposed approach also helps to provide dynamic adaptations by providing the functionality of platform independency. The routing protocols are selected from the routing layer and services are offered by the transport layer. Thus the routes are taken seamlessly from the routing layer to the transport layer through middleware build upon as cross-layer approach.

1.7.5 Service Discovery and Routing Protocols

Service discovery protocols acts as a substratum in the smart space for discovery of service. Service discovery protocols considered are – Service Location Protocol, the Universal Plug and Play Protocols and Pervasive Discovery Protocol. This Service discovery protocol differs from the normal existing protocols due to seamless property added to it and the intelligence provided by the sensor nodes. The routing protocols considered are Adhoc On-demand Distance vector (reactive routing) and Destination Sequence Distance Vector routing (proactive routing).

1.7.6 Peer to Peer Clustering

The nodes in the smart space form clusters. There could be any number of clusters in each of the smart spaces. The important

phenomena dealt with in the architecture are that, only the cluster heads of each cluster remains stationary. All other nodes are moving. When ever a service is to be provided to the user, the request profile of the user seamlessly gets routed to the cluster head of the matching service. The cluster head will know the location of the node which has moved away from the cluster. Ant colony algorithm with honey bee optimization in clustering is applied to obtain optimization while routing. The advantage of clustering is that the search becomes easier and is less time consuming.

1.7.7 Security

Security is an important feature of all networks and especially demands high recognition in wireless adhoc networks, as there are no fixed infrastructures in this set up. As the devices moves across the networks, there are heavy chances of attacks which demands security. Security in the proposed architecture is obtained by the digital signature scheme thereby checking the validity of the user.

1.8 Organization of the Thesis

The thesis is organized as follows.

Chapter 2 discusses on the design of mobile dynamic reflective context aware middleware for protocol interoperability in heterogeneous smart spaces. It also deals on the quantitative performance of the proposed middleware in comparison to the REMMOC architecture.

Chapter 3 discusses on the design of smart spaces at the transport layer of the protocol stack. It deals on the design components and on the security aspects.

Chapter 4 deals on the interoperability feature which is an important property of the middleware. Interoperability is studied in two aspects – interoperability of service discovery protocols across heterogeneous smart spaces at the middleware and interoperability of the routing protocols at the middleware through cross-layer approach.

Chapter 5 deals with the cross layer approach the binds together the routing layer with the transport layer in the protocol stack through the concept of interoperability at the middleware but without traversing the middleware layer.

Chapter 6 discusses in detail the service discovery and routing protocol along with their functionality and performances.

Chapter 7 provides a detailed description of architecture of peering nodes along with clustering algorithm and also deals with service discovery and matching algorithms.

Chapter 8 discusses on the qualitative evaluation of the designed architecture taking an application of smart campus and the study of the various metrics of protocols on the middleware. The result is compared with the performance of message oriented middleware with CAR protocol.

Chapter 9 concludes the work done in the thesis. It discusses the research results of the thesis and also suggests points that can be further explored.