International Journal of Computer Science and Engineering (IJCSE)
ISSN: 2278-9960
Vol.1, Issue 2 Nov 2012 1-10
© IASET

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

A FRAMEWORK FOR CONTEXT DRIVEN SERVICE PROVISIONING IN MANETs

[1]K. PONMOZHI & [2]R.S. RAJESH

[1]Research Scholar Mother Teresa University Kodaikannal, India

[2]Department of Computer Science and Engineering, Manonmanium Sundaranar University, Tirunelveli, India

ABSTRACT

Adaptability is one of the challenges of today's mobile distributed systems. Context-awareness has been recognized as an important enabler for adaptability in service provisioning in MANETs. We are trying to showcase the influence and its management in the issue of service provisioning, namely discovery, selection, reselection and rediscovery. We have designed a framework to provide service provisioning in focus to the context manager component which will tackle the context accessing, interpreting, and decision making activities. We describe how this framework can support adaptability to user profile and preferences, network and device capabilities.

KEYWORDS: Mobile Ad hoc networks, Context-management, Policy-based management, Service provisioning

INTRODUCTION

Service provisioning obtains increasing attention due to the fast growth of telecom technologies and high-quality requirements of the user. Services include those provided by the devices or the public services such as web services. Trends in information and communication technologies propose interoperable environments, where information is shared by operators of the information in a transparent manner [1]. The availability of resources and services strongly depends on the formation of the underlying network. Since ad hoc network is dynamic in terms of mobility services and resources are not offered permanently and stationary. Potential high level and ambitious demands of users, context awareness and limitations on the resources imply the need for an adaptive service provisioning system. Context information must be gathered and used in different provider and user domains. The process of detecting and providing services is called 'service discovery and provisioning'. When a service is offered by multiple nodes in an ad hoc network, the specific matching between client nodes and service providers typically refers to as service selection, has a crucial effect on the performance of the wireless multi-hop ad hoc networks. Over time, changes in network topology degrade the optimality of service selection requiring clients to continuously re-evaluate their choices of a server, a process we refer to as reselection. As new nodes may come and leave the network the availability of the service providers may change, the active probe for the availability of service providers is termed as re-discovery. Any adaptive service provisioning is expected to provide (i) support for service discovery and selection depending on current context (ii) support for service reselection and rediscovery based on the changing context, (iii) adaptability to limited capabilities (iv) support for user profiling and preferences (v) service downloading and configuration of the services on the terminal (vi) Support for QoS management. In order to facilitate these activates, we need to access the current context. Context encompasses all of the information relevant to an interaction between a service and its set of users including the participants as well. Therefore, all the stakeholder in service provisioning namely service, service provider, consumer profile and preferences, device capabilities and current status, network status, has to be accessed in order to adapt to the context. In general the context can be divided into static and dynamic where as the static contexts, which once defined will not change during the execution of the service/application, can be represented in the form of profiles. Whereas, Dynamic contexts, which will change even after the execution starts, can be represented as the form of policies. It possible to make interoperable environment of stakeholder together by agreed common policy mechanism [2]. Policies are used to specify the adaptation to be used on certain conditions. Policy based context information support for service management has been studied in the work [3], in which the information required to trigger the service management activities are assumed. We aim to provide the functional context management aspects to facilitate the management of service provisioning. The context management module is responsible for collecting all the dynamic contexts, their processing, and enforcing the defined policies if the conditions are met. So far different context modelling and frameworks has been proposed in the literature where the context management has been separated from the implementation task of the application. In this paper we propose a context model suitable for representing information and managing services in cross-layered environments and using policy-based management for integration of context information in adaptation of service provisioning.

Contexts in MANETs

Context-aware systems can provide users with better service based on analyzing physical context and personal context such as user preferences, user activity and so on. The main task of any system with context aware is to acquire and utilize the context information in order to provide services that are relevant to the place, time, user etc. Context can be used in many ways, as in [14] context values are used to handoff the load to other APs (application provider). In the case of service provisioning in MANETs which are based on the wireless medium, the framework/middleware which provides service provisioning has to be context aware. In context aware service provisioning, it is essential to enable the dynamic retrieval of available services, while increasing the automation in service selection and binding. Realizing context-awareness has proven to be a difficult problem on many levels: first what constitute context information, many researches provided different definition, and some approaches context as an abstract concept while others came from the desire to be able to technically represent context information. Second what and how should be adapted when context changes and where do the context definitions and adaptation rules come from? Third problem is how to represent and process contexts, the related knowledge and adaptation rules? We address the second and third issues in the next subsections.

Layers of Context Information Processing

There are three layers in context processing as depicted in fig.1. In the three levels of context management process, the lower layer is dedicated to signal processing and techniques used to recognize contexts, the middle layer is the core of context modelling , where multitude of representation and reasoning have to be applied. The top layer deals with firing context dependent actions and adaptation. Applications can use provisions of the middleware to access the context information from context storages if needed. The main use of the middle layer is to eliminate direct linking of context providing components and the consuming components.

Table 1: Types of Context and the Stakeholder which Produces it

Type stakeholders	User (examples)	Network & Device	Service Provider
Sensed	Location	SNR value, Link Quality	Moving Speed, current load
Derived	Activity	Route quality	Reliability (based on speed, memory capacity etc.)
Explicitly provided	Profile and preferences	Protocols, Device profile	Service profiles, preference

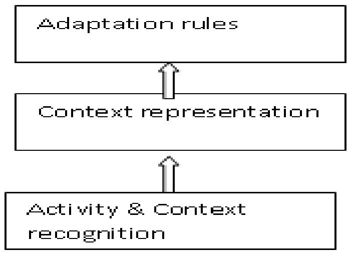

Fig. 1: Layers of Context Processing

Context Types and Sources

Generally, according to the method of retrieving the information the context can be divided into three [4] (i) Sensed Context, (ii) Derived context, (iii) Explicitly provided context. Sensed context is a type of context information that is obtained through low-level sensors. The most used sensor technologies in context acquisition are GPS, RFID, Bluethooth. Derived context in turn refers to context that can be computed on the fly based on the information retrieved and sensed. Explicitly provided context means context information, which is explicitly entered into the application by the user such as profile and preferences.

The explicitly provided context is called metadata. The framework we designed is specifically focused on user-centric service provisioning. Some examples and type are showcased in table. 1. We focus the representation of service/user/device capabilities and requirements. While selecting service, the comparison between service request and service profile is performed by considering single capabilities and requirements. The device profiles are also consulted for refined search. We allow the user to personalise the service discovery by specifying the preferences such as specifying the priority order among the requested capabilities. *Service profile* consists of identification information, functional and non-functional requirements and capabilities. *User Profiles and preference* composed of identification, capabilities and requirements. The user requirements describe user-defined constraints that must be always satisfied during service provisioning. Preferences allow the user to specify their choices on the resulted service. The preferences can be specified in the form <name, type, weight, value>. Where 'name' name of the requirement specified like 'color'. Type is the data type of the requirement. We permit the user to specify three type like Numeric, Boolean and String. The requirement is classified as 'mandatory' and 'secondary' requirements. To represent if the requirement is mandatory we can specify the weight as 1 and other values are considered as the priority of that requirement. Value represents the expected value for that requirement. We use the same mechanism to specify the requirement for the service provider as well as the user. *Device profile* describes the device's capabilities such as its technical characteristics, and supported software and hardware. Device requirements specify the technical requirements the other devices which access this device service should met with. We have defined the types of the context.

Context Managing Framework Architecture

Context management comprises of all the phases from acquisition at lower layers to the offering to the applications/other entities which require the context information. Context aware functionality should only have to produce and send context data and declare their interest to receive them from the support middleware, while an internal middleware function takes over and transparently executes specific management operations [5]. The 'Explicitly provided' Context information can be from many sources such as sensors, databases, or from lower layer protocols. The format of these data may vary depending on the source as well as their capability to store or maintain the data will also vary. So we need to change the formats, store the collected data for later use in an abstract way useable by the middleware. The static data such as user profiles and preferences will be parsed and stored in the database by the parser module. The dynamic context data will be accessed by the use of adaptors. For further details refer to our work [15]. We defined adaptors for each of the source of information. Since the amount of context data that can be gathered is large, it is in efficient to gather all at all times. Also maintaining all them may not be possible. There may be situation when a particular data is not needed. So it is better to collect data only when needed. Adaptors will be called whenever a particular data is needed. Fig. 2. Shows the context acquisition and the interaction among parser & policy enforcer for adaptation.

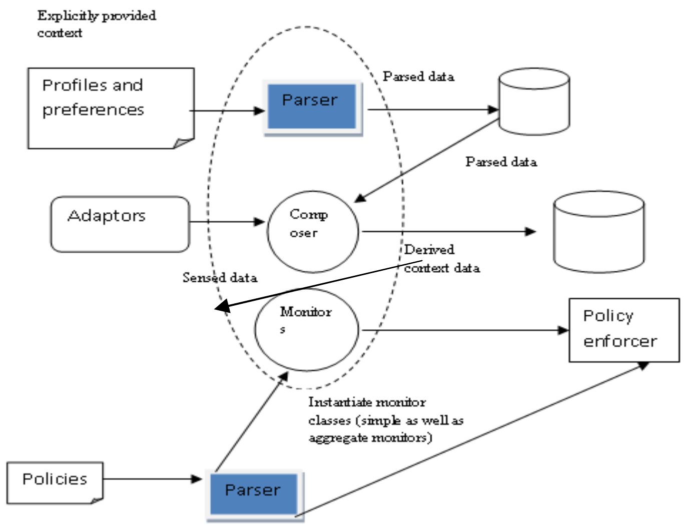

Fig. 2: Interaction among Components of Adaptation

As the gathered data are not readily usable, they have to undergo a simple or complex process to derive new knowledge. A simple process is like converting into the data format required which will be done at the adaptors itself before storing them in the database. Whereas, combination or aggregation of different context may also be needed wherein, this can be done by the use of monitors. Aggregator operation [6] with the goal of merging more context data in a new high-level one is essential even if they are resource demanding, because the context changes claim for continuous updates and we have to handle it automatically. In order to create proper triggers for appropriate actions we define monitors. Monitors in our architecture are simple classes which are capable of raising events when the given threshold value of the context they are monitoring. The threshold value to be monitored can be passes as parameters. If we need to monitor for more than one context data then we can use the aggregateOr or aggregateAnd classes. We have define three types of monitors namely Providers status monitor, Internal Status monitors, network Status Monitors. They can call the necessary adaptors for the current value of the context data.

POLICY MANAGEMENT

Self-management has acquired a lot of importance from technology focusing on distributed networking. This can be realised by providing Context information in the policies which are necessary conditions to trigger adaptation actions in service provisioning, so that we can manage the service provisioning in self-managed way without the need of manual interference. This enables the middleware to take into account the changes in the various stakeholders for adaptation. We aim to develop a middleware framework for adaptive service provisioning by using context information and adaptation information provided in the form of policies. In service provisioning context can be (i) used to adapt to the user preferences (ii) adapt to terminal capabilities (iii) used to expand service request to provide more relevant information that is not explicitly specified by the users (iv) Services could behave differently based on the context, so, context information can be used for service matchmaking (v) used to describe the users' preferences to different services (vi) Used to further categorize service for retrieving better results. For example services can be grouped based on their location. Our objective is create a framework to enhance the service provisioning by means of policies in addition it takes into account the variations in the service operation and performance.

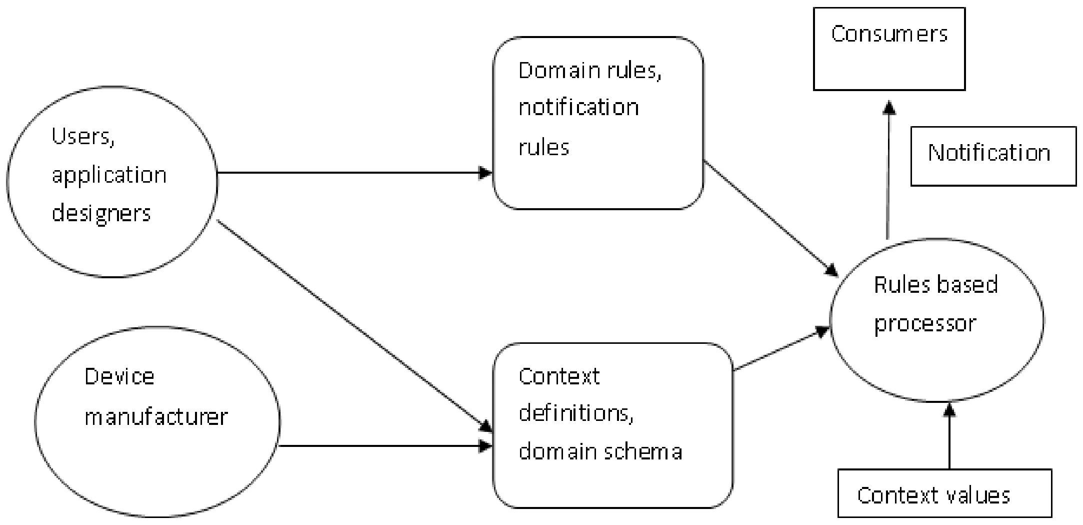
Users, application designers
Device manufacturer
Domain rules, notification rules
Context definitions, domain schema
Consumers
Notification
Rules based processor
Context values

Fig. 3: Policy Management (Providers and Consumers)

The central concept of policy based management is a reliable service that has the knowledge of the configuration policies and that is able to take suitable actions when the context change events are detected. Fig 3. Showcase the policy providers and general processing of the policies. Policies are identified at various stages of service provisioning, from service design phase to deployment phase. Enforcement of policies requires identification of rules outlining actions to be performed under different conditions. These conditions are characterised by events. In table.2 we show different policies and the components affected by them.

In this section we define a set of modules and mechanisms needed to support policy-based management of service provisioning in MANETs. Whenever a node receives an advertisement of new service, that service has to checked if it is the better service provider than the one already selected according to the policy of re-selection. Similarly when a service provider is not available due to mobility, or failure appropriate action has to be taken to tackle the situation based on the policy specified. The Monitor module is the collection of three types of Monitors namely i) Internal status Monitors (ii) Network status monitors (iii) QoS Monitors

Internal status monitors are responsible for collecting the internal resources data. Since many of the services depend on the internal status of the client device. QoS requirements of services have to be satisfied in some of the applications. The parameters such as bit rate, bandwidth delay are to be considered. So we provide a set of adaptors for accessing the values of them and monitors for the expected value of those parameters. The policy enforcer module knows the context change events that are considered critical for policy enforcement and it instruct the Monitor modules about its interest in the detection of such events. The function of the 'Monitors' is to detect these critical changes in the system environment and notify the policy enforcer of such changes. The policy enforcer is responsible for evaluating the policies and enforces them when critical change occurs. The policy enforcer receives new_ services_found event, when the advertisement module informs it, or re-discover event when failure of any service provider is sensed by the monitors. On receiving such events, the policy enforcer identifies policies that depend on these change events and identifies the modules which are to be affected by these events. It also executes the actions required for enforcing the policies.

The above functions performed by three different kinds of components that define policy enforcement conditions, policy enforcement actions and monitoring module. The policy enforcement conditions define the events that should trigger executions of the policy enforcement actions. The enforcement actions first identify the set of components (classes) that are affected by a given event according to the policies. Then it invokes the appropriate classes to realise the necessary changes.

Programming Model

Appropriate abstractions and programming models play a crucial role in the development of flexible context-aware applications [7]. Most context-aware applications are still programmed with embed use of context information directly into score code which leads to static behaviour. To support an asynchronous style of programming in which actions are invoked automatically invoked in response to context changes we use triggering mechanism. We followed Event-Condition_Action model specified in the form of polices, in which each trigger includes a precondition on the invocation of the specified action that is evaluated upon detection of the trigger event. The interaction among the monitor module and policy enforcement is depicted in fig. 4

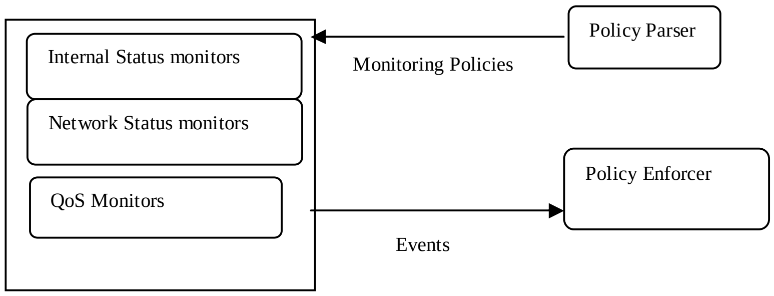

Fig. 4: Interaction among Monitors and Policy Enforcer

Service Provisioning Architecture

This architecture aims to provide functionalities like (i) propagating service information across the network, (ii) match service discovery requests with the advertisements, provide the application with the appropriate information for selection based on the current context,(iii) monitor the changes in the network & adapt based on the policy specified for reselection and rediscovery, and (iv) notify the application for corrective action if the specified polices cannot be executed. In order to do these we split the design to have the following components.

Service Discovery and Selection

All the services advertised in the network will be stored in the table of overlays. When a client application needs service it invoke the interface function provided by specifying the service type to discover. The module resource_manager.get (service type); the service_type will be hashed and the corresponding overlay nodes will identified. In order to reduce the time for discovery and increase performance we sort the service providers based o their metrics.

The matching module will consult with the context manager for the context information and does the selection of the server. The context manager will provide both the static and dynamic context values.

As we have given the user the ability to specify their mandatory and secondary attributes with preferences values the valuation will be done based on the weight age calculated.

Following the service discovery, depending upon his/her request either the list of services or the finest service will be given to the user. From that the user can select the desired service. The binding policy for that service will activated following the selection. Different steps in service provisioning are shown in fig. 5.

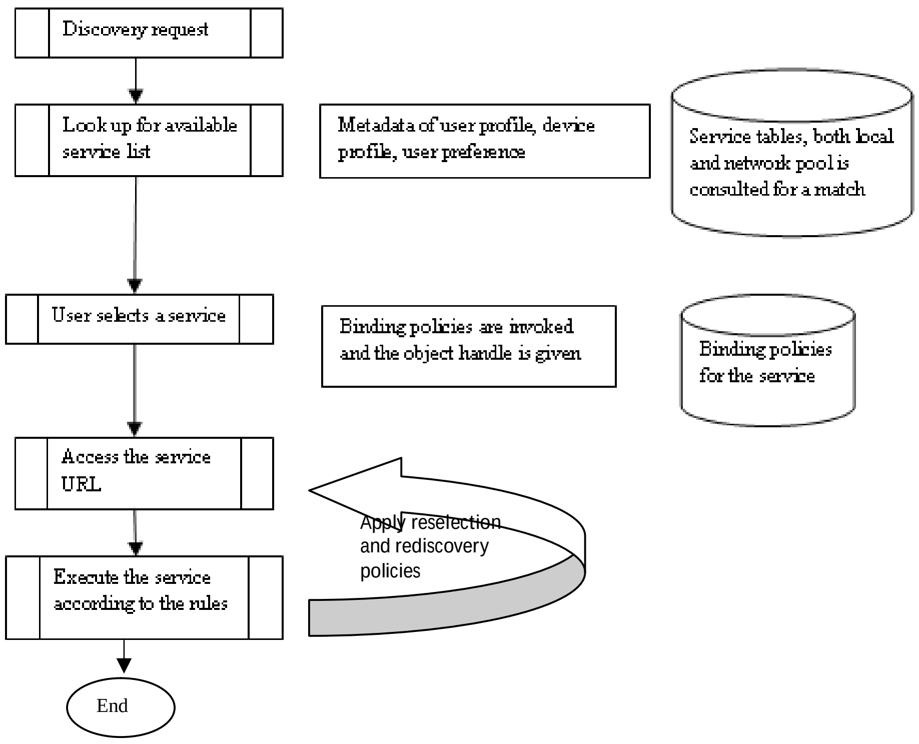
Discovery request
Look up for available service list
Metadata of user profile, device profile, user preference
Service tables, both local and network pool is consulted for a match
User selects a service
Binding policies are invoked and the object handle is given
Binding policies for the service
Access the service URL
Apply reselection and rediscovery policies
Execute the service according to the rules
End

Fig. 5: Service Discovery and Provisioning

Service discovery and provision needs the following phases 1) service discovery 2) service use 3) service termination. Support for all these functionalities in a protocol results in improved performance, because of consistency and completeness of the discovery/provision procedures. During service provisioning there are occasions where clients need to control the server to change service execution parameters. In addition there are circumstances under which servers must notify clients about the status of the service being offered. The control and notification signalling provides a bi-directional reliable link(s) between the client and the server, which improves the performance of service provisioning. There are different approaches for awareness from the status of a service. The service status can be obtained by periodically polling, in which a client regularly asks the service status. Another approach is notification, in which the client subscribes to a service status event and any changes on the subscribed event are notified to the subscribers of that particular event.

Reselection & Re-Discovery

To optimize the performance, MANET clients need to constantly re-evaluate their choice of a service provider. Re-evaluation has two components: reselection and re-discovery. Reselection reconsiders server selection based only on the current entries in the network pool of resources. Rediscovery involves probing the network for up-to-date information about available service providers. In designing a re-evaluation policy, application developers need to determine when to do reselection and rediscovery. These will be specified in the form of polices. These policies also are taken care of by our parsers.

The simplest reselection policy is do reselection in reaction to change in the service table, which stores the list of service in the network. We call this as reactive policy. (i) finding new server entry (ii) a change of service-specific metric, (iii) only when there is no valid route to the server (iv) when the route to the current server breaks.

The rediscovery can be (i) do rediscovery when the route breaks (ii) try rediscovery after all known routes to the current server. Apart from the rediscovery and reselection policies, other two categories of policies used in our architecture is binding and implicit policies.

Explicit Policies	Affected by events
Service re-selection	No-routes to the server, change in service table, explicit server request for termination
Service re-discovery	No-routes to the server and no server information in the service table.
Binding policies	Once the server is selected binding may depend on the factors like memory capacity and the like which may be provided as binding policies
Implicit Policies	For Service type different QoS parameters has to be fixed

Service adaptation before deployment of a service ensures that the service is tailored to the capabilities of the device [8], service adaptation can also be activated during the execution of a service. Each service specifies constraints that define working conditions that guarantee proper execution of the provided functions of the service. A context change causing certain constraints to be violated will then trigger the runtime adaptation. To not overcomplicate the self-adaptation of our system, the adaptation triggered only by changes in resources and service requirements. This information is usually readily available and requires no intensive processing.Our current implementation assists applications in selecting best available server, it is not aware of the actual client-server communication and does not provide additional support for migrating client session between servers. The stateless service migration may be simple such as opening a new connection to the server but, stateful services requires guaranteeing that application-specific state is consistent across the servers. Application state and connection migration is outside the scope of our study.

Experimental Environment

We have simulated our prototype with ns-2. The characteristics chosen are physical radio transmission range of 250m with a raw capacity of 2 Mb/s. IEEE 802.11 as MAC protocol. We use 100 nodes randomly placed in the rectangular area of 300m X 1000m. Nodes move following random waypoint model with no pause time at the speed in the interval 0 to 2 m/s. Among the nodes we used 4 servers and 50 clients with a Constant bit rate of 7.5 packets/sec. We have user DSR routing protocol. We have used only three rediscovery policies namely fast which triggers service rediscovery immediately after the current route to the server breaks, medium triggers service rediscovery only if no route exist to the current server, and it tried all cached routes, and slow which triggers rediscovery only when there are no valid routes to any of the servers that offer the service. Fig. 6 shows the medium achieves best performance, validating the policy of trying the entire cached routes o a given node before issuing a route rediscovery, whereas slow suffers network congestion as it issues rediscovery as soon as the route breaks.

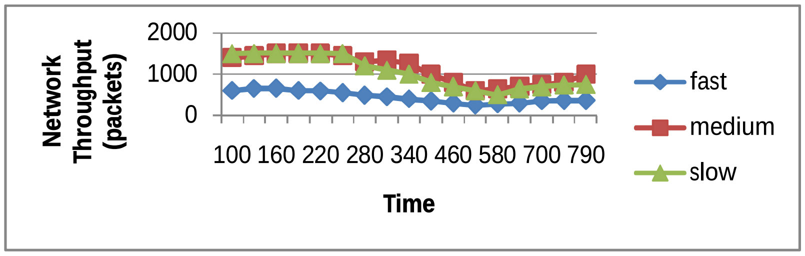

Fig. 6: Effectiveness of Rediscovery Policies with 4 Servers and 50 Clients

Related Work

A framework dynamic provisioning for non-technical service client has been proposed in [9]. Service requirements are expressed using goals as an abstraction, and ontologies to specify domain specific knowledge. In [10] authors defined Coordination and monitoring APIs to provide contextualized services in smart environments. In [11] it designs a scalable and efficient geographic routing and service provisioning framework for MANETs. The frame work facilitates self-configuring, distributed and hierarchical structure, adaptive routing protocols to meet different routing requirements in service provision scenario. In [12] author defines a framework for service provisioning in intermittently connected MANETs, relies on flexible addressing scheme, content-based management of messages and asynchronous communications. [13] Presents a context-aware service provisioning model based on the concept of migratory services. Service migration is triggered by the context changes of nodes.

CONCLUSIONS AND FUTURE WORK

In the computing era of pervasive computing, context awareness being a prime issue demands a generic context-aware system that gathers, interpret contextual data and facilitates the process of smart service provisioning to a mobile user. The process starts with context acquisition, followed by the representation using standardized language and subsequent interpretation and decision making on the basis of users' and the services' context and ends in the delivery of appropriate service to the user.

We proposed a service oriented framework that realises context-awareness in pervasive computing environments. The model facilitates the delivery of finest services to the users by interpreting contextual data. We use context information to stimulate policy-based management for adaptive service provisioning. Since both context and polices are described in XML they can foster to better interoperability. We described parts that are necessary for context realisation and adaptation in service provisioning. As we have not completed the work, we will continue implementation work. From that we will learn whether our approach is efficiently fulfilling the tasks and maybe make smaller refinement to our architecture if needed.

The prime constraint in the system is the availability of the context sensing devices or the sensor devices. Though some of the context may be derived from history, we have not included as it is considered as storage overhead. The dynamic contextual data of the user as well as the service are sensitive information. The security of the information is essential by restricting unauthorized usage; this will be considered in future.

REFERENCES

1. J. Kephart and D. Chess, The Vision of autonomic computing, Computer vol.36, no. 1 pp 41-50, January 2003.

2. Ankur Gupta, A.K.Vatsa, Policy Based Fast Handoff Mechanism for MANET, World of computer science Information technology journal, vol 1, No.8, 339-345, 2011
3. S. Davy, K. Barrett, M. Serrano, J. Strassner, B.Jennings and S. Van der Meer, Policy interactions and management of traffic engineering services based on ontologies, Latin American Network Operations and Management Symposium, September 2007
4. Mostefaoui, G.K., Pasquier-Rocha, J., and Brezillon, P. (2004) Context aware computing: A guide for the pervasive computing community. In ICPS'04 Proceedings of International Conference on pervasive services pp 39-48
5. Paolo bellavista, Antonio corradi, Mario fanelli, and luca Foschini, a survey of context data distribution for mobile ubiquitous systems, ACM Computing Surveys, authors version to appear March 2013
6. LI, F., Sanjin, S., and Schahram, S. 2010. COPAL: An adaptive approach to context provisioning. In Proceedings of the IEEE 6th International Conference on Wireless and Mobile Computing, Networking, and Communications (WiMob'10) 286-293
7. P.J. Brown, J.D. Bovey, and X.Che., Context-aware applications: From the laboratory to the marketplace. IEEE Personal Communications, 4(5):58-64, October 1997
8. Wagelaar, D: Context-driven model refinement, in: Proceedings of the MDAFA 2004 workshop, Linkoping, Sweden(2004)
9. Luiz Olavo Bonino da silva santos, vikram sorathia, Luis Ferreria Pires, Marten J. Van Sinderen, Towards a Conceptual Framework to Support Dynamic service proviosning for non-technical service clients,Journal of software, vol. 6, No.4, Aril 2011, pp. 564-573
10. Pascal Bruegger, Benjamin Hadorn, Beat Hirsbrunner, SSP: Smart Service Provider A Smart Environment Providing Contextual Services on Android Mobile Devices, International Conference on Ubiquitous Intelligence and Computing,Springer LNCS, UIC , October 2010
11. X.Xiang and X.Wang, "A scalable Geographic Service Provision Framework for Mobile Ad Hoc Networks", Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communiactions,2007.
12. N. Le Sommer, "A framework for Service Provision in Intermittently Connected Mobile Ad hoc Networks", IEEE 2007.
13. Riva, T. Nadeem,C, Brocea and L.Iftode, "Context-Aware Migratory Services in Ad Hoc Network", IEEE Transactions on Mobile Computing, Vol. 6, No.12, December 2007
14. Abhijit Sharma, Shantanu Joshi and Sukumar Nandi, Context aware mobile initiated handoff for performance improvement in IEEE 802.11 Networks, International journal of Computer Networks and Communications(IJCNC) vol.3, No.3, May 2011.
15. K. Ponmozhi and R.S.Rajesh, Cross layer Information in MANETs:why and how?, CiiT International journal of wireless Communication vol 4, no. 6, April 2012, pp 294-302.