Ad hoc resource binding in MANETs

K.Ponmozhi,
Research Scholar of Mother Teresa Women's University

Dr. R.S. Rajesh,
Department of Computer science & Engineering,
Manonmanium Sundaranar University,
Tirunelveli

ABSTRACT
Mobile ad hoc networks consisting of highly dynamic, decentralized and self-organizing network of autonomous nodes, which interact as peers have become popular, distributed environments. The possible mobility of users, terminals, and even service components requires solutions to handle properly the links to the resources in response to the mobile entity migration. Binding decisions may depend on dynamic deployment condition and should be determined at service provision time. In this paper we propose a framework to predict the resource movement and the different types of binding, policies to represent the ways to realize dynamic binding to other equal resource.

General Terms
Mobile Ad hoc Networks, Resource sharing, Dynamic binding.

Keywords
Resource sharing, Resource binding, dynamic binding.

1. INTRODUCTION
A Mobile ad hoc network consists of wireless mobile nodes dynamically forming a temporary network without the use of existing network infrastructure or centralized server. In these kinds of networks, the classical distinction between clients and servers as far as physical devices are concerned is replaced by the paradigm of co-operation between a large set of potentially heterogeneous devices. The physical topology of any MANET is in constant state of flux due to node's mobility. Mobility encourages the development and deployment of new classes of information services that can take into account the relative position of users, service components and available resources. Thus, when a mobile entity changes its location, it is necessary to adapt a suitable strategy to update its references to the needed resource. A mobile entity could take the needed resources with itself; or it could transform its references to resources in the origin locality into remote resource references when re-connected at the destination locality; or it could re-qualify its old resource bindings to new suitable resources in the new hosting locality. The choice of the most proper re-binding strategy requires the visibility of the location of the mobile device and the resources, and also depends on dynamic deployment conditions, and therefore should be decided at service provision time. Towards this in this paper we allow to specify the binding strategies as high level policies, and they can be modified and activated dynamically with no impact on the implementation of service component.

The paper is organized as follows section 2 talks about the resource binding, section 3 elaborates the architecture, section 4 concludes with future direction of study.

2. RESOURCE BINDING
A mobile application may employ various services some of which may be used complementary where as others may exists only locally. [12] Service provision in MANETs is opportunistic There are no fixed, well-known service providers. Any node can be a service provider for its own benefit and for as long as it participates in the MANET, or as long as she desires to be a service.

Thus binding of services to applications must be resolved dynamically at runtime as the location of the user and other context information cannot be known as a priori. Thus, dynamic binding is necessary for runtime service composition, and in a changing context. In case of a mobile node with no constraint on its local resources and if it wants to work on a information resource it can copy it and process independently, but if the node is having a strict limitations on its local resources like memory, it can copy it to the nearby node and make a remote reference to it. In another case where an information service, different versions of the same functionality is available depending upon its locality then the middleware should connect to the apt instance of the service in the case of node movement, which expects a node to disconnect and rebind to a new service instance. Therefore, a mobile node can refer to resources through various types of binding. Below are the four types of binding categories [1, 3], each of these type specifies how the mobility of the node affects its bounded resources.

Resource movement strategy: This strategy transfers bounded resources along with the node, e.g. internet connection along with node with WiFi connectivity. This strategy requires the handling of modifications of other possible client bindings to the moved resources.

Copy movement strategy: This strategy copies /transfers bounded resources along with the node e.g. copy of a map into the rescue personnel's node.

Remote reference: Rather than moving the resources, this strategy modifies the node's binding to refer to the resource remotely. This requires communication with the remote execution environment hosting the information resources. When coping to the node is not possible we can store the information to the nearby node which does not have any memory restrictions.

Rebinding : This strategy binds the node to the equivalent resources available in the new locality e.g. connecting to the different instance of the info services.

Since the mobility of the nodes in MANET is a norm rather than exception, and mobile applications require greater binding flexibility than provided in the conventional approaches in which developers hard code strategies into the service logic we should consider binding of resources to the node dynamically to accommodate different deployment scenarios.

In this paper we investigate the resource management for MANETs, and identified challenges include a need to provide support for identification of alternative resources and provision to have dynamic binding to the resource. The requirements lead us to the architecture presented below.

3. ARCHITECTURE

The Resource Manager simplifies the interaction between the application core and complex services on mobile hosts as illustrated in Fig.1. The programmer provides an interface that the service must obey along with a set of policies that dictate the manner in which services are chosen. Policies simply specify the properties of a service that are of importance to the application, e.g., spatiotemporal stability. These policies are specified as a set of weights on all properties of a service. Once a service is chosen, the connection between the application and the service is maintained transparently until it is not required anymore. Note that the middleware is free to choose more than one service to fulfill the needs of a single interface specified by the programmer. In the rare cases that an exception occurs, it is propagated back to the client application. The net result is that an application can interact with a changing set of mobile services as if they were a single static service.

The proposed middleware facilitates any application to discover service components, possibly negotiate service tailoring to fit device characteristics, and the current execution context. We facilitate dynamically program the binding strategies for the application agents. This means biding strategies can be defined and changed even at runtime without affecting the application code.

We separate service logic from binding management so that the developers can code, change and reuse service components & Administrators can express binding strategies at a high level of abstraction in terms of declarative policies. The separation of concerns in the framework we propose is obtained via the adoption of a policy based approach. Polices are rules governing choices in the behavior of a system separated from the components in charge of their interpretation [6]. We use XML to specify the policies[14] and profiles. A policy in XML has the *Target* section which describes the storage location, or the class, whose method that has to be invoked. The *Events* section describes a logical combination of events (e.g. time-based even, context-based events) once that happen, triggers the enforcement of the actions defined. The *Action* section describes the actions to be enforced, once the *conditions* are satisfied. When applied to binding decisions, the event clause captures a change relevant to binding management that operations and the action clause specifies the method that allows setting the type of binding strategy to activate at the event's occurrence. Policies can express a relationship between a domain of subjects and a domain of (target) managed resources. The subject of a policy determines the entities, which are granted permissions to perform actions on the target resources. Both subject and target refer to objects that can be loaded by java class loader. E.g policy specification

```
<?xml version="1.0"?>
  <policy1>
 <target> ….</target>
<subject>…. </subject>
<event>…..</event>
<action>…..</action>
<condition>…</condition>
</policy1>
```

Context awareness – is the knowledge of application specific attributes, such as preferences, level of trust, subscribed services, and access device characteristics- and location awareness – is the knowledge of the physical position of the user or device connection to the network infrastructure. Available resources depend on location information - are the characteristics needed for this separation to be effective.

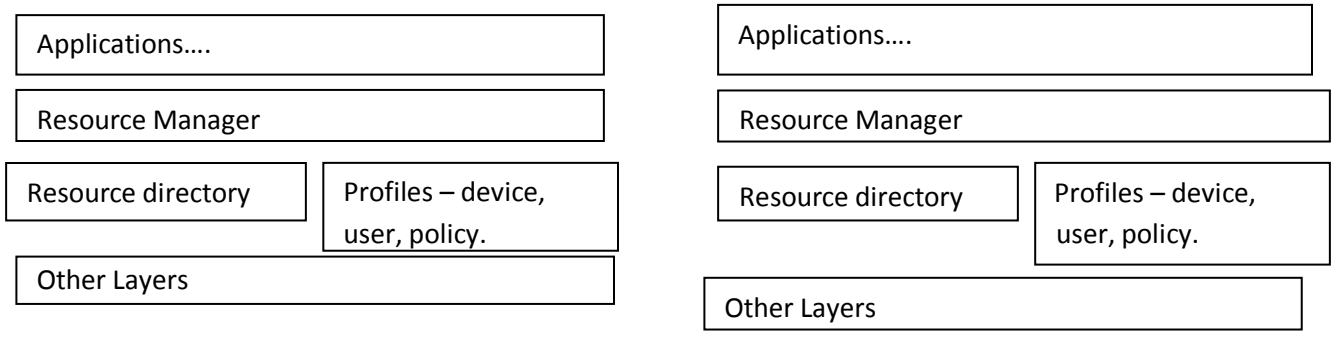

Fig 1. The General Architecture

To support dynamic binding, we use metadata. Meta-data provide information about users, devices, and resources and about the preferred binding strategies; middleware facilities perform runtime binding management actions based on metadata and context and location visibility. We use two types of metadata profiles describe user requirements and resource properties. In this section we describe our initial design of Resource Manager (fig. 2.) and the interaction among its sub-components. We assume the existence of prediction module which will predict the future location of the nodes and topology information which has been elaborated in our earlier paper [13].

Components of Resource Manager

This is an interface between the application layer and the other modules. It accepts resource request queries from the application(s), and sends back the results to the application(s). See Fig 2

Resource Directory

Below the Resource Manager is the traditional Resource directory consists of local and network resource details. The Resource directory is a repository for resource advertisements. Each host has a local resource pool which is shared across hosts within communication range. The Network resource pool is a repository of knowledge that is provided by the local host or obtained from other hosts via the communication layer. It should be noted that any knowledge within the knowledge base is shared freely with any other interested host in the MANET.

Local resource pool: Maintains the list of its own sharable resources with the preference profiles of them. It also maintains the status of them. When receives query from the resource manager it check for the availability in the local pool, compare with the profile and status then returns the resource descriptor- an object with the same methods and constructors as that of the requested resource - to the resource manager.

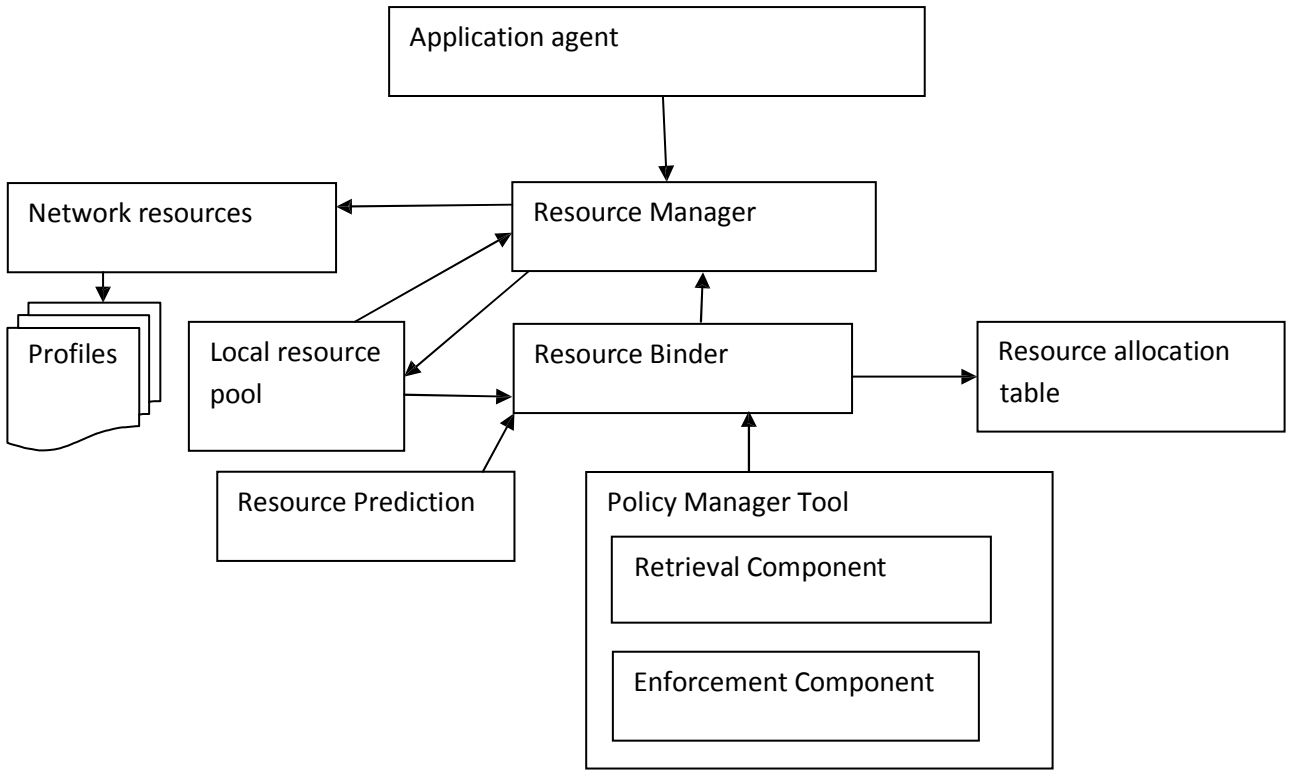

Fig 2. The Components of Resource manager

Network Resource Pool – such as the number of neighbors, the distance between neighbors, and the preference profiles. Every node has to maintain the list of nodes that can share their resources, with their descriptions [2]. It maintains a table for each kind of resource and list of nodes that has the resource and the statistics about the nodes. Whenever a request is received by this module it checks and returns the node_id. Now the resource manager has to make a request to the node for its confirmation of availability if the time of beacon packet stored is grater than the specified time interval. - Which is not shown in the diagram – for details refer [10].

Resource availability / prediction: In MANET scenarios, peers have to exchange information needed to carry out assigned works. Such an exchange can occur only if peers are connected and also they are in the network. In these cases if the two nodes are going to disconnect the assigned work could be terminated leading to a situation of stalling in the application/ system. Bridging/alternate actions has to be taken to avoid these situations and to support for time planed execution of a process. In choosing a bridge / alternative resource the context and the availability, the resource description etc. has to be considered. This module will trigger resource discovery when the resource this node refers to is moving. Once the resource manager identifies a resource to bind, that information will be passed on to the binder module.

The smooth running can be ensured by reselecting new resources to replace if the already assigned resource is no longer available resources. Each node will have separate component for dynamic resource management.

The alternate resource identification will be initiated in one of the following

- By the node using the resource (client node mobility)
- By the node providing the resource (serve mobility)

Server node mobility

Phase I

- When the server node detects that the node it services has not sent any hello packet for a threshold period of time it sends a warning packet & wait for ($) time
- If it is not receiving any acknowledgement, it changes the status to "available" in the resource allocation table for all the resources it has allocated for that node.
- Sends this de allocation message back to the client

Phase II

- In the case of its movement the server send all the nodes to whom it serves a packet of "de allocation"
- The clients will initiate the resource searching

Client node mobility

- Prepares the list of resources from other nodes from the network allocation table
- Check the RDF of the resource

If it can be copied,

- Check the client nodes status
- Check the WSDL provided by the server node
- Get the copy of the resource
- Change the applications reference to the new instance of the resource

If the resource cannot be copied

- Check for equivalence in the network resource table
- When the equivalence is found
- Send request query for the needed resource, if the requesting node is ready to serve it will send the profiles to the requester
- If the resource can be allocated the new resource can be bound to the application
- The reference to the previous resource can be changed and made it to refer to the new resource.

b. Resource Binder

We need supporting facilities for adaptive binding to resources. This module is responsible for managing the binding. We had one resource table for an application. Resource table records the list of resources used by the application.

Reference object implements binding mechanism accordingly to the required resource binding strategies. When the strategy to be applied is remote reference the reference object uses RMI to refer to the resource. Where as, if the strategy is copy movement then the object uses serialization mechanism to convert the objects and transmits them. In the other end, it will be de serialized and stored in the resource table. In the case of resource movement Applications which are using this resource will be informed – this information is available in the resource pool - .

In the case of rebinding the binder searches the availability of the resources in the network, by sending information to the network resource module. With the availability of this node-id the resource manager will ask for the confirmation from the node's local pool and make the binding. The information in the

resource table will also be altered based on this – details can be found in paper [10].

Binding decision is done once in every context/topology change triggered by the prediction module. We use obligation policies which represents the action to be taken when a particular condition/event occurs. Events are related to generic changes in the system and can model any variation in the environment state.(i.e) changes in the binding strategies can be triggered by variations in user/device location, by resource method invocation. Binding policies define the circumstances that trigger a binding change, a new type of binding to exploit and dynamic conditions that must hold in the deployment for resource binding to take place. Binder is responsible for managing the bindings; policy-enforcer enables the binder to perform the low-level operations according to the desired binding strategies.

In this section we focus on the idea behind the binding of resources dynamically. We permit the designer to provide the default binding as declarative policies. The choice of proper binding strategies depends on several factors like access-device properties, runtime conditions, management requirements and user preferences. To support dynamic binding we use two categories of metadata. (i) Policies they are used to specify and control choices in resource binding strategies. They represent which binding strategy should be used and when. (ii) Profiles describe the user requirements and resource properties. (iii) Device profile specifies the access device's hardware and software characteristics. Resource profile describes resource interfaces as well as the properties that could be useful for binding decision such as whether the resource can be copied or migrated. We use XML based standard formats for profile representations.

At the first request the resource descriptor – an object that has the same methods and constructor interface of the requested resource- is issued. This can be requested using the resource.get (resource_name) function. When the resource is identified from the local pool the appropriate resource_id, & its descriptor is issued. Afterwards the application can access these resources directly by using the reference objects without the influence of the binder. The default binding strategy for that resource is identified by calling the function getpolicyprofile() method which will return the policy file for this resource. The default binding for the I/O resources are remoterefernce where as the files can be specified as copy/ move type binding. For hardware resources like printer or internet facility the binding will be rebind. For the software services the default binding will remote reference. policy_enforcer(resource_id, policy_filename). The policy_filename is the metadata filename of the policy for this resource. Which will be parsed by the parser and the values will be supplied to the policy_enforcer. If it is the resource of other user then the preference profiles of that resource will be parsed and given to this policy_enforcer. Based on the parsed details the policy_enforcer does the binding and adds them to the resource table. It also instantiate one reference object for the resource. The resources that are used by an application are maintained in a table. Each entry in the resource table has resource id, the resource name in use, its descriptor, binder policy, reference object, the binding object encapsulates the low level functions to be done for the current binding policy.

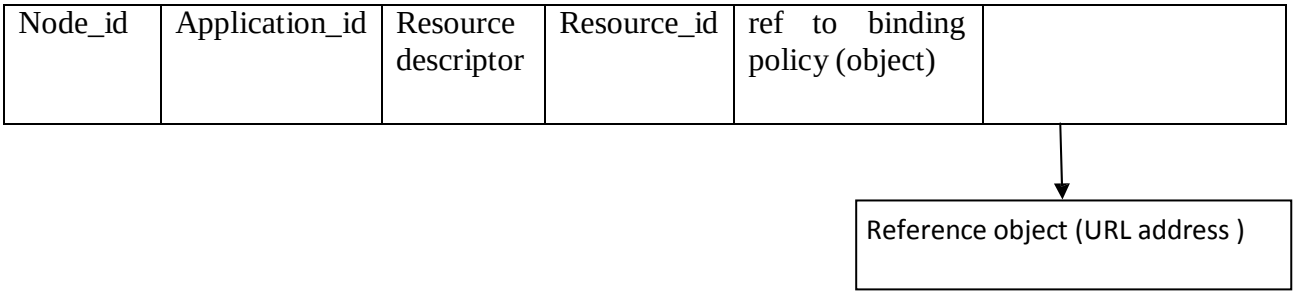

Fig 3. The structure of the resource table

For resources like I/O devices the binding to the devices will be voided and new resource which is identified if any will be rebind to the application. In the case of remote reference strategy, the java RMI is used, to make remote reference. When ever the device movement is identified, the Resource_Diconnect (resource_id) will be issued the policy-enforcer checks for possible policy to be enforced and if it is resource copy/move strategies, the resource object is converted in to stream of bytes & java serialization and de serialization mechanism is used, and updates the resource table accordingly. The resource is passed as a stream of bytes to the new execution environment and rebuild through de serialization – these functions are encapsulated in the binding policy objects- . Whenever we dynamically readjust the binding due to node movement the binder updates the resource table accordingly.

4. CONCLUSION & FUTURE WORK

The complexity of developing and deploying mobile applications over the Internet dictates a separation of concerns between resource-binding strategies and application logic implementations. Only this clean isolation permits the necessary flexibility and reusability of middleware and service

components. Novel and programmable middleware solutions, integrated with different types of high-level metadata, can provide management configurability while hiding low-level mechanisms and implementation details from service developers and system administrators.

In this paper we have used XML for Meta data representations. XML parser is the required element that needs t be there in the nodes along with the other java files. Sessions of the applications are not taken into consideration now. The session maintenance for the rebind able services may be implemented in future. In the case of modifiable resources the conflict resolution has to be devise.

5. REFERENCES

[1] A. Fuggetta, G.P. Picco, and G. Vigna, , 1998 "Understanding Code Mobility," IEEE Trans. Software Eng., vol. 24, no. 5, pp. 342–361

[2] G. Kiczales, J. Irwin, J. Laming, J. Loingtier, C. Lopes, C. Maeda, and A. Mendhekar, 1996, "Aspect Oriented programming", In special issues in Object Oriented programming Max Muewhaeuser (general editor) et. Al. in.

[3] L. Cardelli, "Mobile Computation," 1997, Mobile Object Systems: Towards the Programmable Internet", LNCS 1222, J. Vitek and C. Tschudin, eds., Springer-Verlag, , pp. 3–6

[4] SUN Microsystems. 1998, Remote Method Invocation.

[5] W. Grosso, R. Eckstein(eds.), 2001, Java RMI, O'Reilly.

[6] M. Sloman, 1994, "Policy Driven Management For Distributed Systems", Journal of Network and Systems Management, vol.2, No. 4, Plenum Press.

[7] P. Bellavista, A. Corradi, and C. Stefanelli, 2002, "The Ubiquitous Provisioning of Internet Services to Portable Devices," IEEE Pervasive Computing, vol. 1, no. 3, pp. 81–87

[8] P. Bellavista, A. Corradi, and C. Stefanelli, 2001, "Mobile Agent Middleware for Mobile Computing," Computer, vol. 34, no. 3, pp. 73–81.

[9] World Wide Consortium, Composite Capabilities/Preferences Profile, http://www.w3.org/Mobile/ccpp.

[10] R.S. Rajesh and K. Ponmozhi 2009,"Applying p2p in MANETs" in International Conference on control ,automation, communication and energy conservation, vol 1, pp 66-70.

[11] Ines Houidi, Wajdi Louati, , Djamal Zeghlache, Panagiotis Papadimitriou and Laurent Mathy, 2010, "Adaptive Virtual Network Provisioning" VISA , September 3, New Delhi, India ACM 978-1-4503-0199-2/10/09, pp 41-48

[12] Christopher N. Ververidis and George C. Polyzos ,2008, "Optimizations for Charged Service Provision in Mobile Ad Hoc Networks" 978-1-4244-2100-8/08/$25.00 ©2008 IEEE awareness in to MANET" at the National Conference on "Explorations & iNnovations In Advanced Computing", organized by Department of Information Technology, National Engineering College, Kovilpatti, India.

[14] David W Chadwick, Linying Su, Romain Laborde. "Coordinating Access Control in Grid Services" Concurrency and Computation: Practice and Experience, Volume 20, Issue 9, Pages 1071-1094, 25 June 2008.