A System Architecture and Implementation for Meshed Multicast in MANET using SIP

Ying Yu

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

December 2004

Your file Votre référence
ISBN: 978-0-494-16247-7
Our file Notre référence
ISBN: 978-0-494-16247-7

ABSTRACT

A System Architecture and Implementation for Meshed Multicast in MANET using SIP

Ying Yu

Mobile Ad-Hoc Network (MANET) is an active topic of research for its potential of providing pervasive services anywhere and anytime, even though some challenges need to be handled first. In contrast to conventional networks, ad hoc networks are formed by a set of devices that communicate without using a pre-existing network infrastructure.

Multicast is an efficient transmission scheme for supporting group communication in networks. The concept of overlay networks enables multicast to be deployed as a service network rather than a network primitive mechanism, allowing deployment over heterogeneous networks without the need of universal network support.

In this thesis, we propose an overlay multicast framework to handle multicasts in MANET environment in a flexible way. Our approach is using SIP to discover peers to set up a meshed overlay network first, then overlay multicast trees are set up on demand. To cope with the bandwidth limitation problem, the Dominating Set Based and Multi-Recipient routing are adopted, so that the virtual meshed network topology gradually adopts the changes in the underlying network in a distributed manner and the multicast tree is adapted to the updated topology information accordingly. A prototype is built to evaluate the feasibility and performance of the proposed framework.

ACKNOWLEDGEMENTS

First I would like to express my deepest appreciation to my supervisor, Dr.Anjali Agarwal, for her invaluable help, patience, guidance and support. I would also like to thank all the professors with whom I have interacted during my studies at Concordia University.

I would also like to thank my friends for their help, companion and fellowship. In particular, thanks to Ms. Bin Qi and Mr. YuYong He.

Finally, I express my deepest gratitude to my parents; without their love, support and encouragement I would never have reached this level.

Table of Content

List of Figures

List of Tables

List of Abbreviations and Symbols

ALM	Application Layer Multicast
GMSNG	Group Member and SIP Non-Gateway node
GLSNG	Group Leader and SIP Non-Gateway node
GMSG	Group Member and SIP Gateway node
MANET	Mobile Ad-hoc NETwork
MN	Mobile Node
NS	Non-SIP node
SG	SIP Gateway node
SIP	Session Initiation Protocol
SNG	SIP Non-Gateway node
UA	User Agent
URI	Universal Resource Identifier

Chapter 1 Introduction

1.1 Mobile Ad-Hoc Network

Mobile ad-hoc network (MANET) [1] is an active topic of research for its promise to provide ubiquity services anywhere and anytime, including providing a networking platform in the absence of a fixed, central-managed infrastructure. MANETs replace the centralized hierarchical administration structure of contemporary networks with a distributed approach to routing, to security, to management and to application.

MANETs are useful in many application environments and do not need any infrastructure support. Collaborative computing and communications in smaller areas (buildings, organizations, conferences, etc.) can be set up using MANETs. Communications in battlefields and disaster recovery areas are other examples of application environments. Ad hoc network has also been regarded as an important part of the 4G Wireless System. The goal of 4G is to provide a seamless and ubiquitous communication environment so that users can accomplish their tasks and access information at any time, anywhere, and from any device [2]. Ad hoc networks can be used to extend base station's coverage to complement the deficiencies of the infrastructure-based network. The increasing use of collaborative applications and wireless devices may further add to the needs and uses of MANETs.

Mobile devices (also called as Mobile Nodes) form a MANET when they wirelessly communicate with other nearby mobile devices without the support of fixed infrastructure.

The mobility of nodes means that a MANET's topology changes frequently and dynamically, where each node communicates over wireless channels, moves freely and may join and leave the network at any time. Without the aid of any centralized administration to support the monitoring of these network topology changes, mobile nodes must themselves track changes in a decentralized manner.

Another serious impediment of MANET is the limited area that can be covered by mobile application components using a wireless transmitter, hence a node communicates directly with nodes within wireless transmission range and indirectly with all other destinations using a dynamically determined multi-hop route, relying on other nodes to act as routers [3]. Thus, multiple network hops may be needed for one node to exchange information with another mobile node. Each node behaves as a router as well as an end host, so that the connection between any two nodes could be a multi-hop path supported by other nodes.

1.2 Overlay multicast (Application Layer Multicast)

Network-level IP Multicast [4] communication was proposed a decade ago to enable one-to-many or many-to-many communication as opposed to one-to-one supported by unicast. Multicasting allows a sender to transmit voice, video and text to multiple receivers simultaneously. When sending the same packet to multiple receivers, multicast improves bandwidth utilization and involves lower host/router processing.

However, the use of multicast in applications has been limited because of the lack of wide scale deployment and the issue of tracking group membership [5]. There are a large number of applications whose requirements are substantially different from the design point of IP

multicast. Such applications include video-conferencing, multi-party games, private chat rooms, web cache replication and database/directory replication. These applications usually contain a small number of group members, and the groups (e.g. multi-party games) are often created and destroyed relatively dynamically.

In order to meet the requirements of the emerging applications, a solution is needed for multi-sender multicast communication, which scales for a large number of communication groups with small number of members, and does not depend on multicast support in the routers. Application Layer multicast (ALM, sometimes, also called as Overlay Multicast) [6] addressed these concerns and is proposed as an alternative approach.

In ALM, the overlay network is built on top of the physical level, providing simple primitives to hide the complexity of the underlying physical network topology, allowing any peer to uniformly address any other peer on the network. Because an overlay network forms a logical network consisting of multicast member nodes, the underlying network looks at the data exchange between member nodes as a unicast communication. The change in physical topology is transparent to the overlay members and they are required to provide multicast functionality only [7].

Since the multicast support shifts from core routers in IP multicast to end systems in ALM, end systems now implement all group communication functionalities, including membership management, packet replication and distribution. Furthermore, it allows more flexibility in customizing some aspects, e.g. data transcoding, error recovery, flow control, scheduling, differentiated message handling or security, on an application-specific basis [8].

A straightforward advantage of using multicast in the application layer is that multicast

applications can be executed in networks that do not support IP layer multicast. It also allows firewalls to handle better the traffic created by the multicast application [9]. Thus, ALM offers accelerated deployment, simplified configuration and better access control at the cost of additional (albeit small) traffic load in the network, because in an ALM session, only the source and the destination nodes of the multicast session can sit on the end points of the paths. All the participants are connected via a virtual multicast tree, that is, a tree that consists of unicast connections between end hosts. The shift of multicast to end systems introduces certain performance penalties, such as duplicate packets on physical links and larger end-to-end delay than IP multicast [8].

In summary, the ALM approach differs in many respects from traditional IP multicast routing:

- A forwarding node in the overlay topology can be either an end host, a dedicated server within the site, or an edge router. On the contrary, traditional multicast trees only include core routers.
- With an overlay topology, the underlying physical topology is completely hidden. A directed virtual graph is created between all the nodes.
- In traditional multicast, the membership knowledge is distributed in the multicast routers. In an Overlay Multicast group, membership knowledge is known either by a rendezvous point, the source, or everybody, or is distributed among members.
- The overlay topology is potentially under complete control.

As it was mentioned earlier, in overlay multicast group, each member node needs to keep track of other members in its neighborhood. This can be done by a query to its "routing table", or by a periodic neighbor discovery operation. Each node records the status of its virtual

neighbors in the overlay topology. The entries are the link state information of all group nodes obtained from virtual neighbors [6].

1.3 Overlay multicast (Application Layer Multicast) in MANET

Multicasting has become increasingly important in MANET because of the need for collaborative applications among a group of mobile nodes. It is always advantageous to use multicast rather than multiple unicast, because the bandwidth comes at a premium in MANET. Therefore, combining the features of a MANET with the usefulness of multicasting, it will be possible to realize a number of envisioned group-oriented applications [10].

Several multicast routing algorithms such as MAODV [11], ODMRP [12], AMRIS [13], AMRoute [14], CAMP [15] have been proposed to support multicast in MANETs. These routing algorithms work at network layer. As demonstrated in these protocols, the absence of a fixed infrastructure and the dynamical changes of connectivity and network topology make robustness as a key issue for multicasting in MANET.

MANET and overlay multicast systems both exhibit a lack of fixed infrastructure and possess no a-priori knowledge of arriving and departing users [16]. As we mentioned earlier, in Application Layer Multicast (ALM), the virtual topology can remain static even though the underlying physical topology is changing. This function is suitable for applications used in the typical, dynamic MANET. Moreover, as shown in Figure 1-1, it needs no support from the non-member nodes, i.e. all multicast functionality and state information are kept within the group member nodes only (like the peers in Overlay Network of Figure 1-1). Due to this common nature, overlay multicast seems natural and attractive to be deployed for MANET

applications.

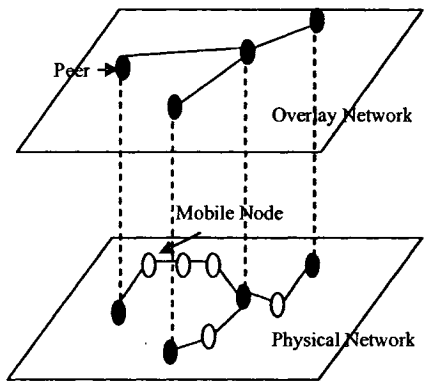

Figure 1-1 Concept of virtual topology for overlay multicast in MANET [6]

1.4 Session Initiation Protocol (SIP) in Mobile Ad-Hoc Network

The Session Initiation Protocol (SIP) [17], defined by IETF, is an application-layer control protocol that can establish, modify, and terminate multimedia sessions. These sessions can be multimedia conferences, Internet telephone calls and similar application consisting of one or more media types such as audio, video, whiteboard etc. Moreover, the leading wireless technical forums (3GPP [18], 3GPP2 [19], and MWIF [20]), have agreed upon the SIP as the basis for session (call) management of the Mobile Internet within UMTS. Therefore, SIP will certainly be an integral part of the mobile Internet's protocol architecture.

SIP is designed in a modular way so that it is independent of the type of session established and of the lower-layer transport protocol deployed beneath it. SIP messages and the sessions they establish can pass through entirely different networks.

Because of their mobility, nodes in MANET sometimes are configured into a meshed

topology. Therefore, SIP sessions in MANET could use full-meshed architecture. That is, in a full mesh, every participant builds a signaling leg with every other participant and sends an individual copy of the media stream to the others. This mechanism only scale to very small groups, but requires no network support. Proposals for setting up and tearing down SIP call-leg meshes have been made [21]. There is also some work done on full-mesh conferencing [22].

The current SIP procedures are designed to be well suited for users within infra-structured networks. But in ad-hoc networks, because there is no fixed, managed infrastructure, there is no dedicated proxy, redirect and registration servers and users are not aware of each others' presence and do not know others' identities. Thus the common SIP procedures are not suitable for ad-hoc network users. Not much work has been done to address this issue. A framework to use SIP in ad-hoc networks was proposed [23].

1.5 The motivation of our proposal

As technologies of wireless communication improve, more and more software services and mobile devices will exist in large numbers and will operate in a networked world where they may not be sure about the other services and devices nearby beforehand, or about the state of the network neighborhood as time goes. In such uncertain environments, individual components will need to discover and maintain awareness of their surroundings and to configure and adapt themselves in response to changing situations.

The rapid growth of computation and network resources for end-hosts has enabled peer-to-peer (P2P) computing and networking [24]. Multipoint communication achieved by

the end-hosts makes Application Layer Multicast (ALM) increasingly recognized. Multicast is used for neighbor and service discovery in MANET because of [25]:

- the broadcast nature of wireless communication environment;
- the infrastructure-less feature of the mobile ad hoc networks;
- the one-to-many character of service advertisement and discovery.

Mobile ad hoc systems currently developed adopt the approach of not having a middleware, but rather rely on each application to handle all the services it needs. This constitutes a major complexity/inefficiency in the development of MANET applications [26]. The middleware layer operates between the networking layers and the distributed applications, with the aim to build on top of raw network services, higher level mechanisms that ease the development and deployment of applications.

Among middleware services, service discovery and location play a relevant role in ad hoc environments. Upon joining a self-organizing network, mobile nodes should be able to explore the environment to learn and locate the available services. Current solutions on service discovery assume (more or less) a fairly stable network since they are based on centralized service registries. Several architectures for service distribution in ad hoc networks have been proposed in literature. Many of them focus on the mechanisms for describing and matching services, as well as the user interfaces for discovery applications [27].

Jini [28] and Service Location Protocol [29] provide the function of service advertisement and discovery for wired networks. As mobile device have become more and more powerful, it is natural to find a way to provide the service and corresponding neighbor discovery in MANET.

Our goal is to investigate the use of SIP methods to support adapting the network topology changes and provide a platform for service location and discovery in ad hoc networks.

SIP is adopted into our proposal because it is a very flexible protocol and can be easily extended [30]. A user with multiple devices like a cell phone, desk phone, PC client and PDA can rely on SIP to seamlessly integrate these entities for increased efficiency and productivity. SIP is differentiated from similar communications protocols by its wide industry support, providing a practical means of multivendor integration at the highest level of the protocol stack – the application layer [31]. The "Event" header and the idea of Event package could provide a way for SIP User Agents to get some service or event state notification, which could be used for service advertisement and discovery.

Furthermore, since SIP is designated as the call control signaling for 3G Multimedia services, more and more applications might be designed using SIP for 3G devices. So by using SIP middleware for mobile nodes in MANET, we may adopt the same application structures and functionalities for other kinds of mobile devices.

1.6 Assumptions in our framework

Our approach is based on some of the assumptions made for Mobile Ad hoc NETwork (MANET). They are:

- An IP level connectivity exists in the ad-hoc networks. It is a valid assumption considering most of the modern devices run operating systems that provide automatic configuration techniques [27].
- Each node selects unique URI and IP address by which it will be known in ad hoc

networks. This allows each node to be recognized by all other nodes in ad hoc networks as a single entity regardless of which network interface they use to communicate with it.

- There exists an underlying unicast routing protocol that can be utilized for IP unicast communication between the neighboring multicast nodes.
- All nodes wishing to communicate with other nodes within the ad hoc network are willing to participate fully in the operations of the networks. In particular, each node participating in the network should also be willing to forward packets for other nodes in the network.
- Nodes do not continuously move so rapidly as to make the flooding of every individual data packet the only possible routing protocol.

The application registration and notification mechanisms defined herein are not intended to be a general-purpose infrastructure for all classes of application. For that reason, we provide a SIP-specific middleware framework for applications in MANET, which is not so complex as to be unusable for simple features, but which is still flexible enough to provide services.

Because a MANET is a collection of nodes forming a temporary network without the aid of any centralized administration, where each node communicates over wireless channels, moves freely and may join and leave the network at any time, SIP Registrar, Proxy and Redirect servers are not required in our approach.

1.7 Introduction to SIP-based Overlay Multicast Framework

In this thesis, we propose a middleware system: SIP-Application Layer Multicast (SIP-ALM),

which works between the application and network layers of mobile nodes in MANET. Our goal is to implement fundamental functions in a middleware and to provide some basic functions for applications to be used in MANET networks.

As shown in Figure 1-2, SIP-ALM is an application level group communication middleware, which does not rely on network infrastructure support and thus, allows accelerated deployment and simplified configuration at the cost of a relatively small increase in traffic load. It is used to build an ad-hoc overlay network allowing users to know each other, especially to inform users of which multicast groups to join. It also provides a platform for service discovery and location, for applications, such as entertainment; military and post-disaster rescue applications, used in MANET.

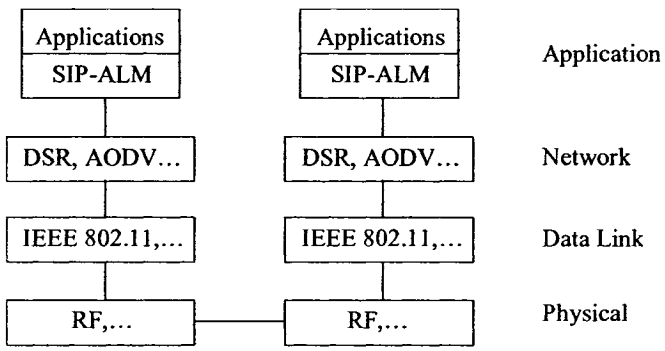

Figure 1-2 Concept of our framework

This middleware system uses SIP messages to organize the overlay multicast groups for the applications in MANET, which hides underlying network inconsistencies from distributed applications and provides the applications with consistent views of a distributed execution. Our proposed framework does not require any particular feature on participating devices except the implementation of our middleware in mobile nodes (MNs). It provides the following:

- allows peers to discover each other;
- allows peer to establish a meshed MANET;
- allows peer to organize Application Layer Multicast (ALM) groups in that MANET.

As shown in Figure 1-3, our middleware functions are divided into three parts:

- SIP Overlay Mesh function handles the creation and maintenance of the SIP overlay meshed network;
- SIP ALM Group function is in charge of creating and maintaining SIP ALM group;
- Node List function is used to manage the node information.

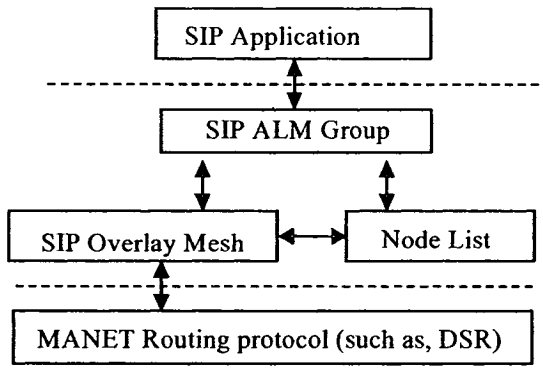

Figure 1-3 Concept of our framework

Here we will describe the main ideas of the system briefly. The more detailed information will be introduced in Chapters 2, 3 and 4.

Many of the proposed multicast routing protocols [10] construct multicast trees over which information is transmitted. Using multicast trees is evidently more efficient than the brute force approach of sending the same information from the source individually to each of the receivers. However, due to the frequent and hard-to-predict topological changes of ad-hoc networks, maintenance of a multicast tree to ensure its availability could be a difficult task.

Thus the idea of creating a multicast mesh is proposed.

A meshed network is a subset of the network topology that could provide multiple paths between multicast senders and receivers. Redundancy in the paths helps maintaining routes between senders and receivers, even if a mobile node in the mesh moves. Since a multicast mesh could provide rich connectivity among group members, it improves the robustness and survivability of the multicast in MANET. The middleware we propose has two main functions: overlay meshed network construction and multicast group routing. It creates a SIP overlay meshed network first, and then builds a multicast group and transfers application data to all group members in a session.

Generally, an Application Level Multicast consists of three fundamental functions [24]:

- End-host discovery
- Overlay network construction
    - The joining end-host saves received requests and responses in the meshed host lists to keep trace of the neighboring nodes logically.
    - Optimize the overlay network by maintaining and optimizing an overlay network routing information
- Multicast routing
    - Build multicast tree, perform routing (by the host list)
    - Communicate with neighbors on overlay network
    - Manage the information of neighbors

The SIP-ALM middleware constructs an overlay structure among participating Mobile Nodes (MNs) in a self-organizing and fully distributed manner. MNs gather information of network

topology, builds application layer multicast groups and transfers application data to all multicast group members. Thus we divide the states of the MNs in SIP-ALM as Non-SIP node, SIP node and SIP-ALM node, as shown in Figure 1-4.

The Non-SIP node is the kind of mobile nodes, which does not have the SIP functionality or does not start the functionality yet. SIP nodes are those nodes, which started the SIP-ALM middleware. At this point, SIP nodes create or join in a SIP meshed network, but do not attend any multicast group yet.

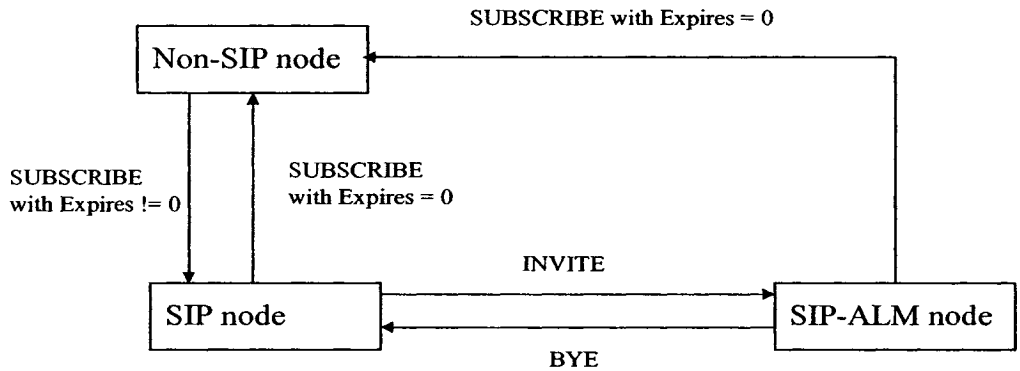

Figure 1-4 State diagram of the mobile nodes in SIP-ALM.

SIP-ALM node (a SIP-ALM group member) shows that the mobile node is part of some SIP-ALM group. Thus, it receives and sends data as it would in an IP multicast session; in addition, it also forwards data to designated adjacent neighbors. Data eventually reaches all group members through this relaying process, by the assumption that all members are cooperative. In our access control module, the assumption that a group does not contain any malicious member who intentionally blocks the data flows, is reasonable for our targeted applications such as video conferencing, web replication, etc. Besides forwarding data on the data plane, a group member may also monitor the performance of unicast paths to and from a

subset of other group members. This may be achieved by periodically sending probes to these members and measuring an application level performance metric; in the current implementation the round trip response delay.

Furthermore, it is expected that this mechanism would not be used in applications where the frequency of reportable events is excessively rapid. A SIP network is generally going to be provisioned for a reasonable signaling volume; sending a notification every time a MN's connection changes by one hundredth of a second could easily overload such a network [32].

Maintaining a routing tree for the purpose of multicasting packets when the underlying topology changes frequently can incur substantial control traffic [15]. The mesh-based approach is motivated by the need to support multi-source applications. Single-shared trees are not optimized for the individual source and are susceptible to a central point of failure. An alternative to constructing shared trees is explicitly constructing multiple overlay trees, one tree for each source. However, this approach needs to deal with the overhead of maintaining and optimizing multiple overlays [33].

We decide to use core-based multicast group management to control the requests to be sent toward the core instead of being flooded to the entire network. Also in MANET, frequent topology change and dynamic group membership often lead to substantial signaling overhead in maintaining the global multicast session state information. When replicated unicast is used for signaling transport, the number of messages between two SIP MNs is in accordance with the number of recipients that are accessible over the second MN. In contrast, using multicast only a single SIP message is sent between two SIP MNs.

The ALM group leader handles ALM group member registration and maintains the multicast

group. In order to achieve the latter, the leader performs the important functions of ensuring connectivity among group members of this multicast group when members join and/or leave the session and when network or host failures occur.

1.8 Organization of the thesis

The thesis is organized into six chapters as follows:

- Chapters 2 to 4 describe how to realize our proposed SIP-based application layer multicast framework.
- In Chapters 2 and 4, we detail the procedures of using SIP SUBSCRIBE, NOTIFY, INVITE and BYE methods to setup and maintain the overlay meshed network and multicast groups respectively.
- In Chapter 3, we introduce the lists of node information, which are used in our implementation to handle the membership transition in the network.
- Chapter 5 is concerned with detailing the requirements to successfully implement a SIP ALM simulation.
- Finally, Chapter 6 concludes the discussion with some insight for future work.

Chapter 2 The SIP-based Overlay Mesh Network

2.1 Introduction

SIP is a text-based client-server signaling protocol used to initiate, maintain and terminate communication sessions. It also offers a discovery capability. If a user wants to initiate a session with another user, SIP must discover the current host(s) at which the destination user is reachable. This discovery process is frequently accomplished by SIP network elements such as proxy servers and redirect servers which are responsible for receiving a request, determining where to send it based on knowledge of the location of the user, and then sending it there. To do this, SIP network elements consult an abstract service known as a location service, which provides address bindings for a particular domain [17].

Normally, registration creates bindings in a location service for a particular domain that associates an address-of-record URI with one or more contact addresses. A proxy server that is responsible for routing requests for that domain then typically consults this location service. Thus, when a proxy for that domain receives a request for Request-URI to match the address-of-record, the proxy will forward the request to the contact addresses registered to that address-of-record.

However since MANET is an infrastructure-less network without centralized administration structure, there are no pre-defined proxy servers and redirect servers. The discovery process needs to be done in a different way.

Because MANET is on-the-fly network, the binding of a location service for contact addresses is only important for the neighboring nodes. Since the MNs in MANET could move freely, the most important thing for routing request in MANET is to store the routing information, record the neighbors for each MN. We propose the saving of these bindings in the host lists of the neighboring mobile nodes (MNs), as discussed in Chapter 3. Therefore each MN knows how to route the request. When a MN receives a request, it checks its host lists for the shortest route – the route with the smallest number of hops – to the destination.

As we mentioned in Section 1.7, our framework allows users to discover each other, to establish a meshed MANET network and to organize Application Layer Multicast (ALM) groups in that MANET network. It does not require any particular feature on participating devices except the implementation of our middleware as SIP User Agent (UA). Registrar, Proxy and Redirect servers are not required in our approach.

The SIP-ALM middleware framework we propose has two main functions: overlay meshed network construction and multicast routing. We discuss the overlay meshed network construction in the following sections. The multicast routing function will be explained in Chapter 4. In Chapter 3, we introduce the host lists used for managing node information.

2.2 Network Architecture

In our proposal, one of our goals is to create an effective middleware system. Thus, we need to choose a proper network architecture to achieve that. As displayed in Figure 2-1, there are several alternative network architectures, such as full mesh, tree and clustering. We will introduce them briefly and explain our choice for our SIP-ALM structures.

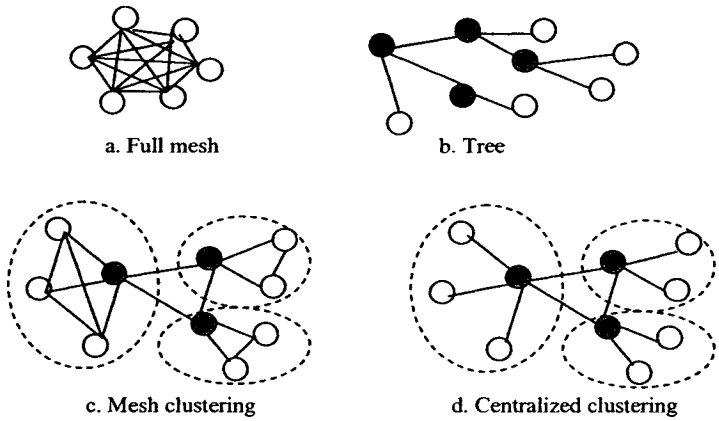

Figure 2-1 Network Architectures

2.2.1 Full Mesh

As shown in Figure 2-1a, full mesh is a fully distributed architecture. There is no control point. Nodes have the same responsibilities and they are directly connected to each other. Full mesh directly reflects the peer-to-peer communication idea. It is applicable to some small-scaled cases where there is no control node available.

2.2.2 Tree

The typical tree structure includes a root, some branches and some leaves. An example of tree structure is shown in Figure 2-1b.The root is the ancestor of all the leaves. The leaves refer to all the nodes except the root. Each leaf has a father and may have several sons. The branches are the connections between nodes.

One of the advantages of this architecture is that information can be efficiently propagated through it. A tree algorithm used in ad hoc networks is presented in [34]. A new node can easily join in just by creating a connection with any node in the tree. Through a learning

procedure, all the ancestors will have the knowledge of the new node. This is done by propagating information back to the root through the branches. "Node-leaving" is also not a big problem. When any father node leaves, it just "delivers" its sons to its father. Another advantage of this architecture is that it can be scalable. There is no limit in the growth of a tree. In addition, there is much less connections between nodes than those found in full mesh. However, the tree is an asymmetric architecture. The root plays a critical role for maintaining the tree. It is subject to heavy burdens if the tree is big. It is hard to determine the most powerful node as a root because in MANET, newly joined node may be more powerful than the pre-existing one. In addition, it is difficult to reorganize the whole tree when a new node joins and the old root cannot handle the tree anymore.

2.2.3 Clustering

Clustering, shown in Figure 2-1c and d, is a kind of architecture that divides a set of nodes into different clusters and provides connections between them. Each cluster has a cluster head, which is responsible for maintaining a list of cluster members and also connecting to other clusters. Clustering is often used in communication networks [35] to handle the issues of distributed computing. Similar to the tree, clustering can be easily scaled. It organizes large number of nodes in small clusters and each cluster is self-governed. Clustering also provides possibility for efficient information propagation. Since the cluster heads are all connected, they can propagate information to each other. All cluster members can then get the information from their cluster heads.

Compared with the tree architecture, clustering is more symmetric. That is, instead of

maintaining the table of all nodes, the burden of the root-node is shared by several cluster heads. If new nodes are added and the cluster head is no longer capable of maintaining them, it just splits the cluster into two smaller clusters. There is then the possibility to select the most powerful nodes as cluster heads.

In our opinion, two possible types of clustering: centralized clustering and mesh clustering can be used in our signaling system. Clusters in centralized clustering are organized in a centralized manner. Heads are the center points of each cluster. In contrast, clusters in mesh clustering are organized in a full mesh manner. Nodes have full mesh connections with each other.

If we compare the two, the meshed one has a better fault tolerance while the centralized one needs less signaling connections. Because of the mobility of the mobile nodes in MANET, we use the mesh clustering architecture in our approach.

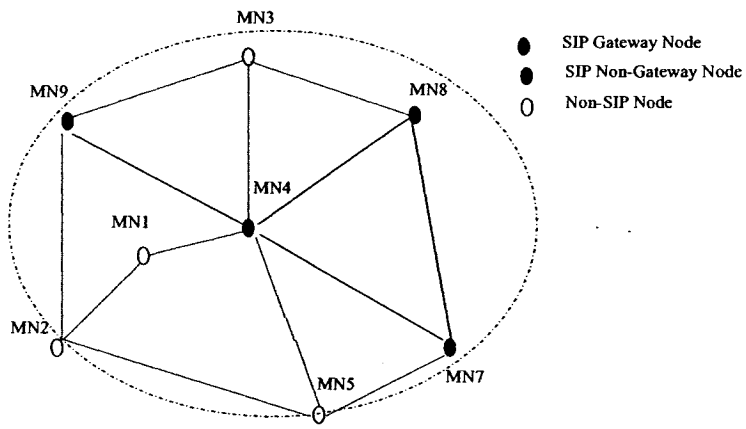

Figure 2-2 An Example of the Network Structure in Our Framework

This kind of hierarchical network structure is used to reduce the amount of the transferred routing information. Here, cluster header is called as gateway node, while cluster members are called as non-gateway nodes. In our approach, gateway nodes are SIP gateway nodes.

Similarly, non-gateway nodes are SIP non-gateway nodes. Figure 2-2 displays an example of the network structure used in our proposed framework.

2.3 The gateway node decision process

In SIP, every request is transferred in a request/response model, that is, whenever there is a request sent out, a response would be sent back to confirm this transmission. Since non-gateway nodes and gateway-node are always one-hop neighbors, it is not required for the non-gateway nodes to send out all its neighbor information.

In our approach, once becoming operational in a SIP mesh, every MN initially sets itself to be a gateway node. Then it broadcasts a request to join its one-hop neighbors. If no response is received, it is the gateway node. Otherwise, it changes itself to non-gateway node and sets the MN that has sent the correspondent response message as its source gateway node. If two MNs respond, the one arrived first is its source gateway node.

The gateway selection process for each node existing in the meshed network begins when there is a change in neighborhood. Each node will decide whether it should still be serving as a gateway node, or become a non-gateway node by checking its members with its neighbors. We will discuss the process in details in next section.

2.4 Creating an overlay-meshed network

The function of overlay meshed network construction is subdivided into the following [22]:

- Maintain and optimize an overlay meshed network including components;
- Communicate with other members;

- Manage the information and host lists of members.

RFC 3261, the core SIP specification, defines that a SIP request consists of a request line, header fields and a message body. The header fields contain information about call services, addresses and protocol features. The message body may contain anything. The core SIP also defines six different method types, including INVITE, ACK, CANCEL, OPTIONS, REGISTER and BYE. Due to the extension of the usage in SIP, a set of SIP extensions are defined, for example, SUBSCRIBE and NOTIFY are defined in RFC 3265 [32]; INFO is defined in RFC 2976 [36]; PRACK is defined in RFC 3262 [37]; UPDATE in RFC 3311 [38]; MESSAGE in RFC 3428 [39] and REFER in RFC 3515 [40].

Here we describe our choices for the SIP methods to achieve those functions in our framework.

2.4.1 The chosen SIP methods

In SIP scheme, a SIP UA sends REGISTER request as the first step to register to a network. Generally REGISTER method is used to register to the service and verify the identities of all the elements in the network [17]. In infrastructure networks, a REGISTER request can add a new binding between an address-of-record and one or more contact addresses. A client can also remove previous bindings or query to determine which bindings are currently in place for an address-of-record. So the most important function of REGISTER is to bind the current location with the user's URI. But because MANET is only a temporary, infrastructure-less network, this function may not be as useful as in the infrastructure network.

On the other side, SUBSCRIBE/NOTIFY methods in SIP are designed for event notification

to keep subscribed information updated. SIP UAs can subscribe to a remote SIP server and request event notification from it. In case of an event, remote server will alert the MNs that an event has occurred. In this way, SUBSCRIBE/NOTIFY requests are used to keep updated information in the network. Because in MANET, it is critical to keep the updated information of the MNs in the network, we use SUBSCRIBE/NOTIFY requests in our framework, instead of REGISTER request, to set up the overlay network.

Here we explain how SUBSCRIBE/NOTIFY requests are used to create and maintain a SIP meshed network.

2.4.2 Overview of Operation

The general concept of SUBSCRIBE/NOTIFY is that entities in the network can subscribe to resource or call state for various resources or calls in the network, and those entities (or entities acting on their behalf) can send notifications when those states change [32].

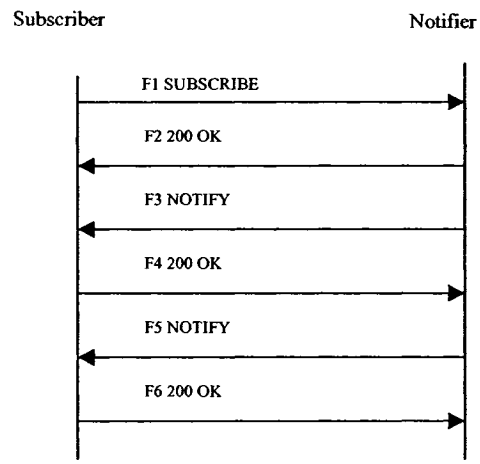

Figure 2-3 A typical flow of SUBSCRIBE/NOTIFY messages in SIP

A typical flow of messages is demonstrated in Figure 2-3. The transferred messages between a subscriber and a notifier are:

- F1 SUBSCRIBE – Request state subscription
- F2 200 OK – Acknowledge subscription
- F3, F5 NOTIFY – Return current state information
- F4, F6 200 OK – Acknowledge state information

Each node in our framework implements a neighbor discovery function started by sending out a SUBSCRIBE message. The meshed network members periodically exchange host information with their neighbors in a non-flooding manner so that the virtual meshed network may constantly adapt to the changes in the underlying network topology. Therefore, by looking at its own host lists, a member gets a view of the entire topology. This information will be used to build multicast trees in ALM groups. We will discuss it in Chapter 4.

Usually multicast registrations are addressed to the well-known "all SIP servers" multicast address "sip.mcast.net". SIP MNs listen to that address and use it to become aware of the location of other local users. This is appropriate for the MANET use. When a new SIP MN wants to find other mobile nodes around, it sends SUBSCRIBE request with some well-known multicast address. (In our implementation, we add "MN_user" as part of the Request-URI.) When the neighboring mobile nodes, listening at the default 5060 port, receive such a request, they save node information of the sender of this SUBSCRIBE request. They search the host lists for current stored information. If the relevant record does not exist, it will be created. Then the MN returns a 200 (OK) response. The response contains *Contact* header field values enumerating the URI of the responding MN. Upon receiving the 200 OK response, the initiating MN stores the URI of the responding MN, as its one-hop neighbors. The initiating MN waits to receive the corresponding NOTIFY message, it then updates its

own host lists. Therefore the original SIP MN becomes aware of the location of other local users. Now it becomes a member of the SIP meshed network.

In general, registration records in host lists are soft state and expire unless refreshed, but they may also be explicitly removed. A MN requests the immediate removal from the meshed network by specifying an expiration interval of "0" in a SUBSCRIBE request. We will describe the procedure of using SUBSCRIBE/NOTIFY to manage the membership of SIP meshed network in more details in the following sections.

Generally SIP itself does not provide services. But it makes available to other protocols and applications a set of primitives useful to implement services such as service location and discovery platform. The "Event" header [41] added to the headers of SUBSCRIBE/NOTIFY messages could be used to define event packages or services. In this way, the list of users reported to the subscriber represents the complete user list. For example, the "Event" header could be as follows:

Event: conference;recurse;type="membership,general"

Upon receiving the SUBSCRIBE message sent out on a SIP overlay network by a new node, each receiving node performs service matching, contained in "Event" header.

Therefore, if we want to have the function of service discovery, we use the "Event" header as part of the subscription information. Thus, a SIP meshed network is created among the MNs supporting the same event packages or services. It is done in the following way:

- A SIP user agent has the feature of organizing, attending and leaving MANET. It sends out a SUBSCRIBE request to the mobile nodes in its neighborhood when needed.

- The neighboring nodes listens to such SUBSCRIBE request. If they also have such feature, they store its information into their host list and reply with NOTIFY response with its own host lists information.

Because in this thesis, we only propose to provide a basic middleware framework, we do not discuss in detail the "Event" header, but only use the "presence" and " leaving_SIP_network" values of "Event" header for the purpose of managing a SIP meshed network.

2.4.3 The SUBSCRIBE method

In brief, when a SIP mobile User Agent wants to join an ad-hoc network, it generates a SUBSCRIBE request and simply broadcasts it to search for the participating mobile nodes. All existing SIP MNs should listen to a given port (5060, by default) and use it to receive the SIP messages.

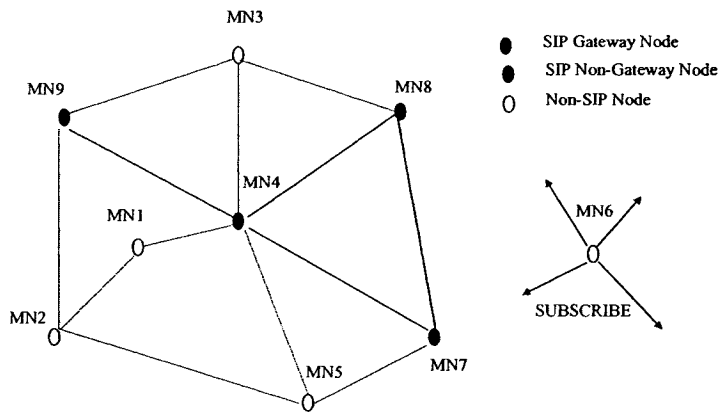

Figure 2-4 Joining of a new node (MN6) into a SIP meshed network

For example, as illustrated in Figure 2-4, when a SIP mobile User Agent (MN6) wants to join the SIP overlay network, it simply broadcasts a SUBSCRIBE request to search for the nearest participating mobile nodes with "Expire" header that includes the length of the valid

subscription.

The generated SUBSCRIBE request is shown in Figure 2-5. In the headers of SIP SUBSCRIBE requests, a SIP request is routed according to its Request-URI. We use "MN_user" here as a well-known parameter. The receivers of such request resolve it as a request for the node information in the overlay network.

```
SUBSCRIBE sip:MN_user@example6.com SIP/2.0
Call-ID: 1537778812@example6.com
CSeq: 1537778812 SUBSCRIBE
From: <sip:S6@example6.com:5070>;tag=f3433
To: <sip:MN_user@example6.com:5060>;tag=t1234
Via: SIP/2.0/UDP 132.205.46.227:5070;branch=z9hG4bKc34225244c03edd0a7c90f94e738137f
Max-Forwards: 1
Expires: 3600
Event:  presence
Require: ad_hoc_list
Accept: application/cpim-pidf+xml
Contact: <sip:S6@example6.com:5070>
Content-Type: application/adrl+xml
Content-Length: 0
```

Figure 2-5 An example of SUBSCRIBE request

A MN only needs to ask for the node information from the SIP MNs around it. Therefore, "Max-Forwards" is set to 1, meaning this request is only for the one-hop neighbors. Other SIP MNs' information is obtained from the replied NOTIFY messages, as described later.

"Event" header specifies the type of event this SUBSCRIBE request subscribes to. Because this request is used to join a SIP meshed network, "presence" is used in this case.

"Expires" header defines the duration of the subscription. For registering in a SIP meshed network, default value "3600" is used.

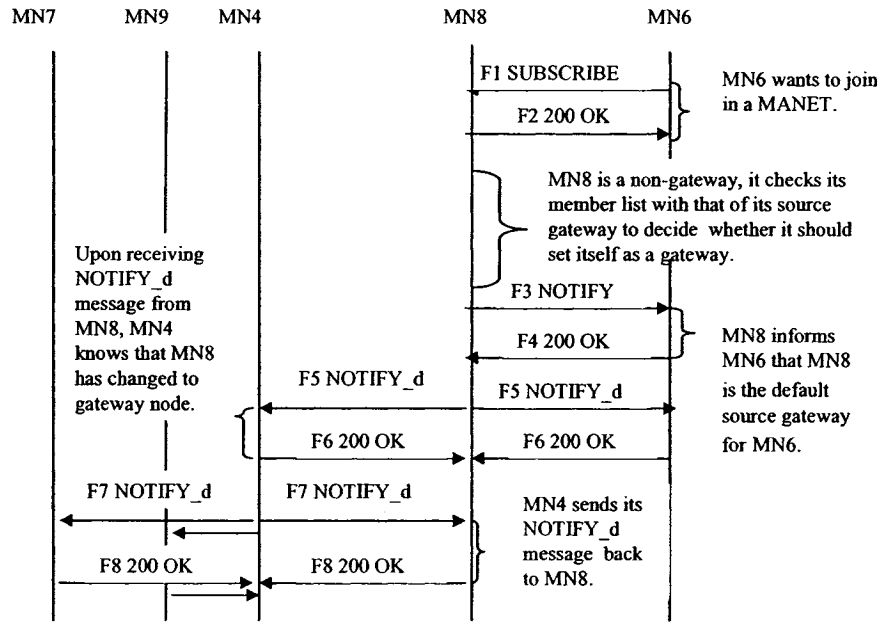

Figure 2-6 Message sequence for peer discovery and gateway node decision

Figure 2-6 describes the sequence diagram for peer discovery and gateway node decision process. Here, MN6 is the new MN that wants to join the network, while MN9, MN8 and MN7 are the participating non-gateway nodes. MN8 receives the SUBSCRIBE request sent by MN6. Thus no SIP response is sent back by any other nodes except MN8. Then MN8 compares its one-hop neighbors with those of MN4, which is MN8's source gateway node. Because MN8 and not MN4 is able to connect to MN6, MN8 assigns itself as a gateway node. MN8 then responds with NOTIFY message back to MN6. MN8 informs its source gateway, MN4, about that change by sending differential NOTIFY messages. The difference between NOTIFY and differential NOTIFY messages will be discussed in the following sections.

After MN6 joined the mesh network, MN8 becomes a gateway node. Thus there are two clusters, as shown in Figure 2-7. According to clustering approach, MN8 will communicate

with MN7 via MN4.

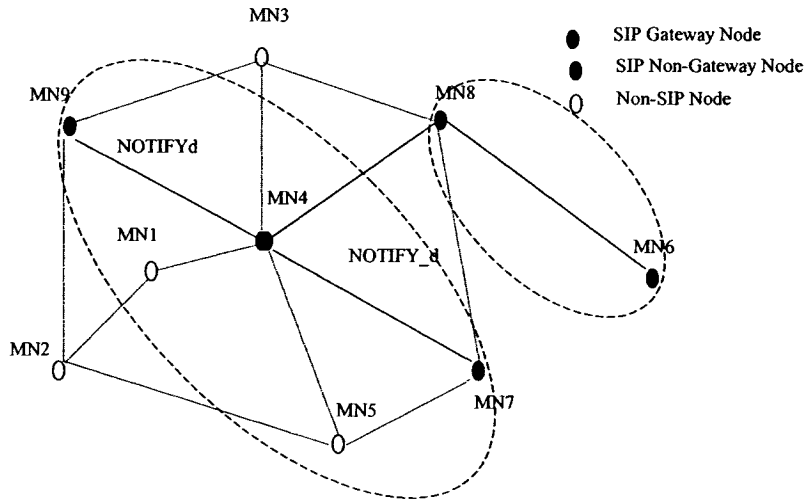

Figure 2-7 The network structure after MN6 joining

In our approach, each mobile node in the SIP meshed network contains a profile list of its own information. It also contains host lists that include the information about other members and active ALM groups of the network.

Participating nodes on receiving SUBSCRIBE requests store the new node's information, including the identity of the new node, in its SIP-Meshed Network Node List or Non-Gateway Node Member List.

The SUBSCRIBE request is confirmed with a 200-class response to indicate that the subscription has been accepted, and that a NOTIFY will be sent immediately. Non-200 class responses indicate that no subscription or dialog has been created and no subsequent NOTIFY message will be sent. We assume that all the MNs in the meshed network are willing to participate fully in the protocols of the network; we do not focus on non-200 responses in our study currently.

200-class responses to SUBSCRIBE requests will not generally contain any useful information beyond subscription duration. It is obligatory for 200-class responses to SUBSCRIBE requests contain an "Expires" header [32]. Their primary purpose is to serve as a reliability mechanism. The "Expires" header in a 200-class response to SUBSCRIBE indicates the actual duration for which the subscription will remain active (unless refreshed). In our proposed network, the NOTIFY messages are important to let other MNs know its status. In our framework, we decide that, unless the MN leaves the network, the NOTIFY message will keep being sent out.

2.4.4 The NOTIFY method

As shown in Figure 2-6, upon receiving a SUBSCRIBE request, a participating MN responds with a 200 OK message. The participating MN then immediately constructs and sends a NOTIFY request to the subscriber.

Information about the available nodes, routing information and the active ALM group information is included in the content of this NOTIFY message. This information will be used to build multicast trees later. Upon receiving the NOTIFY message, the attached information will be stored in the corresponding host lists. By doing so, the joining node, for example, MN6 in Figure 2-4, gets and stores other MNs' information and becomes a member of the virtual meshed, overlay network. The structure of generated NOTIFY request from MN8 is shown in Figure 2-8. <ProfileList> sector contains the information of the NOTIFY sender. <HostList> sector contains all the nodes information in the mesh network.

```
NOTIFY sip:S6@example6.com:5070 SIP/2.0
Call-ID: 1537778812@example6.com
CSeq: 1537778812 NOTIFY
Max-Forwards: 1
Event:  presence
Require: ad_hoc_list
Accept: application/cpim-pidf+xml
Contact: <sip:S6@example6.com:5070>
To: <sip:S6@example6.com:5070>;tag=f3433
From: <sip:S8@example8.com>;tag=f1222
Via: SIP/2.0/UDP 132.205.46.227;branch=z9hG4bK0260c8e68ce8c5235c03b84dfd1ce005
Content-Type: ad_hoc_list/adrl+xml
Content-Length: 783

<ProfileList>
    <node  uri="S8@example8.com"  ip="10.20.20.8"  gateway="S8@example8.com"  Event-ID="null"
GroupMembers="null" leader="false" changes="Y" time="null"/>
</ProfileList>

<HostList name = "S8@example8.com">
    <node  uri="S9@example9.com"  gateway="false"  Event-ID="null"  alive="true"  changes="N"
from="S4@example4.com" hops="2" time="19:1:56.356"/>
    <node  uri="S6@example6.com"  gateway="false"  Event-ID="null"  alive="true"  changes="Y"
from="S6@example6.com" hops="1" time="19:2:5.372"/>
    <node  uri="S7@example7.com"  gateway="false"  Event-ID="null"  alive="true"  changes="N"
from="S7@example7.com" hops="1" time="19:1:56.841"/>
    <node  uri="S4@example4.com"  gateway="true"  Event-ID="null"  alive="true"  changes="N"
from="S4@example4.com" hops="1" time="19:1:56.356"/>
</HostList>
```

Figure 2-8 An example of NOTIFY request from MN8 to MN6

It is said [32] that if all subsequent NOTIFY messages that correspond to the SUBSCRIBE message contain the same "Call-ID", a "To" header "tag" parameter which matches the "From" header "tag" parameter of the SUBSCRIBE, and the same "Event" header field, but that do not match the dialog would be rejected with a 481 response. But because we use NOTIFY message to find the neighbors' information, we would like to know which MNs

respond to that SUBSCRIBE request. Thus, we use the responding MN's URI, instead of the predefined URI in the To/From field of the responding NOTIFY request.

The subscriber can expect to receive a NOTIFY message from each node that has processed a successful subscription or subscription refresh. A NOTIFY does not terminate its corresponding subscription; in other words, a single SUBSCRIBE request may trigger several NOTIFY requests.

Once the notification is deemed acceptable to the subscriber, the subscriber returns a 200 OK response to this NOTIFY message. Upon receiving such 200 OK message, the gateway node knows that this receiver is still its one-hop neighbors in the meshed network.

```
NOTIFY sip:MN_user@example8.com SIP/2.0
Call-ID: 285530902@example8.com
CSeq: 285530902 NOTIFY
From: <sip:S8@example8.com>;tag=f3433
To: <sip:S4@example4.com>;tag=t1234
Via: SIP/2.0/UDP 132.205.46.227;branch=z9hG4bK2a79df0f822766da5e53d8d610c4df34
Max-Forwards: 1
Event:  Presence
Require: ad_hoc_list
Accept: application/cpim-pidf+xml
Contact: <sip:S8@example8.com>
Content-Type: application/adrl+xml
Content-Length: 181

<HostList uri="S8@example8.com">
<node uri="S6@example6.com" gateway="false" Event-ID="null" alive="true" changes=" "
from="S6@example6.com" hops="1" time="19:2:5.716"/>
</HostList>
```

Figure 2-9 An example of NOTIFY_d request (when MN6 joins the meshed network)

The periodic NOTIFY messages (called NOTIFY_d message in short) only contain the differential host information. Ideally, in a stable network where the topology and membership

remain unchanged, only the first NOTIFY message needs to contain a content body with the node information and all subsequent NOTIFY_d messages would contain no body at all. Reporting only the differences significantly reduces the message size. The example of NOTIFY_d message from MN8 is shown in Figure 2-9.

2.5 Leaving an overlay meshed network

Generally speaking, when a node wants to leave a SIP meshed network, it sends out a "leave" message. The directly connected mobile nodes make the corresponding operations, in order to adapt to the membership change and the network topology change.

In our framework, the periodic NOTIFY message is used as a secondary mechanism to detect an absent peer. A SUBSCRIBE message with Expire equals to 0 is defined as the preferred way for a MN to leave the network.

We explain the different scenarios for different leaving behaviors in Section 2.5.1 and Section 2.5.2.

2.5.1 Graceful leaving

When a member wants to leave a mesh gracefully, it notifies its neighbors and the information is propagated to the rest of the mesh. In RFC 3265 [32], un-subscription is handled in the same way as refreshing of a subscription, with the "Expires" header set to "0". A successful un-subscription also triggers a final NOTIFY message with the "Subscription-state" set as "terminated". However in our case, this final NOTIFY message is not needed, since un-subscription is only required when the mobile node turns off and shuts down the application.

Thus in our framework, a natural way for a MN to leave the proposed network gracefully is a SUBSCRIBE with the "Expires" of 0 to unsubscribe from the neighboring MNs.

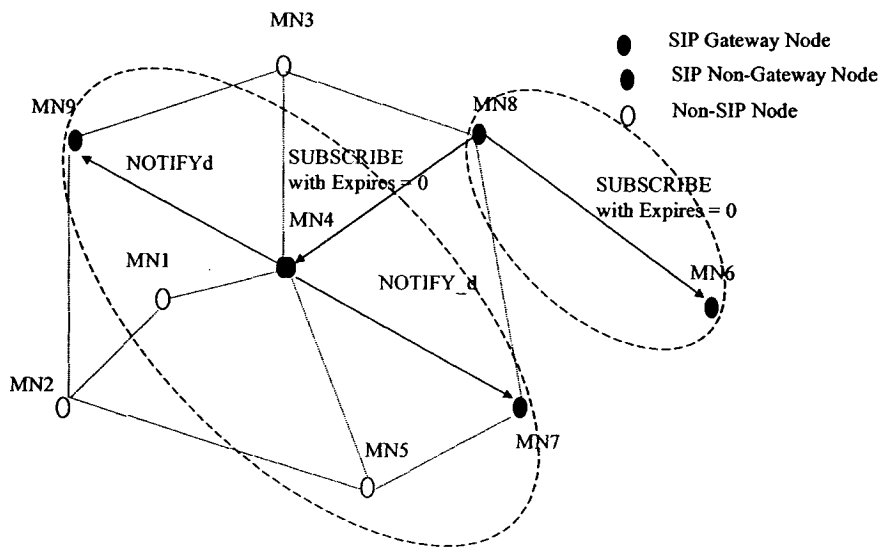

Figure 2-10 Leaving of a mobile node (MN8) from a SIP meshed network

As shown in Figure 2-10, a MN that wants to leave an overlay meshed network sends out a SUBSCRIBE request, with "Expire" equals to 0. It is a broadcast message sent from the departing MN to its neighboring MNs. The MN is then removed from the host list of neighboring nodes and is excluded from further forwarding computations. The departing MN removes all information associated with the connections in the meshed network. MN6 now needs to find its new gateway node.

Figure 2-11 shows an example of SUBSCRIBE request for a node leaving SIP meshed network. The SUBSCRIBE request contains the "Event" field set as "leaving_SIP_network" while the value of "Expires" set to "0".

Figure 2-12 shows an example of propagating the information about MN8's leaving, in this case, to other remaining nodes. MN7, the receiver of the NOTIFY_d message, needs to delete

the record of MN6 and MN8 from its host list.

```
SUBSCRIBE sip:MN_user@example8.com SIP/2.0
Call-ID: 1141673825@example8.com
CSeq: 1141673825 SUBSCRIBE
From: <sip:S8@example8.com:5070>;tag=f3433
To: <sip:MN_user@example8.com:5060>;tag=t1234
Via: SIP/2.0/UDP 132.205.46.227:5070;branch=z9hG4bKc9000b00d61397cf95f98e43ea6135d0
Max-Forwards: 1
Require: ad_hoc_list
Accept: application/cpim-pidf+xml
Contact: <sip:S8@example8.com:5070>
Content-Type: application/adrl+xml
Event:  leaving_SIP_Network
Expires: 0
Content-Length: 0
```

Figure 2-11 An example of SUBSCRIBE request for a node (MN8) leaving SIP meshed network

```
NOTIFY sip:MN_user@example4.com SIP/2.0
Call-ID: 238268084@example4.com
CSeq: 232112101 NOTIFY
From: <sip:S4@example4.com>;tag=f3433
To: <sip:S7@example7.com>;tag=t1234
Via: SIP/2.0/UDP 132.205.46.227;branch=z9hG4bK42e93f1878eea834326303480fb9b2a5
Max-Forwards: 1
Event:  Presence
Require: ad_hoc_list
Accept: application/cpim-pidf+xml
Contact: <sip:S4@example4.com>
Content-Type: application/adrl+xml
Content-Length: 329

<HostList uri="S4@example4.com">
  <node  uri="S6@example6.com"  gateway="false"  Event-ID="null"  alive="true"  changes="Delete"
from="S8@example8.com" hops="2" time="19:3:40.372"/>
  <node  uri="S8@example8.com"  gateway="true"  Event-ID="null"  alive="true"  changes="Delete"
from="S8@example8.com" hops="1" time="19:3:40.372"/>
</HostList>
```

Figure 2-12 An example of NOTIFY_d request (when MN8 leaves the meshed network) from MN4 to MN7

2.5.2 Detection of abrupt disconnection

An abrupt disconnection occurs when a MN moves out of the connection of any other node in the network. This is detected through MNs monitoring each other's status using a ping-timeout mechanism. NOTIFY_d messages are sent periodically and receivers respond with 200 OK messages. If a node in the host lists fails to send or respond NOTIFY_d messages in a predetermined period, it is assumed to have left the network ungracefully. Then the node will be removed from the host lists. It needs to broadcast the SUBSCRIBE request again to restore the connection to the network.

If a SIP gateway node has not sent out NOTIFY_d message or a SIP non-gateway node has not responded to the periodic NOTIFY_d messages in a pre-defined period, the other nodes consider the node as left. Thus the record is marked as "Delete", the change is propagated in the next NOTIFY_d message transmission, and finally the record is removed from its host list.

2.6 Maintaining an overlay meshed network

When a mobile node is a SIP gateway node and it has members around, it will keep sending NOTIFY differential messages to its one-hop neighbors periodically. All the network topology changes are propagated in the content of NOTIFY_d message.

2.7 Summary

In this chapter we proposed a mechanism of using SIP SUBSCRIBE and NOTIFY requests to create, maintain and leave a SIP meshed network in MANET. The detailed message formats

for SUBSCRIBE and NOTIFY messages are shown for MNs to join and leave the proposed network. The idea of gateway nodes and non-gateway nodes is also introduced to reduce the maintenance traffic. The following chapter will describe the mentioned host lists in detail.

Chapter 3 Node Information Management

3.1 Introduction

In an overlay network, node connectivity is mapped by a logical connective graph, and not by a physical connective graph. The former is a sub-graph of the latter. Both graphs share the same nodes, but the logical graph is missing some of the links, as illustrated in Figure 1-1.

Since there is no fixed infrastructure in MANET, all mobile nodes are required to compute, maintain and store routing information [42]. A node's location is distributed throughout the network and maintained in nodes, which act as location servers for other nodes. A source node uses this location information to forward packets to a destination node. Intermediate nodes make local decisions to forward those packets.

To store the network and multicast group information, we propose the use of host lists in our framework. To handle mobile node failures and a node leaving a group, these host lists are refreshed periodically. The basic host lists in our approach are:

- The Profile List, which is used by every node in the meshed network for storing its own information, such as, the group membership of the multicast groups it is in and the connected source gateway;
- The Open Neighbor Set, which is used to store the information of other nodes in the meshed network;

In our framework, each node maintains the routing information, and refreshes it at regular

intervals. Because of the low bandwidth, convergence to new, stable routes after dynamic changes in MANET may be slow and expensive. For updating routing tables, each node transmits control packets periodically, which constitute an inefficient use of network capacity. To solve this problem, organizing members into hierarchies of clusters could achieve better scaling properties. A concern with hierarchy-based approaches is that they complicate group management, and need to rely on external nodes to simplify failure recovery [33]. Jie Wu [43] proposed Dominating-Set-Based Routing to localize the routing information for adapting to changes such as quick host movement. In this scheme, the Mobile Nodes (MNs) are divided as gateway nodes and non-gateway nodes. For gateway nodes, the SIP-Meshed Network Node List is divided into the two following lists:

- The Non-Gateway Node Member List is used by hosts to store the information of its members, which are the non-gateway nodes in its neighborhood, in the network;
- The Gateway Node Member List is used to store the information of other gateway nodes in the network.

Host lists are explained in more details in the following section.

3.2 The Host Lists

We use an example of SIP-ALM graph. To make it simply, we re-draw Figure 2-7 as Figure 3-1, to describe the host lists. In this meshed network, MN4 and MN8 are working as gateway nodes; MN6, MN7 and MN9 are working as non-gateway nodes.

3.2.1Profile List

The profile list is stored in every mobile node. It contains its own information and the

information of the attending multicast groups. It consists of information of active multicast groups joined by this host. For example, the profile list of MN6 is shown in Table 3-1:

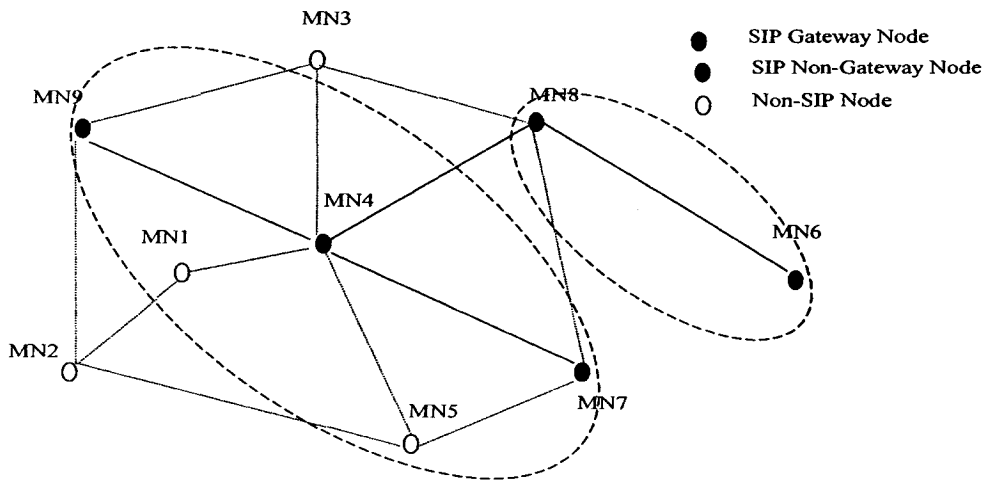

Figure 3-1 An example of SIP-ALM graph

Table 3-1 MN6's Profile List

Host Name (URI)	IP Address	Gateway	Event -ID	Group members	Leader
S6@example6.com	10.20.20.206	S8@example8.com			

Each SIP MN gains a SIP identity, which is a special type of Uniform Resource Identifier (URI), called SIP URI, the "Host Name" field stores the MN's SIP URI. "IP Address" may also be used to address that MN.

If the MN is a gateway node, the MN's host name will be filled into the "Gateway" field. If not, the host name of its source gateway node will be stored in this field.

If the MN is an ALM group member, it will save the attending group's identity in the "Event-ID" field. The "Group members" field will list the host names of all the group members in that group. The "Leader" field is marked as "N", if the node is not the leader of the ALM group. If the MN is the leader of an ALM group, in addition to saving the group's

identity and its group members in the corresponding fields, it will mark "Y" in the "Leader" field. If the MN does not attend any ALM group, the last three fields will be empty. We will illustrate these three fields in Chapter 4.

3.2.2 Non-Gateway Nodes Lists

3.2.2.1 SIP-Meshed Network Node List

The SIP-Meshed Network Node List only exists in non-gateway nodes. It contains all the mobile nodes information in the SIP-meshed network. The one saved in MN6 is shown in Table 3-2. It is created after receiving NOTIFY message from its source gateway, MN8.

Table 3-2 MN6's SIP-Meshed Network Node List

Host Name (URI)	From	Gateway	Event-ID	Hops	Changes
S9@example9.com	S8@example8.com	N		3	
S8@example8.com	S8@example8.com	Y		1	
S4@example4.com	S8@example8.com	Y		2	
S7@example7.com	S8@example8.com	N		3	

The "Event-ID" field is only filled for the record of the leader of that multicast group, instead of marking it in all the group members. We will discuss it in detail in Chapter 4.

The neighbor information is updated periodically for its successful communications in MANET. As we know, frequent topology change and dynamic group membership could lead to substantial signaling overhead in maintaining such routing information. Except for the first registration information, we only transfer the differential routing information in the periodical NOTIFY_d messages in our approach. Thus we add the "Change" field to show whether a change has been made in that record during the interval.

Generally, NOTIFY_d message is sent periodically. During the time, if there is any change in that record, the Changes field will be set to "Y"; and when that MN leaves the mesh, it will be

marked as "Delete". So only the record with Changes field as "Y" or "Delete" will be sent out in the next NOTIFY_d message.

3.2.3 Gateway Nodes Lists

3.2.3.1 Non-Gateway Node Member List

The Non-Gateway Node Member List is stored in gateway nodes only. It contains the information of that gateway node's members. It is the list of non-gateway nodes in its neighborhood. For example, the Non-Gateway Node Member List s of MN4 and MN8 are shown in Table 3-3 and Table 3-4 respectively:

Table 3-3 Gateway MN4's Non-Gateway Node Member List

Host Name (URI)	From	Event-ID	Changes
S9@example9.com	S9@example9.com		
S7@example7.com	S7@example7.com		

Table 3-4 Gateway MN8's Non-Gateway Node Member List

Host Name (URI)	From	Event-ID	Changes
S6@example6.com	S6@example6.com		

3.2.3.2 Gateway Node Member List

The Gateway Node Member List is also stored in gateway nodes. It contains the information of other gateway nodes in the network. The Gateway Node Member List s of MN4 and MN8 are shown in Table 3-5 and 3-6:

Table 3-5 Gateway MN4's Gateway Node Member List

Host Name (URI)	Members	Event-ID	Changes
S8@example8.com	S6@example6.com		

In the next section, we discuss how these host lists are used to compute, maintain and store routing information in the meshed network.

Table 3-6 Gateway MN8's Gateway Node Member List

Host Name (URI)	Members	Event-ID	Changes
S4@example4.com	S7@example7.com S9@example9.com		

3.3 Membership Management

It is important to keep the routing information updated in MANET. A location update mechanism is required to exchange host information within a group of nodes. At the same time, the location updates also serve as group membership refreshments to detect dead or unreachable nodes. A node will be excluded from a source node's destination list when its membership is not refreshed over a certain time period [44].

A hybrid approach to location/membership update may be used, which includes in-band update and periodic update. In-band update takes place when a node has data packets to send. Consequently, other members of the multicast group will learn the sender's location. Periodic update kicks in when a node has no data packets to send for an extended period of time. In that case it has to send out a special null packet to inform other nodes of its current location [44]. Since other nodes use the host list information to construct the packet distribution tree, up-to-date host list information will improve the optimality of the trees. We only discuss signaling part in this thesis, therefore we only describe periodic update here.

As our middleware framework is divided as overlay meshed network construction and multicast routing, we next describe the network membership management in Section 3.3.1. The multicast membership management will be discussed in Chapter 4.

3.3.1 Network Membership Management

As we mentioned earlier, in our meshed network, there are two kinds of mobile nodes: gateway nodes and non-gateway nodes. Different type of nodes contains different host lists. The idea is, to reduce the burden of gathering and maintaining routing information. In our framework, it leads to a more efficient use that only a subset of nodes sends the periodic beacon message to keep the node information updated. These nodes in this subset act as gateway nodes, which are used for routing by the rest of the nodes not in this subset [45].

Generally, if a meshed network member receives a SUBSCRIBE message from a new neighbor, the node information is added to the SIP-Meshed Network Node List (for non-gateway nodes) or the Non-Gateway Node Member List (for gateway nodes).

It is then decided whether its status of gateway nodes or non-gateway nodes should be changed. In our framework, only gateway nodes send NOTIFY_d messages periodically. Its members will use the 200 OK messages (which is mandatory in SIP for SIP requests) as its "Alive" messages.

New node receives a copy of the existing host information for the SIP-Meshed Network Node List (for non-gateway nodes) or the Non-Gateway Node Member List (for gateway nodes) via the NOTIFY message. After that, it will get updated node inforamtion from its source gateway by NOTIFY_d message.

In our approach, every node in the meshed network is gateway node initially. After receiving NOTIFY message, the new node changes its status to Non-gateway. After receiving a SIP SUBSCRIBE message from a MN that wants to join the mesh, the receiving node adds the

sending MN into its neighbor list. If the receiving node is a non-gateway node as illustrated in Figure 2-6, a Gateway Node Decision process is triggered. It compares its SIP-Meshed Network Node List (including all the 1-hop neighboring nodes) with that in its own gateway node. If there exist two unconnected neighbors, the node designates itself as a gateway node. Otherwise, it changes itself to a non-gateway node. The decision is made each time a MN joins or leaves the mesh.

A soft state approach is used to maintain the overlay network membership and associated routes. Thus, the host lists are refreshed from time to time to maintain up-to-date membership information by sending and receiving the differential routing information periodically to the surrounding peers and its members. This is done by periodically sending the SIP NOTIFY_d message with differential routing information out to neighbors.

3.4 Summary

In this chapter, we introduce the host lists used in our framework. Besides Profile List, non-gateway nodes also contain Open Neighbor Set, which stores all the nodes in the meshed network. For gateway nodes, the SIP-Meshed Network Node Listis divided into Gateway Node Member List and Non-Gateway Node Member List . The management of the network membership is also introduced here.

The following chapter will focus on the issues about creating and maintaining Application Layer Multicast group in MANET.

Chapter 4 ALM Group Management

4.1 Introduction

Through the combination of wireless communication ability and mobile devices, applications of self-organizing, infrastructure-free, mobile networks are being developed in many domains: educational, industrial, commercial and military. The most characteristic operation in these domains is multicast, where messages are sent from one node to multiple recipients [46].

Because a Mobile Ad Hoc network (MANET) is a collection of nodes forming a temporary network without the aid of any centralized administration, where each node communicates over wireless channels, moves freely and may join and leave the network at any time, full-meshed multicast groups are more suitable for providing reliable multicast data transfer. In full-meshed multicast groups, each participant has a signaling connection with all other participants.

SIP-ALM takes the centralized control approach to maintain the consistency and efficiency of multicast groups. This design choice is made for admission control and reduced overhead when a new member joins the ALM group. Thus, there is a leader for each group who is responsible for the maintenance of the group and the reception of new joining participants. The leader also sends the list of participants to all group members when there is any change in the ALM group.

The function of multicast routing is subdivided into the following [47]:

- To build multicast trees and perform routing;
- To communicate with neighbors on the overlay network;
- To store and manage the information of neighbors.

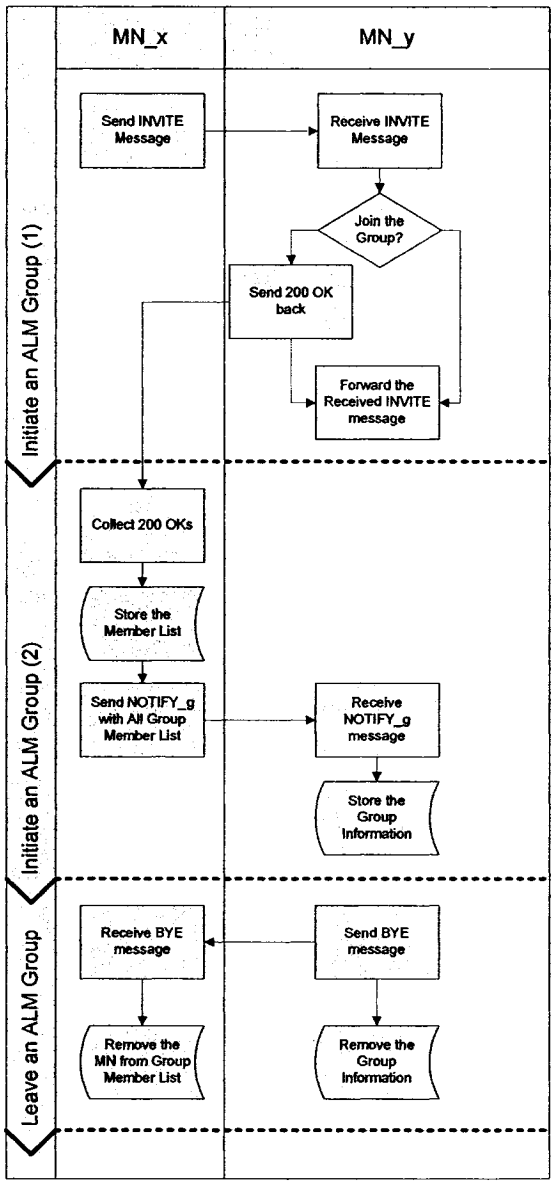

Figure 4-1 The flow chart of the management in an ALM group

As shown in Figure 4-1, in our proposal, the brief procedures of creating and maintaining an ALM group are:

- A MN multicasts an INVITE request to the known SIP MNs.
- In the headers of the INVITE request, a new field "Event-ID" is added to identify the ALM group it is initiating.
- If other SIP MNs in the MANET want to join the group, 200 OK response will be sent back, and the INVITE request will be forwarded to its neighbors. Otherwise, a MN only works as a router to forward such request.
- Upon collecting 200 OK response within a certain period, the leader will generate and send a NOTIFY message with the information of all the group members to its members.
- Such NOTIFY message is sent only when the group membership is changed, that is, a MN joins or leaves the group.
- If a MN wants to leave an ALM group, it will multicast a BYE message to all the group members.
- The abrupt disconnection of a MN is detected on the meshed network layer, whenever a group member notices some group member has left the network without notice, it multicasts the membership change to all the group members.

The leader of an ALM group is the MN that initiates the ALM group formation or another MN that replaces the initiator in case of the original one disconnected. To distinguish ALM groups, a Group-ID is assigned to each ALM group, when an initiation message is

constructed.

We explain the above procedures in more details in the following sections.

4.2 Initiating an ALM group

4.2.1 The INVITE and NOTIFY messages

When a meshed network member wants to initiate an ALM group, the MN needs to send an INVITE request to the MNs in the virtual, meshed MANET. It becomes the leader of the ALM group. Its function includes admission control for group members and to maintain the group membership. The initiated INVITE request asks each accepting MN to establish a multicast session.

In our framework, the initiating node has the knowledge of the network topology; hence to prevent flooding of messages in mobile ad-hoc network, it decides the multicast routes of transferring control messages based on the meshed network topology. That is, when a MN wants to send some data to multiple MNs, it finds the routes from the information in its host lists. If the source is a non-gateway node, it will send the multicast data to its 1-hop neighbors and its source gateway node. The gateway node will check its own host list and forwards the data to its neighbors and its gateway nodes. The process continues until data is reached to all the destinations.

For example, as shown in Figure 4-2, MN4 multicasts the INVITE request to initiate a multicast group to all the members in its host lists (MN6, MN7, MN8 and MN9, in this case). Because MN7, MN8 and MN9 are one-hop neighbor nodes of MN4, so MN4 sends out three copies of INVITE request. MN6 will receive the forwarded INVITE request from MN8.

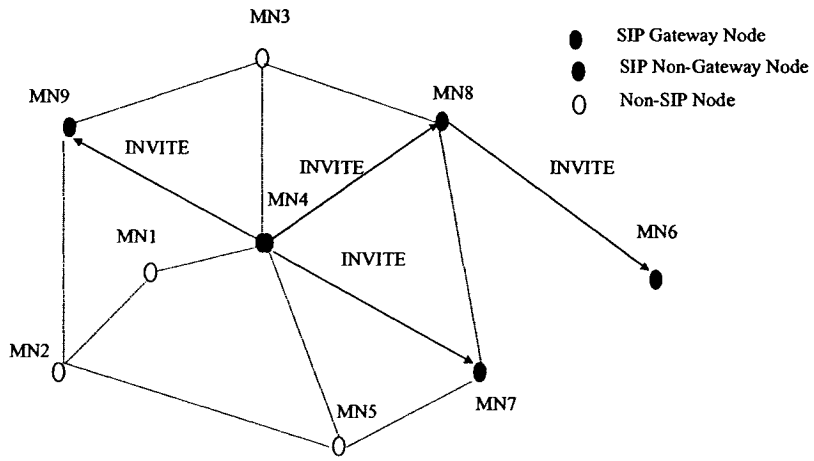

Figure 4-2 MN4 initiates an ALM group

INVITE sip:S4group@sip.manet.example4.com SIP/2.0

CSeq: 33 INVITE

From: <sip:S4@example4.com:5070>;tag=f3433

To: <sip:S4group@sip.manet.example4.com:5060>;tag=t1234

Via: SIP/2.0/UDP 132.205.46.231:5070;branch=z9hG4bKb6a85a70b229aa4601068a43ba12937f

Max-Forwards: 70

Contact: <sip:S4@example4.com:5070>

Forward-To: S8@example8.com S6@example6.com

Event-ID: 839872666@example4.com IM

Call-ID: 839872666@example4.com

Content-Length: 0

Figure 4-3 An example of INVITE request

The generated INVITE request to MN8 is shown in Fig 4-3. The Request-URI field is constructed using the predefined part: "group@sip.manet". For example, "S4group@sip.manet.example4.com" shows that this INVITE request is an ALM initiation request sent from MN4. When a MN receives an INVITE request with this kind of Request-URI, it deduces that this request is a multicast group initiation message and is for every MN in the network. If the MN wants to accept this request, it will reply 200 OK back to

the originator of this request. If there are other SIP URIs in Forward-To field, then it will add its URI in the Record-Route [17] field and forward this message to other nodes, which are listed in the Forward-To field. If it does not want to accept the request, it only forwards this message to other listed nodes and also adds its URI in the Record-Route field. Forward-To field [47] will be discussed shortly.

The main tasks of signaling (that is SIP in this case) are initiation, maintenance and termination of the communication. Signaling information is usually distributed to all participants in a multiparty communication. Because of its size, SIP signaling information should be distributed in the most efficient way.

A multi-recipient request [48] is proposed to contain SIP URIs of each request recipient in the message headers. The request is sent over unicast in Overlay Multicast network and do not require any support on the network layer, but only on the application layer.

The "Forward-To" header is proposed as a multi-recipient request for multicasting SIP messages to many recipients. In "Forward-To" header, the SIP URIs of all the recipients are listed. If the request for some of the SIP URIs should be routed to the same next hop server, the request is sent as a multi-recipient request with corresponding SIP URIs in the "Forward-To" header field. Thus on the same connection between two MNs, the same messages only traverse once. The signaling traffic would be reduced.

Thus, when a MN is asked to send a message to MN addresses, it looks in its local host lists to see if it has routes to those MNs. That is, when a MN receives a message, the detailed process is the following:

- Perform a host list lookup to determine the next hop for each of the destinations listed

in the header of the message.

- Partition the set of destinations based on their next hops.
- Replicate the packet so that there is one copy of the message for each of the next hops.
- Modify the list of destinations in each of the copies so that the list in the copy for a given next hop includes just the destinations that ought to be routed through that next hop.
- Send the modified copies of the message to the next hops.

The route is ordered from the next hop to the final destination MN. Because the routing information may become obsolete at any point, in order to complete the message delivery, the route discovery is done hop-by-hop. That is, each router checks its own host list to find the next hops for the destination peers and if the message is received by one destination peer, the address will be removed from the messages' destination list. Because the messages will go along the multicast tree and be checked at each hop, there should be no loop and recurrent messages generated. In the example shown in Figure 4-3, because the communication from MN4 to MN6 is via MN8 and MN8 belongs to the same route, the Forward-To header is: *Forward-To: S8@example8.com S6@example6.com.*

In this case, upon receiving the INVITE message, MN8 modifies the Forward-To field and forwards the message. As shown in Figure 4-4, each forwarded INVITE request also contains a record in the "Record-Route" field listing the address of each intermediate node through which this particular copy of the INVITE message has been forwarded.

When a node receives the INVITE message and wants to join, it returns a 200 OK message to

the initiator of the ALM group using the route listing in the Record-Route; when the initiator receives this 200 OK message, it caches this host in its profile list to be used in sending subsequent packets to this destination.

INVITE sip:S4group@sip.manet.example4.com SIP/2.0
CSeq: 33 INVITE
From: <sip:S4@example4.com:5070>;tag=f3433
To: <sip:S4group@sip.manet.example4.com:5060>;tag=t1234
Via: SIP/2.0/UDP 132.205.46.231:5070;branch=z9hG4bK13dc0102e05c5311e81817c49d00e25e
Contact: <sip:S4@example4.com:5070>
Event-ID: 839872666@example4.com IM
Call-ID: 839872666@example4.com
Record-Route: <sip:S8@example8.com:5600>
Max-Forwards: 69
Forward-To: S6@example6.com
Content-Length: 0

Figure 4-4 An example of forwarded INVITE request (by MN8)

In addition, this receiving node appends its own address to the route record in the INVITE message and propagates it by transmitting it as a local multicast packet (with the same Call-ID).

For the headers of the INVITE request we propose to add a new field called "Event-ID". This field contains the identity of the initiating ALM group for which this MN is the leader. The "Event-ID" represents an ALM multicast group assigned upon creation.

Because there is no centralized administration in MANET, it is not possible to acquire some assigned distinguished Event-ID from some certain server. Since the Call-ID of the SIP requests in SIP sessions is generally unique, in our approach, we use the Call-ID of the first INVITE request and the provided service type to construct the value in Event-ID field. For

the example shown in Figure 4-3, the Event-ID is: "*839872666@example4.com IM*", where "*839872666@example4.com*" is the Call-ID of the INVITE request sent by S4@example4.com to initiate an ALM group; "IM" is the service name provided by that group. The Event-ID value is used for routing group control messages and is used by MNs for sending "Join" requests. We will describe this more shortly.

4.2.2 Maintaining Group Membership

Membership of an ALM group is dynamic over the lifetime of a group due to processes joining, leaving, failing or disconnecting. All the group members are responsible for tracking the changes to the group and reporting these changes to the members of the group. It is done by sending NOTIFY_g message with a list of the current members of the group to the members of the group.

To make the multicast groups in MANET robust and efficient, group members create multicast trees over unicast tunnels [14]. Each node is aware of its group neighbors and forwards data on the tree links to its neighbors.

When there are changes in ALM group membership, the group member, which notices them, sends a NOTIFY_g message to all the group members. Therefore all the group members keep the updated group information about that group. The initiator of an ALM group waits for a certain time to collect the 200 OK messages. It stores the names of the MNs, which responded with 200 OK messages, as the multicast group members. When the timer is up, the group leader sends a NOTIFY_g message (the Event header is set as: "ALM Group Membership".) with the group member list to all the group members. By now, an Application Layer Multicast

group is set up. An example of NOTIFY_g message is shown in Figure 4-5.

```
NOTIFY sip:ALMGroupMembers@example4.com SIP/2.0
Call-ID: 839872666@example4.com
CSeq: 41 NOTIFY
From: <sip:S4@example4.com>;tag=f3433
To: <sip:ALMGroupMembers@example4.com>;tag=t1234
Via: SIP/2.0/UDP 132.205.46.231;branch=z9hG4bKc1ecae4fc55a1630a497bbc3cecf6a71
Max-Forwards: 70
Require: ad_hoc_list
Accept: application/cpim-pidf+xml
Contact: <sip:S4@example4.com>
Forward-To: S8@example8.com S6@example6.com
Content-Type: application/adrl+xml
Event-ID: 839872666@example4.com IM
Content-Length: 175

<GroupMembers Event-ID="839872666@example4.com IM">
      <member=S6@example6.com/>
      <member=S4@example4.com/>
      <member=S8@example8.com/>
      <member=S9@example9.com/>
</GroupMembers>
```

Figure 4-5 An example of NOTIFY_g request

4.2.3 Node information about ALM group

As we mentioned in Chapter 3, there are some fields in host lists about ALM group information. Below we use the example of Figure 4-2 to describe them. Here we assume that MN6, MN8 and MN9 accepted to join that ALM group. If any SIP node does not accept to join a group, it acts as a forwarding node that forwards messages to its neighbors.

The examples of Profile List are shown in Tables 4-1 and 4-2. Because they are in the ALM group, *839872666@example4.com IM* is filled into the Event-ID field and it lists all the group members in the Group Members Field. MN4 is the leader of ALM group, so its Leader field is

"Y". The Leader field of MN6 is "N" instead.

Table 4-1 MN4's Profile List

Host Name (URI)	IP Address	Gateway	Event-ID	Leader	Group members
S4@example4.com	10.20.20.4	S4@example4.com	839872666@example4.com IM	Y	S6@example6.com S8@example8.com S4@example4.com S9@example9.com

Table 4-2 MN6's Profile List

Host Name (URI)	IP Address	Gateway	Event-ID	Leader	Group members
S6@example6.com	10.20.20.6	S8@example8.com	839872666@example4.com IM	N	S6@example6.com S8@example8.com S4@example4.com S9@example9.com

The "Event-ID" field is only filled for the records of the leaders of multicast groups, instead of marking it to all the group members. Therefore, the Non-Gateway Node Member List (shown in Table 4-3), and the Gateway Node Member List (shown in Table 4-4) do not have Event-ID field marked. MN6's SIP-Meshed Network Node List (shown in Table 4-5) has a value in Event-ID field for MN4 record.

Table 4-3 MN4's Non-Gateway Node Member List

Host Name (URI)	From	Event-ID	Changes
S9@example9.com	S9@example9.com		
S7@example7.com	S7@example7.com		

Table 4-4 MN4's Gateway Node Member List

Host Name (URI)	Members	Event-ID	Changes
S8@example8.com	S6@example6.com		N

Table 4-5 MN6's SIP-Meshed Network Node List

Host Name (URI)	From	Gateway	Event-ID	Hops	Changes
S9@example9.com	S8@example8.com	N		3	
S8@example8.com	S8@example8.com	Y		1	
S4@example4.com	S8@example8.com	Y	839872666@example4.com IM	2	
S7@example7.com	S8@example8.com	N		3	

A MN may obtain the list of the active ALM groups (identified by Event-IDs and the leaders of the groups in the record) from the periodic NOTIFY_d message. The leader of each group

is responsible for admission maintaining the group membership. The leader is also responsible for updating the list of group members upon any change in membership.

4.3Joining an ALM group

When a SIP MN wants to join an ALM group, it checks its host lists for the active group information. Because it has the information of ALM group leaders, it knows where to send the "Join" request.

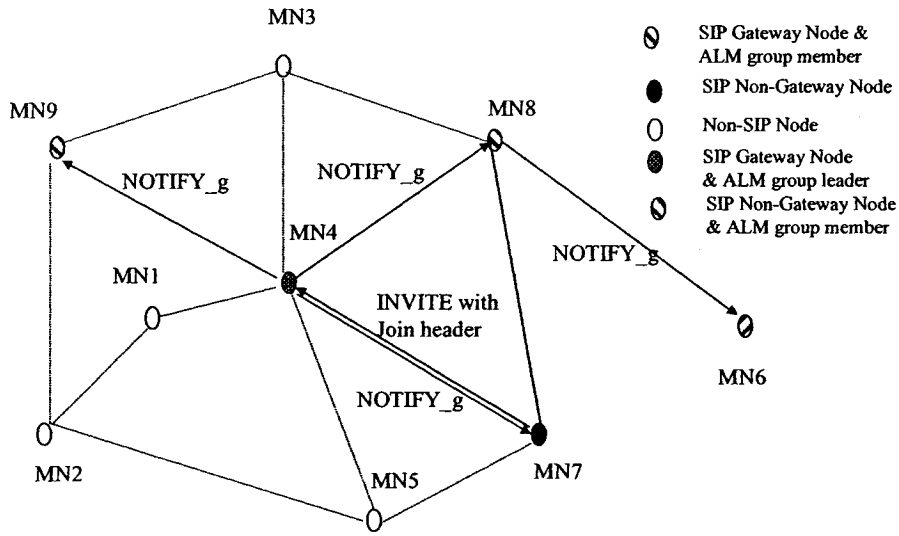

Figure 4-6 MN7 wants to join into an existing ALM group

For example, MN7 wants to join an ongoing ALM group, as shown in Fig. 4-6. MN7 sends an INVITE message with *Join* parameter to the leader of that ALM group, MN4 in this case. The header of INVITE includes the "Join" header [49] with the value of "Event-ID", which is the ALM group identity that the MN wishes to join.

The "Join" header was proposed for logically joining an existing SIP dialog with a new SIP dialog. This is especially useful in peer-to-peer call control environments. The "Join" header

contains information used to match an existing SIP dialog (call-id, to-tag, and from-tag). Here, because we use the value of "Event-ID" to describe different ALM groups, we put the value of Event-ID in the header. For the example as illustrated in Figure 4-7, the header of INVITE contains the identity of the ALM group that the MN wishes to join (as: *Join: 839872666@example4.com IM,* where " *839872666@example4.com IM*" is the Group ID of that active ALM group.

INVITE sip:S4@example4.com SIP/2.0
Call-ID: 2084073988@example7.com
CSeq: 48 INVITE
From: <sip:S7@example7.com:5070>;tag=f3433
Via: SIP/2.0/UDP 132.205.46.231:5070;branch=z9hG4bKcfef3a841608b21e008f4af27cf9342a
Max-Forwards: 70
Contact: <sip:S7@example7.com:5070>
To: <sip:S4@example4.com:5060>;tag=t1234
Join: 839872666@example4.com IM
Content-Length: 0

Figure 4-7 An example of INVITE request with a "Join" header

Because the ALM group leader is responsible for the admission control, the INVITE request with a Join header is addressed to be sent to that group leader. The Request-URI is addressed to S4@example4.com for this example.

Upon receiving an INVITE with a Join header, the leader MN attempts to match this information with a confirmed or early ALM group. If there is some control policy, the leader also will check whether the request should be accepted. The leader in this case sends a NOTIFY_g message as shown in Figure 4-8 to all the group members in that group including the new group member. Otherwise if the request is not accepted, the leader rejects the

INVITE and returns a 481 Call/Transaction Does Not Exist response.

```
NOTIFY sip:ALMGroupMembers@example4.com SIP/2.0
Call-ID: 839872666@example4.com
CSeq: 50 NOTIFY
From: <sip:S4@example4.com>;tag=f3433
To: <sip:ALMGroupMembers@example4.com>;tag=t1234
Via: SIP/2.0/UDP 132.205.46.231;branch=z9hG4bK8ff246f384580add94b7979bad32d1f4
Max-Forwards: 70
Event:  ALM Group Membership
Require: ad_hoc_list
Accept: application/cpim-pidf+xml
Contact: <sip:S4@example4.com>
Forward-To: S9@example9.com
Content-Type: application/adrl+xml
Event-ID: 839872666@example4.com IM
Content-Length: 202

<GroupMembers Event-ID="839872666@example4.com IM">
      <member=S6@example6.com/>
      <member=S4@example4.com/>
      <member=S8@example8.com/>
      <member=S7@example7.com/>
      <member=S9@example9.com/>
</GroupMembers>
```

Figure 4-8 An example of NOTIFY_g request after a Join, sent to ALM group members

4.4 Leaving an ALM group

As SIP MNs in meshed network, an ALM group member may also leave the group gracefully or abruptly. We will discuss these two cases in the following sections.

4.4.1 Leaving an ALM group gracefully

In our framework, a MN that wants to leave an ALM group sends a BYE message via its gateway node to all the other group members in that group. Upon the reception of this message, all the MNs remove it from the Group Members field in its Profile List.

For example, when MN9 wants to leave the multicast group, it constructs a BYE request and sends it to all the members of that ALM group. The generated BYE request is shown in Figure 4-9.

BYE sip:ALMGroupMembers@example4.com SIP/2.0
Call-ID: 839872666@example4.com
CSeq: 61 BYE
From: <sip:S9@example9.com:5070>;tag=t1234
To: <sip:ALMGroupMembers@example4.com>;tag=f3433
Via: SIP/2.0/UDP 132.205.46.231:5070;branch=z9hG4bK1b05d7f0939f27a8273bbb6dcfcd4946
Max-Forwards: 70
Contact: <sip:S9@example9.com:5070>
Forward-To: S4@example4.com S6@example6.com S8@example8.com S7@example7.com
Event-ID: 839872666@example4.com IM
Content-Length: 0

Figure 4-9 An example of BYE request

Here "ALMGroupMembers@example4.com" demonstrates that this BYE request is for members of ALM groups. Since the sender of this request knows all the group members, it will construct BYE requests to all listed group members. "S4@example4.com", "S7@example7.com", "S6@example6.com" and "S8@example8.com" which are listed in the "Forward-To" header, are all the other members of the ALM group "*839872666@example4.com IM*", which the MN wants to leave.

Upon receiving the BYE message, a MN performs the following steps:

- It analyses the Request-URI first. Because it is a BYE message, it checks the SIP URI list in the Forward-To field.
- If it is one of the recipients, it removes its own SIP URI from the list in the "Forward-To" header and modifies its profile list accordingly.
- The MN then checks its host lists to decide the next hops for forwarding such

message. The message to the same SIP URIs will be sent only once using the multicast recipient SIP extension. Each copy has a different Forward-To header field depending on the recipients that are accessible to that MN.

Figure 4-10 displays the forwarded BYE request from MN4 to MN8 and MN6. MN4 generates another BYE request to MN7.

BYE sip:ALMGroupMembers@example4.com SIP/2.0
Call-ID: 839872666@example4.com
CSeq: 61 BYE
From: <sip:S9@example9.com:5070>;tag=t1234
To: <sip:ALMGroupMembers@example4.com>;tag=f3433
Via: SIP/2.0/UDP 132.205.46.231:5070;branch=z9hG4bK1b05d7f0939f27a8273bbb6dcfcd4946
Contact: <sip:S9@example9.com:5070>
Event-ID: 839872666@example4.com IM
Record-Route: <sip:S4@example4.com:5600>
Max-Forwards: 69
Forward-To: S8@example8.com S6@example6.com
Content-Length: 0

Figure 4-10 An example of forwarded BYE request

Therefore, all the group members know the membership changes by receiving and processing received BYE requests.

4.4.2 Leaving an ALM group abruptly

It is also possible that a MN is disconnected without sending a BYE message, that is, it moves out of the connection with other nodes, or the wireless link of that MN is broken down. Under these situations, the participants in that MANET do not receive any NOTIFY_d or 200 OK messages from that MN. After a given amount of time, they assume that MN has left the mesh and mark its record as "Delete". For gateway nodes, after sending NOTIFY_d message, the MN's record is removed from the gateway nodes' host lists. The non-gateway nodes will

remove the record when reading it.

When an ALM group member receives the updated information, it checks whether the MNs marked as "delete" is a member of the ALM groups it is in. If the result is yes, it will send the information in a NOTIFY message to all other group members immediately. So other group members can remove it from their member lists. If no, only the normal operations are needed in the mesh part.

If a MN detects a connection failure, the detecting MN will try to connect to another route based on its host lists. If it loses the connection with other MNs totally, it will re-send SUBSCRIBE request to get the network information first. Then it re-joins the ongoing ALM groups and so on.

4.5 Routing in ALM

A basic design aspect, in supporting the delivery of multicast traffic to mobile nodes, is identifying the node to join the multicast tree associated with the requested group.

Due to the underlying mesh, there is no need for frequent tree readjustments, thus providing robustness in a high mobility environment. Per-group trees are embedded in the overlay mesh, and are formed as the union of the overlay routes from the group members to the root of the tree. Group trees can be used to multicast messages to the group by checking the route hop-by-hop to its destination. The multicast transfer could achieve low delay and introduces low link and node stress [50].

The proactive table-based routing schemes require each of the nodes in the network to maintain host lists to store the routing information, which is used to determine the next hop

for the packet transmission to reach the destination. Our framework attempts to maintain the table information consistent by transmitting periodical updates throughout the network.

When a source sends out control information, it is forwarded hop-by-hop from the source to the rest of the multicast group via unicast routing. This packet forwarding process is guaranteed loop-less because a destination address will be taken out of the list whenever the packet has reached the destination, therefore, it will not go back to that node again.

We choose to construct the multicast tree hop-by-hop due to the following reasons:

- It allows the intermediate nodes to utilize the latest location information of the destination nodes in computing the tree;
- By caching a previously computed tree, the computation will not be duplicated when the locations of the nodes have not changed between data packets.

When a MN receives a message, it checks its destination list. If this MN is one of the destinations, it will store the message for further processing. Then remove itself from the destination list and forwards the message to next hop according to the routing information in its host lists.

Routing together with gathering and maintaining routing information poses a heavy burden on the mobile devices. The Dominating Set Based Routing [51], where only gateway nodes need to keep routing information, and the non-gateway node is adjacent to at least one gateway node, is used to improve the performance.

4.6 Summary

In this chapter we propose a mechanism of using SIP INVITE, BYE and NOTIFY_g requests

to initiate, maintain and leave a SIP ALM group in MANET. The detailed message formats for INVITE, BYE and NOTIFY_g messages are illustrated here. The two new headers: Forward-To and Event-ID, are also introduced to reduce the maintenance traffic and describe ALM groups. The following chapter describes the simulation and verification of our proposed system.

Chapter 5 Simulation and Verification

5.1 Introduction

The SIP-ALM framework is motivated by the need to support group communication among a small group of hosts without relying on the IP multicast model. In this scheme, participants of a multicast group are connected via virtual multicast routes, i.e. routes that consist of unicast connections between end hosts. Each device installs a SIP-ALM application that facilitates human interaction to initiate and manage multicast group membership.

We implement our framework as a proof of concept. Thus we can demonstrate the basic functions of the system and can verify the integrity of the system. The program of verifying our SIP-ALM framework is entirely relying on Java technology. The SIP part of the implementation is using Java SIP Toolkit (implementing JAIN SIP API 1.1 specifications [52]). The other components are coming from the Java domain. In this chapter, we discuss some detailed information about the simulation and verification of our framework.

5.1.1 JAIN SIP

We selected Java because of its inherent portability to all the machines and the extensive offer of packages, though the executable code is slower than what can be achieved by coding in C. The Java language has been introduced by Sun Microsystems in 1995 with the purpose of deploying dynamic content into web pages. The main idea was that of creating web pages that

not only contained static text and images, but also dynamic multimedia (video and animations) supported by interactivity. Java has then extended its application to web server programming and to software targeted for consume electronics (cell phones, PDAs, etc.). Java is a C/C++ based language.

The JAIN SIP APIs specifications have been proposed by Sun Microsystems to define the entire set of Java interfaces for the SIP protocol. It provides an application layer interface for sessions that use the SIP protocol for control signaling. The JAIN SIP architecture is shown in Fig 5-1. The main elements are:

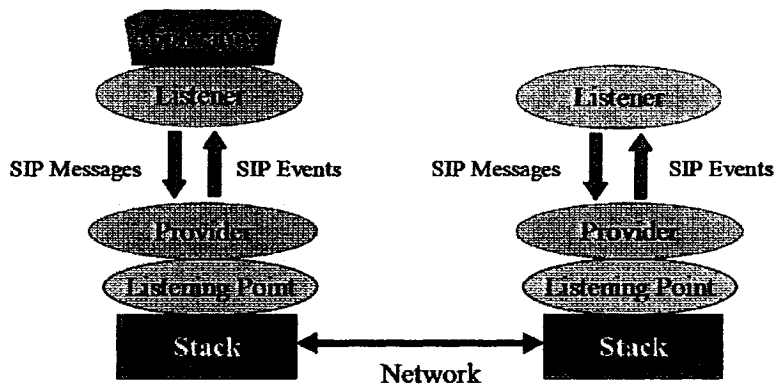

Figure 5-1 JAIN SIP Architecture [53]

- JAIN SIP Stack: this interface defines the entry point to accept SIP requests from lower network layers;
- JAIN Listening Point: a Java representation of the port that a SipProvider messaging entity uses to send and receive messages. A SipProvider as a message entity may only have a single Listening Point.
- JAIN SIP Provider: it is the entity which routes the input events towards the appropriate JAIN SIP Listeners;

- JAIN SIP Events: the SIP messages are encapsulated as object events and relayed from the SIP Provider towards the SIP Listener.
- JAIN SIP Listener: it is the entity that develops the application and manipulates directly the events sent to the JAIN SIP Provider;

JAIN SIP is based on the SIP Factory model. It means that the introduction of an element allows calling any proprietary implementation of the JAIN interfaces. This architecture permits any application to obtain a proprietary JAIN SIP object, which refers to a particular software implementation, by setting the appropriate implementation path.

The reference model is reported in Fig.5-2. Our simulation works mainly in the Setup Function part.

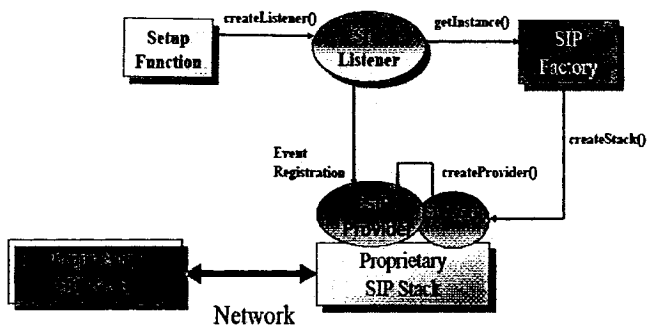

Figure 5-2 Creation of a JAIN SIP object [54]

5.2 Simulated System Architecture

As shown in Figure 5-3, the proposed SIP-ALM architecture is comprised of two layers: SIP Mesh layer and ALM Group layer. The SIP Mesh layer has functionalities of registering to other MNs and accepting registration. It also maintains the membership of SIP meshed network and keeps the routing information updated.

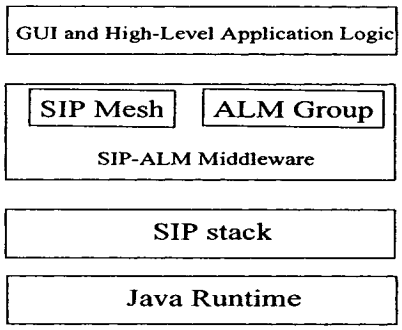

Figure 5-3 SIP-ALM Architecture

The ALM Group layer is primarily responsible for providing a totally ordered multicast to the upper layers based on simple send and receive operations to and from the underlying network. As displayed in Figure 5-4, ALM Group Layer obtains the basic routing information from SIP Mesh Layer. The creation of ALM groups is used to support the applications/services on the upper layer. The same layer exchanges information with other peers.

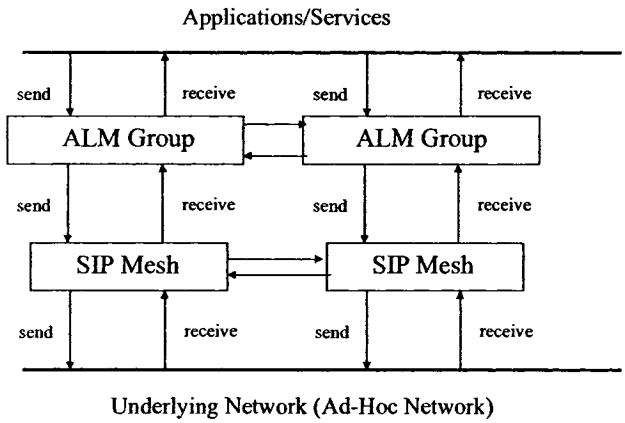

Figure 5-4 SIP-ALM Structure

In our application simulation, we have the following packages:

- userInterface: handles all the user interface, window, screens.
- debug: contains a class DebugUtilities to print out all the execution result in a log file.

- control: contains the class of calling the classes to execute SIP requests and responses; control the operations of writing to and reading out from the memory.
- sipSessions: contains the classes of creating and processing SIP messages used in our framework, such as INVITE, NOTIFY, SUBSCRIBE and BYE requests.
- nodeMag: handles different data structures to store the node information, and the operations to execute on those data structures.

Next, we explain some primary classes, as shown in Figure 5-5, in the following sections.

5.2.1 The MNScreen class

The MNScreen class is the main class of our SIP-ALM simulation. When activating this class, a window will be popped up. It provides the user-friendly interface for the users. Users select a name for the simulation file to start the verification. It is also possible to check the MN information here. It is the starting point of our simulation.

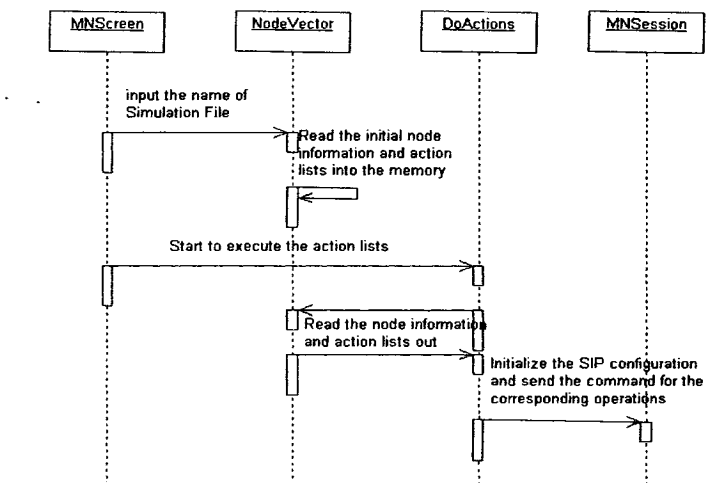

Figure 5-5 UML sequence diagram of starting operations

5.2.2 The NodeVector class

The NodeVector class is called for parsing input files, which are in XML format, and saving the information into the different data structures in the memory. It is also called for reading or storing the node information from/to the memory. Because we assume that our middleware would be installed on mobile devices, to use vector, instead of databases, for handling the routing information in the memory should reduce the valuable storage place and some processing time for mobile nodes. It is for meeting the need to limit class instances on a poorly performing environment such as a PDA.

The singleton pattern has been chosen for this class. The factors that led to this design are the need of a single point of access to the host information in the vector for the integrity of the information. When the constructor is called at the beginning, the object is already built. After that, the same instance is always returned for manipulating.

5.2.3 The MNSession class

The MNSession class is another primary class of our implementation. It is used to create the SIP object, just as shown in Figure 5-2. The MNSession is the connecting point between the High Level API and the SIP Stack. The Provider/Listener model, as shown in Figure 5-1, is used for sending/receiving of the SIP requests/responses. The MNSession class provides a path for user to execute the service operations via SIP protocol. Each SIP request must be sent through an instance of this class. Operation commands are coming from the DoActions class, which executes the action lists in the simulation files.

By analyzing the operation commands, MNSession triggers the corresponding request

creation classes as shown in Figure 5-6.

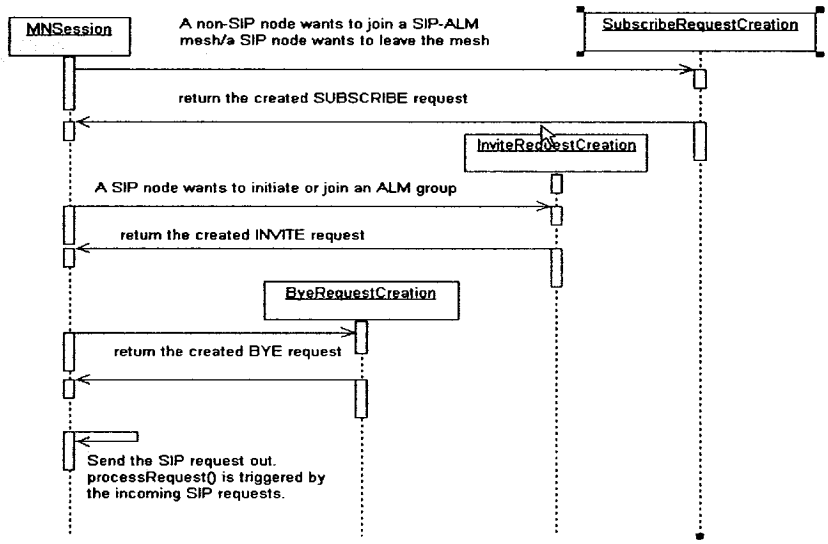

Figure 5-6 UML sequence diagram of creating SIP Request

This class has the role of managing incoming requests from the SIP stack by creating SIP Listener, Stack and Providers, as shown in Figure 5-2. The MNSession class also provides methods for the stack to inform a user of incoming SIP requests/responses/timeouts.

The MNSession class must implement all the methods defined in the SipListener interface. When one of the methods inherited from the SipListener interface is invoked, the MNSession instance sends the request to the class that effectively manages that type of communication.

The methods of this class that manage incoming messages are the processRequest() and the processResponse() method. The processRequest() method is called every time a new SIP request is received by the stack. This action is performed by invoking the processRequest() method on MNSession and passing it a parameter of type SipEvent. The SipEvent class is a container for all the information regarding a SIP event. SIP events, for example, are the

receipt of a request, a response or a timeout. By performing checks on the attributes of the SipEvent object the MNSession class can understand if the message is directed to the current MN or not. The first check that is done in the processRequest() method is by checking the type of request. The requests handled in our implementation are SIP INVITEs, SIP BYEs, SIP SUBSCRIBEs and SIP NOTIFYs. The processResponse() method manages received responses as well.

An incoming request is processed by a server transaction, and then it is passed to our SIP-ALM application. The MNSession class first processes any requests by analyzing the SIP methods of those requests. Then by calling the corresponding request processing class, the MNSession class triggers other classes to determine what to do and where to route the request. An outgoing request for each next-hop location is processed by its own associated client transaction. The MNSession collects the responses from the client transactions and uses them to send responses to the server transaction.

Here we next explain the relevant operations used for different SIP request in more details.

5.2.3.1 SIP INVITE

A SIP INVITE request is sent to invite members to a new group or join an existing ALM group. In the first case, such SIP INVITE request contains "group@sip.manet" in its Request-URI, and a new Event-ID header is added. In the setting up of an ALM group, the user that receives an invitation will see a pop-up window on the display. The users then decide whether to accept such initiation.

For the latter case, the request-URI of such SIP INVITE request will address to the leader of that ALM group. There will be a Join header in such SIP INVITE request, whose value is the Event-ID of that group. The flow chart of how SIP INVITE requests are processed is displayed in Figure 5-7.

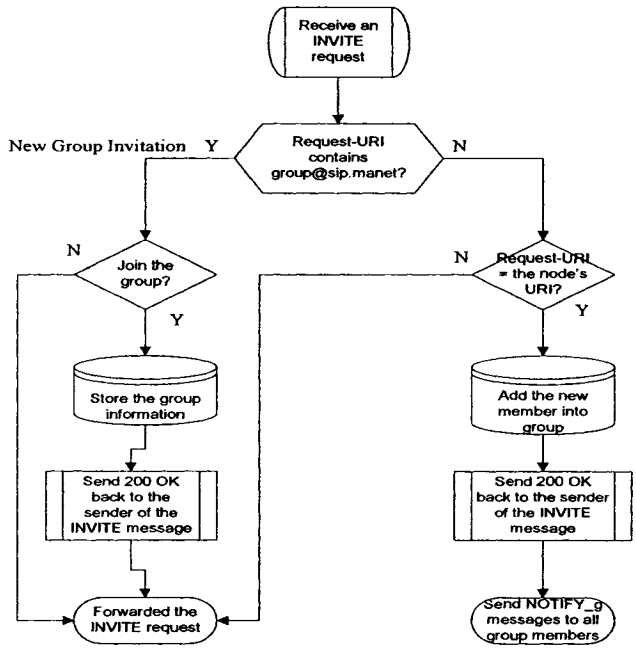

Figure 5-7 Flow Chart of processing SIP INVITE Request

The process of forwarding request follows these steps:

- Copy request - The copy of the received request contains all of the header fields from the received request.

  The Request-URI in the copy's start line should not be replaced. In some circumstances, the received Request-URI is placed into the target set without being modified.

- Max-Forwards - If the copy contains a Max-Forwards header field, the proxy must decrement its value by one (1).

- Record-Route – Generally, a Record-Route header field is inserted to remain on the same path of future requests in a dialog created by this request (assuming the request creates a dialog). It might be a mechanism to keep the route unique, by using SIP URIs for unicast routing.

  Because in MANET, mobile devices work as end hosts, as well as routers, this field is not mandatory in our approach.

- Add a Via header field value - A Via header field value must be added into the copy before the existing Via header field values. The MNs in SIP mesh choosing to detect loops have an additional constraint in the value they use for construction of the branch parameter.

  Loop detection is performed by verifying that, when a request returns to a router, those fields having an impact on the processing of the request have not changed. The value placed in this part of the branch parameter should reflect all of those fields (including any Route, Proxy-Require and Proxy-Authorization header fields).

- Forward Request to the next recipient.

The sequence diagram of processing received INVITE request is shown in Figure 5-8. When the receiving INVITE request is analyzed as a Group Invitation message. If the receiver accepts it, an instance of UpdateGroup class is called to update the host information, and then dump it to the memory. If there are SIP URIs in the Forward-To header, no matter whether the receiver accepts it, an instance of InviteRequestForwarding class is called to modify and forward the INVITE request.

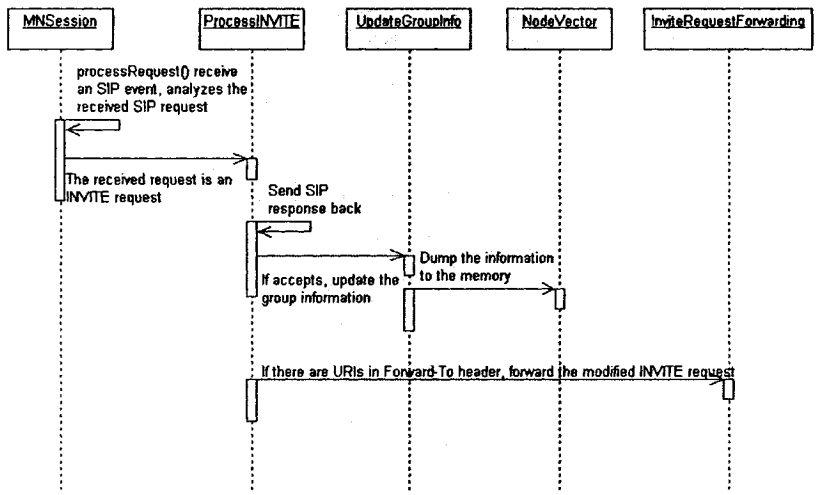

Figure 5-8 Sequence Diagram of processing received INVITE message

5.2.3.2 SIP SUBSCRIBE

SIP SUBSCRIBE request is created and sent when a MN wants to join or leave a SIP meshed network. The difference of the SUBSCRIBE requests between those two cases is: for the join case, the value of "Expires" should be some value other than "0", and the value of the "Event" header should be "presence"; for the leave case, the value of "Expires" will be "0", and the value of the "Event" header will be "leaving_SIP_network".

As we mentioned earlier, the MNSessions listen to SIP requests sent by other MNs. When received by the processRequest(), SIP SUBSCRIBE requests are mapped by the MNSession instance to the ProcessSubscribeRequest class. The flow chart of how SIP SUBSCRIBE requests are processed is displayed in Figure 5-9.

When the receiving SUBSCRIBE request is analyzed as a "Join" request, some classes are called consequently as shown in Figure 5-10. An instance of UpdateHostList class is called to update the host information, and then dump the changes to the memory. It also verifies

whether this receiving MN is a gateway node. If it is, NotifyRequestCreation is called to create and send NOTIFY message. Then PresenceService class is called to send NOTIFY_d message periodically.

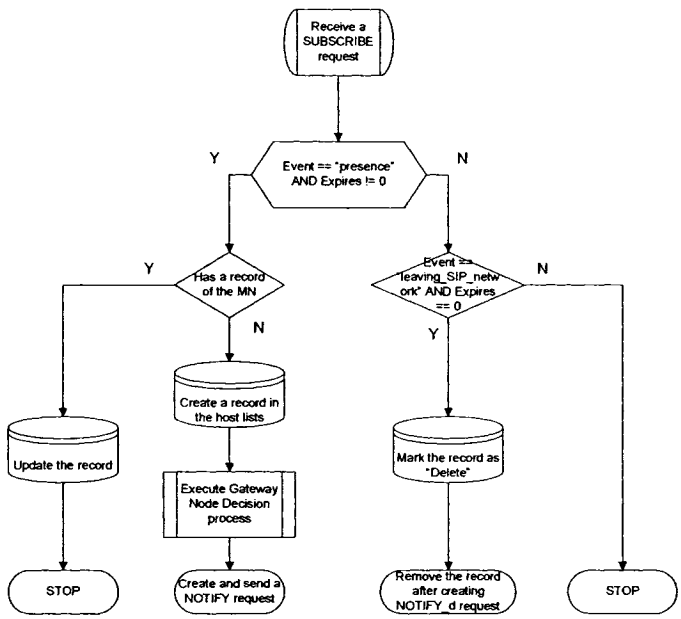

Figure 5-9 Flow Chart of analyzing received SUBSCRIBE message

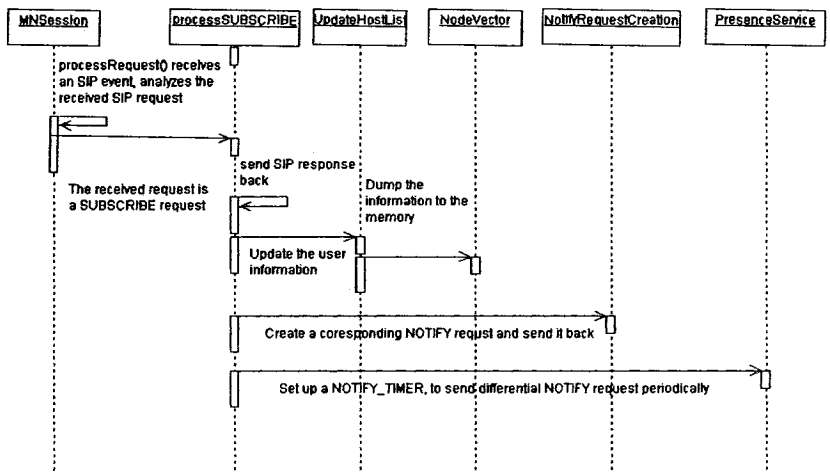

Figure 5-10 Sequence Diagram of processing received SUBSCRIBE message for joining

5.2.3.3 SIP NOTIFY

As we mentioned in the previous chapters, there are three kinds of NOTIFY requests in our framework:

-
- On the receipt of a SIP SUBSCRIBE, the SIP-ALM application sends SIP NOTIFY requests back to the initiating MN. In the content of this NOTIFY request, it contains all the member list of this local MN.
- If a MN is a gateway node, the PresenceService instance sends SIP NOTIFY_d requests periodically. Unlike the NOTIFY request in the first case, its contents only contain the differential information of the MN's member list.
- When a group member notices that another group member has left the ALM group, or a group leader received an INVITE request with Join header, it will update its host lists and send a NOTIFY_g request with the entire group member list to all the group members.

When received by the processRequest(), SIP NOTIFYs is mapped by the MNSession instance on the ProcessNOTIFY class. The Request-URI header is checked and the corresponding operation is executed, as shown in Figure 5-11.

The sequence diagram of processing received NOTIFY message is shown in Figure 5-12. ProcessNOTIFY is called to analyze the type of received NOTIFY message. For the normal NOTIFY message and NOTIFY_d message, the content of the received message is analyzed and stored in the memory. If it is a NOTIFY_g message, an instance RequestForwarding class is called for forwarding such message.

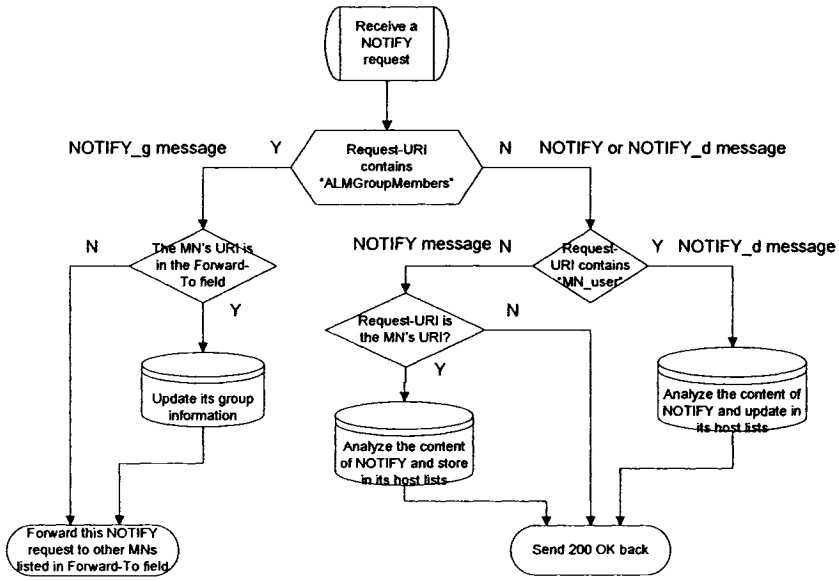

Figure 5-11 Flow Chart for processing received NOTIFY message

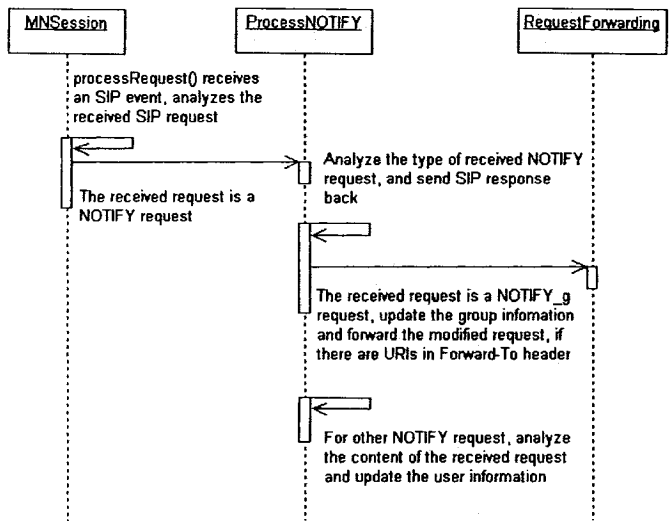

Figure 5-12 Sequence Diagram of processing received NOTIFY message

5.2.3.4 SIP BYE

A BYE request is constructed by an ALM group member for leaving that group. It has "ALMGroupMembers" as part of its Request-URI, and all other group members are listed in "Forward-To" header.

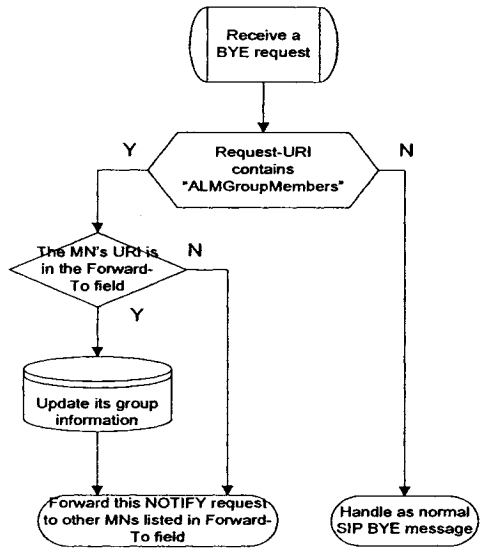

Figure 5-13 Flow Chart for processing received BYE message

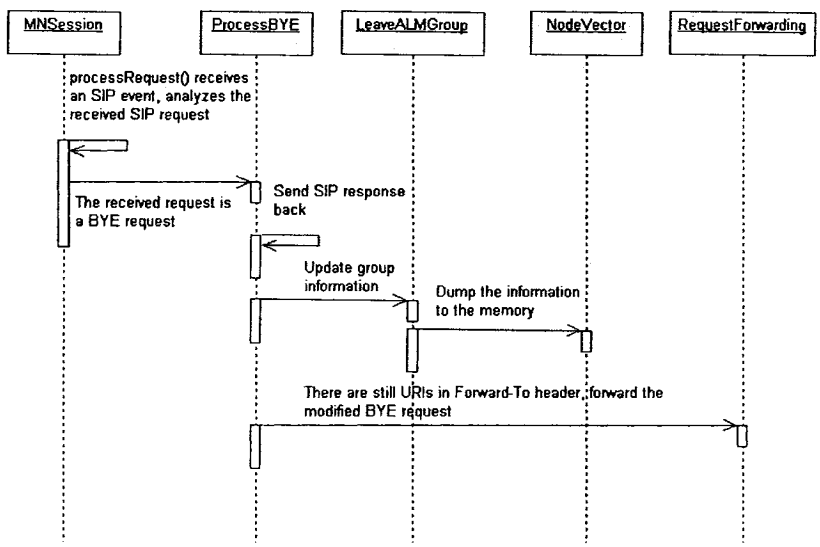

Figure 5-14 Sequence Diagram of processing received BYE message

When a MN receives a BYE request, the procedure, shown in Figure 5-13, is executed to

terminate the involvement of the ALM group. The relevant classes are called sequentially, as shown in Figure 5-14.

5.3 Verification of our approach

Implementing and testing an actual ad-hoc network is costly due to high complexity, required hardware resources and inability to test them under a wide range of mobility scenarios. Not only that the implementation will involve sophisticated system-level programming, but also that a thorough test requires deployment of large-scale test-bed in various mobility patterns [55].

There have been previous attempts [56] [57] to describe full mesh conferencing for SIP by Jonathan Lennox and Henning Schulzrinne. In [57], it is said that the primary difficulty in verifying the full mesh protocol is that its behavior depends strongly on the order in which events occur. We have similar issue in our case. For example, a MN should join a SIP mesh first, and then it could initiate or join an ALM group. Therefore, we verify our proposed framework by setting up different scenarios simulated as operating in MANET.

In the testing, the functionalities of our simulation were verified by logging and comparing to the expected results. By the log file, the applications' progress is followed and any incorrect functionality may be found.

Now we start to introduce the designed test cases and sequences used in our framework in the following sections.

5.3.1 Introduction to the Node States

As we discussed in Chapters 2, 3 and 4, there are three kinds of node in our proposed framework. As shown in Figure 5-15, the nodes are divided as: Non-SIP node, SIP node and SIP-ALM node.

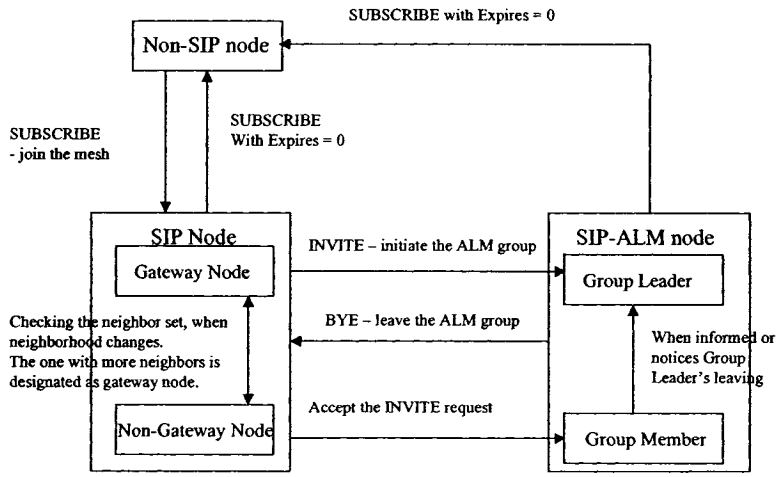

Figure 5-15 The state transition diagram of the verifier

A more detailed state transition in our proposed SIP-ALM framework is drawn in Figure 5-16. The notation for expressing transmissions and receptions is: a question mark (?) identifies an input event - the reception of a message; an exclamation mark (!) identifies an output event - the transmission of a message [58].

The states mentioned in Figure 5-16 are:

- NS means Non-SIP node,
- SNG means SIP Non-Gateway node,
- SG means SIP Gateway node,
- GMSNG means Group Member and SIP Non-Gateway node,
- GLSNG means Group Leader and SIP Non-Gateway node,
- GMSG means Group Member and SIP Gateway node,

- GLSG means Group Leader and SIP Gateway node.

Generally, as listed in Table 5-1, twenty-two state transition events could be executed to transfer one state to another. And also, a state transition happens in a certain order.

As we could see from Figure 5-16 and Table 5-1, behaviors of the proposed framework depend strongly on the order in which events occur. Thus, we decide to design every test case correspondent to each event and then put the simulated actions in different sequences to simulate different scenarios for protocol verification. In this way, the verification is optimized in order to keep verification tractable on standard hardware.

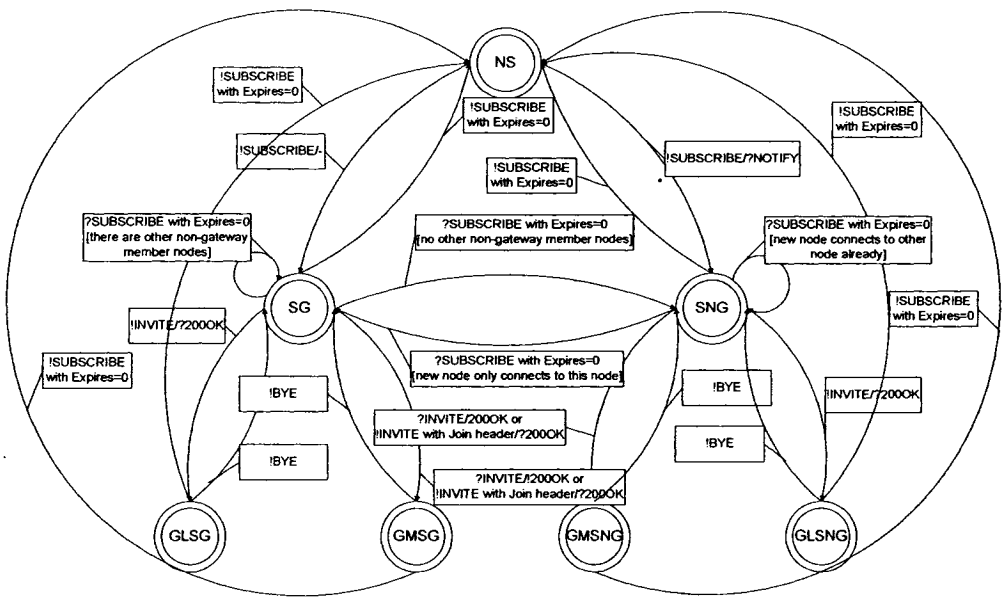

Figure 5-16 The State Transition in SIP-ALM framework

The number of members in the simulated meshed network and ALM group is not more than six, and most of the actions are done among five members.

Table 5-1 List of State Transition Events

ID	State	Sending Message	Receiving Message	Additional Condition	Successor State
1	NS	SUBSCRIBE	-	-	SG
2	NS	SUBSCRIBE	NOTIFY	-	SNG
3	SG	SUBSCRIBE with Expires=0	-	-	NS
4	SNG	SUBSCRIBE with Expires=0	-	-	NS
5	SG	-	SUBSCRIBE with Expires=0	No other non-gateway member nodes	SNG
6	SG	-	SUBSCRIBE with Expires=0	There are other non-gateway member nodes	SG
7	SNG	-	SUBSCRIBE with Expires=0	New node only connects to it	SG
8	SNG	-	SUBSCRIBE with Expires=0	New node connects to other node already	SNG
9	SG	INVITE	200 OK	-	GLSG
10	SG	200 OK	INVITE	-	GMSG
11	SG	INVITE with Join header	200 OK	-	GMSG
12	GMSG	BYE	-	-	SG
13	GMSG	SUBSCRIBE with Expires=0	-	-	NS
14	GLSG	BYE	-	-	SG
15	GLSG	SUBSCRIBE with Expires=0	-	-	NS
16	SNG	INVITE	200 OK	-	GLSNG
17	SNG	200 OK	INVITE	-	GMSNG
18	SNG	INVITE with Join header	200 OK	-	GMSNG
19	GMSNG	BYE	-	-	SNG
20	GMSNG	SUBSCRIBE with Expires=0	-	-	NS
21	GLSNG	BYE	-	-	SNG
22	GLSNG	SUBSCRIBE with Expires=0	-	-	NS

5.3.2 Test Actions Design

Based on those possible actions, certain test actions are designed to verify each state transition path. Table 5-2 lists all the simulated mesh actions executed by the verifier. Mobile Nodes are named 1, 2, 3, ... , in order. In Column "Testing State Transition Events" of Table 5-2, the

testing events are listed for each action.

Table 5-2 SIP-ALM scenarios explored to be verified

ID	Source	Source State	Receiver	Receiver State	Action	Expected Result[2]	Testing State Transition Events
0	Simulation1.xml				Refresh		
1	4	NS			JoinMesh	4-SG	1
2	4	SG			LeaveMesh	4-NS	3
3	9	NS	4	SG	JoinMesh	4-SG 9-SNG	2
4	9	SNG	4	SG	LeaveMesh	4-SG 9-NS	4
5	4	SG	9	SNG	LeaveMesh	4-NS 9-SG	3, 7
6[1]	4	SG	9	SNG	NoResponse	4-SG	
7	4	SG	9	SNG	NoRequest	9-SG	
8	8	NS	4	SG	JoinMesh	4-SG 8-SNG	2
9	7	NS	4	SG	JoinMesh	4-SG 7-SNG	2
10	7	SNG	8	SNG	JoinMesh	7-SNG 8-SNG	8
11	6	NS	8	SNG	JoinMesh	6-SNG 8-SG	2, 5
12	6	SNG	8	SG	LeaveMesh	6-NS 8-SNG	
13	8	SG	4	SG	LeaveMesh	4-SG 8-NS	3, 6
14	4	SG			StartALM	4-GLSG 6-GMSNG 8-GmSG	9, 10, 17
15	4	SG	7	SNG	StartALM[3]	4-GLSG 9-GMSNG 7-SNG	
16	4	SG	8	SG	StartALM	4-GLSG 7-GMSNG 8-SG 6-GMSNG	
17	7	SNG	4	GLSG	JoinALM	4-GLSNG 7-GMSNG 9-GMSNG	18
18	9	GMSNG	4	GLSG	LeaveALM	4-GLSG 7-GMSNG 9-SNG	19
19	8	GMSG	4	GLSG	LeaveALM	4-GLSG 9-GMSNG 8-SG	12
20	6	GMSNG	8	GMSG	LeaveMesh	4-GLSG 6-SNG 7-GMSNG 8-GMSNG	20
21	5	NS	7	SNG	JoinMesh	5-SNG 7-SG	2, 5
22	9	SNG			StartALM	4-GMSG 9-GLSNG 8-GMSG	16
23	8	GMSG	4	GLSG	LeaveMesh	4-GLSG 8-NS	13

24	8	SG	4	GLSG	JoinALM	4-GLSG 8-GMSG	11

Note:

1 - The scenario is that MN4 receives no response from MN9 to its NOTIFY_d. We still keep the state in MN9, because for MN9, it still receives NOTIFY_d message from MN4.

2 – Because the combination of actions is different, the result could be different too. But the states of the main actors, which are mentioned as the sender and receiver of an action, should be checked for the expected result.

3 – For "Start ALM" cases, if no receiver is listed, it means all the mesh network members accept such invitation. For example, in Action 15, since MN7 is the receiver, it refuses such invitation whereas other non-receiver nodes accept the invitation.

As we could see State Transition Events 14, 15, 21 and 22 are not listed in Table 5-2. The reason is when a group leader leaves the group, there should be one group member designated as group leader. Further research is needed to select an algorithm for choosing the new group leader, when the old one is leaving.

Also there are some additional test casts, like 6, 7, 15 and 16. For test cases 6 and 7, they simulate the detection of abrupt leaving. For test cases 15 and 16, they verify the execution when one node does not agree to join an ALM group.

5.3.3 Test Sequences Design

As we could see from Figure 5-16, there are pre-conditions for testing state transition events. For example, for the state transition NS -> SNG, there should be a SG node to receive SUBSCRIBE messages and respond with NOTIFY messages. Thus to test some designed test cases, several tested actions should be run beforehand to meet those pre-conditions. That is the reason we design action sequences for ordering purpose to cover the typical scenarios of operations executed in our framework.

The executed test sequences in our verifier are listed in Table 5-3. The last test action in each

test sequence is the one need to be verified. The pre-conditions should be met before executing the last test action; they correspond to the source state and receiver state of that test action in Table 5-2. The testing paths are also listed to demonstrate the coverage of our verification.

Table 5-3 Test Sequence Lists

ID	Test Sequence	Pre-Condition	Path Taken	Expected Result	Note
0	1	4-NS	1	4-SG	
1	1 -> 3	4-SG	2	9-SNG	
2	1 -> 2	4-SG	3	4-NS	
3	1 -> 3 -> 4	4-SG 9-SNG	4	4-SG 9-NS	
4	1 -> 3 -> 5	4-SG 9-SNG	3, 6	4-NS 9-SG	
5	1 -> 3 -> 6	4-SG 9-SNG		4-SG 9-SNG	There is no MN9 record in MN4's host lists.
6	1 -> 3 -> 7	4-SG 9-SNG		4-SG 9-SG	There is no MN4 record in MN9's host lists.
7	1 -> 3 -> 8 -> 9 -> 10	4-SG 7-SNG 8-SNG	6	4-SG 7-SNG 8-SNG	To create a mesh in a clustering, the state of S8 does not change
7	1 -> 3 -> 8 -> 11	6-NS 8-SNG	6	6-SNG 8-SG	
8	1 -> 3 -> 8 -> 9 -> 10 -> 11	4-SG 6-NS 7-SNG 8-SNG	6	4-SG 6-SNG 7-SNG 8-SG	To create two clusters, MN7 connects to MN8 via MN4.
9	1 -> 3 -> 8 -> 9 -> 11 -> 12	6-SNG 8-SG	5	6-NS 8-SNG	Also checks whether the information of MN6's leaving transfers to other MNs
10	1 -> 3 -> 8 -> 9 -> 11 -> 13	4-SG 8-SG	3	4-SG 8-NS	Since MN8 will not send NOTIFY_d message periodically, MN6 should turn to be SG.
11	1 -> 3 -> 8 -> 9 -> 11 -> 14	4-SG 6-SNG 7-SNG 8-SG 9-SNG	7, 8, 15	4-GLSG 6-GMSNG 7-GMSNG 8-GMSG 9-GMSNG	
12	1 -> 3 -> 8 -> 9 -> 11 -> 15	4-SG 6-SNG 7-SNG 8-SG 9-SNG	7, 8, 15	4-GLSG 6-GMSNG 7-SNG 8-GMSG 9-GMSNG	
13	1 -> 3 -> 8 -> 9 -> 11 -> 16	4-SG 6-SNG 7-SNG 8-SG 9-SNG	7, 8, 15	4-GLSG 6-GMSNG 7-GMSNG 8-SG 9-GMSNG	
14	1 -> 3 -> 8 -> 9 -> 11 -> 15 -> 17	4-GLSG 6-GMSNG 7-SNG 8-GMSG	16	4-GLSG 6-GMSNG 7-GMSNG	

		9-GMSNG		8-GMSG 9-GMSNG	
15	1 -> 3 -> 8 -> 9 -> 11 -> 14 -> 18	4-GLSG 6-GMSNG 7-GMSNG 8-GMSG 9-GMSNG	17	4-GLSG 6-GMSNG 7-GMSNG 8-GMSG 9-SNG	
16	1 -> 3 -> 8 -> 9 -> 11 -> 14 -> 19	4-GLSG 6-GMSNG 7-GMSNG 8-GMSG 9-GMSNG	10	4-GLSG 6-GMSNG 7-GMSNG 8-SG 9-GMSNG	
17	1 -> 3 -> 8 -> 9 -> 11 -> 14 -> 20	4-GLSG 6-GMSNG 7-GMSNG 8-GMSG 9-GMSNG	18	4-GLSG 6-NS 7-GMSNG 8-GMSNG 9-GMSNG	
18	1 -> 3 -> 8 -> 9 -> 10 -> 11 -> 21	5-NS 7-SNG	6	5-SNG 7-SG	Check the host lists about gateway nodes MN4, MN7 and MN8
19	1 -> 3 -> 8 -> 9 -> 10 -> 11 -> 22	4-SG 6-SNG 7-SNG 8-SG 9-SNG	8, 14, 15	4-GMSG 6-GMSNG 7-GMSNG 8-GMSG 9-GLSNG	Also need to check forwarded INVITE message
20	1 -> 3 -> 8 -> 9 -> 11 -> 14 ->23	4-GLSG 6-GMSNG 7-GMSNG 8-GMSG 9-GMSNG	11	4-GLSG 6-GMSG 7-GMSNG 8-NS 9-GMSNG	
21	1 -> 3 -> 8 -> 9 -> 11 -> 16 -> 24	4-GLSG 6-GMSNG 7-GMSNG 8-SG 9-GMSNG	9	4-GLSG 6-GMSNG 7-GMSNG 8-GMSG 9-GMSNG	

When executing these test sequences, aside from verifying the state transition to be executed correctly, we also need to verify if the transferred network and group information is handled correctly. For example, when a new node joins the meshed network, all the nodes in the meshed network should contain its record with correct topology information. That is the reason that we have different test cases for same state transition event and run different test sequences for the same testing paths. Checking the log after executing each test sequence does the verification of node information.

Because we use test sequence to manage the state transition, it is done step by step. The remaining case is to consider simultaneous action of the meshed network.

The only possible affected scenario is: when two SIP nodes initiates SIP function simultaneously, they both think themselves as Gateway nodes and thus, responding to the SUBSCRIBE request with NOTIFY message. If there are no other members under those two nodes, then both of them change themselves to Non-Gateway nodes. This situation could be changed when one node has other members. So this node sets itself as Gateway node and sends with NOTIFY_d message periodically. So the lack of verifying simultaneous actions in our framework does not affect the functional verification very much.

5.3.4 The Verification Framework

To set up the simulation environment, the simulator maintains two items: firstly, the state of the system, describing which MNs are in the SIP mesh network, the states of these MNs and each MN's knowledge of the network topology; and secondly, a list of pending actions which are to be executed, consisting of actions such as, joining the mesh, leaving the mesh, a MN inviting other MNs to the group, joining existing groups, and leaving the group.

To start the test sequence for verifying our framework, we decide to use a Simulation File in XML format, which has become a standard for data exchange.

5.3.4.1 The format of Simulation File

An example of simulation file is displayed in Figure 5-17; it is divided into host lists part and action lists part.

As we mentioned earlier, the order in which events occur is quite important for verifying our framework [56]. Thus, to simulate a particular scenario, the initial states of mobile nodes, such as, which mobile nodes are in the simulation, each node's status, are set up. In Figure

5-17, the host lists part is used to initialize the position and state of each MN in the network, which creates a node graph as Figure 2-4. The host lists part consists of:

```
<?xml version='1.0' encoding='us-ascii'?>

<SimulationFile>

<HostTable>

<NODES>
    <node id="9" uri=" " position="(40,40)" />
    <node id="1" uri=" " position="(90,100)" />
    <node id="2" uri=" " position="(40,160)" />
    <node id="3" uri=" " position="(160,20)" />
    <node id="4" uri=" " position="(160,160)" />
    <node id="5" uri=" " position="(240,240)" />
    <node id="6" uri=" " position="(500,140)" />
    <node id="7" uri=" " position="(320,200)" />
    <node id="8" uri=" " position="(300,40)" />
</NODES>

</HostTable>

<Action>
    <issue id="0" source="Simulation1.xml" action="refresh"/>
    <issue id="1" source="4" sourceState="NS" receiver=" " receiverState=" " action="JoinMesh" result="4-SG"/>
    <issue id="2" source="4" sourceState="SG" receiver=" " receiverState=" " action="LeaveMesh" result="4-NS"/>

    <issue id="3" source="9" sourceState="NS" receiver="4" receiverState="SG" action="JoinMesh" result="4-SG
9-SNG"/>
    <issue id="4" source="9" sourceState="SNG" receiver="4" receiverState="SG" action="LeaveMesh" result="4-SG
9-NS"/>
    <issue id="5" source="4" sourceState="SG" receiver="9" receiverState="SNG" action="LeaveMesh" result="4-NS
9-SG"/>
</Action>
<Sequences>
    <issue id="0" actions="1 2 3 4 7" />
</Sequences>
</SimulationFile>
```

Figure 5-17 An example of a Simulation File

- <HostTable> - represents that its content will be stored into the corresponding host lists in Vector.
- <NODES> - The content in this block lists the Node ID, position and URI, if it has. All the information is used to draw the node graph in the main screen.
- If the Node ID starts with "S", it has the SIP-ALM function and is in a SIP mesh. If the Node ID starts with "G", it is in one or more ALM groups.

By reading the position of each node, a node graph will be drawn in the "SIP-ALM Node Graph" area. As shown in Figure 5-18, each dot in the graph is a node. The node in the lightest color is a normal MN without SIP-ALM function. The node may be a normal MN without SIP-ALM function. It may be part of the SIP meshed network and may belong to ALM groups.

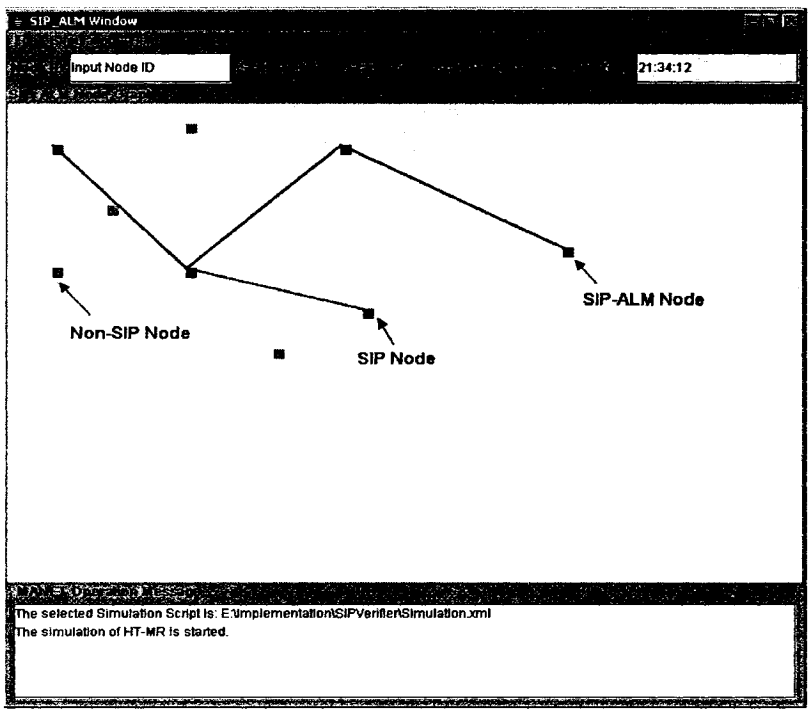

Figure 5-18 The network topology used in our simulation

An action list is also given for the actions to be taken. The action lists part consists of:

- <Action> - The content in this block lists the available actions.
- <Sequences> - Lists the executing sequence of actions

The test sequence is picked up from Table 5-3.

5.3.4.2 The result verification

Once an execution sequence has completed, the verifier validates the resulting state, as shown in Figure 5-16. A resulting state is valid if it is the result listed in the last action item. Then the verifier chooses the next execution sequence from the list, and continues until all actions have been exhausted. If the resulting state is not what expected, the verifier prints an error message and the exact sequence of actions and states that led to this outcome.

5.4 ANALYSIS AND RATIONALE

By different combination of execution sequences, our verification covers the typical scenarios of operations executed in our framework. However, these simulations do not fully explore the possibilities of the network status, as the potential size of the SIP mesh is, of course, unlimited. In this section we will attempt to justify the belief that the cases considered adequately cover all the possible ways that a network or group membership changes can interact.

5.4.1 The result verification

Once an execution sequence has completed, the verifier validates the resulting state, as shown in Figure 5-16. A resulting state is valid if it is the result listed in the last action item. Then the

verifier chooses the next execution sequence from the list, and continues until all actions have been exhausted. If the resulting state is not what expected, the verifier prints an error message and the exact sequence of actions and states that led to this outcome.

5.4.2 Framework Correctness

In our verification, the first point to consider is to ensure that knowledge of a member joining and leaving the SIP mesh is always noticed to all other members of the mesh. In the execution sequence 1 -> 3 -> 8, this is clear. Once MN4 receives SUBSCRIBE request from MN8, it will update its member list and send its host and member information in the responding NOTIFY request. Thus, MN8 will get MN9's information. Because MN4 works as a gateway node, it sends NOTIFY_d request periodically. In response to these NOTIFY_d requests, MN9 gets MN8's information, etc.. And when executing sequence 1 -> 3 -> 8 -> 11, MN8 works as a gateway node too, it forwards MN9's information to MN6 and forwards MN6's information to MN4. Then MN4 forwards such information to MN9. Thus eventually, all the network members know that a node has joined the mesh and the route connecting to it.

Leaving from the mesh is similar. SUBSCRIBE with Expires = 0 messages are sent to every direct neighboring MNs by leaving MN. Then the leaving information will be forwarded by gateway nodes subsequently. If other neighboring nodes send data due to its out-of-date host lists, the leaving node will send back a SIP 401 Gone response back to keep the sender of data from sending it repeatedly.

Another point to consider is to ensure group messages are forwarded as expected. In the execution sequence 1 -> 3 -> 8 -> 9 -> 11 -> 16, MN8 forwards INVITE request to MN6,

even though it decides not to join. MN8 performs gateway function for its member nodes.

And as we mentioned in Chapter 4, in ALM groups, to keep the knowledge of group membership, a NOTIFY_g request or BYE request will be sent out to all the group members. For example, in the execution sequence 1 -> 3 -> 8 -> 9 -> 11 -> 15 -> 17, MN7 sends INVITE request with "Join" header to the existing ALM group leader: MN4. After accepting such request, MN4 sends NOTIFY_g requests to all group members. The knowledge of MN7 joining the ALM group is propagated to all other members in the ALM group.

Similarly, departure from the ALM group is straightforward. In the execution sequence 1 -> 3 -> 8 -> 9 ->11 -> 14 -> 18, BYE requests are sent to every group member and group leader of the ALM group. Because Event-Ids distinguish instances of ALM group memberships, there is no ambiguity between near-simultaneous departures and re-connections, and thus the analyses of these two scenarios can be considered independently.

5.5 SIP-ALM performance

Several performance metrics have been defined to characterize performance and impacts on the network [59]. Since we implement an application layer meshed network and mainly focus on managing and maintaining the routing information, we only discuss control overhead in this thesis.

5.5.1 Data load of control overhead

Maintaining the overlay topology has a cost. The metric of control overhead considered is in terms of control information exchanged (number of messages processed).

Aside from sending SUBSCRIBE/NOTIFY requests for joining or leaving a SIP meshed

network, our proposed framework requires the exchange of periodic SIP messages in a limited bandwidth environment. In general, the protocol overhead is measured mainly due to routing and group management associated with the SIP-ALM application. Therefore, the protocol overhead increases mainly proportionally to the size of the meshed network and that of ALM groups.

Packet sizes for different messages are noted, as listed in Table 5-4.

Table 5-4 Packet sizes of SIP messages in SIP-ALM

Name of SIP Message	Packet Size (bytes)
SUBSCRIBE	464
200 OK for SUBSCRIBE	337
NOTIFY with 1 profile list and 1 neighbor list	640 (including content length: 193)
NOTIFY with 1 profile list and n neighbor list	600 + (n * 40)
200 OK for NOTIFY	273
INVITE	416
200 OK for INVITE	375
ACK	349
INVITE with Join	395
200 OK for INVITE with Join	336
BYE	373
200 OK for BYE	290

As we see from Table 5-4, the packet size of SIP messages is noticeable. Thus, we try to find ways to save resources on the limited bandwidth:

- In our approach, we use NOTIFY_d messages as "Keep Alive" messages. It also performs neighbor discovery information distribution and is sent out periodically. So except for the first NOTIFY message, the periodic NOTIFY_d messages only convey nodes differential routing information in meshed network. This reduces signaling

traffic substantially.

- We also introduce Dominating Set Routing to reduce the number of exchanged NOTIFY messages. Since not all the network members need to update all other members' information periodically and SIP messages are transferred in Request/Response way, non-gateway nodes use the responding 200 OK message as their "Alive" beacon messages in our framework. It allows us to reduce the number of transmitted messages and the burden of messages in the ad-hoc network and therefore improve the effective use of bandwidth.

In the following section, we make two kinds of comparisons. One is to compare our hierarchical system with the flat topology approach. The purpose of this comparison is to evaluate the scalability of our architecture. The other kind of comparison is to compare two different ways of SIP message transmission in multi-hop environment. It is used to evaluate the optimization of the control message transmission.

5.5.1.1 Traffic of periodic messages

In response to neighbor discovery by sending out NOTIFY and NOTIFY_d request, a node may learn and cache mesh connections to other MNs. This allows the reaction to routing changes to be much more rapid, since a node with multiple routes to a destination can try another cached route if the one it has been using should fail. This caching of multiple routes also avoids the overhead of needing to perform Route Discovery each time a route in use breaks.

In our approach, the SIP nodes in MANET would implement NOTIFY/200 OK scheme as

some form of neighbor discovery to identify nearby mobile nodes with SIP application. This neighbor discovery scheme is made proactive where it periodically checks for neighboring devices or user initiated where the user set off the search.

As we listed in Table 5-2, the size of the headers in NOTIFY message is around 447 bytes. The size of the content including one profile list and one neighbor list of a NOTIFY message is 193 bytes. Its size including one profile list and four neighbor lists is 314 byte. So if all the neighbor information is included in every NOTIFY message, the size of this NOTIFY message would be large. To reduce some transmission load, except for the first time, only differential information will be sent in NOTIFY messages. Thus, when creating a NOTIFY message, the entire host lists is checked. Only the records marked with "Y" in the "Changes" field, will be attached in the content of the NOTIFY message.

Compared to the normal NOTIFY message, the differential NOTIFY message reduces the transferred message size as the size of the mesh network increases. It avoids excessive traffic overhead of periodic transferred message since only updated node information is distributed. Furthermore, using the Dominating Set Routing, the hierarchical network makes the packet transmission effectively.

For example, there are N MNs in the mesh, so if there is no designated gateway nodes, to exchange the topology information, N NOTIFYs will be sent. As the request/answer model of SIP, N 200 OK messages will be transferred too. If there are M gateway nodes among those N MNs, there will be M NOTIFYs sent by M gateway nodes, N 200 OK messages as we suppose that all the MNs in the mesh get noticed. Thus by using the idea from Dominating Set Routing, the number of transferred NOTIFY message is reduced by N-M.

Figure 5-19 displays the different number of messages by using and not using the idea of gateway node.

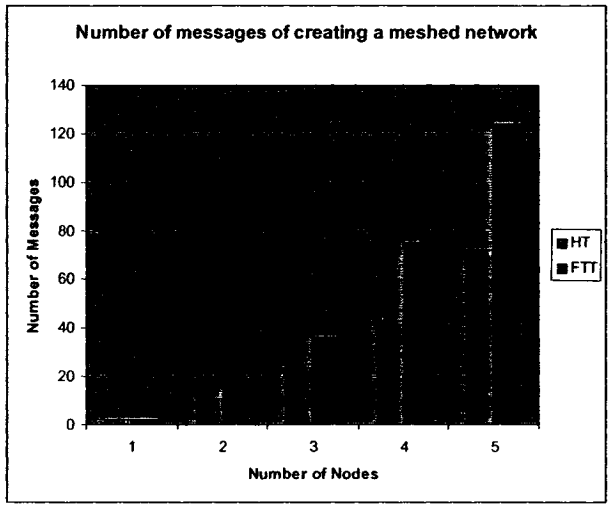

Figure 5-19 Number of messages vs. number of nodes when creating a network (HT means Hierarchical Topology; FTT means Full-Flat Topology)

As demonstrated in our simulation, the number of transferring SIP messages for meshed network management is reduced noticeably as the size of meshed network increases. The bigger the size of that meshed network, the more effective it works.

5.5.1.2 Traffic in the group

The reason of using a group leader to handle joining requests is for the purpose of admission control and to limit the control traffic needed for receivers to join multicast groups. Except for the group members, the MNs in the mesh only store the ongoing ALM group information in its leaders' record. Otherwise, for keeping the integrity of the mesh information, changes in multicast group membership have to be disseminated together with routing-table updates. MNs hear such information from their neighbors and remember which neighbors belong to which groups. Furthermore, every change made in the group membership will cause the

control information to flood between the gateway nodes in the mesh. This is because the changes in group membership are important for the ALM group and should be informed as soon as the changes are noticed. Actually this information is not mandatory for non-group members. So even though sending join requests directly to group leaders need some extra overhead than sending it to the nearest group members, its overall traffic would be less.

As we mentioned earlier, in ALM groups, BYE and NOTIFY_g requests are sent to all group members. Since the senders of those two messages are members of ALM groups, they have the list of all group members already. The messages could be sent in a unicast way or in a multi-recipient way. In the unicast way, the sender sends the request to all the destinations directly. In the multi-recipient way, when a MN receives such requests, it will check whether it is the receiver of these requests, from the values in the "Forward-To" header. If not, the MN will check its host lists and forwards such requests out.

With "Forward-To" header, the overall signaling traffic drops as the number of recipient increases [60]. Instead of using unicast path between any two MNs, no matter how many intermediate nodes it has, a multicast tree provides a single virtual path between any two routers in the tree, the minimum number of copies per packet are used to disseminate packets to all the members of a multicast group. For example, for a tree of N routers, only N-1 links are used to transmit the same information to all the nodes in the multicast tree in a network with point-to-point links.

The improved performance is shown in Figure 5-20. Thus, we deduce that in our implementation, the more the number of group members is, the more efficient signaling transfers by using "Forward-To" header.

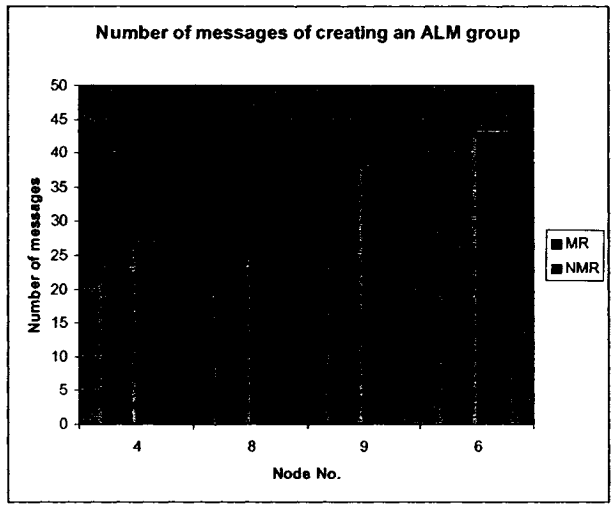

Figure 5-20 An example of overall signaling traffic with/without "Forward-To" extension (MR means Multi-Recipient; NMR means Non-Multi-Recipient)

5.5.2 Message Comparison

AMRoute [14] is an ad hoc multicast protocol that uses the overlay multicast approach. It creates a per group multicast distribution tree using unicast tunnels connecting group members. The protocol has two main components: mesh creation and tree creation. Only the logical core node initiates mesh and tree creation.

AMRoute uses five control messages for signalling purposes and one data message format:

- JOIN_REQ: is broadcast periodically by a logical core for detecting other group segments.
- JOIN_ACK: is generated by a tree node in response to receiving a JOIN_REQ from a different logical core.
- JOIN_NAK: is generated by a tree node if its application leaves the group.
- TREE_CREATE: is generated periodically by a logical core to refresh the group spanning tree.

- TREE_CREATE_NAK: is generated by a tree node to convert a tree link into a mesh link.
- DATA_MESSAGE: is generated by a sender on receiving data from a multicast application.

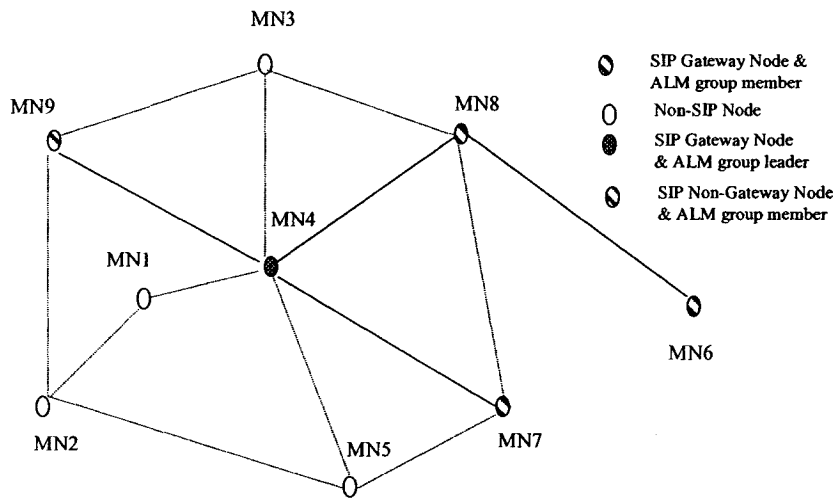

Figure 5-21 A virtual ALM group

Thus to initiate a multicast group in Figure 5-21, there are four JOIN_REQ, four JOIN_ACK and four TREE_CRATE messages in AMRoute. The initiating messages in our proposed framework are 4 INVITE and 4 200 OK messages.

There are two timers for a logical core in AMRoute to send JOIN_REQ and TREE_CREATE messages periodically. If we adopt the value of JON_REQ_SEND and TREE_CREATE_SEND timers to 10 s [14], to keep the multicast group and routes, there will be four JOIN_REQ and four TREE_CREATE messages.

In our proposed framework, since MN4 and MN8 are gateway nodes, there will be five NOTIFY_d and five 200 OK messages to keep the node information. The one additional NOTIFY_d message is because there are two gateway nodes, two NOTIFY_d messages

transfer in the link between MN4 and MN8.

Thus, the number of signalling messages in AMRoute and SIP-ALM are comparable. Moreover, in our proposed framework, those periodical messages do not only maintain multicast group membership, but also meshed network information. Thus nodes in the meshed network could organize more than one multicast groups with the same maintenance messages, instead, in AMRoute, those periodical messages sent by the logical core is only used to maintain that multicast group. As shown in Figure 5-22, as the number of multicast groups in the same meshed network increase, for AMRoute the number of periodical messages increases; for SIP-ALM, the number of periodical messages remains the same.

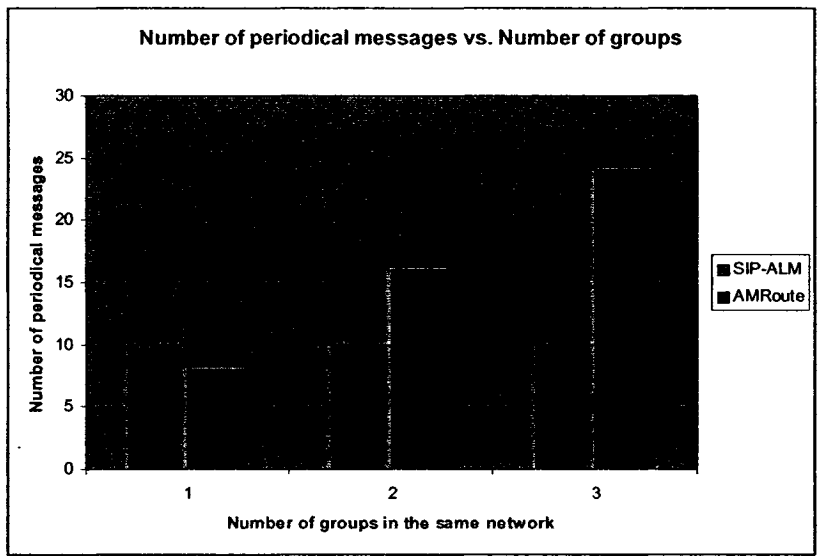

Figure 5-22 An example of overall signaling traffic of maintaining multicast groups

5.5.3 Metrics on end host performance

For the metrics of end host performance, it is the question of how soon the changes in network membership can be found. It could be thought as the impact of the node mobility in

MANET. As we could imagine, as the node moves, it could be in and out of the connection with other nodes. In this case, we will discuss about the ungraceful leaving, and the leaving information transferred in the network.

In our approach, we use 2*NOTIFY_TIMER as a timeout timer to detect the ungraceful leaving of MNs. Whenever a group member notices the ungraceful leaving of another group member, it will send a NOTIFY_g message to inform all the group members. Then there will be no data sent to it, until it re-joins. So the worst case is the packets sent during (2*NOTIFY_TIMER + transmission time to source) are lost.

It is important to note that with higher density of nodes in a given area, there is more overlapping between the coverage areas of nodes, which would result in better (multi-hop) communication between member nodes and result in lower latency.

For the time of joining an ongoing ALM group, it is up to the number of hops between the joiner and the group leader. It could be designed as any group member that may receive such request, and then it will inform all the group members of this new member. But to think about the possible admission control and the volume of maintaining information, we decide to use group leader to handle this kind of request.

5.5.4 Replicate Detection

It is also possible that a MN received a message more than once. To eliminate the redundant requests, the common way is to keep a record in MNs for the receiving requests. Once a request is received, the MN will compare it with its records to decide whether this request is received before or not. To use this method, it needs large memory resources to keep the

records, if this network is an active network.

In SIP, Via header records the intermediate router the SIP message has gone through. Thus here, when MNs receive SIP requests, we check the IP addresses in the Via header. If the local address is not in that field, that means the MN has not received such request before, so it will process the received message. Otherwise, the received message will be discarded. Because the simulation is done in one computer, the added IP address would be the same one. Thus, we did not install the function in our verifier currently.

Also, for the SUBSCRIBE/NOTIFY message, we use the propagation control. Generally, the propagated messages are associated with a Time To Live (TTL). In SIP, the parameter of MAX-FORWARD is used to limit the number of elements a SIP request can traverse. Each time a propagated message is received by a peer, its parameter is decreased by one. If the request contains a Max-Forwards header field with a field value of zero (0), the message will not be forwarded. Thus, SUBSCRIBE/NOTIFY message is only transferred to the neighbors, with MAX-FORWARD equals to 1.

For the SIP messages used in maintaining ALM groups, because all the recipients are listed in the "Forward-To" header, each receiver will remove its URI from that field. So the duplication is prevented.

5.6 Summary

We implemented a SIP-ALM verifier that simulates the scenarios in creating, maintain and leaving SIP meshed network and ALM group. We used JAIN SIP as the basis of the SIP extension. We created test action list to verify the functionality of our proposed network.

Performance measurements are made and preliminary results are obtained from the simulation. We compared the meshed clustering structure with flat topology in our architecture and evaluated the multi-recipient method used in transferring multicast signaling.

Through the work, we can see that our proposed system is feasible to manage application layer multicast group in ad hoc networks. By using mesh clustering structure, the number of transferring SIP messages for meshed network management is reduced noticeably as the size of meshed network increases. The bigger the size of that meshed network, the more effective it works. In our proposed system, by adding "Forward-To" header in multicast signaling messages, the overall signaling traffic increases slightly compared to the increase of the number of recipient.

In this chapter, we also compare our proposed framework with AMRoute theoretically. In the same meshed network, the number of periodical messages of maintaining one multicast group in AMRoute and in SIP-ALM is comparable. Moreover, our proposed framework provides better performance when there are more than one multicast groups in one meshed network.

Chapter 6 Conclusion and Future Work

6.1 Contribution

Multicast is an efficient transmission scheme for supporting group communication in networks. The concept of overlay networks enables multicast to be deployed as a service network rather than a network primitive mechanism, allowing deployment over heterogeneous networks without the need of universal network support. In this thesis we have presented a SIP-ALM middleware framework to be used for applications in ad-hoc networks. The objective of this framework is to allow the users of ad-hoc networks to communicate with each other and exchange application data by using SIP methods in an efficient way. In this thesis, we have:

- Studied the performance difference with AMRoute;
- Analyzed possible network structures: full mesh, tree and clustering, in order to reduce signaling volume;
- Implemented the SIP-ALM framework as an extension to SIP, one new header was introduced for the group management purpose;
- Built a prototype to evaluate the feasibility and performance of the proposed framework.

6.2 Conclusion

The main methods used in transferring SIP messages and routing information in our approach are:

- SIP SUBSCRIBE/NOTIFY messages to convey nodes routing information in meshed network and multicast group. These members exchange information with the nodes nearby and organize host lists in the memory.

  Mobile Nodes are divided as gateway nodes and non-gateway nodes, as introduced in Dominate Routing Set. Thus not all the network members need to record all other members' information, where only gateway nodes store all the routing information and send NOTIFY message.

- NOTIFY message performs the routing (including neighbor discovery) information distribution using periodic updates, without using sequence numbers for updates

- Updating routing table in the host lists is based on the shortest path. Each node selects the shortest path to a destination in term of number of hops to reach that destination. Upon the expiration of the lifetime of route without reception of another NOTIFY/200OK message from the concerned node, this node has to be removed from the host lists.

- Multi-recipient way is used in SIP messages transferred among group members. With all the destinations listed in the "Forward-To" header field, the same packets only traverse a path once. Thus signaling traffic is reduced.

SIP messages inherently require a large amount of data to be exchanged. However, the

methods we adopt allow us to reduce the number of transmitted messages in the ad-hoc network and therefore improve bandwidth use and decrease the collision probability. It also resolves scalability issue due to the fact that it is based on multicast.

A SIP-ALM verifier is implemented to simulate the scenarios in creating, maintain and leaving SIP meshed network and ALM group. The meshed clustering structure is compared with flat topology in our simulation, and the multi-recipient method used in transferring multicast signaling is also evaluated. Through the work, we can see that our proposed system is feasible to manage application layer multicast group in ad hoc networks. By using mesh clustering structure, the number of transferring SIP messages for meshed network management is reduced noticeably as the size of meshed network increases; by adding "Forward-To" header in multicast signaling messages, the overall signaling traffic increases slightly compared to the increase of the number of recipient.

We also compare our proposed framework with AMRoute theoretically. In the same meshed network, the number of periodical messages of maintaining one multicast group in AMRoute and in SIP-ALM is comparable. Moreover, our proposed framework provides better performance when there are more than one multicast groups in one meshed network.

6.3 Future Work

Meeting requirements for the general problem set of application is far too complex. In this paper, we gave out the scheme to create and maintain multicast group problem using SIP in an ad-hoc environment. Thus it leaves the simulation work to address the security issue and evaluate the performance concerning the new scheme in future work. In all, our goal is to

design a new application scheme, which is an efficient framework to find, locate and evaluate the service in vicinity required by client and fit for high dynamic environment.

The multicast transmission of SIP messages is still under research. Also further development and implementation of this middleware may provide detailed information in using SIP methods in ad-hoc networks. Here are some issues suggested for the future work.

6.3.1 Internet Interconnection

In our framework, we set up a subset organized by gateway nodes to share network topology. The gateway node is assigned according to the number of its member nodes. Currently, there is another kind of ad hoc network, which has one or more gateway to make connections between the Internet and the ad hoc network. The existence of such gateway nodes could allow packets to transparently be routed from the ad hoc network to nodes in the Internet and from the Internet to nodes in the ad hoc network and achieve the seamless interoperation between an ad hoc network and the Internet [61].

6.3.2 QoS Issues

Mobile multi-hop wireless networks introduce unique issues and difficulties for supporting QoS in MANET environments. The issues are itemized as follows [62]:

- Node mobility: mobility of the nodes creates a dynamic network topology. Links will be dynamically formed when two nodes come into the transmission range of each other and are torn down when they move out of range.
- Route maintenance: the dynamic nature of the network topology and the changing behavior of the communication medium make the precise maintenance of network

state information very difficult. Thus, the routing algorithms in MANETs have to operate with inherently imprecise information. Furthermore, in ad hoc networking environments, nodes can join or leave at any time. The established routing paths may be broken even during the process of data transfer. Thus, the need arises for maintenance and reconstruction of routing paths with minimal overhead and delay.

SIP can be used to initiate a session that uses some other conference control protocol. Since SIP messages and the sessions they establish can pass through entirely different networks, how to realize the resource reservation capabilities is still under research

6.3.3 Security Issues

The nature of the provided services makes security particularly important. To that end, SIP provides a suite of security services, which include denial-of-service prevention, authentication (both user to user and proxy to user), integrity protection, and encryption and privacy services.

To verify whether the MNs are the right users, some security methods could be adopted in different level. For example, the MANET registration policy could be used during the subscription period.

Even though there are some ideas about authentication in SUBSCRIBE process, like acting as a proxy, using a "401(Bad Request)" response, or a "407 (Proxy Authentication Required)". If authorization fails based on an access list or some other automated mechanism (i.e., it can be automatically authoritatively determined that the subscriber is not authorized to subscribe), the receiver should reply to the request with a "403 Forbidden" or "603 Decline" response.

Currently in SIP, even if an eavesdropper learns the Call-ID, To, and From headers of a dialog, they cannot easily modify or destroy that dialog if Digest authentication or end-to-end message integrity are used.

But which way is more suitable for MNs in MANET still needs to be considered.

6.3.4 Failure of the leader of an ALM group

When a group leader wants to leave the group, it could send BYE message to the nearest group members. Then the new group leader using the same Event-ID and Request-URI to sends the NOTIFY_g message to all group members.

It is possible that the leader of an active ALM group failed or disconnected abruptly. In the profile list of each ALM group member, the Group Member field contains the list of participants of the ALM group. Since it is based on the meshed network, the failure or the disconnection of the leader of an ongoing ALM group should not disrupt the group. When a group member notices that the leader is "missing", it compiles a NOTIFY_g message and sends to all the group members.

This mechanism permits to keep the multicast group even if the leader fails, and in the case of the fragmentation of the ad-hoc network.

Reference

[1] M.Gerla and J.T.Tsai, "Multicaster, mobile, multimedia radio network", ACM-Blatzer Wireless Networks, Vol.1, Issue.3, 1995, pp: 255-265

[2] B.G. Evan, K Baughan, "Visions of 4G", IEEE Electronics & Communications Engineering Journal, Vol.12, Issue.6, Dec. 2000, pp: 293-303

[3] Parasant Mohapatra, Jian Li and Chao Gui, "QoS in Mobile Ad Hoc Networks", IEEE Wireless Communications, Vol.10, Issue.3, June, 2003, pp: 44-52

[4] S.Deering and D.Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs", ACM Transactions on Computer Systems, Vol.8, Issue.2, May 1990, Pp: 85-110

[5] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec and Antony Rowston, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure", IEEE Journal on Selected Areas in Communications, Vol.20, Issue 8, Oct 2002, pp: 1489-1499

[6] Chao Gui and Prasant Mohapatra, "Efficient Overlay Multicast for Mobile Ad Hoc Networks", IEEE Wireless Communications and Networking (WCNC), Vol.2, Mar 2003, pp: 1118-1123

[7] Li Xiao, Abhishek Patil, Yunhao Liu, Linoel M.Ni, and A.-H Esfahanian, "Prioritized Overlay Multicast in Mobile Ad-Hoc Environments", IEEE Computer Magazine, Vol.37, Issue.2, Feb 2004, pp: 67-74

[8] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma, Marcel Waldvogel, "ALMI: An Application Level Multicast Infrastructure", In Proceedings of 3rd Usenix Symposium on Internet Technologies & Systems, Mar 2001, pp: 49-60

[9] Reuven Cohen and Gideon Kaempfer, "A Unicast-based Approach for Streaming Multicast", IEEE INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol.1, 22-26 Apr 2001, pp: 440-448

[10] Carlos de Morais Cordeiro, Hrishikesh Gossain and Dharma P.Agrawal, "Multicast over Wireless Mobile Ad Hoc Networks: Present and Future Directions", IEEE Network, Vol.17, Issue.1, Jan/Feb 2003, pp: 52-59

[11] E.Royer, and C.E.Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol", Proc. Of the 5th ACM/IEEE Annual Conf. on Mobile Computing and

Networking, Aug 1999, pp: 207-208

[12] Sang Ho Bae, Sung-Ju Lee, William Su, Mario Gerla,, "The design, implementation, and performance evaluation of the on-demand multicast routing protocol in multihop wireless networks", IEEE Network, Vol.14 , Issue. 1 , Jan-Feb 2000, pp:70 - 77

[13] C.W.Wu, Y.C.Tay, "AMRIS: a multicast protocol for ad hoc wireless networks", IEEE Military Communications Conference Proceedings, MILCOM 1999, Vol.1, 31 Oct-3 Nov 1999, pp: 25-29

[14] Jason Xie, Rajesh R.Talpade, Anthony Mcauley and MingYan Liu, "AMRoute: Ad Hoc Multicast Routing Protocol", Mobile Networks and Applications, Vol.7, 2002, pp: 429-439

[15] J.J.Garcia-Luna-Aceves and E.L.Madruga, "The Core-Assisted Mesh Protocol", IEEE Journal on Selected Areas in Communications, Vol.17, Issue.8, Aug 1999, pp:1380-1394

[16] Alexander Klemm, Christoph Lindemann and Oliver P.Waldhorst, "A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks", Proc.IEEE Semiannual Vehicular Technology Conference 2003, VTC2003, 6-9 Oct 2003, Vol.4, pp: 2758-2763

[17] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "Session Initiation Protocol", RFC 3261, June 2002.

[18] 3rd Generation Partnership Project (3GPP), http://www.3gpp.org

[19] 3rd Generation Partnership Project 2 (3GPP2), http://www.3gpp2.org

[20] Mobile Wireless Internet Forum, http://www.openmobilealliance.org/tech/affiliates/mwif/mwifindex.html

[21] R. Mahy, B. Campbell, R. Sparks, J. Rosenberg, D. Petrie, A. Johnston, "A Call Control and Multi-party usage framework for the Session Initiation Protocol (SIP)", draft-ietf-sipping-cc-framework-02.txt, March 7, 2003

[22] Jonathan Lennox, Henning Schulzrinne, "A Protocol for Reliable Decentralized Conferencing", Proceedings of the 13th International workshop on Network and operating systems support for digital audio and video, NOSSDAV'03, June 1-3, 2003, pp: 72-81

[23] Hechmi Khlifi, Anjali Agarwal, Jean-Charles Grégoire, "A Framework to Use SIP in Ad-hoc Networks", IEEE CCECE 2003, Vol.2, 4-7 May 2003, pp: 985-988

[24] NodoKa Mimura, Kiyohide Nakauchi, Hiroyuki Morikawa and Tomonori Aoyama, "RelayCast: A Middleware for Application-level Multicast Services", Proc.3rd International Symposium on Cluster Computing the Grid, May 12-15, 2003, pp: 434-441

[25] Liang Cheng and Ivan Marsic, "Service Discovery and Invocation for Mobile Ad Hoc Networked Appliances", Proc.2nd International Workshop on Networked Appliances (IWNA'2000), Nov.30-Dec.1, 2000

[26] Marco Conti, Giovanni Turi, Gaia Maselli, Jon Crowcroft, Sven Ostring, Pietro Michiardi, Refik Molva, Jose Costa Requena, Piergiorgio Cremonese, Veronica Vanni, Ivan Defilippis, Silvia Giordano, Alessandro Puiatti, " MobileMAN – Architecture, protocols and services", http://cnd.iit.cnr.it/mobileMAN/deliverables/MobileMAN_Deliverable_D5.pdf, MobileMan Technical Report, Oct 2004

[27] Sumi Helal, Nitin Desai and Varum Verma, "Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks", Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC), Vol.3, March 2003, pp: 2107-2113

[28] Sun Microsystems, http://wwws.sun.com/software/jini/

[29] J.Veizades, E.Guttman, C.Perkins and S.Kaplan, "Service Location Protocol", RFC 2165, Internet Engineering Task Force, June 1997.

[30] J.Rosenberg and H.Schulzrinne, "Guidelines for Authors of Extensions to the Session Initiation Protocol (SIP)", draft-ietf-sip-guidelines-07.txt, October 27, 2004

[31] AVAYA Inc., "Evolving to Converged Communication with Session Initiation Protocol (SIP)", http://www1.avaya.com/enterprise/whitepapers/lb2337.pdf, white paper, February 2004

[32] A. B. Roach, "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002

[33] Yang-hua Chu, Sanjay GRao, Srinivasan Seshan and Hui Zhang, "A Case for End System Multicast", IEEE Journal on Selected Areas in Communications, Vol.20, Issue.8, Oct 2002, pp: 1456-1471

[34] Cristian Tuduce and Thomas Gross, "Organizing a Distributed Application in a Mobile Ad hoc Network", Proceeding of the Second IEEE International Symposium on Network Computing and Applications (NCA'03), Apr 2003, pp: 231-238

[35] Barbosa e Oliveira et al., "Evaluation of ad-hoc routing protocols under a peer-to-peer application", in IEEE Wireless Communications and Networking (WCNC), March 2003, Volume: 2, pp: 1143-1148

[36] S.Donovan, "The SIP INFO Method", RFC 2976, October 2000

[37] J.Rosenberg, H.Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP), RFC 3262, June 2002

[38] J.Rosenberg, "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, September 2002

[39] B.Campbell, H.Schulzrinne, C.Huitema, D.Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December, 2002

[40] R.Sparks, "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003

[41] J.Rosenberg, "A Session Initiation Protocol (SIP) Event Package for Registrations", RFC 3680, March 2004

[42] Upkar Varshney, "Multicast Over Wireless Networks", Communication of the ACM, Vol.45, No.12, Dec 2002, pp: 31-37

[43] Jie Wu, "Extended Dominating-Set-Based Routing in Ad Hoc Wireless Networks with Unidirectional Links", IEEE Transactions on Parallel and Distributed Systems, Vol.13, No.9, September 2002, Pp: 866-881

[44] Kai Chen and Klara Nahrstedt, "Effective Location-Guided Tree Construction Algorithms for Small Group Multicast in MANET", Proceedings of IEEE 21st Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2002, Vol.3, 23-27 June 2002, pp: 1180-1189

[45] Jie Wu and Hailan Li, "Dominating-Set-Based Routing Scheme in Ad Hoc Wireless Networks", Wlley Series on Parallel And Distributed Computing, Handbook of wireless networks and mobile computing, 2002, pp: 425-450

[46] Thiagaraja Gopalsamy, Mukesh Singhal, D.Panda and P.Sadayappan, "A Reliable Multicast Algorithm for Mobile Ad hoc Networks", Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), 2-5 Jul 2002, pp: 563-570

[47] Nodoka Mimura, Kiyohide Nakauchi, Hiroyuki Morikawa, Tomonori Aoyama, "RelayCast: A Middleware for Application-level Multicast Services", 3rd International Symposium on Cluster Computing the Grid, May 12-15, 2003, pp: 434-441

[48] Igor Miladinovic, Johannes Stadler, "Multi-Recipient Request in the Session Initiation Protocol", International Conference on Applied Informatics, Innsbruch, Austria, Feb 10-13, AI PDCN 2003, Proceedings of the 21st IASTED International Conference APPLIED INFORMATICS, pp: 801-806

[49] R. Mahy, D. Petrie, "The Session Initiation Protocol (SIP) "Join" Header", RFC 3911, October 2004

[50] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Antony Rowstron, "Scalable application-level anycast for highly dynamic groups", Technology and Business Models, 5th COST264 International Workshop on Networked Group Communications, NGC 2003, Munich, Germany, September 16-19, 2003, pp: 47-52

[51] Jie Wu and Hailan Li, "A Dominating-Set-Based Routing Scheme in Ad Hoc Wireless Networks", Telecommunication Systems, A special issue on Wireless Networks, Volume:18, Issue 1, Sep 2001, pp: 13-36.

[52] Sun Microsystems, "The JAIN APIs: Integrated APIs for the Java Platform", White Paper, 2002

[53] Sun Microsystems, "JAIN Technology: Serving the Developer Community", http://java.sun.com/products/jain/jain0503.pdf, 2003

[54] Sun Microsystems, "JAIN Introduction", jp.sun.com/solutions/infra/jws/integration/pdf/MasafumiWatanabe.pdf, 2004

[55] Yongguang Zhang and Wei Li, "An integration Environment for Testing Mobile Ad-Hoc Networks", Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, MOBIHOC'02, June 9-11, 2002, EPFL Lausanne, Switzerland, pp: 104-111

[56] H.Schulzrinne and J.Rosenberg, "SIP call control services", draft-ietf-mmusic-sip-cc-01.txt, Internet draft, Internet Engineering Task Force, June 1999, Work in progress

[57] Jonathan Lennox, Henning Schulzrinne, "A Protocol for Reliable Decentralized Conferences", NOSSDAV'03, June 1-3, 2003, Monterey, California, USA, pp: 72-81

[58] David Lee, Dongluo Chen, Ruibing Hao, Raymond E.Miller, Jianping Wu and Xia Yin, "A Formal Approach for Passive Testing of Protocol Data Portions", Proceedings of the 10th IEEE International Conference on Network Protocols, ICNP'02, 12-15 Nov 2002, pp: 122-131

[59] Ayman El-Sayed, Vincent Roca, "A Survey of Proposals for an Alternative Group Communication Service", IEEE Network, Vol.17, Issue.1, January/February 2003, pp: 46-51

[60] I.Miladinovic, J.Stadler, "A simulation study of multi-recipient message in the Session Initiation Protocol", The 8th International Conference on Communication Systems, 2002, ICCS 2002, Vol.2, pp: 977-982

[61] David B.Johnson, David A.Maltz, Josh Broch, "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks" in Ad Hoc Networking, edited by Charles E.Perkins, Chapter 5, Addison-Wesley, 2001, pp: 139-172

[62] Parasant Mohapatra, Jian Li and Chao Gui, "QoS in Mobile Ad Hoc Networks", Special Issue on QoS in Next-Generation Wireless Multimedia Communications Systems in IEEE Wireless Communications Magazin, Vol.10, Issue.3, June 2003, pp: 44-52