P2P SIP over mobile ad hoc networks

Thirapon Wongsaardsakul

Ecole Doctorale EDITE

Thèse présentée pour l'obtention du diplôme de Docteur de Télécom & Management SudParis

Doctorat conjoint TMSP-UPMC

Spécialité :

Par
M. WONGSAARDSAKUL Thirapon

Titre
P2P SIP over Mobile Ad Hoc Networks

Soutenue le **4 Octobre 2010 devant le jury composé de :**

Assoc. Prof. Bjorn Landfeldt	Rapporteur
Assoc. Prof. Teerapat Sanguankotchakorn	Rapporteur
Prof. Kanchana Kanchanasut	Examinateur
Prof. Serge Fdida	Examinateur
Prof. Djamal Zeghlache	Examinateur
Prof. Noël Crespi	Directeur de thèse

Thèse n° 2010TELE0021

Acknowledgment

I would like to express my sincere gratitude to Prof. Kanchana Kanchanasut, my advisor, for her constant guidance, insightful ideas, priceless suggestions, and encouragement during the period of my study. Moreover, she gave me a great opportunity to join the DUMBO project in which I implemented and tested my software prototype. The two years I spent in the project were precious as I learnt many wonderful things, gained invaluable knowledge, and developed close-knit friendships.

I am deeply grateful to Prof. Noël Crespi who gave me the chance to work with him and his team at TELECOM & Management SudParis. He also gave me innumerable advice which I found to be very beneficial. I would also like to express my gratitude to Dr. Teerapat Sanguankotchakorn, who served as a committee member for my dissertation, for his invaluable comments and suggestions.

Moreover, I am also indebted to Assoc. Prof. Bjorn Landfeldt, my external examiner, for his valuable comments and suggestions towards the improvement of my dissertation.

I would also like to acknowledge and express my appreciation for the financial support I received from Bangkok University to fund my studies at AIT along with France's cooperation at AIT which provided me with a scholarship for my study in France.

I would like to thank all my friends at intERLab and CSIM for their friendship, support, and encouragement. Special thanks to Mr. Manutsiri Chansutthirangkool for his friendship, help, and ideas. I also wish to thank Mr. Dwijendra Kumar Das and Ms. Nisarat Tunsakul who are and will always be my best friends.

Finally, I would like to thank my beloved family and my wife for their understanding and support when I faced arduous times during the course of my study.

Résumé

Cette thèse propose une nouvelle architecture Peer-to-Peer pour l'établissement de sessions SIP (Session Initiation Protocol) sur les réseaux ad hoc. SIP est un protocole conçu à l'origine sur un modèle centralisé est n'est pas nativement adapté aux réseaux mobiles ad hoc (MANET) en raison de leurs caractéristiques inhérentes de mobilité. Nous avons ciblé nos études sur un mécanisme de lookup distribué Peer-to-Peer (P2P) tolérant aux fautes, même en cas de mobilité des nœuds du réseau.

Cette thèse s'articule autour de quatre principales contributions:

Nous introduisons le concept de Structured Mesh Overlay Network (SMON) : un overlay P2P sur MANET permettant d'effectuer des lookups de ressources rapides dans un environnement ad hoc. SMON utilise une architecture cross layer design basée sur une Distributed Hash Table (DHT) utilisant directement les informations de routage OLSR. Cette architecture cross layer permet d'optimiser les performances du réseau overlay lors d'un changement de topologie du réseau.

La seconde contribution, SIPMON, est un overlay SIP sur réseau SMON. Sa particularité est d'utiliser un DHT pour distribuer les identifiants d'objet SIP dans le réseau overlay SMON. Les expérimentations menées prouvent que cette approche garantit une durée de découverte SIP constante et permet un établissement de session plus rapide entre deux usagers sur réseau ad hoc. SIPMON ne s'applique cependant qu'à un réseau MANET isolé.

Notre troisième contribution SIPMON+ permet un interfonctionnement de plusieurs overlays SIPMON connectés à Internet. SIPMON+ unifie donc les overlays de réseau et permet de joindre un client SIP qu'il soit localisé sur un réseau ad hoc ou sur l'internet. De plus, SIPMON+ permet une continuité de service sans couture lors du passage entre un réseau MANET et un réseau d'infrastructure. Notre prototype a démontré que les performances de temps d'établissement d'appel SIPMON+ étaient meilleures que pour l'approche concurrente MANEMO (MANET for Network Mobility).

Le scénario d'usage principal est la fourniture de services de communication multimédia d'urgence rapidement déployables en cas de catastrophe majeure. Nous avons développé un prototype SIPMON+ totalement fonctionnel de service de communication P2P multimédia. Ce prototype a été expérimenté en situation réelle de catastrophe. Notre prototype sans infrastructure a donné de biens meilleurs résultats que MANEMO en termes de temps de déploiement, de taux de perte de paquets et de temps d'établissement d'appel.

Mots-Clés: MANET, Peer-to-Peer, SIP, Distributed Hash Table, Voix sur IP, la mobilité du terminal, OLSR, Réseau d'urgence.

Abstract

This work presents a novel Peer to Peer (P2P) framework for Session Initiation Protocol (SIP) on Mobile Ad Hoc Network (MANET). SIP is a client-server model of computing which can introduce a single point of failure problem. P2P SIP addresses this problem by using a distributed implementation based on a P2P paradigm. However, both the traditional SIP and P2P SIP architectures are not suitable for MANETs because they are initially designed for infrastructured networks whose most nodes are static. We focus on distributed P2P resource lookup mechanisms for SIP which can tolerate failures resulting from the node mobility. Our target application is SIP-based multimedia communication in a rapidly deployable disaster emergency network. To achieve our goal, we provide four contributions as follows.

The first contribution is a novel P2P lookup architecture based on a concept of P2P overlay network called a Structured Mesh Overlay Network (SMON). This overlay network enables P2P applications to perform fast resource lookups in the MANET environment. SMON utilizes a cross layer design based on the Distributed Hashing Table (DHT) and has direct access to OLSR routing information. Its cross layer design allows optimizing the overlay network performance during the change of network topology.

The second contribution is a distributed SIP architecture on MANET providing SIP user location discovery in a P2P manner which tolerates single-point and multiple-point of failures. Our approach extends the traditional SIP user location discovery by utilizing DHT in SMON to distribute SIP object identifiers over SMON. It offers a constant time on SIP user discovery which results in a fast call setup time between two MANET users. From simulation and experiment results, we find that SIPMON provides the lowest call setup delay when compared to the existing broadcast-based approaches.

The third contribution is an extended SIPMON supporting several participating MANETs connected to Internet. This extension (SIPMON+) provides seamless mobility support allowing a SIP user to roam from an ad hoc network to an infrastructured network such as Internet without interrupting an ongoing session. We propose a novel OLSR Overlay Network (OON), a single overlay network containing MANET nodes and some nodes on the Internet. These nodes can communicate using the same OLSR routing protocol. Therefore, SIPMON can be automatically extended without modifying SIPMON internal operations. Through our test-bed experiments, we prove that SIPMON+ has better performance in terms of call setup delay and handoff delay than MANET for Network Mobility (MANEMO).

The fourth contribution is a proof-of-concept and a prototype of P2P multimedia communication based on SIPMON+ for post disaster recovery missions. We evaluate our prototype and MANEMO-based approaches through experimentation in real disaster situations (Vehicle to Infrastructure scenarios). We found that our prototype outperforms MANEMO-based approaches in terms of call setup delay, packet loss, and deployment time.

Keyword: MANET, Peer-to-Peer, SIP, Distributed Hash Table, Voice over IP, Terminal Mobility, OLSR, Emergency Network.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AODV	Ad-hoc On-Demand Distance Vector routing
CBRP	Cluster-Based Routing Protocol
CN	Correspondent Node
CQSA	Client Query Service Advertising
DHT	Distributed Hash Table
DSR	Dynamic Source Routing
DUMBO	Digital Ubiquitous Mobile Broadband OLSR
EasyDC	Easy Disaster Communication
HA	Home Agent
HQ	Headquarter
MANEMO	MANET for Network Mobility
MANET	Mobile Ad hoc NETwork
MN	Mobile Node
MNN	Mobile Network Node
MPR	Multi Point Relay
MR	Mobile Router
NINA	Network In Node Advertisement
OLSR	Optimize Link State Routing
OON	OLSR Overlay Network
P2P	Peer to Peer
PDSR	Peer Computing based Dynamic Source Routing
RERR	Route Error
RREP	Route Reply
RREQ	Route Request
RTP	Real-time Transport Protocol
SIP	Session Initiation Protocol
SLE	Service Location Extension
SLP	Service Location Protocol
SMON	Structured Mesh Overlay Network
TBRPF	Topology Dissemination Based on Reverse-Path Forwarding routing
TDP	Tree Discovery protocol
URI	Uniform Resource Identifier
VoIP	Voice over Internet Protocol

Chapter 1

Introduction

IP telephony has become one of the most popular applications on the Internet today as it does not require heavy additional investment on end user equipment nor costly traditional telephone infrastructure. Such application involves signaling protocols that initiate, maintain, modify, and terminate communication sessions between users, which have been specified by the IETF RFC 3261 Session Initiation Protocol (SIP). This IP telephony application is particularly attractive as an alternative means of communication when traditional telecommunication infrastructure is not available such as those in rural areas or in post-disaster scenarios. Temporary or local area networks, deploying wireless technologies, can be set up rapidly at low cost for isolated communities to provide IP telephony as well as some other multimedia applications. These networks can eventually become an edge network for the global Internet infrastructure once a point of connection to the Internet becomes available. This dissertation provides a framework to set up IP telephony services based on SIP for these isolated temporary or ad hoc networks which could subsequently be integrated with the global Internet infrastructure. The proposed service relies on a Peer-to-Peer (P2P) SIP over structured overlay mesh network on mobile ad hoc network which will be represented as SIPMON in the entire dissertation.

This chapter provides an overview of the SIP protocol and the networks upon which our framework can be applied. It then provides a broad description of the proposed framework followed by the contributions of this dissertation in the context of related work. Finally, the organization of the dissertation is presented.

1.1 Session Initiation Protocol (SIP)

SIP (Rosenberg et al., 2002) is a signaling protocol which allows users to locate other users and exchange multimedia session parameters, or Session Description Protocol (SDP) (Handley & Jacobson, 1998). SIP handles user mobility by providing a location service, which is a database service. It contains a list of user binding information of address of record or SIP Uniform Resource Identifier (URI) to IP addresses. SIP URI has a similar format the email addresses; thus, a user can use existing email addresses as SIP URI's without having to acquire a new address, e.g. *"sip:user@domain.com"*. The location database is updated upon movement and queried whenever there is a lookup for a SIP object. SIP supports four types of mobility, namely, terminal, session, personal, and service mobility. Nevertheless, the most important one for an IP phone conversation is terminal mobility as a user terminal can move from one location to another while a session's continuity is maintained.

Session setup, one of the SIP operations, is a process of establishing agreed session parameters between a caller and a callee, the so-called "dialing" and "ringing". Before the caller can make a call over the Internet, it must know IP address of the callee. A call invitation, INVITE request, is directly sent from the caller to the callee as a P2P communication.

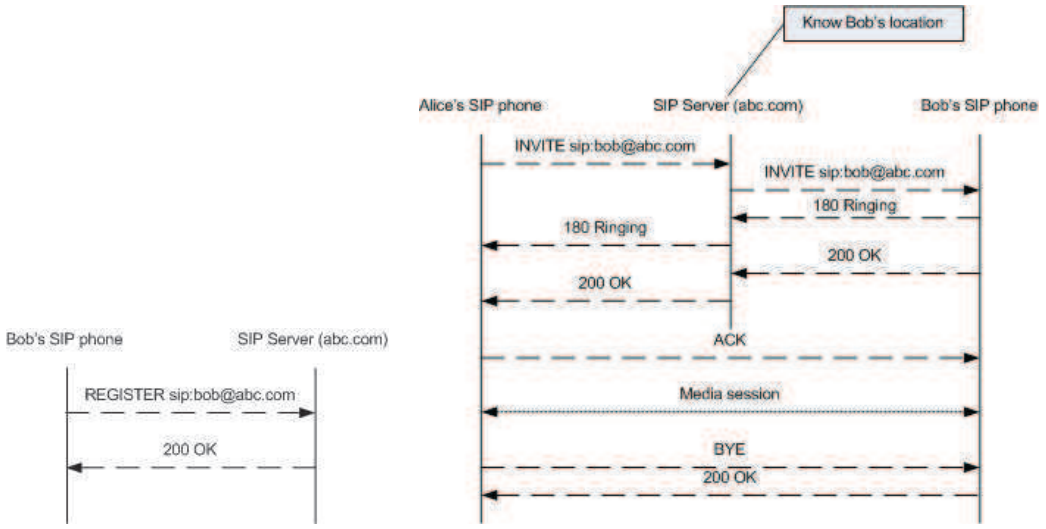
Using the IP address to identify the location of the callee is inconvenient since an IP address is not permanently assigned. Two scenarios can cause IP addresses to change. 1) User mobility, either the caller or the callee moves to a new machine. User mobility can affect session mobility when the user wishes to transfer an ongoing session to another terminal. However in this dissertation, we do not consider session mobility for the case of IP telephony as supported by our framework. 2) Terminal mobility, either machine of caller or callee acquires a new IP address due to a move to another network.

SIP provides users with a location discovery allowing the caller to make a call without knowing the callee's location. Both caller and callee are required to register their identities and addresses or bindings to a centralized SIP server whose address is known to them. They must update this binding to the SIP server whenever they change IP addresses in order to maintain valid binding information at the server. The binding update ensures that SIP users are reachable regardless of their locations. Instead of sending the call invitation to the callee, the caller requests the SIP server to perform user location discovery and forward the INVITE request on its behalf. Fig. 1.1 shows simplified examples of registration and session setup between two end users.

SIP supports terminal mobility, where a terminal changes locations, i.e. moves to other networks, while ongoing sessions are maintained. Once a conversation starts between SIP users A and B, they maintain a SIP dialog and the parameters of their ongoing session. When either one of them changes his/her IP address due to this mobility, it must perform two operations in sequence by sending a new binding update and by sending re-INVITE by using the maintained SIP dialog and parameter information to resume the ongoing session. For example, if Bob's machine moves to a new network during a call, since the address is changed, his binding information is invalid, and the ongoing session will be terminated. First, Bob sends a new binding update containing his new address to the SIP server. Second, he uses the re-INVITE request, without making a new call, to inform Alice about his new IP address to resume the ongoing session as shown in Fig. 1.2. Thus, the user location discovery and registration are very important for SIP operations to address problems with mobility.

1.2 The Internet and Mobile Ad Hoc Networks

The Internet is a network connecting many networks together and now covers the entire globe. It is based on a protocol suite called TCP/IP (Cerf et al., 1974; Postel, 1981). It can be said that the Internet is an infrastructured network as it relies on nodes and gateways which are at fixed locations connecting eventually to the Internet backbones; mobile nodes wirelessly connect themselves to fixed access points or gateways. Another type of network which does not rely on infrastructure or fixed network connectivity is referred to as Mobile Ad Hoc Networks (MANET), or infrastructureless networks. This type of network is a self-configuring network. It is formed by wireless mobile nodes without the use of any network infrastructure. Each mobile node can randomly move and at the same time acts as a router discovering and maintaining route information that is used for multi-hop communication. They can be characterized by unpredictable topology changes, high degree of mobility, low bandwidth and intermittent connections. Routing protocols



(a) Registration

(b) Session setup (adapted by author from Rosenberg et al. (2002))

Figure 1.1: The simplified SIP examples

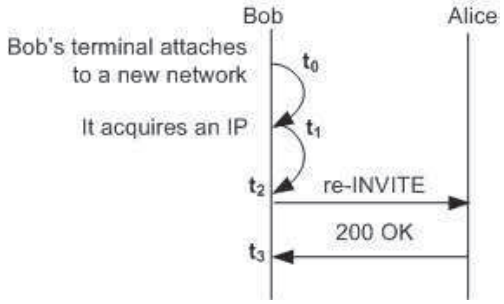


Figure 1.2: re-INVITE request for resuming the ongoing session

for these networks take care of the fragile characteristics of MANET's connectivity. Some of these well-known protocols are the Optimize Link State Routing (OLSR) (Clausen & Jacquet, 2003) and the Ad-hoc On-Demand Distance Vector routing (AODV) (Perkins et al., 2003). Mobile Ad Hoc Networks have been designed to be interoperable with the Internet; thus, they can form edge networks for the Internet and help expand network connectivity far beyond those provided by the infrastructure.

These types of networks are particularly useful in rural areas or for rapid post-disaster environments where communication networks must be set up rapidly to provide communication services among members of the local community. Sometimes only MANETs are deployed while other times an integration of a single or several MANETs with the Internet may be found. We will refer to these types of networks, which require rapid set up and deployment time, as *emergency networks*.

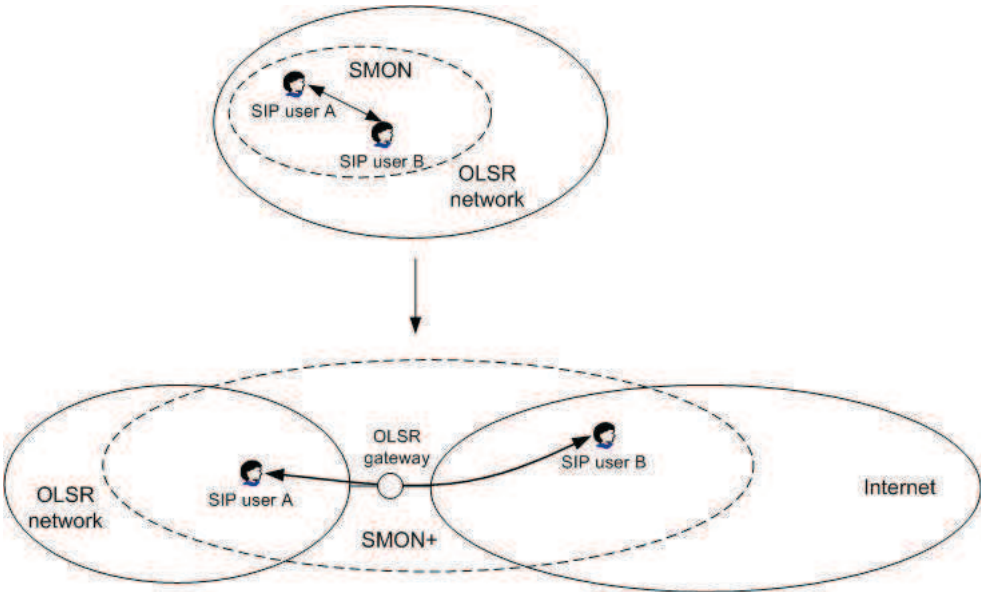


Figure 1.3: SMON+ covering MANET and the Internet

1.3 An Overview of P2P SIP on MANET and the Internet

This dissertation proposes a framework based on the following concepts: P2P overlay network on MANET and a distributed SIP over this P2P overlay structure to prevent problems associated with single point of failure. The P2P SIP overlay network is composed of the proposed P2P overlay network called Structured Mesh Overlay Network (SMON) and a distributed system of SIP servers. SMON is a cross-layer design that can optimize overlay network performance when the network topology is changed by extracting the topological updates from the network layer. Besides, SMON dynamically selects one node which is delegated to broadcast periodically topological update messages. This mechanism significantly reduces control overhead when compared with other SLP based methods. Moreover, OLSR is used for an underlying routing protocol that allows SMON to flood its messages via Multi Point Relays (MPRs), which further reduces duplicated retransmissions during the flooding procedure when compared to regular broadcast techniques (Laouiti, Qayyum, et al., 2002). Each SMON node has its own SIP server that provides P2P user location discovery and registration and handles terminal mobility. The P2P SIP overlay network deploys the concept of Distributed Hash Table (DHT) (Karger et al., 1999) to distribute efficiently SIP object identifications over the P2P SIP overlay network.

To handle the integration of MANET and the Internet in our framework, we propose the stretching of SMON from MANET to cover those peers residing on the Internet hence creating an extended SMON or SMON+. The P2P SIP overlay network can then be expanded to the Internet in order to deliver seamless SIP mobility support for users on both MANET and the Internet, as shown in Fig. 1.3.

A prototype of P2P SIP overlay network with P2P multimedia communication, called Easy Disaster Communication (EasyDC), has been implemented and tested in simulated

post-disaster recovery operations as proof of the viability of our proposed concept and framework.

We evaluate our framework based on its efficiency in providing terminal mobility which is an important criterion for IP telephony, particularly in emergency networks. SIP terminal mobility was something which was not addressed in existing SIP on MANET research (Fu et al., 2005; Leggio et al., 2005; Khlifi et al., 2003; Banerjee & Acharya, 2004; L. Li & Lamont, 2004; Yu & Agarwal, 2005; Castro & Kassler, 2006; Zhang et al., 2006; Stuedi et al., 2007; Fudickar et al., 2009), while handoff delay was used as a performance metric for one-hop mobile network (Nakajima et al., 2003; Jung et al., 2003; Dutta et al., 2004; Yeh et al., 2006; Mohanty & Akyildiz, 2007; Zeadally & Siddiqui, 2007) in infrastructured networks.

Performance evaluation of EasyDC is conducted through simulations and experimentations under an emergency network environment, where a low call setup delay, terminal mobility, and fast network deployment are critically required. Thus we consider post dialing delay, re-INVITE handoff delay, and network deployment as the main performance metrics in our evaluation. The post dialing delay is the difference between the amount of time that a caller uses in sending a call invitation and the time needed to receive a ring in return. The re-INVITE handoff delay is the time used to resume the ongoing session between two end users after the terminal mobility occurs. The results of simulations and experimentations show that the post dialing delay is less than 100 ms and re-INVITE handoff delay is around 32 ms, both of which are considered to be very low delays and are suitable for emergency networks. For network deployment, EasyDC can be practically operated on any computer with a standard wireless interface, and does not require centralized servers. Thus, regular laptops that have EasyDC installed can be setup in order to establish an emergency network within a short period of time.

1.4 Problem Statement

To provide multimedia communications in MANETs for a disaster emergency network, SIP remains an attractive protocol because it is a pre-defined standard signaling protocol which the majority of telephony applications use. However, SIP is designed for infrastructured networks using centralized architecture and is thus not suitable for MANETs due to their high degree of mobility and other constraints such as noise, fading, and interference. If a centralized SIP server on a MANET node fails, the entire community on the network cannot use its services and this is referred to as a single point of failure. A number of research work (Fu et al., 2005; Leggio et al., 2005; Khlifi et al., 2003; Banerjee & Acharya, 2004; L. Li & Lamont, 2004; Zhang et al., 2006; Stuedi et al., 2007) have addressed the issue of deploying SIP as user location discovery on MANETs allowing a SIP user agent to locate another user agent IP address without depending on centralized servers, but these researchers make use of regular broadcast techniques such as Service Location Protocol (SLP) to announce the user location that add high traffic to the network. Moreover, the terminal mobility problem is still not addressed in these approaches.

1.5 Contributions

In this dissertation, we study the problems related to SIP functionalities over MANET. We focus on the distributed resource lookup mechanisms which can tolerate single-point and multiple-point failures and which can effectively handle terminal (or node) mobility. Our target application is IP telephony and multimedia communication in a rapidly deployable disaster emergency network. Emergency workers can carry and communicate using self-configuring and self-healing portable communication terminals. In most emergency networks, terminals may move and are subject to physical failures and radio outages. Starting, maintaining, and recovering communication throughout an emergency mission would prove very challenging. We provide answers to address these challenges vis-*à*-vis disaster emergency communication with the following four contributions.

The first contribution is the design and validation of a novel fast P2P lookup architecture called Structured Mesh Overlay Network (SMON). It enables P2P applications to perform fast lookups in the MANET environment. SMON is a fully connected mesh overlay topology which allows for a constant lookup time when a terminal needs to locate the destination of a specific resource (e.g. a file). SMON utilizes a cross layer design based on the Distributed Hashing Table (DHT) and has direct access to OLSR routing information. SMON allows the resulting overlay network topology to dynamically adjust to the MANET network topology. We validate SMON through the simulations of heterogeneous MANET terminals in two mobility scenarios. One scenario is a combination of heterogeneous mobile nodes with different transmission ranges and bandwidths. The other scenario is a Wireless Mesh Networking (WMN) consisting of a MANET connected to a fixed network via a gateway. We discover that SMON provides the best lookup times as compared to existing work, i.e. passive (push model) and active (pull model) resource discovery protocols not based on the concept of an overlay network.

The second contribution is a distributed SIP architecture running on SMON which we call SIPMON. SIPMON allows for the creation of an overlay network among cooperating SIP peers. SIPMON replaces a centralized SIP server architecture; hence, it is able to better handle single and multiple terminal failures. Our approach extends the traditional SIP user location discovery by utilizing DHT in SMON to distribute SIP object identifiers over the Mesh Overlay Network. It guarantees a constant time on P2P SIP user discovery in the MANET environment. Other approaches use broadcasting techniques to discover the locations of the target users before initiating a SIP request. Broadcasting approaches result in high control overhead and lookup delay since each node must process SIP requests upon receiving them. In our approach, SIP messages are exchanged among SIPMON peers via regular unicast packets. This results in significantly smaller control overhead as opposed to the broadcasting approaches. We evaluate SIPMON through simulations and empirical experiments. We found that SIPMON provides the lowest call setup delay when compared to the existing broadcast-based approaches.

The third contribution is that we study a scenario where an emergency overlay network consists of several participating MANETs and each of these participating MANETs has a public Internet connectivity. SIP users (e.g. emergency workers) positioned inside the emergency overlay network should be able to communicate as long as they are connected to one of the participating MANETs. We assume the uniqueness of SIP user identities, and

moderate user mobility where each SIP user (i.e. on a terminal) may occasionally join or leave one of the participating MANETs. We propose a novel framework called SIPMON+ to support SIP user mobility. SIPMON+ consists of SIPMON co-functioning with an enhanced version of OLSR gateway called Overlay OLSR Network (OON). SIPMON+ relies on the OON layer to exchange OLSR messages among the participating MANETs via the public Internet. Therefore, the reach-ability status of SIP users can be made known across the participating MANETs. Through our testbed experiments, we evaluate SIPMON+ versus MANET for Network Mobility (MANEMO). We find that SIPMON+ has better performance in terms of call setup delay and handoff delay. Apart from this finding, we also show that SIPMON+ is more scalable in terms of control overhead size and the number of SIP users supported by SIPMON+.

The fourth contribution is a proof-of-concept and fully-functional prototype of P2P multimedia communication based on SIPMON+ for post-disaster recovery missions. We use the Vehicle to Infrastructure (V2I) scenario to evaluate three different approaches: MANEMO based OLSR, MANEMO based TDP/NINA (Thubert et al., 2007, 2008), and SIPMON+. We find that SIPMON+ outperforms the two MANEMO approaches in terms of call setup delay, packet loss, and deployment time, all of which are important factors when deploying an emergency network. The reason is because SIPMON+ uses the SIP re-INVITE mechanism to resume an ongoing session when a terminal moves without using centralized servers and IP tunneling mechanism commonly found in the MANEMO approaches.

1.6 Scope of the Study

Although our framework shows its scalability through a simulation study, we are unable to confirm this through experimentations due to the size limitation of our testbed. The network and information security issues are not within the scope of this study.

1.7 Organization of this Dissertation

The chapters of the dissertation are organized as follows.

Chapter 2 reviews the underlying concepts related to the study. It provides an overview of MANET, well-known routing protocols, the P2P overlay network concept, and P2P applications on MANET in particular P2P resource discovery mechanisms. We then present a survey of these discovery mechanisms, some of which are also used to provide discovery techniques in SIP on MANET. We explain mobility and SIP, provide a literature review of P2P SIP over MANET, and concentrate on related research.

Chapter 3 presents SMON, the overlay network framework for P2P applications on MANET. SMON is an improvement of CrossROAD (Delmastro, 2005), which provides resource discovery for P2P application. We propose a new algorithm for overlay creation and maintenance which significantly reduces control overhead, while giving the same lookup success

ratio and delay as CrossROAD.

In chapter 4, we propose the deployment of SIP on top of SMON or SIPMON in order to create a SIP overlay network, which we refer to as P2P SIP, an access point where an existing SIP based application can seamlessly operate with SMON. We show how SIPMON can handle SIP registration and call invitation. The chapter also shows simulation and testbed results.

Chapter 5 explains how extended SMON (SMON+) is built to cover both MANET and the Internet. We show that SIPMON+ provides SIP terminal mobility support for seamless interoperability for MANET and Internet SIP users without implementing any kind of home or foreign agent that exists in mobile IP and IPv6 mobility. We then present comparative results between our scheme and the NEMO oriented MANEMO (Wakikawa et al., 2007). We show how SIPMON+ can scale in scalability analysis subsection at the end of this chapter.

Chapter 6 presents a multimedia communication prototype, called Easy Disaster Communication (EasyDC) based on SIPMON and SIPMON+. We also discuss issues concerning the deployment of an emergency network based on different network protocols for a post-disaster scenario including a comparison of their performance. Moreover, we propose SIPMON+ as a two-tier network in order to provide better management of group mobility.

Chapter 7 summarizes the main highlights of the dissertation and provides recommendations for future work.

Chapter 2

Literature review

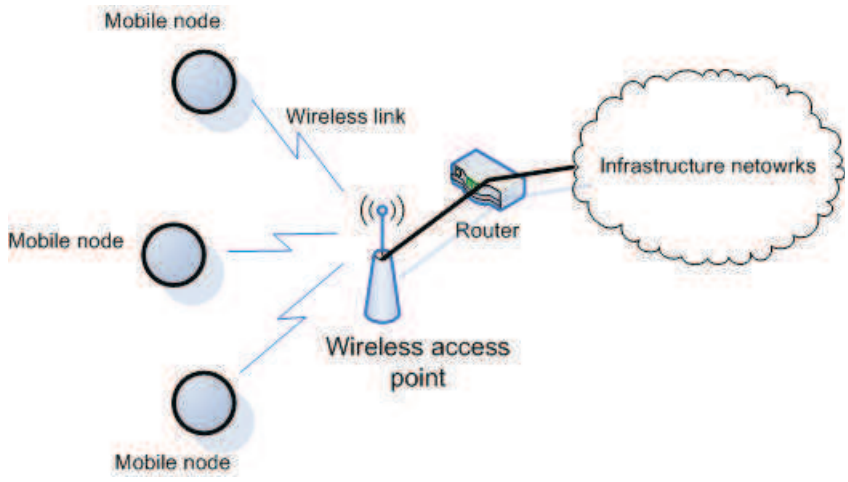
This chapter includes an overview of concepts and technologies essential for this dissertation as well as an extensive survey of related work. Section 2.1 introduces a class of networks called Mobile Ad Hoc Network (MANET) together with a review of some of its well-known routing protocols. We then provide a survey of Peer-to-Peer (P2P) overlay networks and review P2P applications on both the Internet and MANET in section 2.2. In section 2.3, the Session Initiation Protocol (SIP) together with how it can be used to handle mobility is discussed. Finally, we explore the concept of P2P SIP over MANET including more details of related work.

2.1 Mobile Ad Hoc Network (MANET)

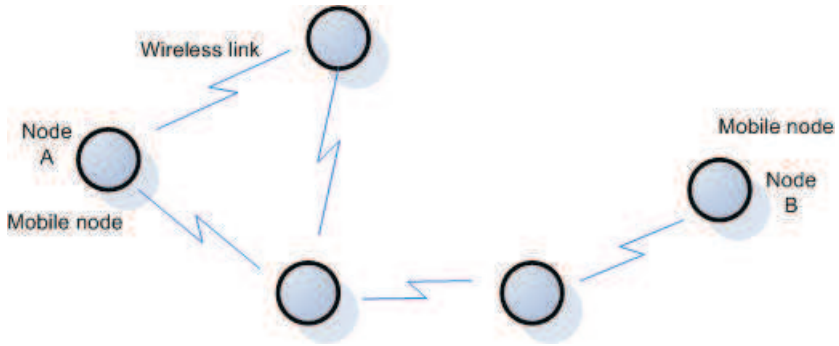
In wireless networks, there are two types of wireless network connections. The first one is an infrastructured network where mobile nodes communicate via a wireless access point which resides on an infrastructure. The access point allows the wireless nodes to exchange data within its radio transmission range, leading to limited distances between nodes and the access point. Another type of connection is through the use of an ad hoc mode with no access points, which means this type of network functions without an infrastructure. It can provide network access in cases when the network infrastructure is not present, such as at disaster sites where the entire network infrastructure has been completely destroyed. By using the ad hoc network, a disaster recovery network can be setup within one or two hours instead of days.

MANET is an infrastructureless multi-hop and self-organizing wireless network made up of mobile hosts as shown in Fig. 2.1. Node A can send a packet to node B by using intermediate nodes to relay the packet without going through any wireless access points. The path, used to forward the packet to the destination, is provided by MANET routing protocols. Every host or node functions as a router where mobile hosts in an ad hoc network are free to move; in other words, the network topology is likely to change frequently. Traditional routing protocols, used in the wired networks, are designed for networks in which most of the mobile nodes do not move frequently. Therefore, a new routing algorithm is needed for a MANET dynamic network environment.

With the increasing availability of wireless devices, MANET has gained more importance with a wide range of interesting applications. In addition, MANET can be easily deployed anywhere with or without a network infrastructure making it attractive for deployment in military battlefields, emergency rescue operations, and for ad hoc vehicle-to-vehicle (V2V) communication. In military battlefields, it is very difficult to create such a fixed network for military communication because military units always move. Moreover, the communication networks should be fast and easily deployable in an ad hoc fashion. Consequently, the MANET is suited for these kinds of operations. MANET allows military units such as soldiers, tanks, vehicles, and headquarters to communicate to one another.



(a) Infrastructured network



(b) MANET

Figure 2.1: Infrastructured Wi-Fi network vs. Mobile Ad Hoc Network

Similarly, emergency rescue operations require rapidly deployable networks for rescue operation communications. In disaster areas, the network infrastructures are likely to be damaged. Instead of setting up a new network infrastructure for rescue operations which could takes weeks or months, MANET can be set up within a few hours or less.

Since an autonomous mobile ad hoc network can be easily and temporarily built by using laptops, PDAs, or wearable computers sharing information between participants in a conference, classroom, or other scenarios, e.g. V2V communication. The example of the MANET application in this category is CarNet proposed by Morris et al. (2000). The main goal of CarNet is to provide a communication system between cars by using wireless networks and a Global Positioning System (GPS) device. The authors introduced a novel scalable routing system, called Gird, using geographic forwarding to route packets from one car to another without flooding the network. A car in the Gird can use other cars as gateways to access the Internet. Moreover, CarNet can be used as an emergency warning system for cars, e.g. intersection collision avoidance or highway-rail intersection warning.

MANET is specified in RFC 2501 (Corson & Macker, 1999). A MANET can operate in isolation or interact with fixed gateways that offer Internet connectivity. A MANET node is generally equipped with wireless capability, which is composed of a transmitter

and a receiver. When the node transmits data, other nodes within its transmission range also receive the data. Each MANET node should perform routing functions in order to forward a packet to a final destination. RFC 2501 also discusses the characteristics of MANET as listed below.

Dynamic topologies Due to mobility of the MANET nodes, the network topology changes randomly, frequently, and rapidly at any time. Moreover, links between nodes may become bidirectional and unidirectional links, which make the network topology unstable.

Limited bandwidth High error bit rate, noise, fading, and other interferences in wireless links lead to limitations in the bandwidth's transmission. A path from a source to a destination in the MANET may consist of multiple physical paths, so it is prone to accumulate more noise. Moreover, many data packets may be lost when the network topology changes.

Energy constrained operation Nodes in the MANET are likely mobile; thus, they must depend on batteries to provide them with power. Moreover, many nodes in the MANET that perform routing functions also require higher power consumption.

Limited physical security Due to the nature of the wireless broadcasting medium. As a result, data can be easily captured by attackers.

2.1.1 Interoperability between MANETs and fixed IP networks

Sometimes, MANET can be attached to the Internet in order to provide a flexible edge network for the Internet. Such interoperability is provided at the network or IP layer. The reason that IP layer should be used in MANET is to provide Internetworking capability over heterogeneous networks (Corson & Macker, 1999). To enable MANET connection to the Internet, all MANET routing protocols adopt IP packets with IP headers for their data packets where their messages get embedded inside IP packets. Fig. 2.2(a) shows how Ad-hoc On-Demand Distance Vector routing (AODV) (Perkins et al., 2003), Topology Dissemination Based on Reverse-Path Forwarding routing (TBRPF) (Ogier et al., 2004), and Optimize Link State Routing (OLSR) (Clausen & Jacquet, 2003) messages are encapsulated in a UDP packet which embedded itself within an IP packet. AODV nodes use UDP protocol to communicate using port number 654. TBRPF and OLSR also use UDP to carry their messages on port number 712 and 698 respectively. On the other hand, Dynamic Source Routing(DSR) (Johnson et al., 2007), does not use UDP as an underlying transport protocol, but DSR defines its new header that must immediately follow the IP header as shown in Fig. 2.2(b). Thus, the mentioned routing protocols are applications that use the TCP/IP stack to exchange its routing messages among MANET nodes.

MANET may have a gateway with two interfaces: an interface to interact with nodes in MANET and the other to connect with an infrastructured network. The interface

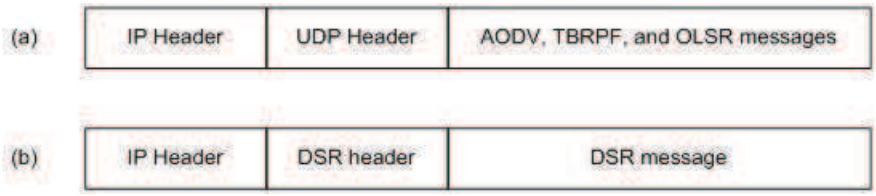


Figure 2.2: Examples of routing message formats

connected to the MANET side operates and interprets MANET routing packets while the other with the infrastructured network does not need to implement this MANET routing protocol. The MANET routing protocol does not perform packet forwarding operations, as it only maintains the routing table. The IP layer uses the routing table to forward an IP packet as described in RFC 1812. For example, upon receiving an IP packet from the MANET side, the gateway uses a destination address indicated in the packet to find the next hop in the routing table in order to forward the packet. If the next hop address belongs to the fixed IP interface or a node in the infrastructured network, the packet is then forwarded to the IP interface. Similarly, the same procedure occurs when an IP packet from the infrastructured network is forwarded to the MANET interface.

2.1.2 MANET routing protocols

Even though this dissertation concentrates on the OLSR used as an underlying routing protocol for our proposed scheme, we will provide an overview of the well-known MANET routing protocols: ABR, FSR, AODV, DSR, ZRP, and TBRPF. Nevertheless, OLSR will be covered in more details.

Originally, MANET routing protocols can be classified into two groups: reactive and proactive routing protocols. Later, hybrid routing protocols combining both proactive and reactive basic properties have been proposed.

Reactive routing protocols This class of protocols maintain the partial routing information upon demand. When a source node wants to send a message to a particular destination, the source node initiates a route discovery process to find the best path to that destination. This approach is based on an on-demand path discovery that floods the query messages as needed. The main benefit of these protocols is the low routing overhead because they do not periodically exchange routing information. However, the main drawback of the protocols is the long delay it takes to establish a connection because reactive protocols need to find and calculate the route before transmission.

1) Associativity-Based Routing (ABR) (Toh, 1997)

A source discovers a destination by flooding a route request message to the entire network. The destination sends a route reply message to the source via a selected path that is the most stable. To select a stable route, every node maintains an associativity table

containing associativity ticks with respect to its neighbor over time. If two nodes are neighbors and they stay connected for a long time, their associativity ticks are high. The associativity ticks can be obtained from periodic beacons or hello messages between a pair of neighbors. At the destination, if there are many paths from the source, a reverse path is chosen based on the highest associativity ticks. The strength of this routing protocol is that a selected path tend to last longer. However, this path is not always the shortest path.

2) Ad-hoc On-Demand Distance Vector routing (AODV) *(Perkins et al., 2003)*

Nodes in the AODV do not need to maintain routes to destinations that are not part of active communications. The two main operations of AODV are path discovery and path maintenance.

Path discovery starts whenever a source node begins to communicate with a destination, and it has no path to that destination in its routing table. The source node will then initiate a broadcast by sending a route request (RREQ) to all neighbor nodes which are within the source node's transmission. The range RREQ is propagated from neighbors to other neighbors, like rumors, until it reaches the destination. While the RREQ travels from the source node to the destination, all intermediate nodes set up reverse paths from themselves back to the source node by recording addresses of the neighbors from which they receive the RREQ.

To prevent routing loop and duplicated update, each node maintains the pairs of destination sequence number and IP address. When a node receives a RREQ packet, it processes the packet if the sequence number the received packet is larger than its own sequence number in the cache.

The other operation is path maintenance. If a route is broken due to the movement of the source node, the source node needs to reactivate path recovery in order to find a new route to the destination. If an intermediate node along the route moves, the upstream node of the broken node needs to transmit a route error packet (RERR) to the source node. When the source node receives the RERR packet, it can initiate a new path discovery process to the same destination.

3) Dynamic Source Routing(DSR) *(Johnson et al., 2007)*

DSR is another on-demand routing protocol based on the concept of source routing. In this concept, every packet carries the complete list of nodes, which the packet should traverse in a packet header. The key ideas of the DSR are to reduce overhead by avoiding unnecessary updates, and to allow a packet to be sent over unidirectional links. For example, a forward path goes between nodes A and F in the following manner: A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ F, while the reverse goes from nodes F to A in the following manner: A $\leftarrow$ B $\leftarrow$ D $\leftarrow$ F. In this example, node C and D have the unidirectional links with node F. The forward path and the reverse path can be different. However, keeping full routing information in the packet header causes extra overhead.

Like AODV, DSR consists of two mechanisms: 1) route discovery where route discovery is based on flooding the network with RREQ packet and 2) route maintenance. The

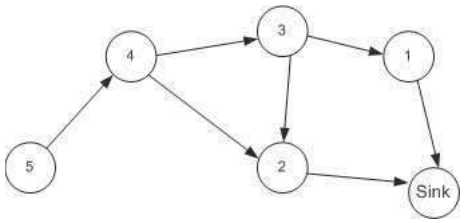


Figure 2.3: An example of a DAG

difference is that the discovered path is stored in the packet header instead of every node along the path. Nodes in the DSR are arranged in a promiscuous mode operation, causing adjacent nodes to overhear the RREQ packet. Therefore, the nodes may cache this overheard route information for possible future use. Moreover, the DSR allows nodes to keep multiple routes to a destination in their caches. When the link to the destination is broken, they can use other routes in the caches without creating a new route discovery. However, a stale cache because of host mobility has a severe impact on the performance of the DSR. The sender may try several stale routes in its cache before finding a good one. Route maintenance is as follows. When an intermediate node detects a broken link to its next hop node toward the destination, it marks this route in its cache as invalid and returns a route error packet (RERR) back to the source node. As the RERR packet arrives, the source node initiates a new route discovery process to the destination.

Apart from the mentioned reactive routing protocols, we briefly give a list of other reactive routing protocols to complete the literature. There are reactive routing protocols based on link reversal routing (LRR) (Gafni et al., 1981). The concept of LRR is that all nodes are represented as a graph, which is converted into a directed acyclic graph (DAG). This DAG contains only one destination or a sink that does not have any outgoing links to its neighbors as shown in Fig. 2.3. When the link from node 2 to the sink fails, node 2 must perform a link reversal algorithm because it becomes the other sink (no outgoing links). Then, node 2 reverses its links. The link reversal process is repeated until there is only one sink left as shown in Fig. 2.4. Reactive routing protocols that use the concept of LRR are Lightweight Mobile Routing (LMR) (Corson & Ephremides, 1995) and Temporally Ordered Routing Algorithm (TORA) (Park & Corson, 1997). Single Stability Adaptive (SSA) (Dube et al., 1997) is similar to ABR, but it uses link stability as a metric for path selection instead. Cluster Based Routing Protocol (CBRP) (Jiang et al., 1999) is a hierarchical routing protocol that divides a network into many clusters. Each cluster contains a cluster head. In CBRP, routing information is exchanged only among clusterheads, leading to the introduction of a lower routing overhead compared with other reactive routing protocols. However, the clusterhead selection is performed every time when the clusterheads become inactive due to mobility. This frequent selection may result in high overhead and route discovery delay.

Proactive routing protocols These protocols are known as table-driven routing protocols; they use concepts of distance vector and link state routing protocols (Sesay et al., 2004). Every node maintains its tables to keep routes to all reachable destinations. In order to keep the routing table up-to-date, routing information is constantly exchanged among nodes. A low delay in setting up a connection between the source and destination

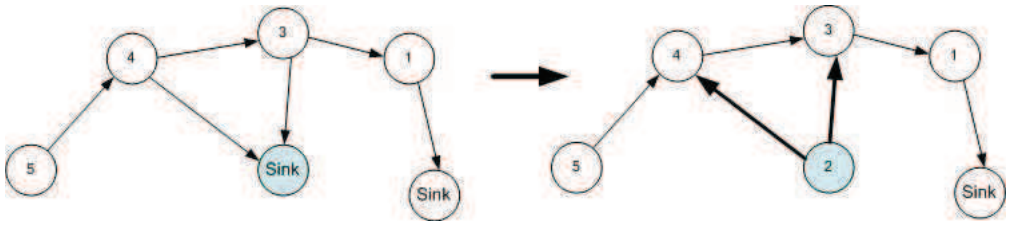


Figure 2.4: Node 2 reverses its links

is the main advantage of these protocols because the routes are computed proactively. On the other hand, the shortcoming is that there is a high overhead of routing information exchanged since the nodes usually broadcast routing information periodically

1) Fisheye State Routing (FSR) *(Pei et al., 2000)*

FSR is based on a link state routing protocol that maintains a full network topology database at every node. FSR uses different routing update frequencies for different types of nodes depending on a fisheye scope distance away from itself. For example, a FSR node uses f_1 update interval for one-hop neighbors and f_2 update interval for two-hop neighbors and so on, where $f_1 < f_2 < \cdots < f_n$. Using different intervals for the routing updates reduces the number of routing overhead compared to the original link state routing protocol, but a route to a far remote destination becomes less accurate.

2) Topology Dissemination Based on Reverse-Path Forwarding routing (TBRPF) *(Ogier et al., 2004)*

TBRPF is a proactive link state routing protocol. Each node computes a source tree to all reachable destinations based on partial topology information. The source tree is calculated by using a modification of Dijkstra's algorithm. In order to minimize the broadcast overhead, each node sends its own partial source tree to neighbors and uses differential HELLO messages to report only changes in the network topology. The TBRPF is composed of two modules: neighbor discovery and routing module. The TBRPF neighbor discovery module is responsible for detecting new neighbors and loss of neighbors. The main feature of this module is to use differential HELLO messages, which report only changes in the status of neighbors, causing smaller sizes of HELLO messages when compared with an old link state routing protocol such as OSPF. Consequently, in order to provide a fast detection of topology changes, the nodes can send HELLO messages more often. The other module is routing. This module allows a node to build a source tree and to report only part of its source tree, called reported subtree, to neighbors. Each node periodically sends reported subtree every 5 s and every 1 s for differential updates. The reported nodes are analogous to multipoint relay selectors in the OLSR, which will be explained next.

3) Optimize Link State Routing (OLSR) *(Clausen & Jacquet, 2003)*

OLSR is a proactive link state routing protocol. OLSR is well suited to large and dense

mobile networks. The purpose of OLSR is to use a selected number of nodes called MultiPoint Relays (MPRs) to relay the broadcast messages in the network. This technique restricts the number of re-transmitters as much as possible by efficiently selecting a small subset of neighbors as MPRs which covers (in terms of one-hop radio range) the same network region which the complete set of neighbors does (Laouiti, Qayyum, et al., 2002). When the node wants to broadcast the packet, it sends this packet to its MPR nodes. Other nodes, which are not MPRs, will never forward the packet. Hence, the number of broadcast packets is reduced significantly. Fig. 2.5 depicts an example of a broadcast packet with and without MPRs. For the network without MPRs, the number of retransmission is nine, whereas three is the number of retransmission with MPRs.

The core functions of OLSR are neighbor detection, topology discovery, and routing table computation. Each node uses HELLO messages to discover its one-hop neighbors. HELLO messages are periodically broadcast by specifying TTL to be 1 so that the HELLO messages will never go beyond one hop. The HELLO messages contain a list of neighbors and their link status. Only bi-directional links are considered as valid links. Moreover, each node can construct a table of two-hop neighbors by using HELLO messages exchanged. Next, each node will perform MPRs selection.

Laouiti, Qayyum, et al. (2002) propose an algorithm to select multipoint relays as follows. Let a set of one-hop neighbors of x be represented by $N(x)$, and a set of two-hop neighbors be represented as $N^2(x)$ and $MPR(x)$ represents a set of multipoint relay nodes of x. Multipoint relays selection algorithm by Laouiti, Qayyum, et al. (2002) is given below:

1. Start with an empty multipoint relay set $MPR(x)$.
2. First select those one-hop neighbor nodes in $N(x)$ as the multipoint relays which are the only neighbor of some node in $N^2(x)$, and add these one-hop neighbor nodes to the multipoint relay set $MPR(x)$.
3. While there are still some nodes in $N^2(x)$ which are not covered by the multipoint relay set $MPR(x)$:
    - For each node in $N(x)$ which is not in $MPR(x)$, compute the number of nodes that is covered among the uncovered nodes in the set $N^2(x)$.
    - Add that node of $N(x)$ to $MPR(x)$ for which this number is maximized.

After MPRs selection is made, each node broadcasts Topology Control (TC) messages in order to create a complete topology table used for routing calculations. This method is similar to the classical link state routing protocol, but it improves the broadcasting technique as previously mentioned.

Finally, each node uses received TC messages that it received to compute its routing table. The routing table is made up of very important information because it helps locate the next hop to a destination. Moreover, the routing table is recalculated when the network topology is changed. Even though the protocol has many functions, only neighbor detection, topology discovery, and routing table computation are the main functions.

The OLSR packet is defined by using unified packet format for all data so that it provides extensibility of the protocol to support backward compatibility. One packet may contain

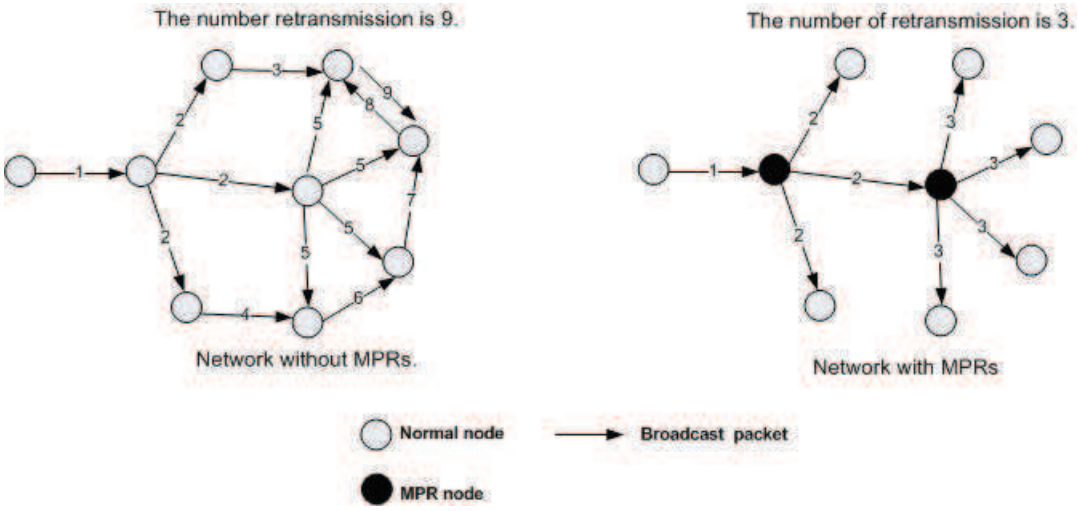


Figure 2.5: The number of retransmissions in OLSR

several messages. The OLSR protocol uses UDP port 698 for communication. The OLSR packet format is depicted in Fig. 2.6. The packet header contains two fields: packet length and packet sequence number. The packet length is the length of the packet. The packet sequence number is used to check duplicate packets. The packet sequence number must be increased by one each time a new OLSR packet is transmitted. The message header is composed of eight fields: message type, *vtime*, message size, originator address, time to live, hop count, message sequence number, and message. The message type indicates the type of message such as HELLO or TC message. The value of the message type ranges from 0 to 127. Next, *vtime* identifies how long a node accepts that this received message is valid. The concept of *vtime* is similar to hold time in routing algorithm. Message size gives the size of the message. The message header, originator address contains the IP address of the original sender, and not the intermediate node. Time to live contains the maximum number of hops that allow the message to be forwarded. Time to live is decreased by one when an intermediate node retransmits it. When it reaches zero, this packet will be dropped. Hop count indicates how many hops the message traverses. Next, the message sequence number is used to make sure that this message is not retransmitted more than once by any node. The last message header field is a message that contains real data. Messages used for constructing a routing table and already defined in OLSR are HELLO and TC messages. The following sections explain these messages and their operations in detail.

Each OLSR node exchanges HELLO message periodically in order to detect neighbor nodes. The format of the HELLO message is shown in Fig. 2.7. The node broadcasts the HELLO message which contains a list of its neighbors and a link status that enables the discovery of one-hop neighbors. The HELLO messages are never forwarded more than two hops, so TTL in HELLO messages should always be zero. Link status can be either an asymmetric or a symmetric link. The asymmetric link is referred as unidirectional link. For example, nodes A and B are physical neighbors of each other. However, node A can send a HELLO message to node B, but it cannot hear the HELLO message from node B as shown in Fig. 2.8.

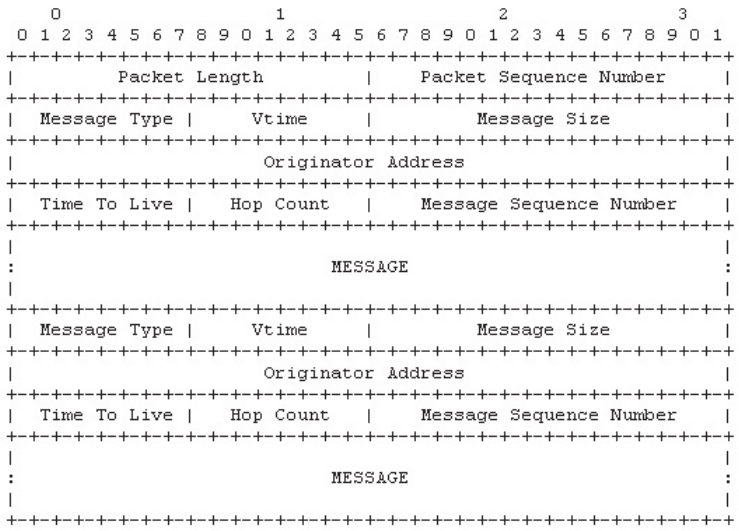


Figure 2.6: The OLSR packet format (source: Clausen & Jacquet (2003))

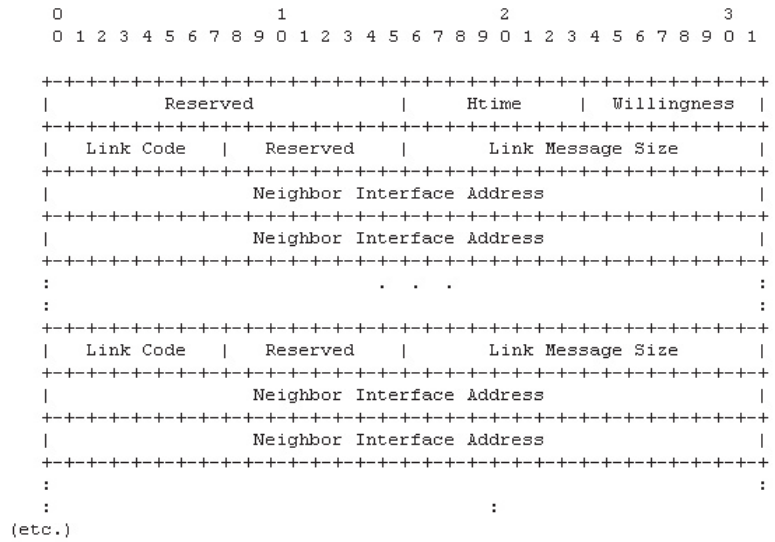


Figure 2.7: The HELLO message format (source: Clausen & Jacquet (2003))

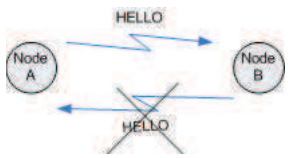


Figure 2.8: An asymmetric link from node A to B in OLSR

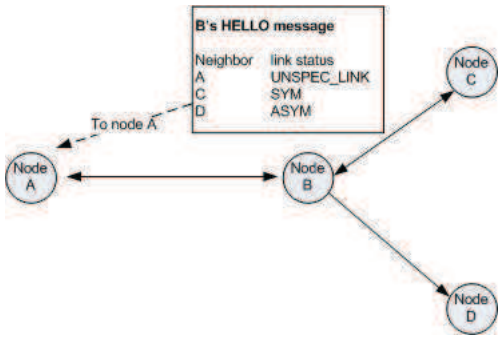


Figure 2.9: Finding two-hop neighbor at node A in OLSR

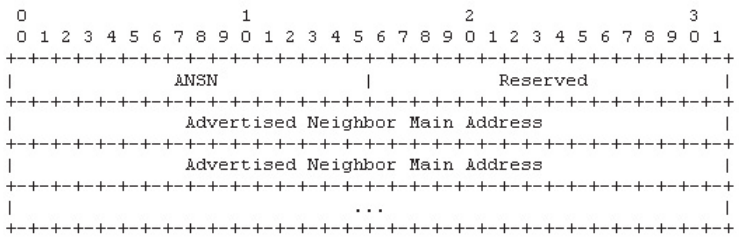


Figure 2.10: TC message format(source: Clausen & Jacquet (2003))

On the other hand, node A can receive the HELLO message from node B, and node B can receive the HELLO message from node A. This link is called a symmetric or bidirectional link. Two nodes in OLSR claim to be neighbors if there exists a symmetric link between them. Moreover, by exchanging HELLO messages, an OLSR node can further know two-hop neighbors based on information in the HELLO messages. Fig. 2.9 shows how node A processes a received HELLO message from node B. After receiving the message, node A knows that node B is a neighbor because it sees itself contained in node B's HELLO message. Therefore, node A changes this link status to an asymmetric link status. Next, node A finds out that node C is a neighbor of node B with symmetric link status. Node A adds node C to its two-hop neighbor sets, while node D is not a neighbor of node B because the link between nodes B and D is asymmetric; as a result, node A does not add node D as its two-hop neighbor. After exchanging HELLO messages, each node uses the MPR selection algorithm as described previously to find its MPR. Only MPR nodes are allowed to forward TC and other OLSR messages.

An OLSR node uses a TC or topology control message for topology discovery purposes. According to one-hop and two-hop neighbor information tables, each multipoint relay node creates and broadcasts a TC message, containing a list of its one-hop neighbors that select this node as their MPR. The TC message floods to the entire network with the help of other MPR nodes. Finally, each node completely knows the network topology. The TC message format is shown in Fig. 2.10.

ANSN is an advertised neighbor sequence number that indicates the freshness of TC message. Every time a node detects any changes in its neighbor set, it increases this sequence number by one. Each node maintains TC's previous information, sequence numbers and senders. Whenever a node receives a new TC message, it checks whether

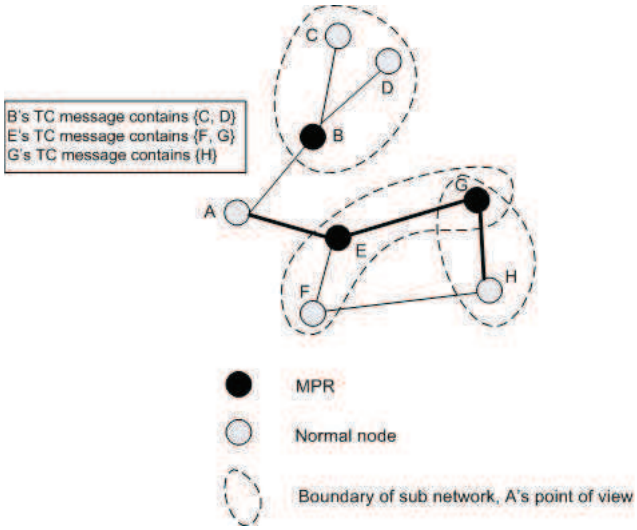


Figure 2.11: An example of OLSR route calculation

the sequence number received and the TC message is higher than the sequence number found in the database. If it is higher, each node will process the TC message; otherwise, it will drop the message. Advertised Neighbor Main Address is the field that contains the address of a neighbor node which selects the sender as its MPR node. After receiving TC messages, each node begins calculating routes to all possible destinations.

The routing table is important information for any kind of routing protocols. Each node maintains a routing table, which is calculated based on received TC messages. The node creates connected pairs of TC senders and advertised neighbor listed in the TC message. For example, node R wants to find possible paths to node Z. Node R extracts pairs of connected nodes in the accumulated TC messages in order to reach node Z. Next, node R selects the path that contains minimal hops toward node Z. The routing table is based on the information in the neighbor and topology table. Hence, the routing table is calculated every time when these tables change.

In Fig. 2.11, after node A receives TC messages from all MPRs, it begins computing route to all destinations. The example shows how node A finds a route to node H based on received TC messages. First, node A looks for node H in TC messages, which is found in G's TC message. Therefore, node A constructs a graph of node G and H as shown in Fig. 2.12 step 1. Next, node A finds G's MPR, which is node E, so it now has the route from node E to H shown in step 2. In the last step, node A knows that node E is its neighbor so that it finally has a complete route from itself to node H, as depicted in step 3. Node A will repeat the same procedure in order to find routes to all nodes.

Besides, the mentioned proactive routing protocols, there are several approaches that use the proactive routing concept. Destination Sequenced Distance Vector (DSDV) routing (Perkins & Bhagwat, 1994) is a distance vector algorithm based on the Bellman Ford algorithm. Each entry in a routing table is labeled with a sequence number specified by a destination. A route with the newest sequence number is used in order to prevent a staled route. DSDV nodes periodically advertise routing updates, resulting in the introduction of high routing overhead. Similar to DSDV, Wireless Routing Protocol (WRP) (Murthy

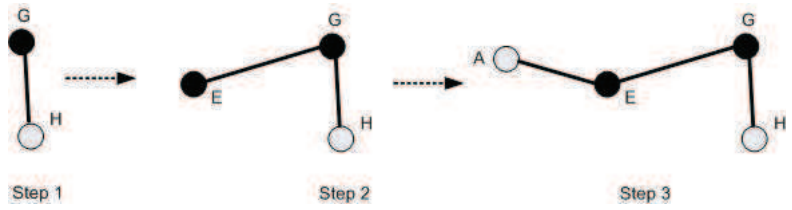


Figure 2.12: An example of OLSR route calculation of Fig. 2.11

& Garcia-Luna-Aceves, 1995) is a proactive routing protocol that solve the count to infinity problem of a distance vector algorithm. Global State Routing (GSR) (T.-W. Chen & Gerla, 1998) is based on a link state routing. In the traditional link state routing protocol, routing messages are flooded to the entire network. However, GSR routing messages are exchanged by only hop-by-hop neighbors, resulting in a reduction in the routing overhead. The size of routing messages is relatively large as the number of nodes grows.

Hybrid routing protocols Hybrid routing protocols are proposed to combine the merits of both reactive and proactive routing protocols. This group of protocols is designed to support a large size of the network by creating some sort of a zone in which a proactive routing is used. Each zone can communicate to one another via a proactive routing. The strength of hybrid routing protocols is a limitation of the proactive overhead because it is confined within the zone. However, determining the proper size of the zone remains an important issue. An example of this class of routing protocol is Zone Routing Protocol(ZRP) (Haas et al., 2002). ZRP organizes its network into zones. Each node defines its zone covering *n-hop* neighbors, where n is a radius of the zone. Within the zone, a proactive routing protocol is deployed to maintain routing information, thus routing broadcast message limited and routes immediately became available within the zone. When it is needed for a route to outsize the zone, a source uses a reactive routing protocol to discover a destination by forwarding a RREQ packet towards all its border neighbors. Upon receiving the RREQ packet, the border neighbors use a reactive routing protocol to flood the network in order to discover the destination. The advantage of ZRP is the limitation of control overhead to the size of the zone. However, zones can be overlapped to each other, if the radius of the zone is not carefully configured and can result in inefficient route discovery process.

In conclusion, for reactive routing protocols, a node is unlikely to maintain correct routing information on all nodes at all times. Instead, the node finds a path, if it does not already exist in its routing table, to a destination where it wishes to communicate with the destination. Finding paths to the destination triggers route discovery is a process that floods a query message in the network. For example, a node in AODV broadcasts RREQ message to all neighbors. The neighbors again forward the RREQ message to their neighbors. Eventually, the RREQ message will reach the destination. The advantage of reactive routing protocols is a lower routing overhead compared with proactive protocols. One shortcoming of reactive routing protocols is the high delay of route-setup. On the other hand, proactive routing protocols try to maintain correct routing information on all nodes at any time. This can be achieved by using regular routing advertisement. The advantage of proactive routing protocols is that paths to all destinations are always available, so the delay of route-setup is typically low. The drawback of the protocols is

a higher routing overhead, compared with on-demand routing protocols, due to periodic routing updates.

We propose the use of OLSR as an underlying routing protocol for our SIP on the MANET framework. By doing so, we expect to minimize the delay that occurs when nodes join an overlay network. When the node makes a request to join the overlay network, it can use the OLSR routing table to maintain associations with other nodes on the overlay network without having to performing the route discovery process to all destinations, something which a reactive routing protocol has to do. The OLSR uses Multi Point Relays (MPRs) to relay a message to all nodes. The usage of the MPR forwarding functionality significantly reduces duplicated retransmissions during the flooding procedure when compared with the reactive routing protocol.

2.2 P2P overlay network

In this section, the concept of the general overlay network is explained. We first consider overlay networks on the Internet or infrastructured networks.

An overlay network is a virtual network of nodes and logical links that is built on top of an existing network with the purpose of implementing a network service that is not available in the existing network (Stoica et al., 2001). The goal of creating the overlay network is to add new services without having to make modifications to existing infrastructure as this would be impractical. Examples of overlay network applications include P2P file sharing, content delivery, routing, security, massive computing, email system, and multicast network. The first benefit of using an overlay network is the ability to support particular services without having to change existing software and hardware. New applications can be deployed on the application layer on top of the network infrastructure. Sometimes overlay networks are used to provide incremental system deployment, for example the 6bone, and legacy IPv6 testbed.

2.2.1 P2P overlay on infrastructured networks

A P2P overlay network is a distributed system, without a centralized administration point, with peers connected together via logical links which are usually TCP connections. Peers on the overlay are nodes that have equal capabilities of being a client as well as a server. The major function of the overlay network is to provide application routing, which is a process that routes messages among peers in the overlay network, providing a substrate for large-scale data sharing, content distribution, and resource searching, etc. In contrast to the IP network, the proximity metric is not taken into account when the application routing is performed on an overlay network; this means that one overlay hop may span across the entire world. The P2P overlay network also supports any applications that require robustness, self-organization, fault-tolerance, and scalability. The first usage of the P2P overlay network was intended to serve as a P2P file sharing application on the Internet, where users downloaded MP3 songs from other users in a P2P fashion. In other

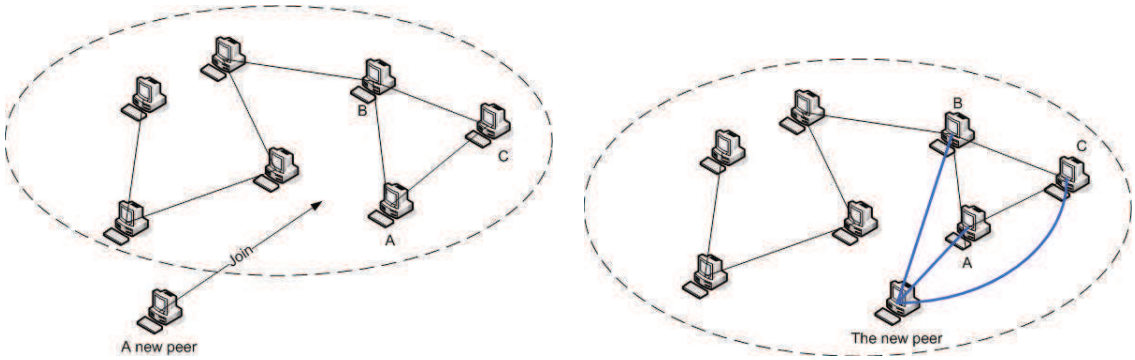
words, P2P file sharing is an application that allows a user to download directly a file from other users' personal computers. For example, the user types a keyword of the file, such as the filename, so that a list of peers who have that file is displayed on the screen. The user selects one of the peers in the list to download the file. The file is copied from the selected peer to the user's machine. While this file is being downloaded, other users can immediately download the file from the user. In order to increase the downloading speed, the user can simultaneously download the file from many peers as well.

P2P overlay networks can be classified into two categories: unstructured and structured overlay networks. In unstructured overlay networks, peers usually use a broadcast mechanism to locate data items. Nevertheless, by using the broadcast method, problems with the bandwidth and scalability emerge. On the other hand, peers in the structured overlay network deploy the same hash functions to find given data items without invoking any broadcast queries. Consequently, the structured overlay network is suitable for applications that require distributed architecture.

1) Unstructured P2P overlay network

An unstructured P2P overlay network is a random graph of nodes with no pre-defined topology or structure. The assumption is that the unstructured P2P overlay network is already created as shown in Fig. 2.13(a). When a new peer joins the overlay, first, it must find a peer that has already joined the overlay. Second, it simply copies their peer's existing link information and uses it to connect to more peers. The joining peer can freely make connections with any other peers. In Fig. 2.13(b), the joining peer detects that peer A is logically the nearest neighbor, so it connects with peer A. The bootstrap procedure for finding nodes that are in the overlay can be achieved in the following ways. The joining peer uses the previous list of overlay peers from the last seen. The joining peer retrieves overlay information about the overlay from a centralized place, such as DNS servers or stable web sites. The next bootstrap strategy is the use of limited broadcast query. The joining peer may broadcast a query message to find existing peers in the overlay network by specifying "time-to-live" in order to limit the distance it takes to forward the query message. After the joining peer has joined the overlay, it copies the existing information about the link overlay from node A so that it knows more nodes in the overlay, namely, nodes B and C. The new node, moreover, tries to connect to as many overlay nodes as possible in order to prevent the problem involving a single point of failure.

Searching for a desired piece of data in the overlay, a peer has to flood a query through the overlay network in order to find potential peers who share the same data as shown in Fig. 2.14. If the number of quires increases, the entire overlay network is eventually overwhelmed by these flooded messages, resulting in high bandwidth consumption. Moreover, an increase in the number of participants in the network also leads to scalability problems. Examples of P2P file sharing applications, Napster (drscholl, 2000), Gnutella (Klingberg & Manfredi, 2002), and KaZaA (Liang et al., 2004) are based on the unstructured overlay. Napster uses a centralized server to maintain a file directory; hence, it is subject to a single point of failure. On the other hand, Gnutella and KaZaA are true distributed P2P systems, where centralized servers are removed, resulting in an increase in the system's robustness. The difference between Gnutella and KaZaA is that Gnutella is a flat network, while KaZaA is a two-tier hierarchical network. In conclusion, scalability is the main problem for unstructured P2P overlay networks.



(a) Joining the existing overlay network (b) Establishing links with peers

Figure 2.13: Operations of joining the unstructured P2P overlay network

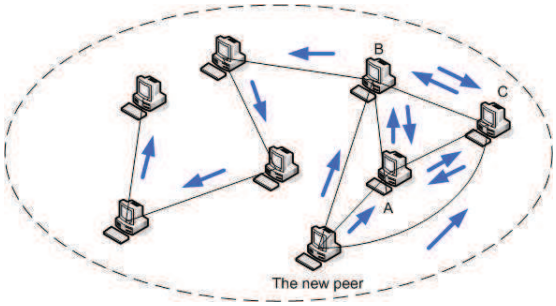
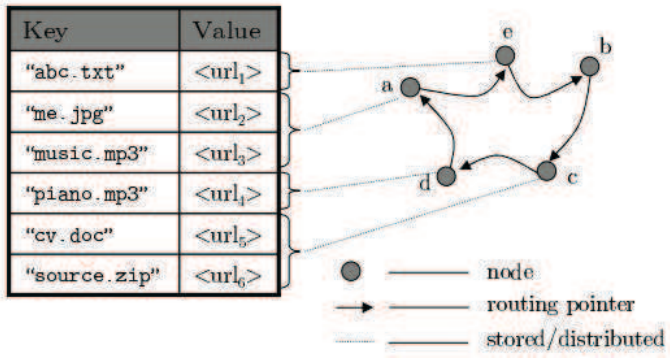


Figure 2.14: Flooding in the unstructured P2P overlay network

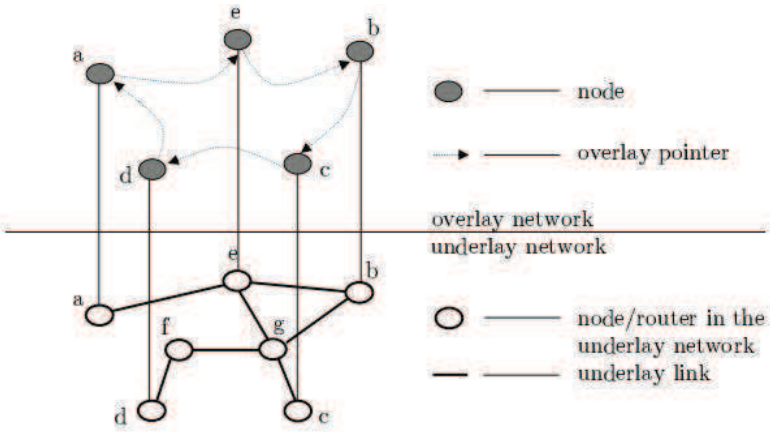
2) Structured P2P overlay network

P2P overlay networks commonly use ***Distributed Hash Tables*** (DHT) to provide a structure for the overlay nodes' network. As the name implies, a DHT is based on the hash table concept, which is distributed among a set of computers, referred to as *nodes* (Ghodsi, 2006; Balakrishnan et al., 2003). Similar to the hash table, each node stores key and value pairs, which is referred as *objects*.

The main operation of DHT is to provide a lookup service, which finds nodes that are responsible for storing objects with given keys. An important property of DHTs is that they can efficiently handle large numbers of data items. Furthermore, the number of cooperating nodes might be very large, ranging from a few nodes to many thousands or millions in theory. Because of the limited storage/memory capacity and the cost of inserting and updating items, it is infeasible for each node to locally store every item. Therefore, each node is responsible for part of the items, which it stores locally. The term structured overlay network is therefore used to distinguish an overlay network created by DHTs from other overlay networks (Ghodsi, 2006). Fig. 2.15(a) shows an example of a DHT mapping filenames to the URLs, which represent the files' current location. The DHT items are distributed to nodes a, b, c, d, and e, and the nodes keep routing pointers to each other. If an application makes a lookup request to node d to find out the current location of the file abc.txt, node d will route the request to node a, which will route the



(a) Example of a DHT mapping filenames to the URLs representing the current location of the files



(b) An overlay network and the underlay network on top of which the overlay network is built

Figure 2.15: DHT and overlay network (source: Ghodsi (2006))

request to node e, which can answer the request since it knows the URL associated with key abc.txt. Note that not every node needs to store items, e.g. node b. Fig. 2.15(b) displays an overlay network and its corresponding underlay network.

We describe examples of well-known structured DHT overlay networks, Chord (Stoica et al., 2001), Pastry (Rowstron & Druschel, 2001), Tapestry (Zhao et al., 2001), and CAN (Ratnasamy et al., 2001) in the following paragraphs.

Chord (Stoica et al., 2001) assigns each node and a key with m-bit identifier based on hash function SHA-1. Each node has an ID by hashing its IP address, and the key is produced by hashing data information, such as file name. The overlay network is arranged in a circle modulo 2^m. Key k is assigned to a node that has its ID equal or greater than k. The node identifiers are represented as a circle of numbers from 0 to $2^m - 1$ clockwise ascending order. Chord routing is based on the concept of DHT as previously discussed. Each node maintains a routing table, called a finger table, with m or $\log_2 N$ entries, where N is the number of nodes. Fig. 2.16 illustrates Chord routing when m is 6. It shows the finger table of node N1, composed of six rows of successors. In Fig. 2.17, N1 wants to

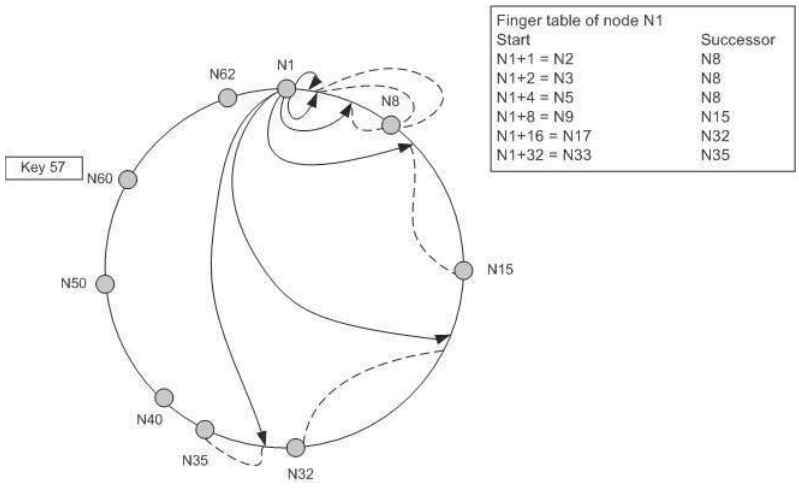


Figure 2.16: Finger tables of N1

find key–57, now stored at node N60. N1 finds out that key–57 belongs to its successor N35. N1, then, sends a query, containing the requested key, to N35. After receiving the request from N1, N35 opens its finger table, and finds out that N60 is responsible for key–57. Thus, node N35 forwards the query on behalf of N1 to N60. The authors of Chord mention that the number of nodes that must be contacted in order to find a desired successor in an N–node network is $O(\log N)$ and $\frac{1}{2}\log N$ on average. To prove that, finding a successor is required at most m steps because $2m$ is equal to N. Therefore, the time to find the successor is $\log N$. When a node joins the Chord overlay network, the successor pointers of some peers must be updated so that the correctness of lookups is guaranteed. Moreover, Chord uses a stabilization protocol running in the background in order to update successful pointers in the finger tables. To improve its robustness, instead of maintaining only the first successor, each peer maintains a list of peers' first r successors. When the first successor peer fails to respond, the peer simply selects the next peer on the list.

Pastry (Rowstron & Druschel, 2001) is a self-organizing decentralized overlay network, where proximities of peers are taken into account. Each peer in Pastry is assigned a 128–bit identifier or node ID, uniformly distributed hash value of its IP address. Pastry uses a ring to arrange nodes in a circular 128–bit identifier space as shown Fig. 2.18. In routing time complexity, Pastry can route any message to a destination in less than $\log_{2^b} \times N$ steps, where b is a configuration parameter, which is normally equal to 4. For example, when $b = 4$ and $N = 106$ peers, the number of routing hops is 5.

Each Pastry node must maintain three tables: a routing table, a neighborhood set, and a leaf set. The routing table is composed of $\log_{2^b} \times N$ rows with $2^b - 1$ entries for each row. Every entry in row n^{th} shares the same prefix of $(n-1)^{th}$. Fig. 2.19 shows an example of the routing table of node 3DA40 (128 bits), when $b = 4$ and $N = 106$. The total entries are 75, $(\log_{2^b} \times N) \times (2^b - 1)$. Each entry contains the IP address of the potential node whose node ID shares the same prefix with proximity consideration. Each peer maintains a neighborhood set, containing the nodes' IDs and IP addresses that are closest to its ID. Usually, the neighborhood set is not used in routing, but it is useful in maintaining

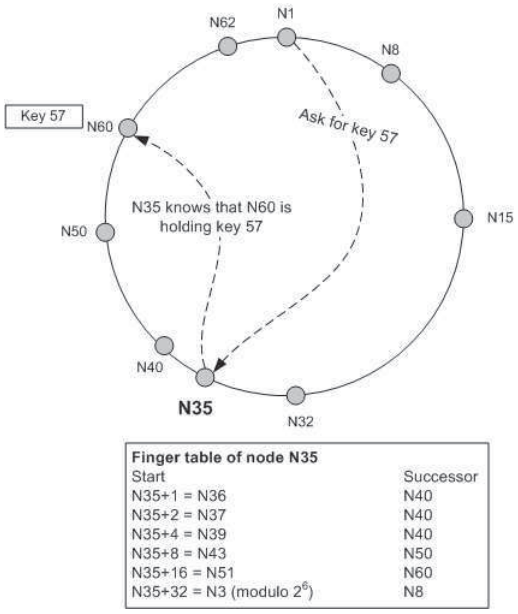


Figure 2.17: Finding key-57 in Chord DHT

locality properties. Next, each peer maintains a leaf set table, consisting of $L/2$ nodes with numerically closest larger node IDs, and $L/2$ nodes with numerically closest smaller node IDs, relative to its ID. L is a configuration parameter with a typical value of 16. The Fig. 2.20 shows an example of a route message in Pastry. Node 2F0DE wants to find key–5F04D, so it looks up at the first row of its routing table to find the node that has the first digit prefix of 5. Therefore, the message is sent to node 5AFE7. After receiving the message, node 5AFE7 again looks for the node that has a prefix beginning with 5F, which is node 5FD89, and sends the message to node 5FD89. This process is repeated at node 5F02F. Finally, the message arrives at the correct node.

Tapestry (Zhao et al., 2001) shares similar properties with Pastry such as proximity

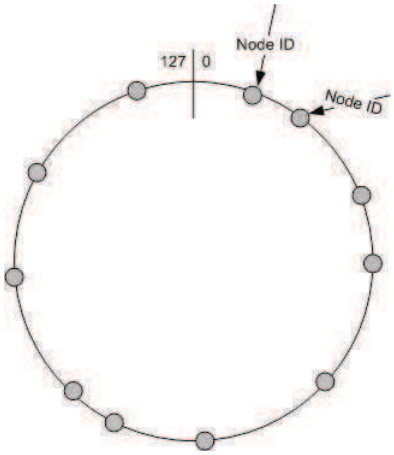


Figure 2.18: A 128-bit circular ID space

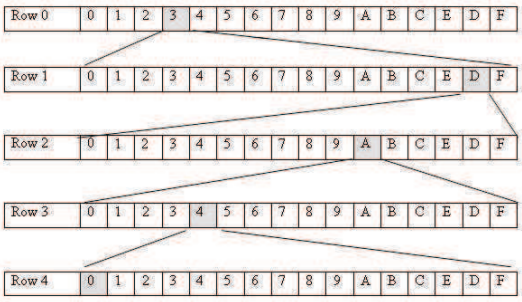


Figure 2.19: Routing table, common prefix of node 3DA40...

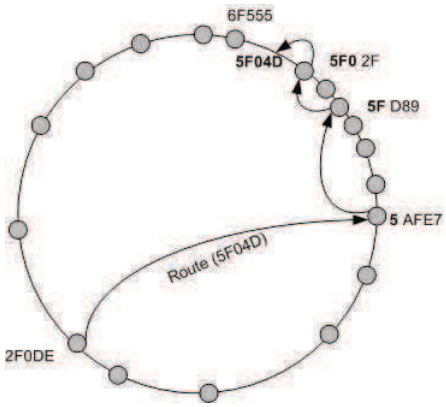


Figure 2.20: An example of route message in Pastry

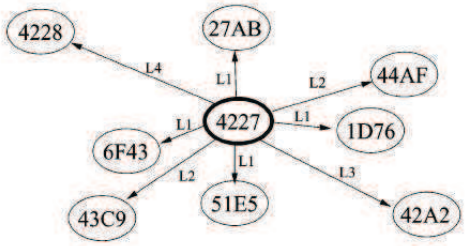


Figure 2.21: Tapestry routing mesh (source: Tapestry (Zhao et al., 2001))

consideration and prefix routing. A Tapestry node is uniformly and randomly assigned a node ID, denoted N_{id}, based on hashing algorithm like SHA-1. Furthermore, the data object has O_G. Every Tapestry message contains application-specific identifier A_{id}, which is used to select a process, or application for message delivered at a destination. In other words, A_{id} is similar to a port number in TCP/IP. The authors define four APIs as follows. PublishObject (O_G, A_{id}) is to publish the object on the local node. In contrast, UnpublishObject (O_G, A_{id}) is to remove the published object. Next, RouteToObject (O_G, A_{id}) is for routing a message to an object's location. The last API is RouteToNode (N, A_{id}, Exact), which routes a message to application A_{id} on node N. "Exact" identifies, such as the destination ID, needs to be exactly matched to deliver the payload. In Tapestry routing, each node maintains neighbor maps, containing links to neighbor nodes that have a prefix that matches with its ID. Tapestry uses neighbor maps to route messages to the destination based on digit matches. For example, to route a message to node 52AF, the node needs to find the longest prefix match and forwards a message to the node that has the closet digit matches. This process is just like the general DHT routing. The message is routed by digit, for example 5*** $\longrightarrow$ 52** $\longrightarrow$ 52A* $\longrightarrow$ 52AF, where * presents wildcards. This method is similar to the longest prefix routing in CIDR IP address allocation. Fig. 2.21 shows the outgoing links of a node 4227. L1 is a link to a node that does not share any digit matches with node 4227, whereas L2 is a link to a node that has a one digit prefix match and so on. Fig. 2.22 shows a path that a message traverses in the example from the Tapestry infrastructure. When a Tapestry node sends a message to a destination, it opens neighbor maps, looks for the longest digit match node, and forwards the message to that node. The process guarantees that any existing nodes in the system will be reached at most $\log_\beta N$ logical hops, where β is the base number, and N is the number of nodes in Tapestry. If the node cannot find any prefix matches, it looks for a node that is numerically closest to the message ID. In Fig. 2.22, node 5230 wants to send a message to node 42AD. It forwards the message to node 400F because it has only has one neighbor node. Node 400F finds out that node 4227 has the longest digit match, so it forwards the message to node 4227. This process is repeated at node 42A2. Finally, the message arrives at the current node 42AD.

CAN (Ratnasamy et al., 2001) is a distributed decentralized P2P infrastructure, designed to be scalable, fault tolerant, and self-organizing. CAN uses a d-dimensional Cartesian coordinate space on a multi-torus (warp around every corner). This coordinate space is a logical space with no relation to any real physical coordinate system. Therefore, CAN does not consider the proximity of the node in the operations. This coordinate space is partitioned into many zones so that each CAN node owns an individual zone. Fig. 2.23

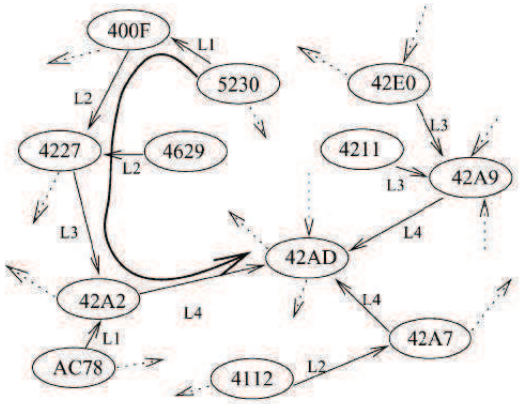


Figure 2.22: Routing path from node 5230 to 42AD (source: Tapestry (Zhao et al., 2001))

shows an example of the two-dimensional $[0, 1] \times [0, 1]$ coordinate space with six CAN nodes. Each node is randomly assigned a unique ID, which is called a key. It gets its key from the uniform distributed hash function. Each node locates itself to the virtual coordinate space by using its key to map the coordinate point in the space. It has its zone which does not overlap with others. Each node must be responsible for storing the object ID that falls in its zone. For example, node A covers the object ID that has a value of $x = 0 - 0.5$ and $y = 0 - 0.5$. The virtual coordinate space is used to keep $(key, value)$ pairs. For example, in order to retrieve the value of key, $K1$, any node applies the same hash function to map $K1$ onto point P in the space, and gets the value from the node that is responsible for that covered point P. A CAN node maintains a coordinate routing table, containing an IP address with virtual coordinate zone for its neighbors that have overlapping coordinates along the x or y axes in case of a two-dimension coordinate space. For example, in Fig. 2.24, nodes B and F are immediate neighbors of node A because node B overlaps on the Y-axis value with node A, and node F overlaps on the X-axis value with node A. On the other hand, node D is not a neighbor of node A since node A does not share an overlapping axis with node D. For a d-dimensional space partitioned into n equal zones or n nodes, the average routing path length is $\frac{d}{4} \times n^{\frac{1}{d}}$ hops, resulting in $O(n^{\frac{1}{d}})$. Each node also maintains 2 x d neighbors. When node A wants to locate the key $(0.8, 0.9)$, it tries to match the key to its ID, which does not match. Then, node A uses the CAN routing table to find a neighbor that has the closest virtual coordinate to the key. Node A finds that node F $(0.5 - 1, 0 - 0.5)$ has an ID that is the closest to the key. Therefore, node A forwards the query for (key, value) to node F. Again, node F knows that node E $(0.75 - 1, 0.5 - 1)$ is the one who is responsible for holding that key $(0.8, 0.9)$, so the query is forwarded to node E directly.

In summary, the P2P overlay networks can be classified into two types: the unstructured and the structured overlay networks. Lookup mechanisms in the unstructured overlay networks are based on the broadcast mechanism, in which a query message is flooded. On the other hand, lookup mechanisms in the structured overlay networks are based on DHT. The unstructured overlay networks are not suited for large-scale networks because of the bandwidth consumed by the broadcasting traffic. Moreover, unstructured overlay networks generally have very poor search efficiency when the number of participants increases. In contrast, the structured overlay networks overcome the limitations of unstructured overlay network by allowing each peer to query for resources based on DHT,

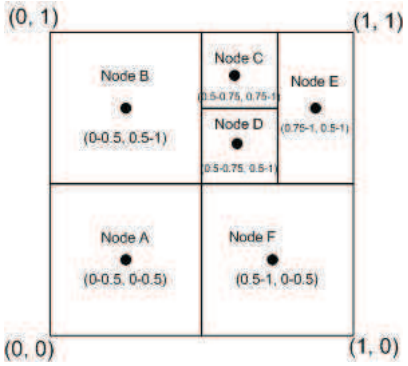


Figure 2.23: A two-dimensional CAN with six nodes (adapted by author from Ratnasamy et al. (2001))

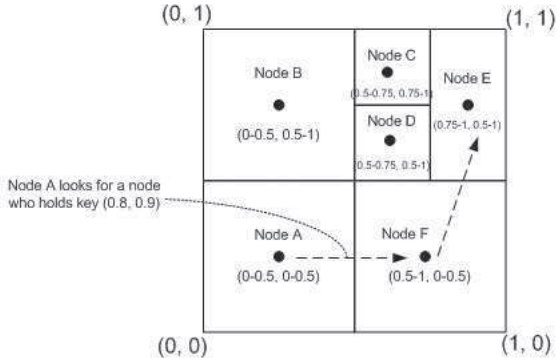


Figure 2.24: An example of CAN routing (adapted by author from Ratnasamy et al. (2001))

which is designed for a large-scale network such as the Internet.

The overlay networks discussed so far have been proposed for wired networks where nodes are not mobile. To deploy overlay networks on MANET, node mobility and a dynamic topology should be included as one of the design requirements. As a result, several overlay networks use a cross-layer approach, which allows them to interact with network layers to improve their overlay network efficiency in such a dynamic topology environment.

2.2.2 P2P overlay over MANETs

The P2P overlay network enables resource discovery on MANET thus MANET nodes can locate desired services and publish their available services on a large scale network. The challenges in providing resource discovery within the MANET are listed below (Sailhan & Issarny, 2005).

- To enable MANET nodes to discovery resources dynamically, while both minimizing discovery traffic overhead and tolerating the discontinuous connectivity of wireless devices.
- To enable service or resource discovery by a general range of devices regardless of their hardware and software platforms.
- To enable resource discovery in a large MANET.
- To enable the bridging of the MANET with infrastructure-based networks such as the Internet.

There are resource discovery mechanisms without the overlay support on MANET where each node knows the addresses of its physical neighbors. Without the overlay support, a source finds resources by broadcasting a query to all other nodes in the network. Only nodes that have these resources reply to the source. These mechanisms do not scale because of the broadcast techniques. Resource discovery mechanisms without overlay support are DEAPspace (Nidd, 2001), L. Li & Lamont (2005) approach, GSD (Chakraborty et al., 2006), Jodra et al. (2006) approach, Konark (Helal et al., 2003), and Splendor (Zhu et al., 2003).

We focus on resource discovery mechanisms on MANET with overlay support which can be divided into two categories. The first category concerns those with overlay support based on layered approaches, for example, Service Rings (Klein et al., 2003b), Lanes (Klein et al., 2003a), and Resource-Aware Overlay Network (RAON) (Lau et al., 2005). The second category are those deploying a cross-layer design. Examples of these include Ekta-RD (Pucha et al., 2004), MAPNaS (Zahn & Schiller, 2005b), Peer Computing based Dynamic Source Routing (PDSR) (Z. Li et al., 2006), and CrossROAD (Delmastro, 2005). Ekta-RD, MAPNaS, and PDSR are overlay networks integrated with reactive routing protocols while CrossROAD is an overlay network organized as a mesh topology and as a cross-layer on OLSR. We provide a brief review of these structured overlay networks in the following paragraphs.

Ekta-RD (Pucha et al., 2004) is a resource discovery application created on top of Dynamic P2P Source Routing (DPSR), an integration of Pastry and DSR. Ekta-RD provides three DHT APIs, *route (message, key)*, *route (message, IP address)*, and *broadcast (message, broadcast address*). These APIs quickly aid a user in the deployment of resource discovery in the MANET. The routing structure of Pastry and DSR are combined into one routing structure, which leads to an optimal system performance. In DPSR, the structures of the routing table and the leaf set table in each node are similar to those in Pastry. Nodes in Pastry maintain pairs of node ID and IP address in the routing table whereas nodes in DPSR keep pairs of node ID and DSR source route. In the original DSR, when a node receives a packet, it finds the next hop in its routing table in order to forward the packet. If there is a path to the destination, it will forward the packet to the next hop. If there is no route to the destination, the node will start the route discovery process in order to find a source route to the destination. The route discovery is based on the sending of a broadcast to a route request packet. DPSR tries to minimize the number of performing route discovery process by replacing it with the unicast prefix routing based on the Pastry concept. Instead of doing the route discovery process, the node performs the prefix routing, which reduces the number of broadcast traffic. However, this prefix routing never gives the optimal shortest path as DSR does. The strength of DPSR is the resource discovery support for P2P applications running on top of MANET. However, the drawback of DPSR is a lookup delay because it is considered as an indirect routing.

MAPNaS (Zahn & Schiller, 2005b) is another P2P based resource discovery approach built on top of MADPastry (Zahn & Schiller, 2005a), which is an integration of Pastry and AODV. The goal of MADPastry is to provide a DHT substrate for practicably sized MANET. Physical locality is one of the main feature in MADPastry. The overlay networks are formed in regards to the physical hop of peers. In MADPastry, random landmarking is used to create multiple Pastry overlay networks as shown in Fig. 2.25. Each node in MADPastry is assigned a node ID by landmark nodes, which temporarily and currently are responsible for landmark keys. The landmark keys are selected in a way that they can separate multiple overlay networks with equal sizes. For example, for networks with an ID base of 16, the landmark keys would be 00 . . . 00, 01 . . . 00, 20 . . . 00, 30 . . . 00 ,. . . , F0 . . . 00, which give 16 overlay networks. Nodes that are responsible for broadcasting landmark keys are chosen from the closet hashing value of their IP addresses. For example, node ID with 0055 will yield to node 0050 because node 0050 is the closet node to the fixed landmark key 00..00. After landmark nodes are defined, these nodes periodically advertised their existences. A new node joins the overlay by measuring the distances to those temporary landmarks, and assigns its overlay prefix the same value as the landmarks' that the node is closest to. After assigning an overlay prefix, the rest of the ID is randomly assigned by using the hash function. Exploit physical locality is the main advantage of MADPastry. Next, the overlay broadcast traffic is limited within a single overlay, similar to the concept of clustering. However, the disadvantage of MADPastry is a delay due to indirect routing and address resolution scheme.

Peer Computing based Dynamic Source Routing (PDSR) (Z. Li et al., 2006) is a P2P cross-layer design that gives an infrastructure for P2P application running over MANET. Each node in PDSR has a unique node ID, which is the hash value of its IP address. PDSR's routing table is similar to that of the routing table in DSR. The dif-

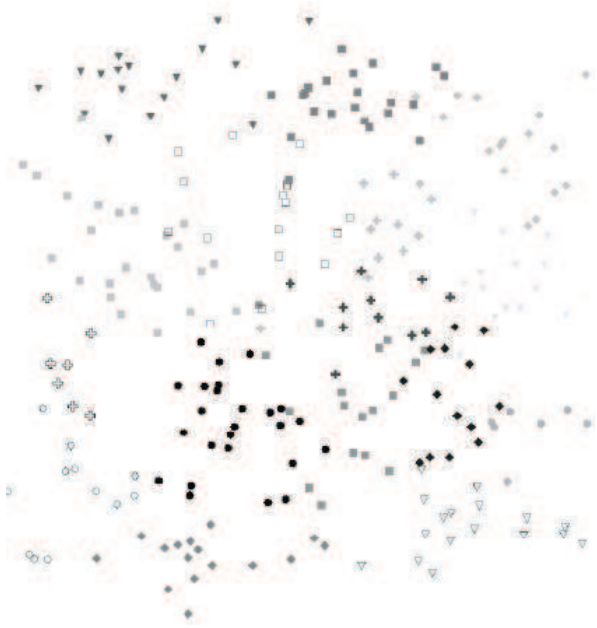


Figure 2.25: Spatial distribution of id prefixes (source: MADPastry (Zahn & Schiller, 2005a))

ference is that the destination and the next hop IP address in DSR routing table are replaced with node IDs. Each node maintains the same parameters similar to Chord. The route discovery algorithm is similar to that in DSR route discovery algorithm. However, a P2P search uses Chord-like algorithms. The authors compare the performance of systems running DSR and PDSR in terms of packet deliver ratio, average end-to-end delay, and routing overhead. PDSR has a higher packet deliver ratio than that of the normal DSR. On average end-to-end delays, PDSR results in a higher delay than DSR because of the indirect routing. One overlay hop may cover many physical hops. The last performance result concerns the routing overhead. DSR generates a higher routing overhead than PDSR. This is because PDSR uses a structured overlay network which minimizes the amount of broadcast traffic when it performs the overlay routing.

CrossROAD (Delmastro, 2005) is a structured overlay network on MANET, which provides service lookups based on DHT without relying on a centralized index server. It is a cross-layer design that can directly collaborate with OLSR. Knowing who is currently taking part in the network, a CrossROAD node can use this information to maintain its overlay network locally and efficiently. In addition, the mesh topology allows a CrossROAD node to communicate with another node via the shortest path given by OLSR routing information. However, every node periodically broadcasts CrossROAD messages to discover other peers on the mesh network, resulting in a high control overhead, which is CrossROAD's main drawback.

In Delmastro (2005), the authors set up experiments on a structured overlay Pastry (Rowstron & Druschel, 2001) on OLSR and AODV networks with TCP connections between overlay nodes. They demonstrated that Pastry on OLSR outperformed AODV due to the required route discovery done in AODV. CrossROAD was subsequently introduced to improve the performance by adopting a cross-layer architecture which allows information

to spread throughout the protocol stack in order for each node to adapt their behavior according to the rapidly changing topology.

CrossROAD, by design, depends on OLSR routing protocol to provide necessary information to maintain its overlay network. Their performance depends on the performance of OLSR's routing layer. OLSR is designed for a dense network but relatively small in size analogous to OSPF. CrossROAD is chosen as our underlying resource discovery for SIP because of the following reasons:

(1) Cross-layer design. Because OLSR provides a complete knowledge of the network topology, CrossROAD can use the routing information to maintain and update its overlay network topology. Therefore, CrossROAD can effectively maintain its overlay network under node mobility.

(2) Fast lookup time. CrossROAD provides constant lookup time $O(1)$ because it uses the mesh overlay network topology, while other approaches are of $O(N \log N)$ because they use the ring overlay network topology based on Chord or Pastry.

2.3 Session Initiation Protocol (SIP)

SIP (Rosenberg et al., 2002) allows the users to locate other users and exchange multimedia session parameters. The session parameters are described in a Session Description Protocol (SDP) (Handley & Jacobson, 1998). SIP supports five operations' establishment and termination of multimedia communications.

First, user location provides user discovery ability for end users. Second, user availability reflects the users' determination or willingness to join the communication. Third, SIP supports user capabilities to identify media and media parameters that are going to be used in the communications. Fourth, session setup provides the establishment of called and calling users. The last SIP support is the session management that is responsible for transferring and terminating existing sessions, modifying session parameters, and bringing up other services. SIP does not provide any real-time data transferring method or any data flow control. Hence, in order to offer an absolute multimedia solution, SIP should be used with other multimedia protocols such as Real-time Transport Protocol (RTP) to transfer real-time multimedia data and control the flow of data. SDP should be used to describe multimedia sessions. SIP components, as shown in Fig. 2.26, consist of User Agent Client (UAC), User Agent Server (UAS), location service, registrar, proxy server, and redirect server. In actual implementation, all components may be grouped into a single unit, called a SIP server, due to the ease of maintenance and configuration.

Both UAC and UAS play important roles for the end users. UAC is an entity that creates a request such as REGISTER and INVITE messages; however, UAS is a server that waits for requests and then replies. Normally, a user agent contains both UAC and UAS. Location service is a database service that is used by a redirect or proxy server. It contains a list of user binding information, addresses, records, or SIP Uniform Resource Identifier (UR). SIP URI has a similar format to email addresses, for example "sip:user@host." A user can use an existing email address as SIP URI without the need for a new address.

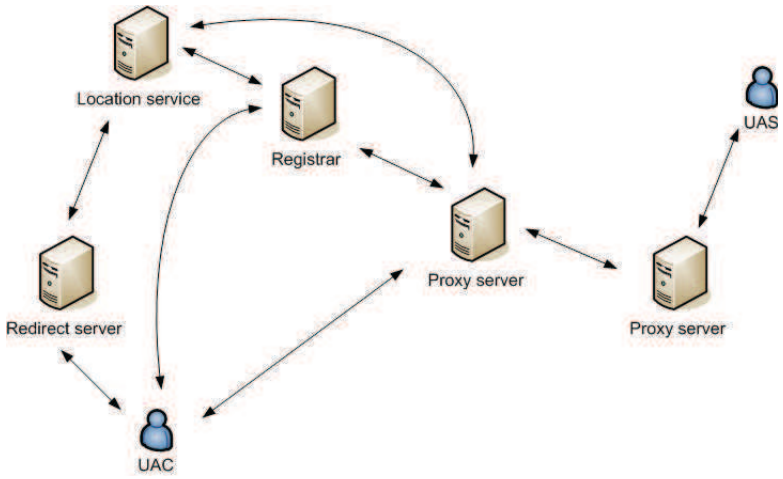


Figure 2.26: SIP components

Layer 4: Transaction User (TU)
Layer 3: Transaction
Layer 2: Transport
Layer 1: Syntax and encoding

Figure 2.27: SIP layer structure

Registrar is a server that accepts registration requests from UAC. Each user must register his/her IP address and SIP URI to the registrar server. This information will be queried from UAC in order to initiate the session. The proxy server is an intermediate server that serves both UAC and UAS and helps make request on others' behalf. Redirect server is a UAS that responses with redirect messages. It is designed to reduce the processing load on the system. In contrast to a proxy server, the redirect server does not route the packet to other servers, but it directly responses to the client with redirect messages only.

SIP structure is composed of four layers, which are syntax and encoding, transport, transaction, and transaction user (TU) layers as shown in Fig. 2.27. These layers are loose couplings from each other. One layer is used to describe a set of functions in that layer, but these layers are not involved in implementation.

In the syntax and encoding layer, all SIP messages are encoded by using an Augmented Backus-Naur Form (ABNF) grammar as defined in RFC 2234, which is also used in other communication protocols such as HTTP.

The transport layer is responsible for the transmission of any requests and responses. In SIP, it is recommended that SIP must be implemented over UDP and TCP, and other transport protocols may be implemented as needed. The transport layer keeps information when connections are opened in records formed by source IP, port, and transport type.

The next layer is the transaction layer that processes requests and responses. A transaction is a request from a client and responses to this request transmitted from a server back to the client. This layer is also responsible for application data retransmissions and for

verifying that there is a match between responses and requests. One client can have many transactions to several servers, where different transactions can be identified by different sequence numbers. There are two modes regarding the transaction layer. A stateful proxy is a logical unit that keeps the client and server transactions; however, transactions are not maintained in a stateless proxy. Therefore, the client that uses stateless does not have this transaction layer.

The last layer is the transaction user or TU, which is only used by a stateful client. When a TU of the client wants to send a request, it creates an object called client transaction instance that will be sent to the transport layer. This instance is used to control the behavior of the created transaction such as transaction cancellation.

A SIP message is either a request or response, and consists of a start-line followed by header fields and the body of the message. The message and header field format is similar to HTTP/1.1 format.

```
generic-message =   start-line
                    *message-header
                    CRLF
                    [ message-body]
start-line      =   Request-Line / Status-Line
```

The start-line, message-header, and CRLF must be followed by a carriage-return line-feed. The start-line can be either a Request-Line that represents a request or Status-Line which corresponds to a response.

A SIP request has a Request-Line for as a start-line. The Request-Line consists of method name, Request-URI, and the protocol version. They are separated by a space (SP) character. The Request-Line as shown below must end with CRLF.

```
Request-Line        =   Method SP Request-URI SP SIP-Version CRLF
```

There are six request methods defined in SIP.

REGISTER This method is used by a client to register its location to the SIP registrar.

INVITE This method indicates that a user or service is being invited to participate in a session. The body of this message also includes the session's SDP message.

ACK This method is an acknowledgement sent by a server to a client to confirm the receipt of an INVITE message.

CANCEL This method is used to cancel a pending request

BYE This method is an terminate an ongoing call.

OPTIONS This method is used to query server capabilities such as encoding, language support, and supported header fields.

Examples of Request-Line are shown below.

```
INVITE sip:thirapon@interlab.ait.ac.th SIP/2.0
```

```
REGISTER sip:thirapon@interlab.ait.ac.th SIP/2.0
```

The SIP response is issued by the server, and it has a start-list with Status-Line instead of a Request-Line. The response contains the status code and the reason string that indicates the condition of the request.

```
Status-Line =   SIP-Version SP Status-Code SP Reason-Phase CRLF
```

An example:

```
SIP/2.0 200 OK
```

The Status-Code is a 3-digit integer that describes the result of the response, along with Reason-Phase that explains the response's in short detail. SIP version 2.0, Status-Code is divided into six types.

Provisional(1xx) This type is to inform a correspondent that the request has been received, and the server is processing the request.

Success(2xx) It indicates that the request is accepted.

Redirection(3xx) It is used to inform a client about another server that should be contacted in order to complete the transaction.

Client Error(4xx) It indicates syntactical errors in the request.

Server Error(5xx) It indicates an error since the server fails to operate the request.

Global Failure(6xx) It indicates errors such as 604 Request-URI does not exist.

Fig. 2.28 illustrates register operation. Alice sends REGISTER message, containing its SIP-URI and IP address, to proxy server abc.com. The proxy replies back to Alice with a 401 unauthorized message, containing an authentication challenge.

```
WWW-Authenticate: Digest
        realm="abc.com",
        qop="auth,auth-int",
        nonce="fa0adf45938cdcde3940498a93fe9b9c9e9f2343",
        opaque="5ccc0983fac049deabdfe39340fffacdde393d"
```

After Alice receives the 401 message, he sends REGISTER message containing his authentication credential to the abc.com again. If Alice's credential is correct, the server accepts her registration by issuing SIP 200 OK to Alice.

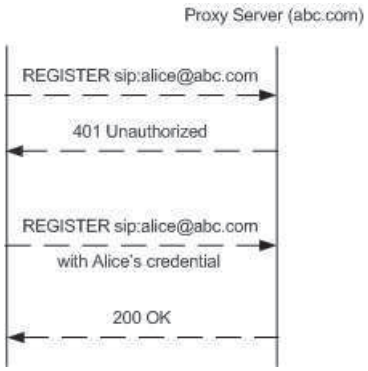


Figure 2.28: SIP Registration

2.3.1 SIP user registration and discovery

SIP user location is one of the main support which allows a user to discover another user's location in order to initiate a session with that user. The user's location includes SIP user registration and discovery. The user creates its binding and sends it to a registrar that performs a front end location service. The user must know the registrar's address in advance, e.g. using pre-configured address or DNS service. The registration process is based on client/server architecture. The user sends a REGISTER request containing SIP URI and IP address to the registrar. The registrar confirms the successful registration by responding with a 200 OK message to the user. The user's binding is kept to the location service. In general, the registrar/proxy is responsible for a particular domain, which is matched with the domain part indicated in a user's SIP URI. For example, "sip:bob@abc.com" implies that Bob is the SIP user on domain abc.com. Bob must send his binding to abc.com's registrar. In a call invitation, the discovery process is performed. Alice wishes to give Bob a call without knowing Bob's location. Alice has to send an INVITE request to Bob's proxy at abc.com. The abc.com proxy retrieves Bob's location from its location service in order to forward this INVITE request to where Bob is.

Fig. 2.29 shows a simplified example of a SIP message flow between Alice and Bob. First, Alice issues an INVITE request containing Bob's SIP URI, "sip:bob@abc.com". After the proxy server receives Alice's INVITE request, it consults with location service to obtain Bob's IP address. Then, the proxy forwards the INVITE on behalf of Alice to Bob; at the same time, it responses with SIP 100 trying to inform Alice that it is trying to reach Bob. When Bob gets the INVITE, Bob fires SIP 180 Ringing and SIP 200 OK responses in sequence to the proxy. After Alice accepts SIP 200 OK from the proxy, Alice knows Bob's IP address. Now, the proxy is no longer in use. Alice can directly rely ACK to Bob. After this point, Alice can establish a session with Bob. At the end, SIP BYE message is used to terminate the existing session.

2.3.2 Mobility and SIP

SIP supports four types of mobility: terminal, session, personal, and service. In this thesis, we focus on providing only terminal mobility supported by our P2P SIP overlay

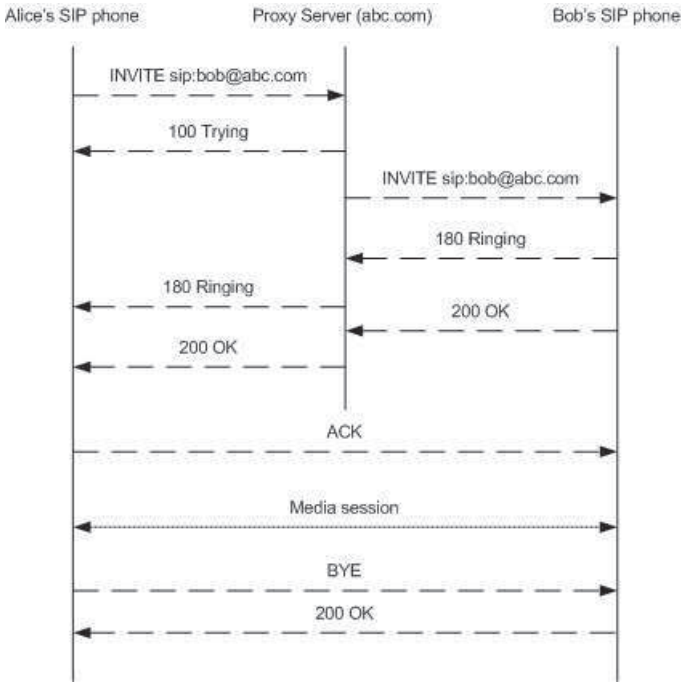


Figure 2.29: A simplified SIP INVITE example (adapted by author from Rosenberg et al. (2002))

network.

a) Terminal mobility

SIP allows a terminal to change its IP address while maintained an ongoing session (Schulzrinne & Wedlund, 2000). Terminal mobility causes SIP pre-call and mid-call mobility. The pre-call mobility is the binding update process, which ensures that Correspondent Node (CN) can reach Mobile Node (MN) after MN moves to a new subnet. The mid-call mobility handles handoff between CN and MN that allows them to resume an ongoing session.

Fig. 2.30 shows SIP binding update when MN acquires a new IP address at a foreign network before making or receiving a call. After binding update is completed, CN can normally give a call to MN at its home network. SIP server, which keeps MP's new binding updates, replies to CN via SIP 302 which moved temporarily and contains MN's new address. CN obtains this new address and sends a SIP INVITE to MN at foreign network as shown in Fig. 2.31.

Once two users start a session, they use a SIP dialog to maintain this ongoing session's parameters. The parameters of the session can be modified within the existing dialog. For example, a caller wishes to change audio encoding or add a new media stream, it can send re-INVITE, containing the new session parameters, to a callee without tearing down the ongoing session. In addition, when the caller changes his/her IP address, re-INVITE can be used to inform the callee about its new IP address as well. Either the caller or callee can modify the ongoing session.

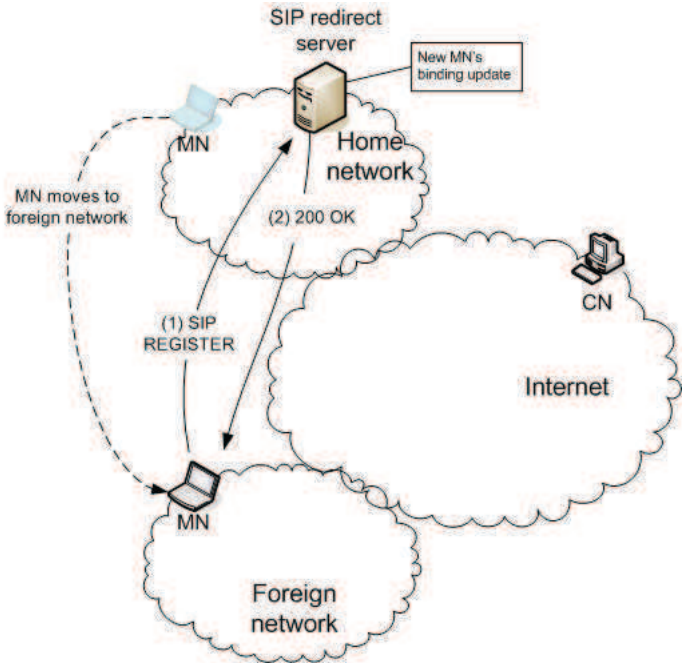


Figure 2.30: SIP binding update when MH moves to the foreign network

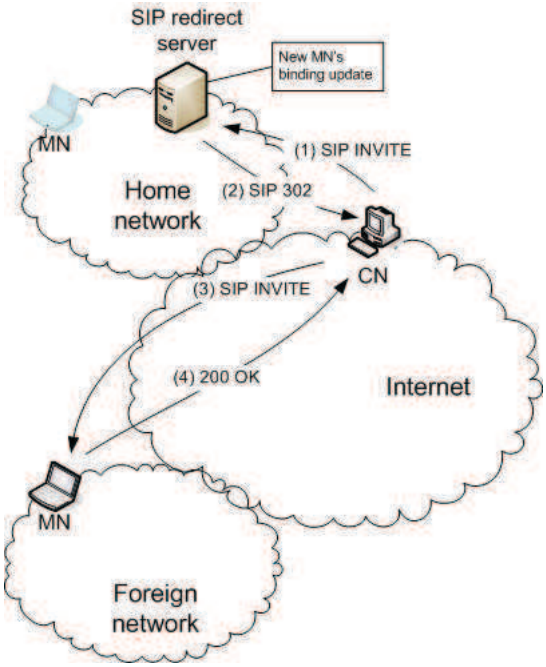


Figure 2.31: SIP INVITE to MN at foreign network after the pre-call mobility is completed

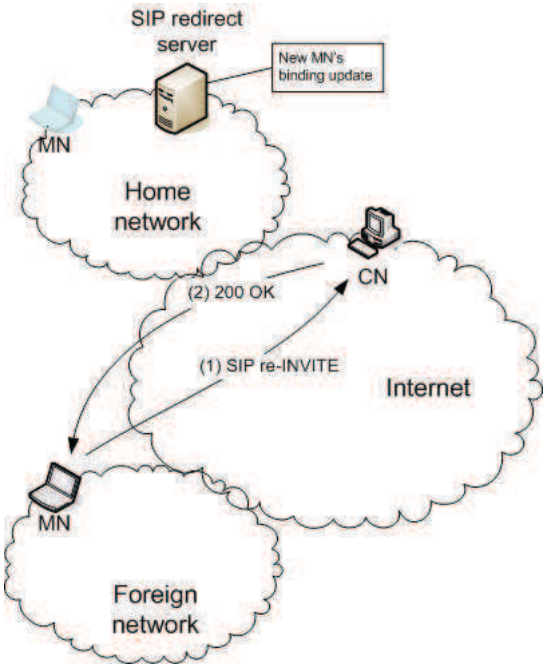


Figure 2.32: SIP re-INVITE for continuing ongoing session in SIP mid-call mobility

The re-INVITE can be used to resume the ongoing session when terminal mobility occurs. The mid-call mobility allows a moving MN to send a re-INVITE request to CN directly to continue the ongoing session without going through SIP servers. The re-INVITE request to CN has the same call ID as previous call sessions setup with a new IP address contained in Contact header of the request. After CN receives the request, it responses with SIP 200 OK to resume the ongoing session with MN as shown in Fig. 2.32.

SIP-based mobility is not suitable for TCP-based applications e.g. FTP that uses IP addresses to maintain a connection between two end points. Changing IP address causes a connection to be cut. However, SIP supports applications based on Real-time Transport Protocol (RTP) that does not use IP addresses to maintain relationship between end points. It uses a random value of 32-bit SSRC identifier. For example, a receiving node can redirect an incoming stream to a new host by informing a sender about its new IP address with a known SSRC value. Changing IP address does not terminate the ongoing stream.

b) Session mobility

A user can transfer an ongoing session to another terminal in session mobility. For example, the user wishes to transfer the ongoing session from a computer to a PDA. The session mobility is supported by using the SIP REFER method (Sparks, 2003). The user sends REFER to inform a recipient about the address of the new terminal where the recipient should transfer the ongoing session to.

c) Personal mobility

In personal mobility, a user can have one SIP URI mapped with different addresses or terminals (one SIP URI → n terminals) or many SIP URIs with one address (many SIP URIs → one terminal). For example in the former case, the user has many devices, e.g. a computer, PDA, PSTN phone, and a wireless device and wishes to be reachable via these devices all at the same time. These devices should ring all together when there is an incoming call. The SIP registrar must be able to know different addresses of terminals owned by the same person. In the latter case, the user has several SIP URIs, for example, user@sipphone.com, user@abc.com, or user@pstn.com, each of which is registered to different SIP registrars, but with the same IP address.

d) Service mobility

Service mobility allows a user to keep information regarding services at SIP servers. Examples of services are phone book, buddy list, and call logs. The user can retrieve the service information at SIP servers while changing to a new terminal.

2.3.3 P2P SIP over MANET

The major problem in implementing SIP in MANET is that SIP is based on a centralized architecture, which is not suitable for an ad hoc environment. SIP provides two main important operations: registration and user discovery. To enable SIP within MANET, these two operations must be presented in MANET as well. In this section, we present a short summary of each existing SIP over the MANET approach.

Early works on SIP over MANET is proposed by Khlifi et al. (2003). It presents a framework for a conference signaling using SIP. This framework allows a MANET user to discover, initiate conferences, and join existing conferences with other users. When the user joins MANET, it broadcasts a REGISTER request containing its SIP URI and IP address to all other users, as shown in Fig. 2.33. Node A broadcasts its REGISTER messages to all users. Other users, then, record this REGISTER information in their cache for future use. Each user has to listen on port 5060 to become aware of broadcast SIP URIs other users' locations. To create a conference session, a leader or user agent that initiates the conference, periodically broadcasts REGISTER request in MANET. The authors propose adding a new field to the header of the REGISTER message. This field is conf-ID containing all users of the ongoing conference that belongs to this leader. When a user agent wants to join the existing conference, the UA sends an INVITE message, containing Conf-ID, to the leader. After receiving the INVITE message, the leader must send all participants other messages in order to inform them about new participants. Distributed registrar is the key advantage of the proposed framework. There is no a centralized registrar server so that the system prevents a single point of failure. However, there are many drawbacks. The main limitation of the proposed framework is that there will be a large number of REGISTER requests exchanged, leading to high overhead. The authors use AODV to be the underlying ad hoc routing protocol, and the RREQ and

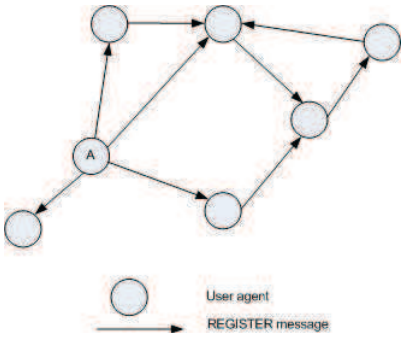


Figure 2.33: Broadcast retransmissions of REGISTER message

RREP message of AODV are replaced by SIP REGISTER request. As a result, every node must run modified AODV, which does not offer backward compatibility with the AODV standard. The next disadvantage is the scalability problem since it is based on pure broadcast. The number of broadcast messages increases largely because of the number of participants. Moreover, this framework does not offer full SIP functions, but it only provides SIP over MANET in conference scenarios.

Next existing work of SIP over MANET is proposed by L. Li & Lamont (2004). SIP is deployed over OLSR based on the use of a cross layer, integrated application and routing layer, to support proxy-based and proxy-less systems. In a proxy-based system, there is at least one SIP proxy server. The SIP proxy server periodically broadcasts its presence so that clients can know the location of the SIP proxy server. All nodes that participate in SIP must register themselves at this SIP proxy server. In contrast, there is no proxy server in proxy-less SIP MANET. Every node broadcasts its SIP URI and location via MPR forwarding at regular intervals. The authors create a new OLSR header, called Service Location Extension (SLE) in order to advertise information about the location of SIP proxy servers. The SLE is divided into two types: Automatic Service Advertising (ASA) for push style discovery and Client Query Service Advertising (CQSA) for pull style discovery. ASA, containing SIP URI and IP address, is advertised by SIP proxy in order to inform nodes in MANET of its presence. CQSA message is a query message, broadcast by a query node. The authors conclude that ASA scheme is suited for proxy-based scenarios, and CQSA is appropriated for proxy-less scenarios where node mobility is high. The benefit of this work is the use of new OLSR header extension that offers the compatibility between non-SIP and SIP nodes in the network. When a non-SIP node received SLE messages, it processes forwarding mechanism as described in OLSR forwarding algorithm. The next advantage is that SIP application layer is integrated into an ad hoc network layer to gain maximum performance for the SIP service discovery process. The drawback of this proposal is that the nodes still use broadcast CQSA message to discovery SIP users.

Banerjee & Acharya (2004) propose a cross-layer designed SIP-based service over Cluster-Based Routing Protocol (CBRP). The authors claim that a proactive routing protocol's pre-computed routes are costly, and a reactive routing protocol has a very high delay when it performs the route discovery process. As a result, the cluster based routing protocol is selected as the underlying routing protocol for SIP because CBRP lies in between proactive and reactive protocols. Instead of flooding SIP messages to the entire MANET, flooding

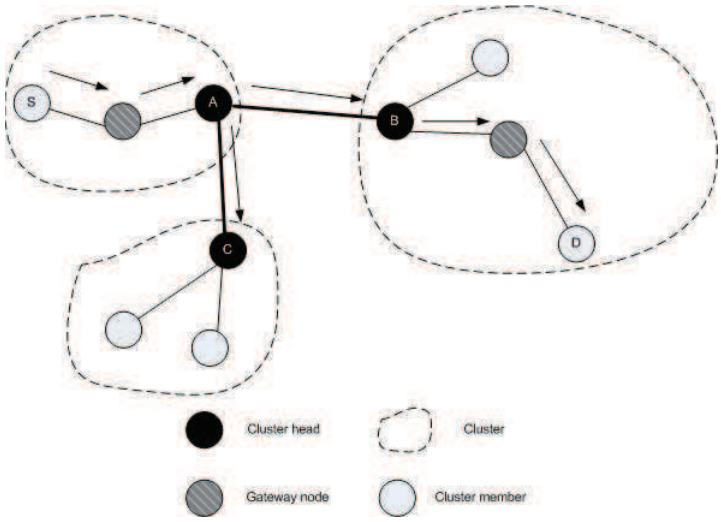


Figure 2.34: SIP endpoint/route discovery

takes place through the clusterhead only. A clusterhead acts as a registrar SIP server, where each node within the cluster registers itself to its own clusterhead. For example, in Fig. 2.34, node S must register to its clusterhead, which is node A. This REGISTER message is propagated throughout clusterheads in MANET. Similarly, node D registers itself by sending its SIP REGISTER message to node B. In addition, the clusterhead node also functions as a SIP proxy in order to route SIP messages. The authors use an integrated approach where SIP is merged into CBRP so that the SIP endpoint can know its clusterhead by retrieving the information from CBRP packets. The simulation results show the latency SIP user discovery, which is between 18–33 s. The strength of this approach is the reduction of the number of broadcast SIP messages because only clusterheads are allowed to forward SIP messages.

Fu et al. (2005) propose a novel signaling system for multiparty sessions in P2P ad hoc networks. The authors try to improve the framework to use SIP in ad-hoc networks (Khlifi et al., 2003) by introducing hierarchical clustering architecture. Like any clustering system, there is a super member, acting as a clusterhead. Every super member must have direct links to all super members of the neighboring clusters. However, normal members should connect to only one super member, like a star topology. Only super members have the right to pass conference information. The topology is shown in Fig. 2.35. The authors prove their concept by conducting a testbed running on eight computers. They conclude that their system generates a lower number of overhead messages as compared to Khlifi et al. (2003). The authors extend the work to support conference in integrated 3G/MANET (Fu et al., 2006) based on the clustering concept.

Banerjee & Acharya (2004) and Fu et al. (2005) approaches are cluster-based SIP registrar/proxy. The main advantage of their approaches is lower signaling overhead as compared to broadcast-based approaches. This is due to the fact that only clusterheads exchange SIP messages. However, there are many drawbacks when using this cluster-based architecture. First, maintaining clusters requires complex tasks such as creating, deleting, splitting, and merging clusters, which results in high power and memory consumption. Moreover, any changes in clusterheads due to the mobility of nodes cause a

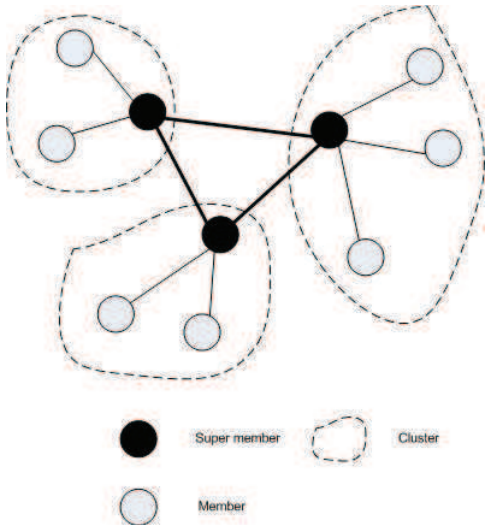


Figure 2.35: Cluster-based architecture

large number of messages to be exchanged in order to repair the broken clusterheads.

The next attempt on SIP over the MANET (dSIP) is proposed by Leggio et al. (2005). The authors mention that Khlifi et al. (2003) and Banerjee & Acharya (2004) focus on how to integrate SIP and ad hoc routing protocol, but they do not pay attention to building complete SIP-related functionalities, such as efficient registration. Therefore, the authors aim to define a framework for generic and flexible deployment of SIP in MANET. In other words, this SIP framework can be applied to any ad hoc routing algorithms. The authors assume that SIP on MANET nodes uses an external registrar located in the Internet. A MANET gateway does not perform any SIP functionalities. In this framework, SIP operations in MANET can be divided into two steps: discovering a SIP user and initiation sessions between users. The first step is very important because when a new node joins MANET, it does not know other SIP users. dSIP uses the pure broadcast technique to distribute a SIP REGISTER request to the whole network. A node in dSIP periodically advertises SIP REGISTER containing its SIP URI along with its IP address or a binding. The rest receiving the REGISTER request keeps this binding in its cache for possible future use. However, the time-limited binding is finally removed from the cache. When the binding is invalid in the cache, the caller has to wait for the next binding advertisement to extract the callee's IP address, which can be used to initiate a call. However, the method relies heavily on broadcast mechanisms, which do not offer scalability.

Manner et al. (2006) uses dSIP(Leggio et al., 2005) for a user discovery mechanism on MANET. However, the authors introduce a SIP gateway, actually a physical MANET gateway that provides connectivity to the Internet. First, the SIP gateway functions as a proxy for MANET nodes. Every SIP on MANET nodes must send REGISTER messages to the SIP gateway. Second, the SIP gateway acts as SIP UAs on behalf of its MANET nodes in order to forward REGISTER requests to an external registrar. However, the SIP gateway does not have the registrar function implemented. All SIP users on MANET must register to external SIP registrars. The SIP gateway also provides NAT capability to its MANET nodes so that none-routable addresses can be assigned within MANET. In other words, the SIP gateway provides NAT traversal for SIP. The authors use their own

modified SLP, allowing nodes to discover its gateway address. The shortcoming of this approach is that the SIP gateway based on centralized architecture, leading to a single point of failure and bottleneck problems.

Castro & Kassler (2006) propose SIP on Internet-connected MANET environment, where centralized registrars/proxies are located in the fixed IP networks similar to Manner et al. (2006). The difference between these two approaches is that Castro & Kassler (2006) provide SIP registrar functionality at the MANET gateway, while Manner et al. (2006) do not.

Zhang et al. (2006) propose an integrated SIP-based session establishment mechanism with DSR extensions. It is a cross-layer design, where SIP request is included in a routing packet. DSR control packets are modified to SIP-RREQ and SIP-RREP, which append four fields: source SIP URI, destination SIP URI, Call-ID, and Cseq. These appended fields are used as a call invitation. Nodes in DSR can learn a new SIP URI and IP address mapping information for other SIP-RREQ and SIP-RREP as well. Due to a long delay in the route discovery process, the average call setup delay in a network of 50 nodes within an area of $1000\,m \times 1000\,m$ is around 5 s. This approach does not provide DSR backward compatibility, which is the main drawback.

Similar to L. Li & Lamont (2004)'s approach, the other SIP on OLSR is proposed by Wang et al. (2006). The authors use OLSR packet's extension to carry a SIP message. The SIP message is disseminated in the whole network through MPR's forwarding technique. However, the authors do not explain how their approach provides SIP register operation, which is one of the SIP on MANET requirements. In addition, their simulation results do not include call setup delay, an important metric for the evaluation. Finally, this approach does not provide a SIP solution in Internet-connected MANET.

SIPHoc (Stuedi et al., 2007) is a middleware infrastructure for SIP on MANET. To discover a SIP user on MANET, SLP is used. SLP piggybacks by appending service information onto routing messages. A node has a routing handler, a software, which intercepts raw routing packets sent from its network layer. Then the routing handler appends piggybacked service information into the raw routing packets. At the receiver's end, its routing handler captures the routing packets to extract the piggybacked service information before passing the routing packets to its network layer. In an Internet-connected MANET environment, a gateway and connection provider software, installed at a MANET gateway, provide a layer two layer tunnel between the Internet access point and nodes within MANET. The layer two tunnel is a bridge, which allows a MANET node to virtually connect to the Internet. The MANET node also can use MANET SLP to discovery its gateway. Once its gateway address is found, the MANET node creates a bridge interface between itself and the gateway. This MANET bridge inter- face is assigned an IP address via DHCP server located in the infrastructure network. Finally, a SIP user on this MANET node uses the bride interface to send a SIP REGISTER request to an external SIP server. One of testbed results shows a call setup delay between an OLSR node and a fixed IP node of around 250 ms. The physical distance between the OLSR node to its gateway is four hops. SIP interoperability between MANET and the Internet is the key advantage of this approach. However, the first limitation is routing backward incompatibility since SLP piggyback method needs to modify routing packets. A bottleneck problem is the other drawback when the number of layer two tunnels increases. N

MANET nodes require N bridge interfaces in their gateway. The tunneling is considered a costly overhead. Later, the authors introduce social networking to MANET by using Mobile Ad hoc Network Directory (MAND) (Stuedi et al., 2008) based on SIPHoc. In their demonstration, they use ten Nokia N810 tablets to form an ad hoc network with AdSocial application allowing a user to distribute and discover other users' profiles.

A SIP-based multicast framework(Yu & Agarwal, 2005) propose to use a multicast overlay network based on mesh clustering architecture. Nodes on this multicast forms clustering network, which is composed of clusterheads or gateway and none gateway nodes. A none gateway node must register itself to its gateway node, which is locally one-hop away. To enable SIP on this multicast network, the authors use SUBSCRIBE and NOTIFY to register via the flooding technique within the multicast network. Likewise, an INVITE request is forwarded to every node in the multicast network. Even though the multicast network is used, exchanging SIP requests is based on the flooding mechanism. Forming the mesh clustering network introduced high control overhead because nodes in the multicast network periodically flood the whole network. Moreover, the clustering architecture is not suitable in MANET, where node mobility causes frequent topology changes. The authors only show a number of messages creating meshes and a clustering network. The result of a call setup delay is not included in their evaluation.

MANETSip (Fudickar et al., 2009) uses On-Demand Multicast Routing Protocol (ODMRP) (Lee et al., 2001) on MANET to build mesh-based multicast routes. After forming the multicast overlay network on OLSR, each node broadcasts SIP REGISTER message via multicast communication. However, creating and maintaining such a mesh multicast network involved in periodic flooding control messages introduces more overhead to MANET. In their evaluation, their testbed includes only three nodes, forming two-hop network, and the average registration delay is 164 ms. Moreover, the authors do not provide the results of the delay in call setup.

In SIP terminal mobility support on MANET, Y. S. Chen et al. (2006) propose MIP6-MANET, IPv6 and SIP-based mobile ad hoc networks. This approach is based on the mobile IPv6 mechanism that ensures a session's continuity, while SIP is used to support binding update between a Mobile Node (MN) and its Home Router (HR). The authors do not focus on distributed SIP on MANET solutions. The MN uses its HR as SIP registrar/proxy based on centralized architecture. Home, foreign, and corresponding routers are implemented with SIP redirect and proxy functionalities. When the MN moves from a home to foreign network, it must send a binding update containing a new Care of Address (CoA) along with a new SIP address to its HR. Keeping SIP up-to-date with its MNs binding information, the HR can forward an INVITE request destined for its MN regardless of the MN's location. Moreover, there is no need to resolve the SIP URI address with a DNS server. The authors modify and redefine the ICMP6 packet to support Destination Sequenced Distance Vector (DSDV) messages and Router Advertisement (RA) along with binding updated messages in MIP6 to provide their modified SIP binding with updated information. Consequently, these modified packets cause backward incompatibility with both standard MIP6 and SIP. Although the re-INVITE mechanism can be used to ensure the session's continuity, MIP6-MANET does not take advantage of this. The session continuity is still handled by MIP6. The handoff delay when the MN moves to the other network is 5000 ms, excluding the triangle routing latency of 500 ms when a tunnel is setup. MIP6-MANET is not P2P SIP on the MANET approach. It is based on

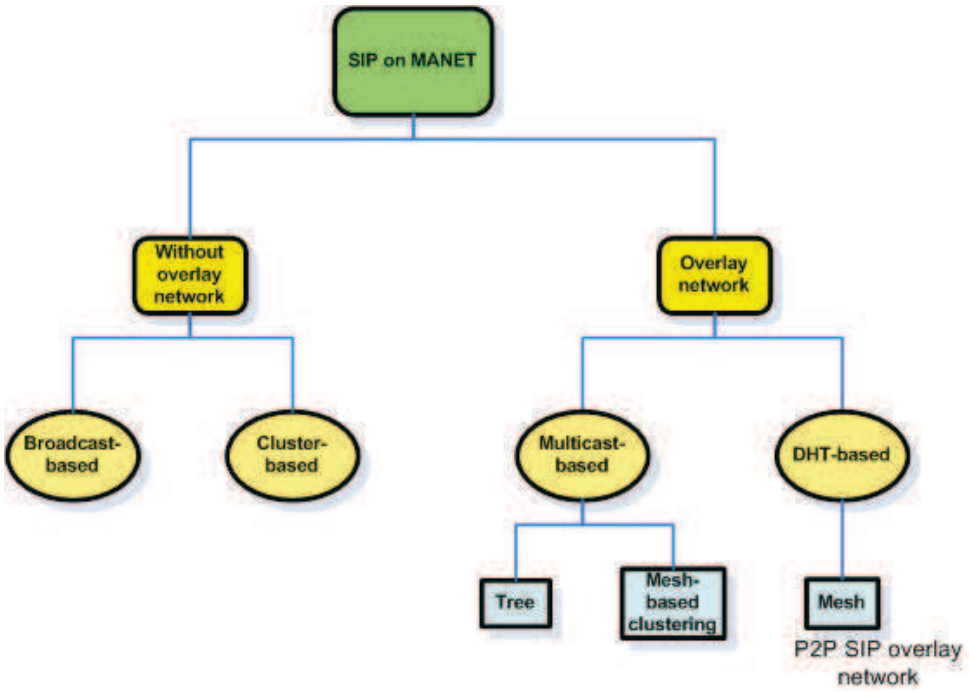


Figure 2.36: The classification of SIP on an isolated MANET

the centralized SIP architecture. It does not provide either P2P SIP registration or user discovery on MANET. However, we include MIP6-MANET to compare with our scheme's terminal mobility handoff.

According to reviewed approaches, some or all MANET nodes have SIP functionalities, which are a registrar and a proxy. A SIP user location can be resolved dynamically within MANET. Moreover, this architecture prevents a single point of failure as opposed to the centralized SIP architecture. Most P2P SIP on MANET approaches use resource discovery mechanisms in section 2.2.2 to provide SIP user location discovery. Thus, P2P SIP on MANET approaches can also be categorized into P2P SIP on the overlay network and without overlay network as shown in Fig. 2.36.

SIP on MANET approaches without overlay network can be grouped into two types: broadcast-based and cluster-based SIP registrar/proxy. Both types can either take the layered or cross-layered approach. For example, in the layered approach, standard or modified SIP messages are exchanged among MANET nodes through resource discovery protocols such as SLP or JXTA on top of the routing protocol. On the other hand, in the cross-layered approach, a routing packet is modified or extended to carry SIP messages.

In broadcast-based SIP registrar/proxy without overlay support (Khlifi et al., 2003; L. Li & Lamont, 2004; Leggio et al., 2005; Manner et al., 2006; Zhang et al., 2006; Castro & Kassler, 2006; Wang et al., 2006; Stuedi et al., 2007), each MANET node is a SIP registrar/proxy. A SIP user on MANET floods its SIP REGISTER to the whole network to register itself. Other users may cache this register information for later use. In user discovery for a call setup, a caller does not know an IP address of the callee. The caller broadcasts a SIP INVITE containing the callee's SIP URI to the network. Every SIP user on the MANET node compares SIP URI in the INVITE request with its own SIP

URI. If they match, it means that this SIP user is the callee. Then, it replies SIP 180 RINGING response back to the caller. In the cluster-based SIP registrar/proxy (Banerjee & Acharya, 2004; Fu et al., 2005), only clusterheads act as registrars/proxies. Nodes within a cluster except the clusterhead are called members of this cluster. After receiving a REGISTER request from its cluster members, the clusterhead forwards this request to other clusterheads. Only clusterheads are allowed to forward the SIP requests. These mechanisms use flooding SIP requests among nodes, resulting in introduction of high overhead to the network leading to problem with scalability.

SIP on MANET with overlay network is classified into multicast-based and DHT-based approaches. Normally, MANET routing protocols do not support application multicast and broadcast sockets. Therefore, multicast routing protocols are proposed to provide multicast socket communication for the applications. MANET nodes form a multicast network based on multicast routing protocols, tree or mesh topology. SIP is deployed on these multicast networks, which allow SIP requests to be distributed via multicast channels. SIP on MANET multicast-based approaches are SIP-based multicast framework(Yu & Agarwal, 2005) and MANETSip (Fudickar et al., 2009). Even though the multicast network allows nodes to exchange SIP requests more efficiently than those approaches without overlay support, creating the multicast network on MANET involves periodic flooding of control messages, which still add high overhead to the network. In addition, SIP lookup time of these approaches depends on the number of nodes participating in the multicast network.

In addition, we discuss the key challenges of P2P SIP on MANET design, which are the SIP user lookup time, the interoperability between SIP users on MANET and Internet, and terminal mobility support.

The lookup time is defined as a number of times that a query message is processed at nodes before it reaches the destination. These P2P SIP on MANET approaches use $O(N)$ or $O(\#clusterhead)$ to locate a SIP user. The approaches with lookup time of $O(N)$ (Khlifi et al., 2003; L. Li & Lamont, 2004; Leggio et al., 2005; Manner et al., 2006; Zhang et al., 2006; Castro & Kassler, 2006; Wang et al., 2006; Stuedi et al., 2007; Fudickar et al., 2009) use flooding SIP requests for registration and user discovery. For example, a node broadcasts an INVITE request to the network. Each intermediate node must process this request by matching its SIP URI and with one in the INVITE request. Banerjee & Acharya (2004), Fu et al. (2005), and Yu & Agarwal (2005) have the search time of $O(\#clusterhead)$, where a member cluster sends a SIP INVITE request to its clusterhead. If its clusterhead cannot find a match the SIP URL in its location database, it forwards this INVITE request to its clusterhead neighbors, while the SIP-based multicast framework and MANETSip have lookup time of $O(M)$, where M is the number of multicast nodes.

Most SIP on MANET approaches provide SIP register and user discovery operations within MANET. However, they do not address how their protocols work in heterogeneous networks to offer interoperability between MANET and Internet SIP users. Leggio et al. (2005), Castro & Kassler (2006), Manner et al. (2006), and Stuedi et al. (2007) provide their solutions for Internet-connected MANET environment. However, they still rely on the centralized SIP registrar/proxy either located at the Internet or a MANET gateway. This centralized registrar/proxy not only causes a traffic bottleneck, where SIP requests are sent to the gateway, but also a problem associated with a single point of failure. In

the centralized architecture, one or a few MANET gateways, called SIP gateways, keep users' SIP binding information on MANET. The SIP gateways also forward received SIP REGISTER requests from MANET users to an external SIP registrar on the Internet. Castro & Kassler (2006)'s approach uses this technique to a register a MANET SIP user to SIP registrars on the Internet. Whereas in Leggio et al. (2005) and Manner et al. (2006) approach, the MANET gateway does not keep SIP binding information, but it forwards the information to an external SIP servers on the infrastructure. SIPHoc (Stuedi et al., 2007) provides a MANET gateway with layer two tunneling capability. Whenever a SIP user on MANET needs to contact a SIP on the infrastructure, this MANET node creates a layer two tunnel or bridge with its gateway to send/receive packets with the fixed IP node. The MANET gateway does have SIP functionality, but it only provides bridge interface with nodes inside and outside MANET.

Finally, none of the approaches, except Y. S. Chen et al. (2006)'s explicitly focus on the SIP terminal mobility solution and measurement even though SIP provides this mobility support.

In this dissertation, we propose P2P SIP on overlay DHT-based mesh network as categorized in Fig. 2.36. Our approach uses DHT to support SIP registration and user discovery operations. Moreover, P2P SIP overlay network is based on fully distributed architecture, which prevents a single point of failure problem. All SIP requests exchanged among the overlay nodes are carried in unicast packets. Our scheme uses the mesh overlay network, which guarantees a constant lookup time. SIP for the heterogeneous networks is supported by our P2P SIP overlay network, sharing the same P2P structured overlay network on both MANET and Internet users as shown in Fig. 2.37. In addition, it provides SIP terminal mobility support for both MANET and Internet users in fully decentralized architecture. A MN on P2P SIP overlay network can roam to any MANETs or fixed networks while it still maintains an ongoing session.

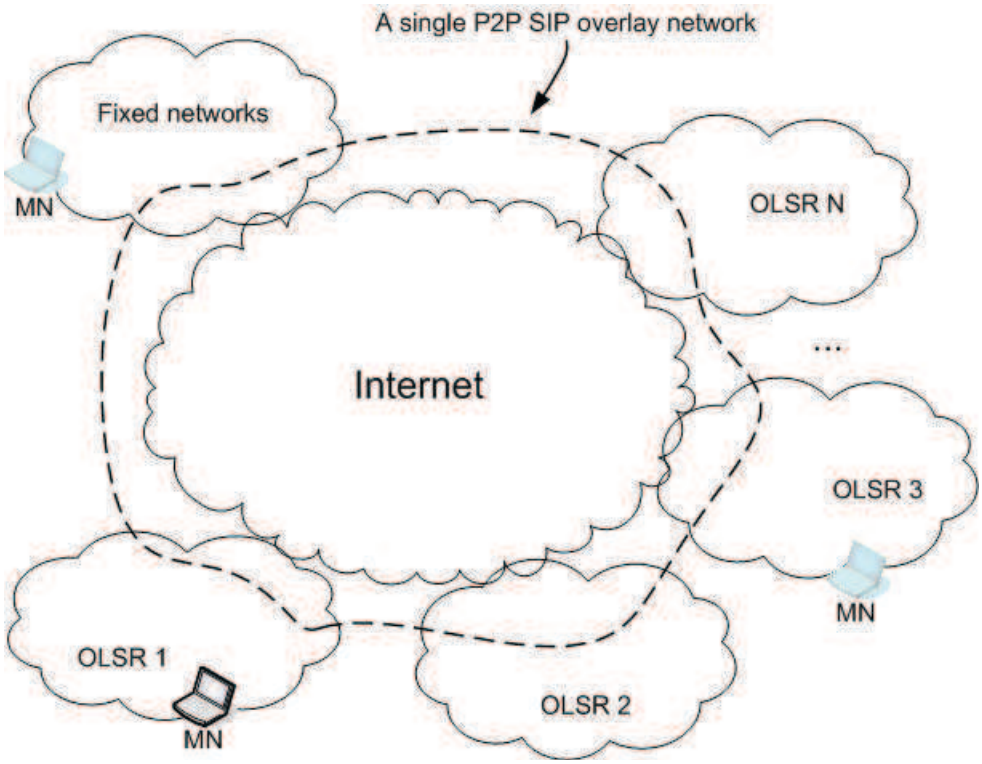


Figure 2.37: A single P2P SIP overlay network covering MANET and Internet

Chapter 3

Structured Mesh Overlay Network (SMON)

This chapter presents Structured Mesh Overlay Network (SMON), overlay network framework for P2P applications on MANET proposed by this dissertation. SMON employs Distributed Hash Table (DHT) on top of Optimized Link State Routing (OLSR), a proactive routing protocol. SMON is an improvement on CrossROAD (Delmastro, 2005), which uses pure flooding techniques to form and maintain an overlay network. In SMON, we introduce a new algorithm for overlay formation and maintenance in order to reduce the number of control overhead. We show through simulations that SMON reduces the number of control overhead by around 95% compared with CrossROAD, while it maintains the same query success ratio and query delay as CrossROAD.

The chapter is organized as follows. We first review the conceptual design of CrossROAD in section 3.1. The next section 3.2 explains how SMON overcomes the limitations of CrossROAD including our proposed algorithm of creating and maintaining SMON on top of OLSR. We include the evaluation of SMON in section 3.3.

3.1 A review of CrossROAD

CrossROAD (Delmastro, 2005) is a P2P structured overlay network on top of OLSR based on DHT, which provides service lookups for P2P applications. CrossROAD organizes its overlay network as a mesh network, which can be created and maintained based on a cross-layer design as described below.

First, the authors propose to use the OLSR messages such as HELLO and TC carrying information that identifies the IP address of a node in the mesh network for node discovery. The following example explains node discovery in CrossROAD. A CrossROAD network contains nodes A, B, and C. Node A broadcasts its PublishService containing its IP address within OLSR. Node B and C receive A's PublishService and know that node A is on the CrossROAD network. Similarly, nodes B and C also advertise their PublishService. All nodes periodically exchange their PublishService messages so that they know all IP addresses of nodes taking part in the same mesh network.

Moreover, CrossROAD uses routing information of OLSR to maintain its mesh overlay network. Since the underlying routing protocol is the OLSR which provides a complete knowledge of the network topology, CrossROAD can directly use the routing information to maintain and update the overlay network topology. For example, node A can remove node B from its overlay network when a route to node B is not presented in its routing table.

When a new node wishes to join the CrossROAD network, it must inform other participants by sending its PublishService message. This PublishService allows other participants to know the IP address of the joining node. At the same time, the joining node

collects PublishService messages from other nodes as well. Each CrossROAD node must advertise its PublishService at the same interval as the routing control messages to ensure synchronization and to make the overlay network consistent. This process is called *push method*, where each node must periodically advertise its own PublishService. Having received PublishService messages from CrossROAD nodes, the joining node extracts IP addresses from the messages and put them in its CrossROAD routing table, which contains hash values of all participants' IP addresses. A CrossROAD node can then use this table to find a destination, which is responsible for maintaining an object with a given key based on DHT. When a node wants to leave the CrossROAD network, it sends a disconnect message, which informs all the other participants.

Due to the mesh topology, a CrossROAD node can use the shortest paths from the routing table to communicate with other nodes searching for an object; hence, it has a low lookup delay. Furthermore, the cross-layer design allows CrossROAD to adjust and update its overlay network topology based on routing information at the network layer to optimize the overlay network performance. However, CrossROAD uses the pure broadcast mechanism to discover participants of the overlay network, resulting in the generation of many redundant message retransmissions which in turn cause high bandwidth consumption and network congestion. Moreover, the joining node must wait for an interval time of advertised PublishService in order to receive all messages from other peers. This may cause a long delay when the joining peer enters the overlay and before they gain a complete list of participants. Besides, the OLSR routing message has to be modified in order to carry the CrossROAD messages. Consequently, this routing message modification does not offer backward compatibility with OLSR standard. No implementation of CrossROAD has yet been discussed.

3.2 Structured Mesh Overlay Network (SMON)

To address the limitations of CrossROAD, we introduce a new algorithm for overlay formation and maintenance in order to reduce the number of overlay broadcast messages and reduce the waiting time of the joint operation. We take into account that our design should provide a backward compatibility with OLSR standards by using the OLSR message extension rather than modifying routing messages such as HELLO or TC to carry SMON messages, which in turn alter the OLSR specification. We define new overlay control messages using message types, which are outside the range of OLSR reserved message types. Finally, we evaluate the SMON concept through simulations and with actual implementation to fulfill all technical details that are not provided in CrossROAD.

SMON, like CrossROAD, uses the information from the OLSR routing table to create its DHT table. This DHT table is adjusted in real-time according to event notifications received when changes take place in the OLSR routing layer such as topology alterations. The mesh overlay network can be simply constructed and maintained if the underlying routing protocol is a proactive routing protocol because every node maintains routes to all reachable destinations. A SMON node can extract the information from its routing table to use in creating and updating the mesh overlay network. Any changes in routing table, such as node disconnection, immediately reflect changes in the DHT table, resulting in

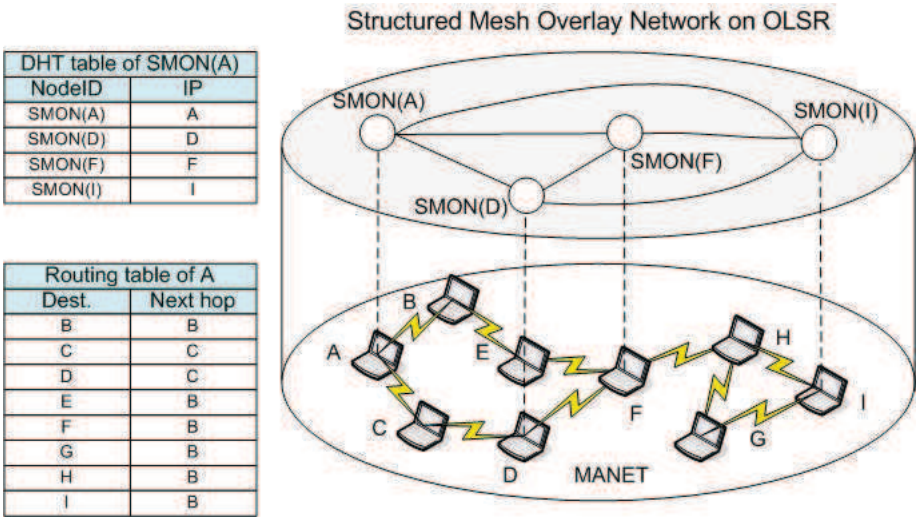


Figure 3.1: An example of SMON with OLSR routing and SMON DHT tables

fast convergence.

The DHT table is very important because peers rely on the DHT table in order to perform overlay functions such as joining, and leaving SMON including P2P registration and search. Each peer or node is assigned a unique ID, which is a hash value of its IP address. Let us call this hash function *SMON*. Thus, the ID is then *SMON(node's IP address)*.

Once SMON is established, a peer can store and search for an item in a distributed manner using a node ID and an object ID. An object ID, a key, can be obtained from a hash value of item properties such as filename or content. The object value is stored at a SMON node whose node ID is the numeric closest to this object ID. Each peer in SMON maintains a DHT table that stores information about all other overlay peers. A row of the DHT table is composed of node ID and IP address of a peer. The DHT table is the subset of the routing table. Not all OLSR nodes have to join SMON as shown in Fig. 3.1. The DHT table of *SMON(A)* contains a list of all members, *SMON(D)*, *SMON(F)*, and *SMON(I)*. This mesh architecture allows a peer to use a direct path to send a packet to a target peer without having it first relayed to intermediate peers. Since OLSR provides the shortest path routes to all destinations (Clausen & Jacquet, 2003; Nguyen & Minet, 2007), a SMON node can also contact all peers via the shortest paths as presented in its routing table. For example, *SMON(A)* sends *SMON(F)* a packet. *SMON(A)* can find the IP address of *SMON(F)* in its DHT table. According to the routing table of A, the packet is forwarded to B, the next hop. Receiving the packet, B forwards it to E. After that, E sends the packet to F, and finally the packet is arrived at *SMON(F)*. The hop-by-hop packet forwarding based on OLSR routing table provides the shortest path from *SMON(A)* to *SMON(F)*. Similar to those existing overlay networks, SMON is just the application overlay network whose DHT table depends on the OLSR routing table. If the routing table is unstable during routing convergence, the DHT table will be unstable as well.

In the following subsections, we will describe operations on SMON, which are join, leave, merge, and split.

3.2.1 Join operation

Nodes wishing to participate must advertise some messages to inform peers in the overlay to recognize their presence and obtain their peers' presence. We term "peer" as one that already is in the overlay network. There are two ways to obtain the presence of peers, either the *push* or the *pull* method. In the push method, a peer periodically broadcasts its presence to others. The nodes wait for at most an interval time of the advertised presence to receive this presence. This may cause a delay when a new node joins the network in order to get to know all available peers in the overlay network. On the contrary, the pull method offers an on-demand peer discovery, where the new node floods a query whenever it needs. Other peers reply to the joining node with their addresses. It is noted that the joining node may or may not continue broadcasting the query even though it has already received other peers' information. However, if there are many nodes that request the peer discovery at the same time, it will initiate the same flooding query messages, resulting in unnecessary increase of control overhead. In order to optimize the way of obtaining the presences of other peers, a combination of push and pull methods can be deployed together in SMON.

We define SMON messages using message types which are outside the range of the message types reserved by the OLSR protocol. This design provides backward compatibility with OLSR standard by not modifying the routing message such as TC to carry SMON messages as in CrossROAD. The messages consist of JOIN, LIST OF ALL MEMBERS, and LEAVE. These SMON messages are embedded within the normal OLSR messages and are treated just like other OLSR messages. If unknown messages are received at nodes, they are forwarded via MPRs according to the OLSR forwarding algorithm. In Fig. 3.2, the first field indicates the type of the message, JOIN, LIST OF ALL MEMBERS, or LEAVE. The second field indicates the number of overlay members whose IP addresses are listed in the third field. The JOIN message is sent by a peer that begins just joined SMON, while the LIST OF ALL MEMBERS message is only advertised by a peer that had been selected as a primary peer. Finally, the LEAVE message is sent by a peer that wants to leave SMON.

When a peer wishes to join SMON, it uses the pull method by broadcasting a JOIN message containing its IP address specified in the list of IP addresses field. Next, the primary peer replies to the joining peer with the LIST OF ALL MEMBERS message. After the joining, peer receives this message. It locally constructs the DHT table, storing overlay peers according to information obtained from the message.

A SMON peer can be either a normal or a primary peer. Both types have the same functions, but a primary peer is the only one allowed to advertise the LIST OF ALL MEMBERS message. The primary peer is dynamically selected by using node ID in the DHT table. We use a simple approach that chooses the primary peer whose node ID is the smallest in the DHT table. Each peer can search for a peer with the smallest node ID by using its DHT table.

Considering that when node mobility does not cause a network partition and assuming that the routing tables of all nodes coverage to the same table. Therefore, all SMON peers have the same values in their DHT table as well. This is because the peers adjust their DHT table based on their routing tables. It means that only one peer is elected as

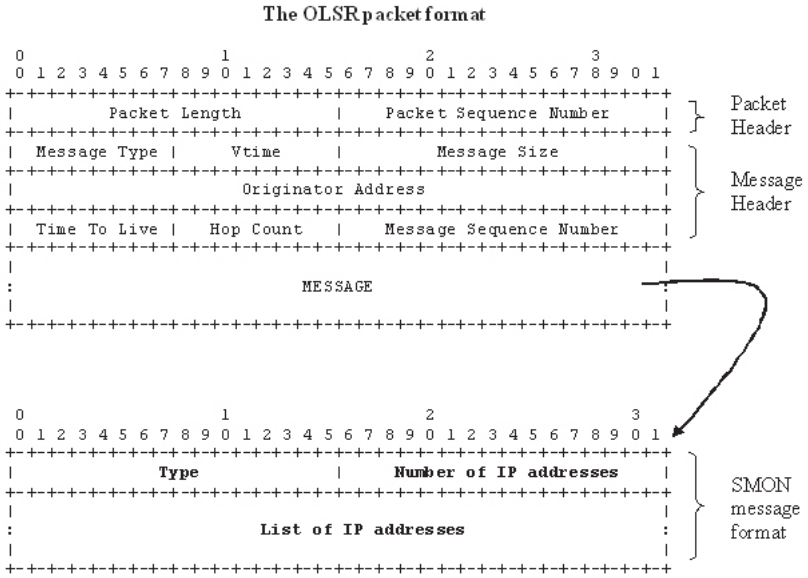


Figure 3.2: SMON overlay control message format

the primary peer for this overlay network. However, when movements take place causing MANET to split or when two or more MANETs are merged, our primary peer selection algorithm can adapt to this situation by re-selecting the new primary node of the resulting MANET as described in section 3.2.3.

The primary peer periodically broadcasts the LIST OF ALL MEMBERS message, informing normal peers about the current list of overlay peers. This periodic advertisement or push method helps each peer to synchronize its own DHT table with others in case of merging and splitting SMON. Each peer checks whether the list of IP addresses in the LIST OF ALL MEMBERS message contains its own IP address. If not, it must send a JOIN message to rejoin the overlay network. If the primary peer gets disconnected from the overlay network, another primary peer must be activated to prevent a single point of failure. Each LIST OF ALL MEMBERS message is stamped with a validation time using *vtime* filed in OLSR message. All normal peers expect to receive the LIST OF ALL MEMBERS message before *vtime* expires. If the primary peer cannot perform the broadcasting of the message within the *vtime* interval, the next primary peer, which is the next smallest node ID, will become active. To do so, each peer has its timer associated with *vtime*. When the timer expires, every node checks whether to receive the LIST OF ALL MEMBERS message from the primary peer or not. If not, every node compares its node ID with the next smallest one in the DHT table. Only the next primary peer (the next smallest ID) starts advertising the LIST OF ALL MEMBERS message. Thus, there is not a single point of failure for a primary peer.

The procedure of joining SMON is shown in Algorithm 1 line 1. When a peer wishes to join SMON, it broadcasts a JOIN message containing its IP address specified in the list of IP addresses field; only a primary peer with the smallest ID answers to the joining peer. After the joining peer receives the message, it constructs a DHT table locally according to the received messages. Moreover, the primary peer periodically broadcasts the LIST OF ALL MEMBERS, informing all other peers about a refreshed list of overlay peers.

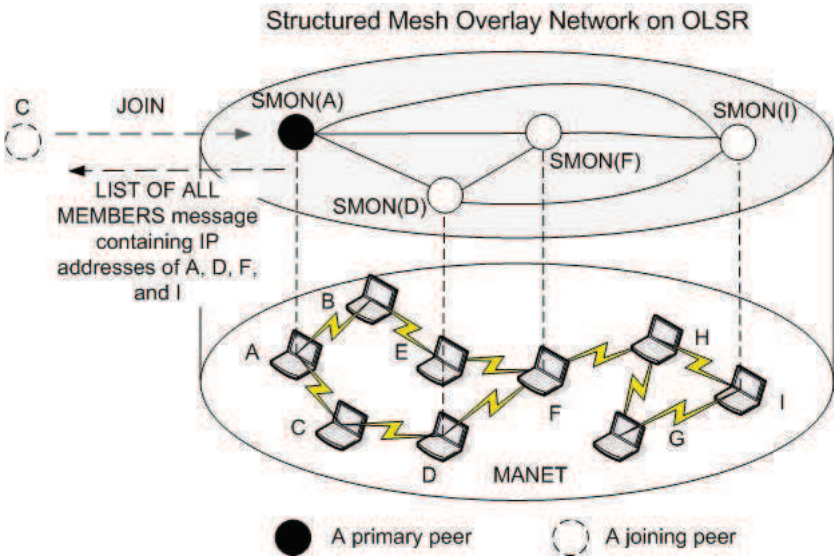


Figure 3.3: A joining peer enters the overlay network

Whenever the primary peer broadcasts the LIST OF ALL MEMBERS message, other peers on the overlay network can also overhear this message. For consistency, each peer checks whether the list of IP addresses in the message contains its own IP address. If not, it must resend a JOIN message to rejoin the overlay network. In Fig. 3.3, assuming that *SMON(A)* is the primary peer, it broadcasts a list message, containing list of IP addresses of overlay nodes. After the joining peer node C receives the list message, it constructs the DHT table locally according to the received message. Moreover, peers *SMON(A)*, *SMON(D)*, *SMON(F)*, and *SMON(I)* must update their overlay information and accept the joining peer into the overlay network by adding *SMON(C)* to their DHT table as shown in Algorithm 1 line 12.

3.2.2 Leave operation

When a peer, denoted as *SMON(X)*, wishes to leave SMON, it has to send a LEAVE message as shown in Algorithm 1 line 24. After receiving the LEAVE message, other peers remove *SMON(X)* from their DHT tables. However, *SMON(X)* can accidentally get disconnected from SMON due to a network partition, interference, or other reasons. *SMON(X)*'s entry should be removed from the DHT table as well even though the primary peer may not receive the LEAVE message. The OLSR of primary peer informs its SMON of the disconnection as displayed in procedure *EventNotification* in Algorithm 1. The primary peer then can remove the *SMON(X)*'s entry from its DHT table.

3.2.3 Merge and split operations

Next, we explain merge and split operations of two overlay networks. Each overlay network has a peer with the smallest ID whose DHT table contains members in its own overlay

Algorithm 1 Joining and maintaining the overlay network

Define:
O is a set of nodes in the DHT table sorted in ascending order according to nodes' hash values
R is a set of nodes in the OLSR routing table
L is a set of nodes in the LIST OF ALL MEMBERS message
n_j is a node in JOIN message
n_l is a node in LEAVE message
n is this joining node

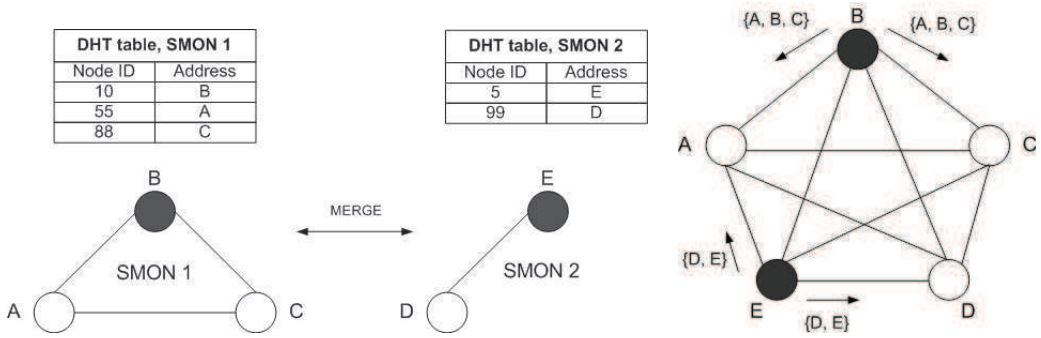
1: **procedure** JoiningOvelayNetwork
2: $O \leftarrow \emptyset$
3: $O \leftarrow O \cup n$ ▷ Add itself to the DHT table
4: Broadcast(a JOIN message)
5: **loop**
6: Sleep(an interval value)
7: **if** n is the first element of O **then** ▷ If this node is the smallest ID
8: Broadcast(a LIST OF ALL MEMBERS message)
9: **end if**
10: **end loop**
11: **end procedure**

12: **procedure** ProcessJOIN_LIST_LEAVEmessage
13: **if** receive a JOIN message **then**
14: $O \leftarrow O \cup n_j$ ▷ Add n_j to the DHT table
15: **else if** receive a LIST OF ALL MEMBERS message **then**
16: $O \leftarrow O \cup L$ ▷ Insert nodes in the message to the DHT table
17: **if** $n \notin L$ **then** ▷ If the LIST does not contain n
18: Broadcast(a JOIN message)
19: **end if**
20: **else**
21: $O \leftarrow O - \{n_l\}$ ▷ Remove the leaving node from the DHT table
22: **end if**
23: **end procedure**

24: **procedure** LeaveOverlayNetwork
25: Broadcast(a LEAVE message)
26: **end procedure**

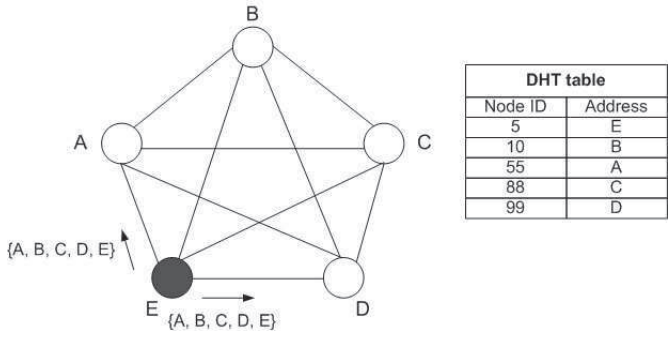
27: **procedure** EventNotification(R) ▷ Called when the routing table changes
28: $O \leftarrow O \cap R$ ▷ Update the DHT table to contain only reachable nodes
29: **end procedure**

network. The set of all peers in the first overlay is denoted by $P = p_1, p_2, \ldots, p_n$, where n is the total number of peers of the first overlay and the set of all peers in the second overlay is denoted by $Q = q_1, q_2, \ldots, q_m$, where m is the total number of peers in the second overlay. We assume that p_1 and q_1 are the peers with the smallest ID of the first and second overlay networks respectively. When these two overlay networks are merging, p_1 receives Q from q_1; similarly, q_1 receives P from p_1. Therefore, a new set of overlay members of p_1 and q_1 is a union set $P \cup Q = p_1, p_2, \ldots, p_n, q_1, q_2, \ldots, q_m$. After merging, either p_1 or q_1 which has the smallest ID continues broadcasting the new list of overlay members $P \cup Q$. The example of merge operation is shown in Fig. 3.4.



(a) Before SMON 1 merges with SMON 2, each SMON has its own primary peer

(b) During merging, the combined overlay network has two primary peers

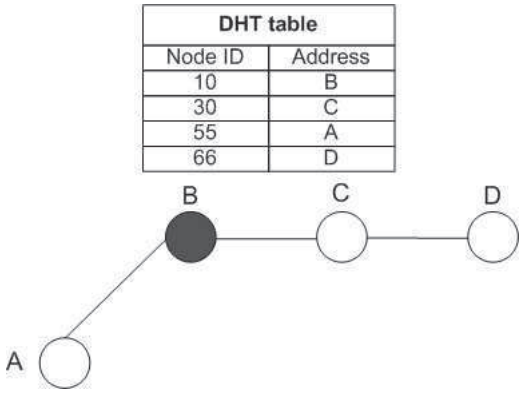


(c) After merging, the merged overlay network has one primary peer

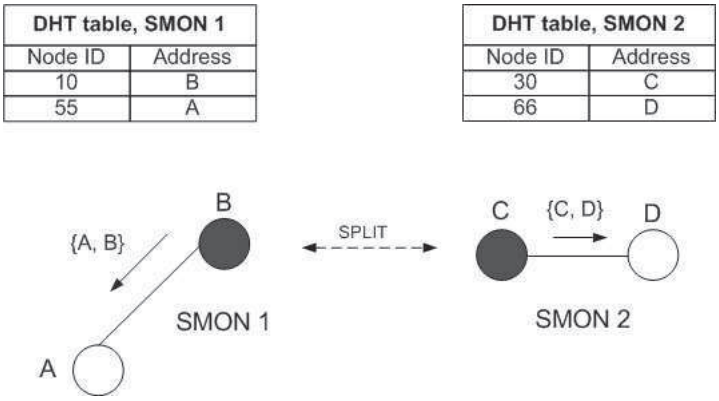
Figure 3.4: A SMON overlay network merges with another overlay network

We are interested in cases when two or more overlays are merging. Many peers can claim to be the smallest ID, resulting in broadcast storm problem. We show that the Algorithm 1 does not break the overlay network consistency. Let S_n denote an overlay network of n^{th} and min (S_n) is a peer that has the smallest ID of overlay network n^{th}. When n overlay networks merge $S_1 \cup S_2 \cup \cdots \cup S_n$, each $min(S_i)$ sends its list overlay member S_i, where $1 \leq i \leq n$. At the overlay S_i, the peer with the smallest ID of S_i receives several messages from $min(S_1), min(S_2), \ldots, min(S_n)$, which may not be in a sequence. It performs a union operation on the received list of overlay member messages $S_1 \cup S_2 \cup \cdots \cup S_n$, which is commutative and associative. The new peer with the smallest ID of merged overlay networks is $min(S_1 \cup S_2 \cup \cdots \cup S_n)$.

In a split operation, an overlay network is divided into two. Consider an overlay network with a set R containing all the peers in the network, we define set $R = r_1, r_2, \ldots, r_i$ where i is the number of peers in the overlay. After a split operation on the overlay network, set R is partitioned into two smaller sets S and T, where $R = S \cup T$ and $S \cap T = \emptyset$. Next, s_1 and t_1, which are peers with the smallest ID, begin advertising the full list of overlay members R within their partitions. Even though the set R is advertised, each peer locally checks the availabilities of the advertised peers by using the information from its OLSR routing table before performing resource registrations and lookups. The example of split operation is shown in Fig. 3.5.



(a) Before SMON breaks



(b) After splitting, each overlay network performs primary peer selection

Figure 3.5: A SMON overlay network splits into two overlay networks

3.3 Evaluation of SMON

We evaluate the performance of SMON by using Network Simulator, NS2.

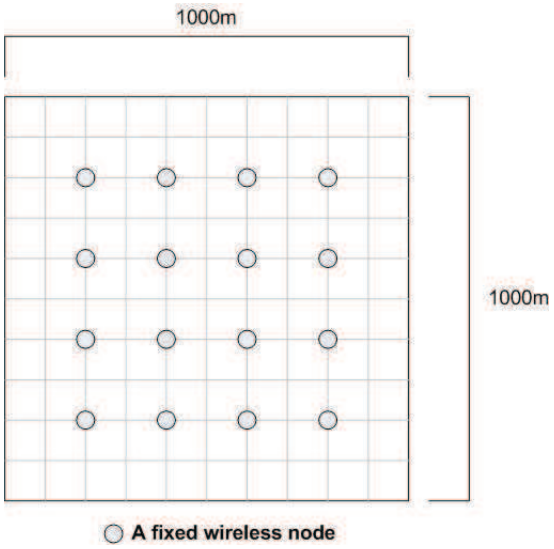


Figure 3.6: Backbone node' positions in the heterogeneous network

3.3.1 Scenario description

We are interested in evaluating the performance of SMON in post-disaster scenarios. In the first scenario, we assume that an existing telecommunication system in an isolated disaster area is not available. Therefore, we deploy a temporary network (MANET) covering the affected area in order to provide communication for recovery operation. In the second scenario, we assume that MANET is used as a last mile network which is connected to an infrastructured network via a MANET gateway. Rescue workers use MANET nodes to communicate with a remote command headquarter (HQ) located in the infrastructure.

The first post-disaster scenario is also called a heterogeneous scenario (Kanchanasut et al., 2007), a mixture of different types of mobile nodes, Personal Digital Assistants (PDAs) (20 nodes with 2 Mpbs and 100 m) and laptops (30 nodes with 11 Mpbs and 250 m) in a disaster area of 1000 m x 1000 m. We setup a temporary backbone network by evenly placing 16 out of 30 laptops in order to have their transmission ranges cover the entire area. These backbone nodes do not move. Fig. 3.6 only illustrates the positions of these nodes. The rest of the nodes are randomly placed in the area. The rescuers carrying PDAs and laptops freely move in any directions by walking. Therefore, a random waypoint mobility with maximum speed of 2 m/s (a fast walking speed), and a pause time of 60 s is used.

In the second scenario, we deploy MANET with a gateway to the infrastructure. One gateway interface is connected to the wired node (HQ), and the other interface is a wireless interface that connects to mobile nodes. The HQ lookups resources in MANET by sending resource queries to the MANET nodes via the gateway. Here, we extend the heterogeneous scenario to include the gateway as shown in Fig. 3.7.

For both post-disaster scenarios, there are 25 CBR flows as background data traffic. The packet size and rate are 512 byte and 3 packets per second. Every peer has one unique service, and the query rate is 2 times per second (the total is 240 queries). In the

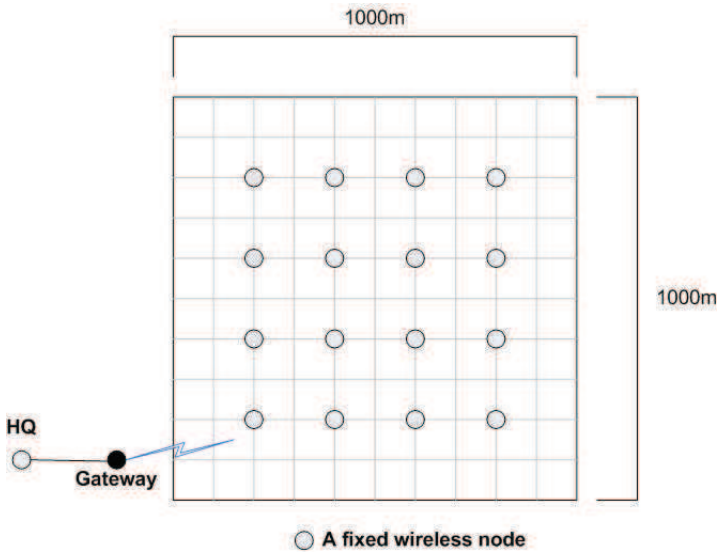


Figure 3.7: Backbone node' positions in the gateway scenario

heterogeneous scenario, each node randomly sends queries for required resources. On the other hand, in the gateway scenario only HQ sends queries to MANET. All simulations run for a duration of 900 s. Results are averages over a set of ten runs.

Furthermore, we evaluate the performance of SMON in the worst case scenarios where all rescuers freely move at different speeds within the affected area without the presence of backbone nodes. These scenarios show the performance of SMON based on a cross-layer design when node mobility causes frequent topology changes. Therefore, the performance of SMON is measured by varying the maximum speed of mobile nodes between 0 m/s (static), 2 m/s (fast walking speed), 10 m/s (slow speed vehicles), 20 m/s (fast speed vehicles), and 30 m/s (very fast speed vehicles) for an insolated MANET. We use the random waypoint mobility model of 50 nodes with a pause time 0 s in the same area of 1000 m x 1000 m square. All mobile nodes are equipped with IEEE 802.11b wireless interface with a bandwidth of 11 Mbps and a transmission range of 250 m. There are 25 CBR flows as background data traffic. The packet size and rate are 512 byte and 3 packets per second. Every peer has one unique service, and the query rate is 2 times per second. All simulations run for a duration of 900 s. Results are averaged over a set of ten runs.

3.3.2 Performance metrics

SMON performance is compared with passive resource discovery protocol (push model) and active resource discovery protocol (pull model). In the passive method, each peer periodically broadcasts its service information and its IP address to others. On the contrary, the active method offers an on-demand service discovery. When a requester wishes to find a service, it floods a query containing the service information to find an address of a peer that has this service. After receiving the query, every peer compares its service information with the one in the query. Only the peers that have this service will reply to the requester. It is noted that the requester continues broadcasting the query because

the requester will need the most up-to-date information about the location of the service. To support both passive and active discovery methods on OLSR, we define a new type of OLSR messages. OLSR nodes use theses new messages to exchange resource information based on passive and active discovery methods. Similar to SMON, these messages are also disseminated to the entire network via MPRs.

The following metrics are used to evaluate these resource discovery protocols. The first metric is the percentage of the increment of control overhead by passive discovery messages, active discovery messages, or SMON messages on OLSR routing messages. The percentage of the increment of control overhead of passive discovery protocol is:

$$C_{passive} = N_{passive}/N_{olsr} \times 100,$$

where $N_{passive}$ is the number of passive discovery messages transmitted by peers, and N_{olsr} is the number of OLSR routing messages transmitted by all nodes.

The percentage of the increment of control overhead of active discovery protocol is:

$$C_{active} = N_{active}/N_{olsr} \times 100,$$

where $N_{passive}$ is the number of active discovery messages transmitted by peers.

The percentage of the increment of control overhead of SMON is:

$$C_{smon} = N_{smon}/N_{olsr} \times 100,$$

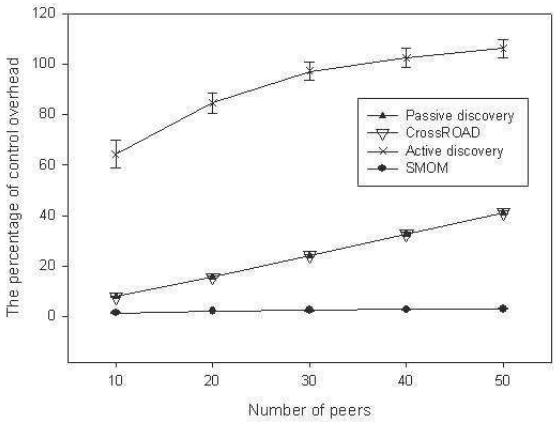
where N_{smon} is the number of SMON messages transmitted by peers. In passive and active resource discovery protocols, we term "peer" as one that generates resource discovery messages. However, in SMON, we term "peer" as one that already is in the overlay network.

The second metric is a query success ratio that is the percentage of success random lookups.

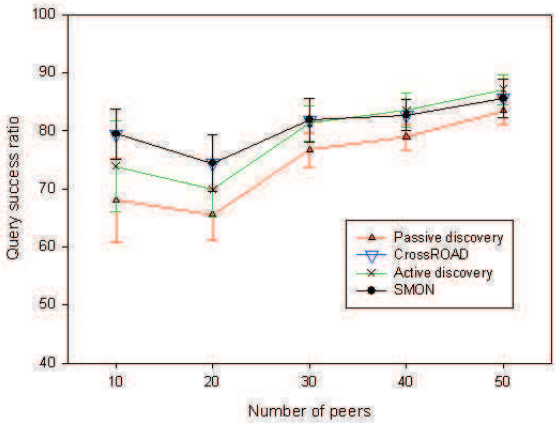
The last metric is an average query delay that is the difference between the time a peer uses in searching and finding a specified service.

3.3.3 Simulation results

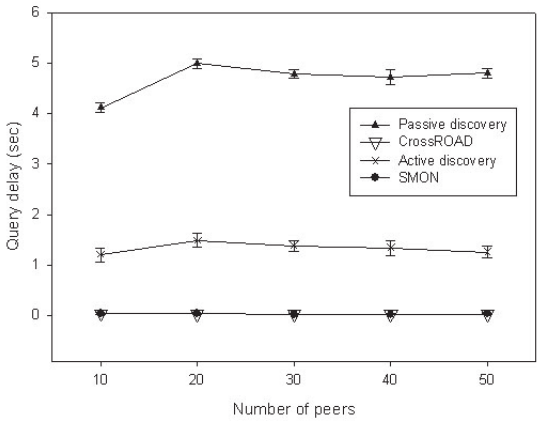
CrossROAD nodes use the passive discovery method to advertise their overlay presences. Therefore, the control overhead of CrossROAD is identical to the control overhead of the passive discovery method. However, for query success ratio and query delay, CrossROAD and SMON are the same because they use the same discovery mechanism.



(a) The percentage of control overhead



(b) Query success ratio



(c) Average query delay

Figure 3.8: Control overhead, query success ratio, and average query delay in heterogeneous network with 95% confidence level

In Fig. 3.8(a), the x-axis represents the number of peers, while the y-axis displays the percentage of the increment of control overhead for all protocols. As the number of peers increases, the percentage of control overhead increases in both active and passive discovery methods, while SMON control overhead slightly increases and remains as constant value. This is because the number of peers has no effect on the number of control overhead in SMON. All lines in the chart are displayed with 95% confidence level. However, confidence the interval of a line with low values (e.g. SMON) cannot be clearly seen when compared to a line with very high values (e.g. Active discovery method). Query success ratios of three protocols are similar as shown in Fig. 3.8(b). It confirms that using the overlay network does not decrease the query success ratio. In Fig. 3.8(c), a peer in the passive discovery model periodically advertises its service information every 10 s. Consequently, the average query delay is about 5 s. In the active discovery model, most of the delay is accumulated from jitter. As a basic OLSR implementation requirement, synchronization of control messages should be avoided. As a consequence, OLSR control messages should be emitted in such a way that they avoid synchronization. To avoid such synchronization,

a node should add an amount of jitter to the interval at which messages are generated and forwarded (Clausen & Jacquet, 2003). Whenever a node has to forward an OLSR message as a query, it should keep the message at least for jitter time, a random value between 0 to $HELLO\_interval/4$. Moreover, each node must process this query before forwarding it to MPRs. On the contrary, the query delay of SMON is the lowest because a service request is encapsulated in a unicast packet, which can be immediately forwarded to a destination.

In the gateway scenario, the wired node is the only one that asks for services via the gateway. Upon receiving the service requests, the gateway sends service queries to MANET on behalf of the wired node. The results from gateway scenario are shown in Fig. 3.9(a), Fig. 3.9(b), and Fig. 3.9(c). The percentage of control overhead is similar to that of heterogeneous scenarios because the number of nodes is the same. The query success ratios of the three models are similar, but SMON has the lowest query delay.

Next, the performance of SMON under different mobility environments is evaluated. Fig. 3.10 illustrates the percentage of control overhead of the three protocols, where the x-axis is the number of peers and y-axis is the percentage of control overhead. CrossROAD and passive discovery method share the same percentage of control overhead.

In the passive discovery method and CrossROAD, changing the maximum speed does not cause an increase in the number of control overhead, but rather an increase in the number of peers which brings about a rise in the number of control overhead. This is because each peer advertises its own service information. The percentage of control overhead of the active discovery method maintains almost the same pattern, no matter what the maximum speed is, except for static cases. The percentage of control overhead of all peers while static, as compared with other speeds, is the highest because the network connectivity stays the same throughout the simulation; as a result, more broadcast messages tend to be successfully sent and received. In SMON, since only the primary peers that have the smallest IDs periodically broadcast the list messages, the control overhead is significantly lower when compared to other protocols. We discover that SMON reduces the number of overlay control overhead 95% on average as compared with CrossROAD when a number of peers are 50.

In Fig. 3.11, the query success ratios of the passive discovery method and the active discovery method are nearly 100% due to the broadcast techniques. A flooding message is reproduced when it traverses MPR nodes. A peer tends to receive one of these duplicated messages even though some of them may be lost due to broken links in high mobility environments.

In SMON, the query success ratio is gradually reduced when the maximum speed increases from 20 m/s to 30 m/s as shown in Fig. 3.11(d) and Fig. 3.11(e). The reason for the decrease is because an OLSR node takes some time to detect neighbor link failures and to send TC messages with the new topology. Data packets that are forwarded along the failed path will be dropped in this transient period. In the simulation, many register, query, and reply messages are dropped due to the transient period, which causes a high degree of message lost. However, the query success ratios reach beyond 90%, and when the maximum speed is below 20 m/s.

Fig. 3.12 explains the query delay of three models by varying speeds of the mobile nodes.

The average delay of the passive discovery method model is about half of the interval time between advertisements. In the active discovery method, the average delay is approximately 1 s due to accumulated jitter at intermediate nodes as already mentioned. Whereas, the average query delay of SMON is below 0.30 s for all maximum speeds.

3.4 Discussions

We have presented the P2P overlay network cross-layer design on OLSR. SMON is the structured mesh overlay network that supports efficient service lookups in P2P environments. We propose a new algorithm of creating and maintaining the overlay network on MANET to limit the number of exchanging overlay messages. From our simulation results, SMON can reduce the number of overlay control overhead significantly, while it still maintains a high query success ratio with low query delay when compared to CrossROAD.

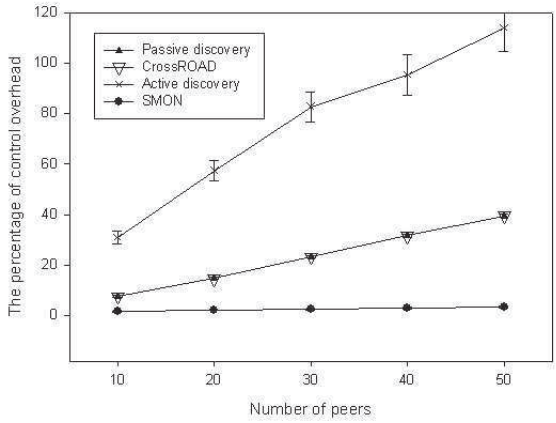
The differences between CrossROAD and SMON can be summarized as follows. First, in terms of the number of overlay broadcast messages, a CrossROAD node periodically broadcasts PublishService messages. With N services, it will advertise N times; thus, a total number of PublishService messages increases according to the number of services. On the other hand, the total number of overlay messages in SMON is lower because the only primary (the smallest ID) is allowed to advertise the list of overlay member messages.

The next difference between SMON and CrossROAD is the compatibility with OLSR. We use extended headers defined in OLSR standard to support the SMON overlay messages. The extended headers allow nodes that do not recognize the new messages to operate with SMON nodes seamlessly.

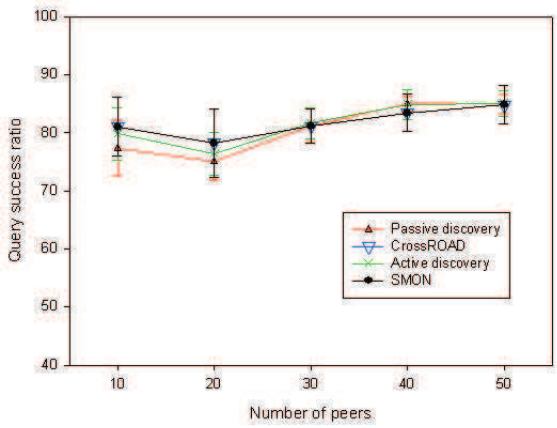
We evaluate the performance of SMON by NS2 in post-disaster scenarios: the heterogeneous MANET and MANET connected to the infrastructure (MANET to HQ communication). We find that SMON provides the best lookup times as compared with passive and active discovery methods, while the SMON control overhead is the lowest.

In the worst case scenarios where all nodes freely move at different speeds in the disaster area, SMON provides the lowest lookup time and control overhead when compared to passive and active discovery methods.

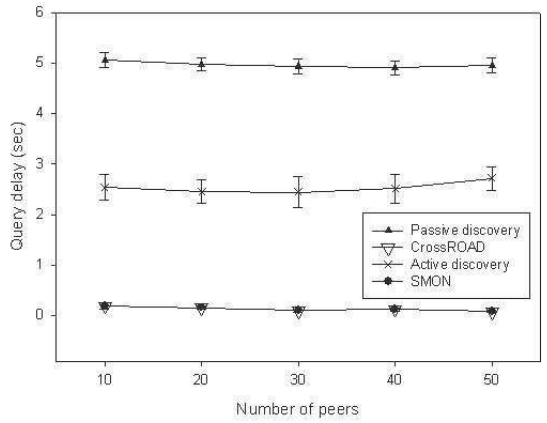
In the next chapter, we will integrate SMON with SIP in order to support SIP functionalities based on P2P architecture.



(a) The percentage of control overhead

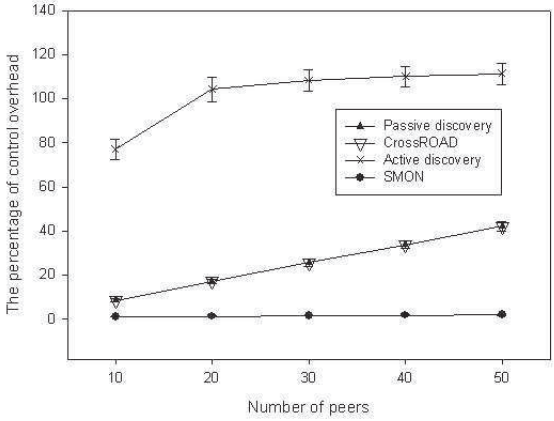


(b) Query success ratio

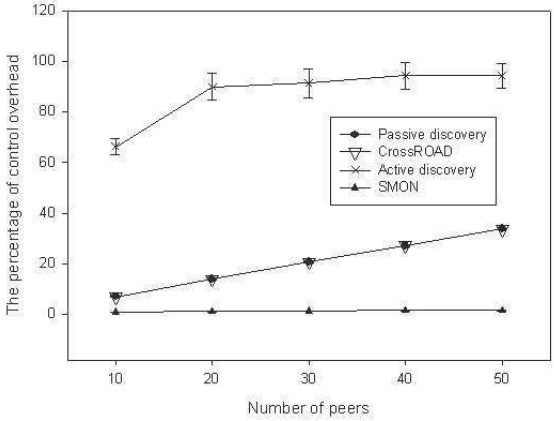


(c) Average query delay

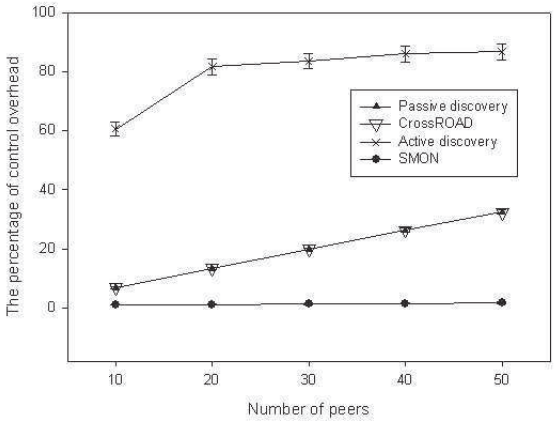
Figure 3.9: Control overhead, query success ration, and average query delay in gateway scenario with 95% confidence level



(a) Static

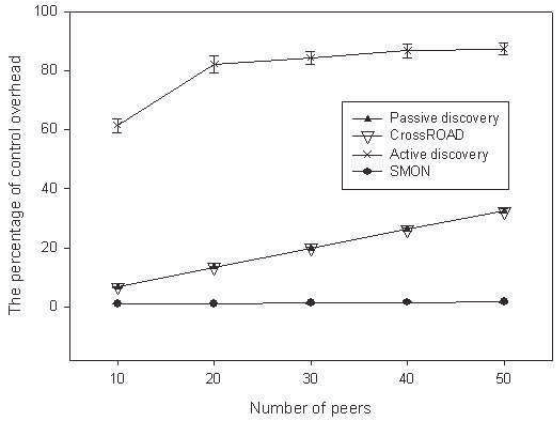


(b) Maximum speed at 2 m/s

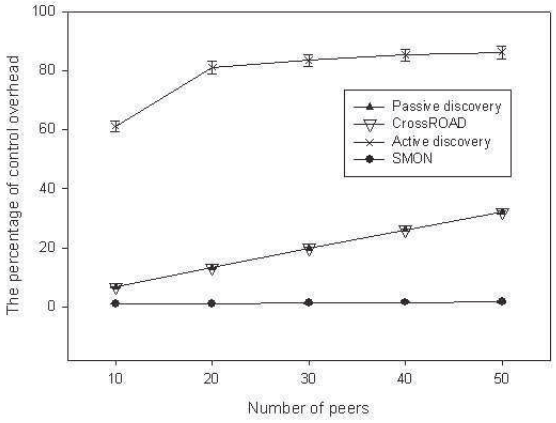


(c) Maximum speed at 10 m/s

Figure 3.10: The percentage of control overhead of all protocols by changing the maximum speed of mobile nodes with 95% confidence level

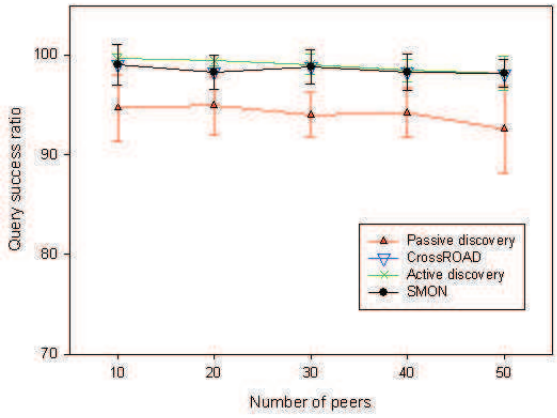


(d) Maximum speed at 20 m/s

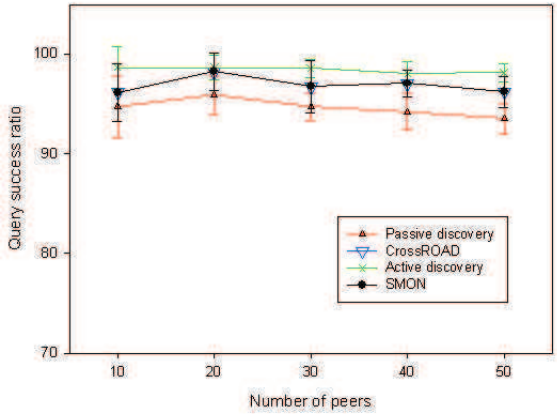


(e) Maximum speed at 30 m/s

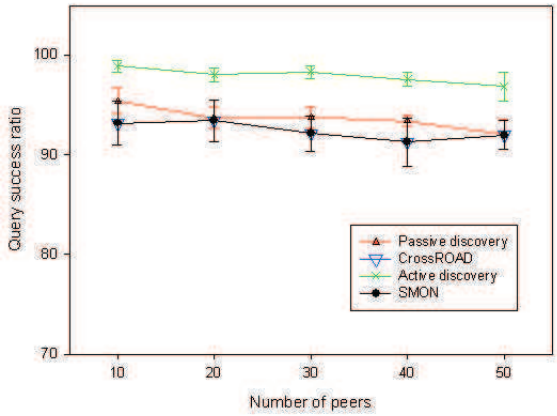
Figure 3.10: The percentage of control overhead of all protocols by changing the maximum speed of mobile nodes with 95% confidence level (cont.)



(a) Static

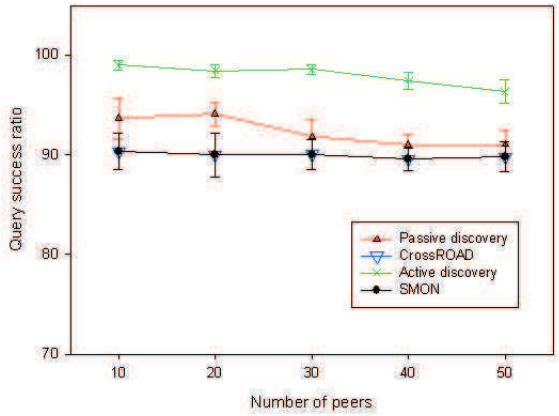


(b) Maximum speed at 2 m/s

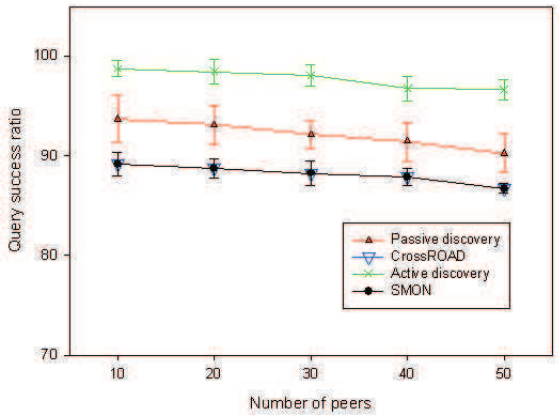


(c) Maximum speed at 10 m/s

Figure 3.11: Query success ratio of all protocols by changing the maximum speed of mobile nodes with 95% confidence level

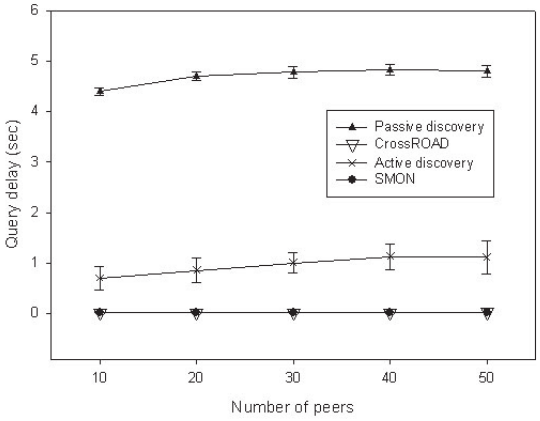


(d) Maximum speed at 20 m/s

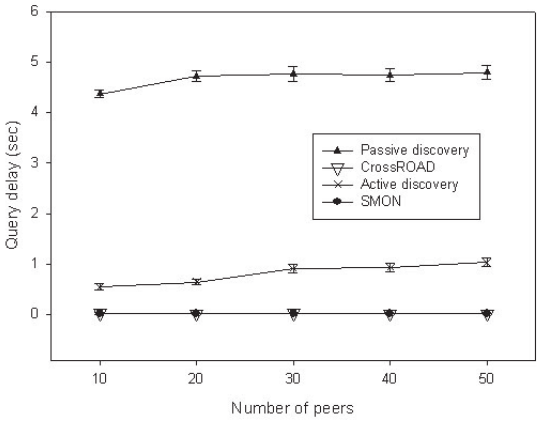


(e) Maximum speed at 30 m/s

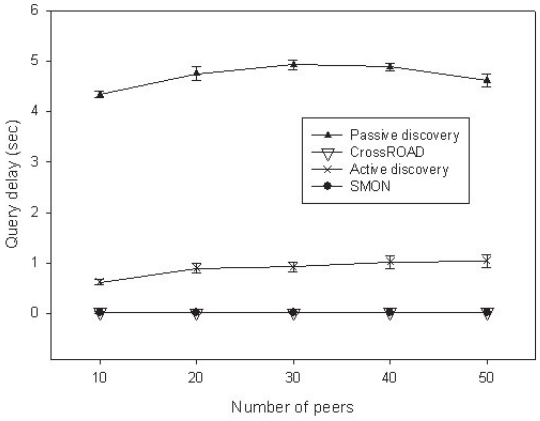
Figure 3.11: Query success ratio of all protocols by changing the maximum speed of mobile nodes with 95% confidence level (cont.)



(a) Static

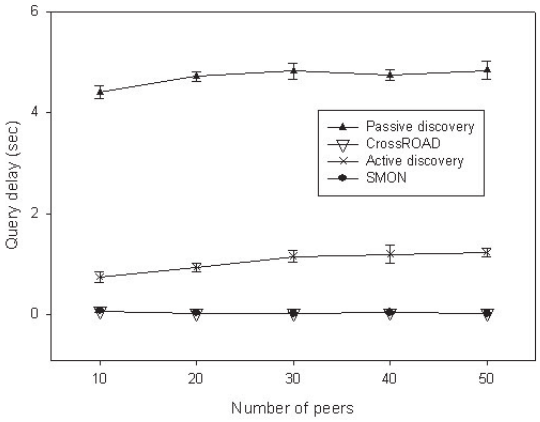


(b) Maximum speed at 2 m/s

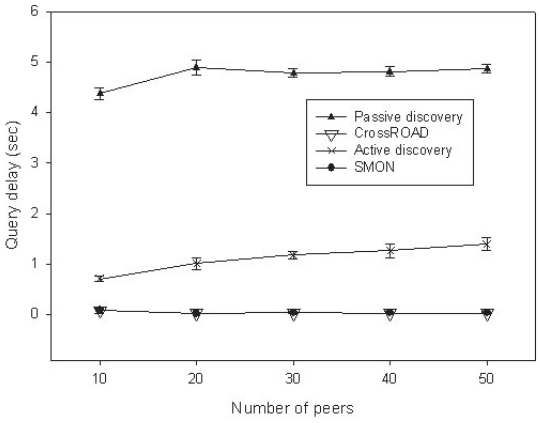


(c) Maximum speed at 10 m/s

Figure 3.12: Average query delay of all protocols by changing the maximum speed of mobile nodes with 95% confidence level



(d) Maximum speed at 20 m/s



(e) Maximum speed at 30 m/s

Figure 3.12: Average query delay of all protocols by changing the maximum speed of mobile nodes with 95% confidence level (cont.)

Chapter 4

SIPMON: P2P SIP on SMON

In this chapter, we focus on using the P2P overlay network to handle SIP user discovery on MANET as opposed to a traditional client and server SIP model. We propose the deployment of SIP over SMON creating an overlay network of SIP servers as peers, called P2P SIP. SMON is responsible for creating and maintaining the overlay network on MANET, while P2P SIP provides two main important SIP operations, which are registration and location discovery. Furthermore, the P2P SIP is an interaction point, where an existing SIP based application can seamlessly operate on SMON. In addition, P2P SIP on SMON handles terminal mobility, which will be presented in more details in the next chapter.

Section 4.1 presents P2P SIP architecture or SIPMON and SMON API, a service access point for P2P SIP. We explain SIP user registration, call invitation, and binding update issues in the P2P SIP overlay network in section 4.2, 4.3, and 4.4 respectively. In section 4.5, an evaluation of P2P SIP on SMON is presented.

4.1 P2P SIP architecture

P2P SIP on SMON is shown in Fig. 4.1. Each P2P SIP node is composed of OLSR, SMON, and P2P SIP. SMON uses OLSR to provide real-time routing information used for updating its overlay topology. When the physical topology is changed, OLSR informs SMON using cross-layer interaction, e.g. OLSR plug-in (Tϕnnesen et al., 2004). On top of SMON, P2P SIP is implemented to handle all SIP requests and responses exchanged among the P2P SIP overlay network. P2P SIP can access DHT table via SMON API. This DHT information is used for P2P SIP registration and user discovery, which are discussed later in the following sections. In a centralized SIP, an outbound proxy parameter of SIP phone is configured to a centralized SIP registrar/proxy. In P2P SIP, it is required that

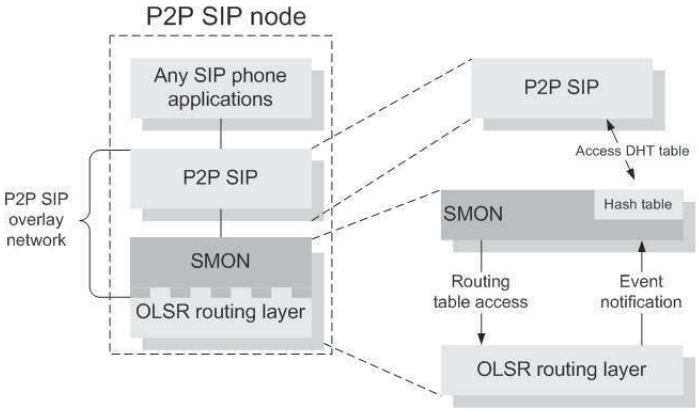


Figure 4.1: P2P SIP overlay network architecture on a P2P SIP node

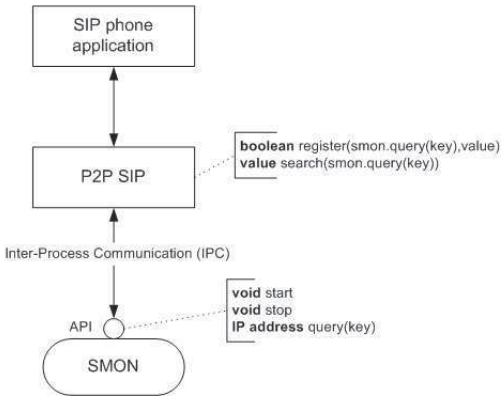


Figure 4.2: SMON API

the existing SIP phone has the outbound parameter pointed to it local address so that all SIP requests are directly sent to its P2P SIP.

SMON provides an API for P2P SIP via Inter-Process Communication (IPC), e.g. TCP connection at the localhost as shown in Fig. 4.2. After establishing communication with SMON, P2P SIP must call $SMON.start()$ to make SMON join the overlay network by executing the procedure JoiningOvelayNetwork in Algorithm 1. After finishing joining the overlay network, P2P SIP can use $SMON.query(key)$ to find a peer's IP address to perform P2P registration or do a search for items, which will be explained in section 4.2 and 4.3. SMON is responsible for maintaining node IDs of peers on the overlay network, while P2P SIP stores and maintains SIP objects, which comprises of SIP URIs and IP addresses, based on their object IDs. When $SMON.query(key)$ is called, SMON looks up its DHT table to find a peer whose node ID is the closet to the key, then SMON returns its corresponding IP address to the P2P SIP.

4.2 SIP user registration operation on SIPMON

On each SMON node, an outbound proxy parameter of a SIP phone is configured by using its IP address, the local address, so that a local P2P SIP can directly intercept SIP messages. When the P2P SIP receives a SIP REGISTER message from its SIP phone application, it obtains an object ID from the hash value of user's SIP URI. The hash function applied to user's SIP URI is denoted as *P2P_SIP(SIP URI)*. Then, it uses SMON to find the address of a peer whose node ID is the closest to this *P2P_SIP(SIP URI)*. The P2P SIP forwards the SIP REGISTER message to that peer as its registrar as shown in Algorithm 2.

The following example explains how a SIP user registers itself to the P2P SIP on SMON. John is a SIP user at *SMON(A)* with SIP URI "John@abc.com". John's SIP phone sends SIP REGISTER to its P2P SIP on *SMON(A)*. Next, P2P SIP of *SMON(A)* determines the object ID from *P2P_SIP("John@abc.com")*, denoted as Obj_{John}. Assume that Node ID is SHA1, and, for simplicity, only the first four hex-digits are displayed as shown in Fig. 4.3. The P2P SIP of *SMON(A)* calls *SMON.query(*Obj_{John}*)* to find an address of the

Algorithm 2 P2PSIP registration

```
1: procedure P2PSIPREGISTER(SIP URI)
2:     loop
3:         object_ID ← HASH(SIP URI)
4:         node_nearestID ← LOOKUP(object_ID)
5:         SEND(SIP REGISTER(SIP URI, node_nearestID))
6:         if WAITSIP200OK then                    ▷ If timeout occurs, loop
7:             return                              ▷ Return if registration is successful
8:         end if
9:     end loop
10: end procedure
```

peer whose ID is the closest to Obj_{John}, which is the address of *SMON(F)*. Therefore, the P2P SIP of *SMON(A)* forwards the REGISTER request to P2P SIP of *SMON(F)*. After receiving the REGISTER request, P2P SIP at *SMON(F)* adds the binding of the John's SIP URI and IP address A to its location database and replies SIP 200OK to P2P SIP of *SMON(A)*. Once this registration is complete, the binding of John and node A is said to be registered at P2P SIP of *SMON(F)*.

4.3 Call setup operation on SIPMON

A SIP user gives a call by sending a SIP INVITE request. However, the SIP user must know the physical address of a target user before making a call. In centralized SIP, the SIP user forwards an INVITE request to pre-configured SIP registrar and proxy that handle the call for the user. In P2P SIP, the INVITE request is processed according to DHT searching. The following steps explain the call setup process using SIPMON as shown in Algorithm 3. When the P2P SIP receives a SIP INVITE request from the SIP phone application, it extracts a target user's SIP URI from the request and uses the hash function *P2P_SIP()* to acquire the object ID. Next, it uses *SMON.query*(Obj_{John}) to find the address of the peer whose node ID is the closest to this object ID. Suppose that SMON returns address of *SMON(X)*, which has the closest ID to the object ID. Then P2P SIP forwards the SIP INVITE message to *SMON(X)*. After receiving the INVITE request, P2P SIP of *SMON(X)* opens its binding database to locate address of the target user indicated in the INVITE request, and uses this address to forward the INVITE request to the target user as shown in Algorithm 4. In other words, *SMON(X)* acts as the SIP registrar and proxy for both caller and callee.

A call setup example (Tom gives a call to John) is shown in Fig. 4.3. Notice that John has already registered to *SMON(F)* according to the previous registration example. To

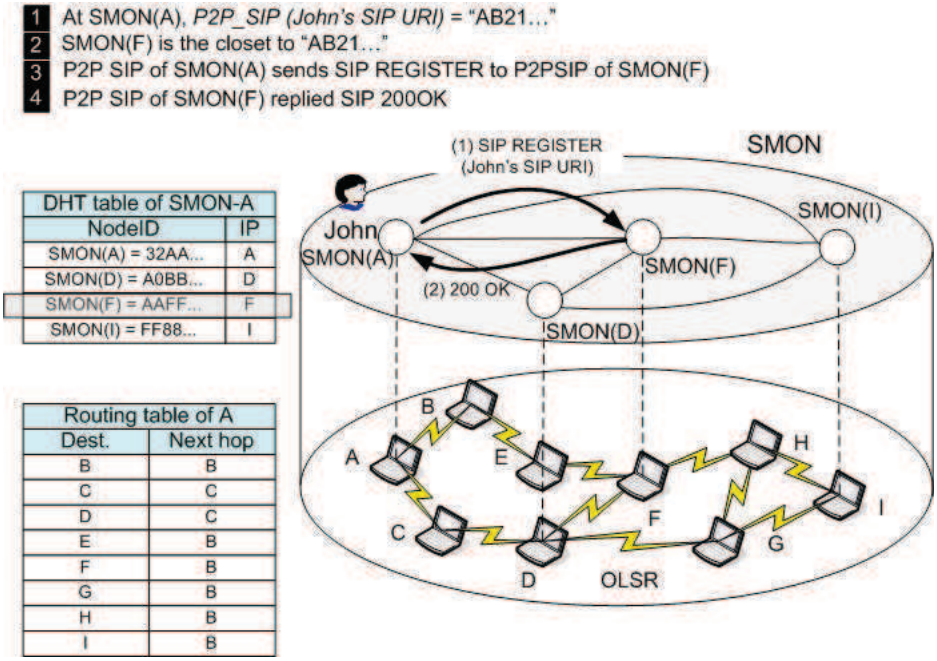


Figure 4.3: SIP user registration example in SIPMON

Algorithm 3 P2P SIP invite sent from a caller

```
1: procedure P2PSIPINVITE(SIP URI)
2:     loop
3:         object_ID ← P2P_SIP(SIP URI)
4:         node_nearestID ← SMON.QUERY(object_ID)
5:         SEND(SIP INVITE(SIP URI, node_nearestID))
6:         if WAITSIPRESPONSE not ERROR then ▷ Error received, loop until timeout
7:             return                          ▷ Return if invite is successful
8:         end if
9:     end loop
10: end procedure
```

Algorithm 4 P2P SIP invite received at peer-X

```
1: procedure P2PSIPINVITERECEIVEDATPEER-X(INVITE request)
2:     target_SIP_URI ← EXTRACT(INVITE request)
3:     address ← LOOKUP(target_SIP_URI)                ▷ Lookup in binding database
4:     if found target's address then
5:         FORWARDINVITE(SIP INVITE(INVITE request, target_SIP_URI))
6:     else
7:         REPLY(SIP 404 Not Found)
8:     end if
9: end procedure
```

make a call, Tom's P2P SIP of *SMON(I)* uses SMON to find an address of a peer in which John may register. SMON returns the address of *SMON(F)* because its node ID is the closest to value of *P2P_SIP(John's SIP URI)*. Next Tom's P2P SIP sends a SIP INVITE to *SMON(F)*. After receiving the SIP INVITE request, P2P SIP of *SMON(F)* looks for John's address in its database, and it forwards a SIP INVITE to John at node A. After that, John's SIP phone replies SIP 180 RINGING back to Tom via P2P SIP of *SMON(F)*. Finally, Tom's SIP phone can establish the call with John's SIP phone. The P2P SIP only handles SIP requests and responses flowing between the caller and callee in the process of SIP user location discovery. Once the caller finishes discovering the callee's IP address, the actual call session is initiated between the caller and callee without involving the P2P SIP.

4.4 SIP registration update operation on SIPMON

A contact header in SIP REGISTER contains expiration parameter to indicate the valid period of time of the SIP URI binding. The default value is set to 3600 s as described in SIP specification [RFC 3261]. Once a SIP user receives a SIP 200 OK response from a registrar, it starts a timer according to the expiration parameter. The SIP user must send a new SIP REGISTER to the registrar before the timer is expired in order to keep its binding valid. However, MANET nodes are prone to disconnection from the network. Consequently, the SIP user has to send a new REGISTER not only when the timer is expired, but also when the network changes. The changes result from network partitions and mobility including node joining and leaving SMON, which bring about a change in the DHT table as well. The modification in the DHT table makes every SMON node perform SIP registration updates according to section 4.2.

Let *SMON(X)* be a peer in SMON that has *SMON(Y)'s binding information*. When *SMON(X)* is unavailable, the P2P SIP of *SMON(Y)* is instantly informed by SMON. It must retransmit SIP REGISTER to another SMON whose node ID is the next closest one.

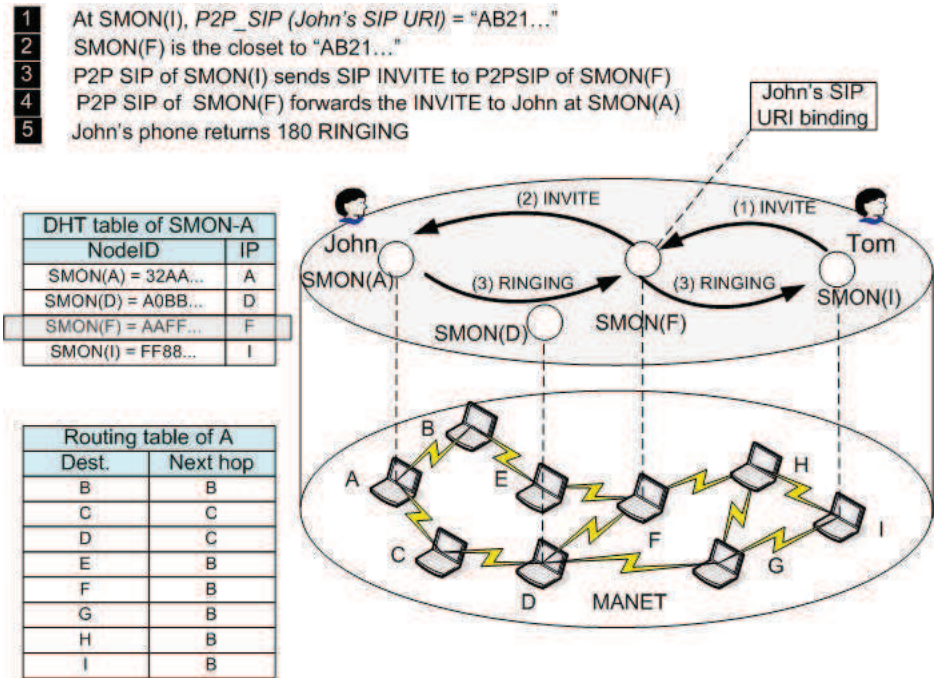


Figure 4.4: A call setup example in SIPMON

If the DHT table is changed due to a new node joining or a node leaving, and *SMON(X)* is not the closest to *P2P_SIP(Y's SIP URL)*, the P2P SIP of *SMON(Y)* must proceed SIP registration again. The following example shows a retransmitting SIP REGISTER when *SMON(F)* disconnects from the network as shown in Fig. 4.5. At *SMON(A)*, SMON detects the topology change. It removes *SMON(F)* from the DHT table and informs this change to its P2P SIP. After receiving the notification from SMON, the P2P SIP of *SMON(A)* performs the registration process again for John, resulting in forwarding a SIP REGISTER request to the next closest node ID, *SMON(D)*. After *SMON(D)* receives the SIP REGISTER request from John at *SMON(A)*, it replies to John with SIP 200 OK.

Next, binding inconsistency problems can occur due to node or user movement. Two cases can cause binding inconsistency. In the first case, a SIP user logs on to a different machine. Both *SMON(F)* and *SMON(D)* have John's binding based on the above example, and *SMON(F)* is partitioned from other peers. If John logs off from *SMON(A)* and logs on to *SMON(B)*, John will send a new SIP REGISTER to *SMON(D)* to update the binding. However, *SMON(F)* still has John's previous binding, which by now is invalid. In the second case, the IP address of John's machine is changed due to SIP terminal mobility. If the IP address of *SMON(A)*, which John logs in to, is changed, John will have to send the new binding update to *SMON(D)*. Similar to the former case, *SMON(F)* has John's wrong binding. Binding inconsistency can be solved by making *SMON(F)* remove the John's previous binding information. When *SMON(F)* is partitioned from *SMON(A)*, the P2P SIP of *SMON(F)* is informed by SMON. It must remove all bindings belonging to peers that cannot be reachable. Therefore, *SMON(F)* removes John's binding from its database because *SMON(A)* is unreachable. Now, only *SMON(D)* has John's valid binding.

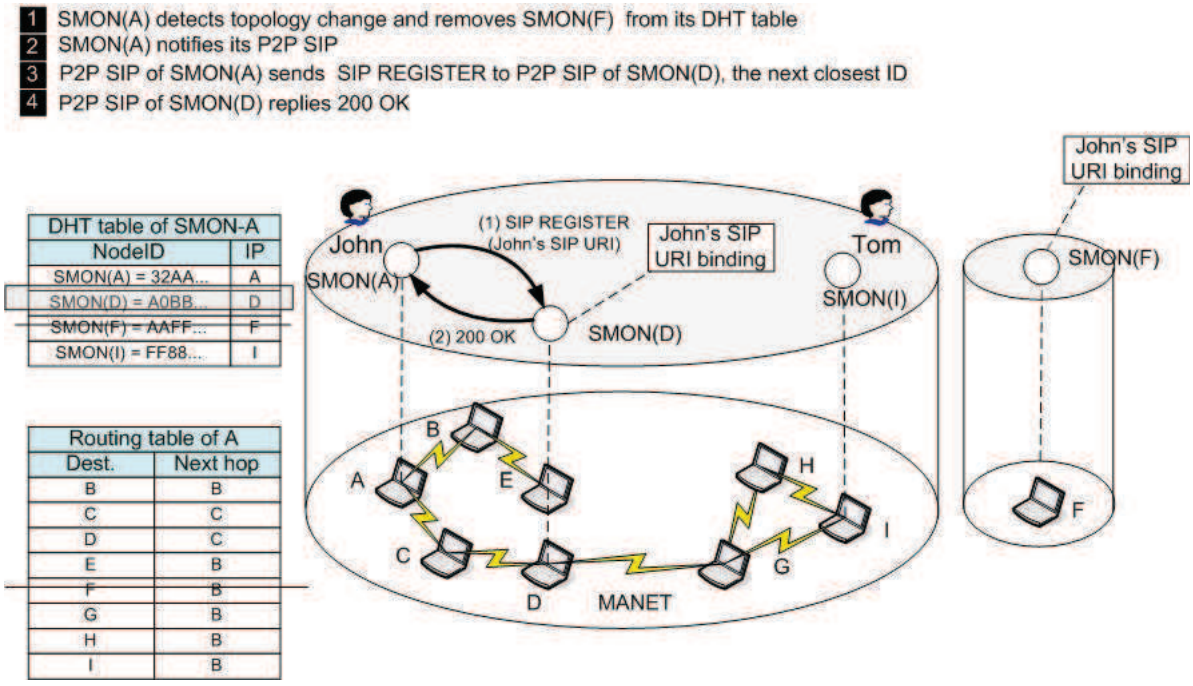


Figure 4.5: Re-registration when node F is partitioned

4.5 Evaluation of P2P SIP on SMON

In this section, we describe simulation and testbed results of SIPMON. We compare SIPMON with two existing approaches. SIP over OLSR (L. Li & Lamont, 2004), as reviewed in section 2.3.3, is based on the use of a cross-layer design to support proxy-less system or Client Query Server Advertising (CQSA) scheme. CQSA is based on the pull method for user discovery. When a SIP user agent wants to make a call, it broadcasts a Service Location Extension (SLE), a new message type in OLSR defined by the CQSA scheme, to query a target SIP user agent's IP address. The target SIP endpoint responds to the query with its IP address. Another scheme we will compare SIPMON with is dSIP (Leggio et al., 2005), which provides SIP user registration and discovery on MANET. Distributed registration is done by periodically broadcasting a SIP REGISTER request to the whole network. We compare our SIPMON performance with the CQSA scheme and dSIP in term of post dialing delay, session setup success ratio, and control overhead.

4.5.1 Simulation scenario description

We evaluate the performance of SIPMON in NS2 with UM-OLSR extension (Ros, 2007) for a post-disaster scenario. We assume a disaster area of $1000\,m \times 1000\,m$ in size. In this area, rescuers carry mobile devices that are capable of handling SIP-based VoIP communication. We consider the scenario where all rescuers freely and randomly move and have VoIP communication to communicate to each other. In this scenario, some of the rescuers may not participate in the SIPMON overlay network, but their mobile devices act as OLSR routers, which help extend the MANET size to cover the disaster area.

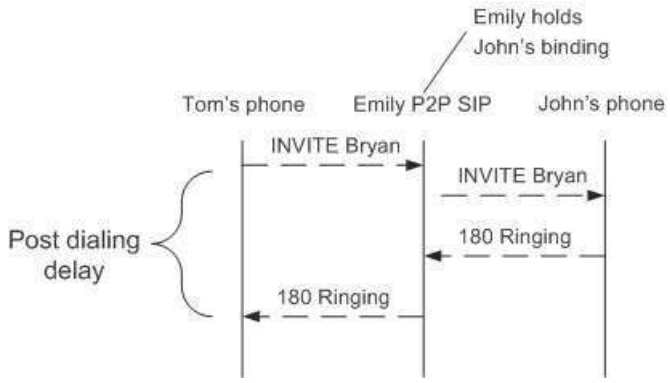


Figure 4.6: Post dialing delay in SIPMON

Therefore, we are interested in measuring the SIP overlay performance by changing the number of peers between 20 and 60. The total number of OLSR nodes is 60. All mobile nodes are equipped with IEEE 802.11b wireless interface with bandwidth of 11 Mbps and transmission range of 250 m. We assume that rescuers randomly move at walking speed. Then, the random waypoint mobility is used at the maximum speed of 2 m/s and pause time of 60 s.

Next, we evaluate the performance of SMON in the worst case scenarios where all rescuers freely move at different speeds within the affected area. We create a second kind of scenario by changing the maximum speed of mobile nodes into 2 m/s(fast walking speed), 10 m/s (slow speed vehicles), 15 m/s (fast speed vehicles), and 20 m/s (very fast speed vehicles).

Each data point represents a result averaged over ten different movement scenarios. Each peer has a unique SIP URI. There are 30 CBR flows, each of which randomly starts from an interval of between 300-400 s as background data traffic. The packet size and rate are 512 bytes and 3 packets per second respectively. All simulations run for a duration of 900 s.

4.5.2 Performance metrics

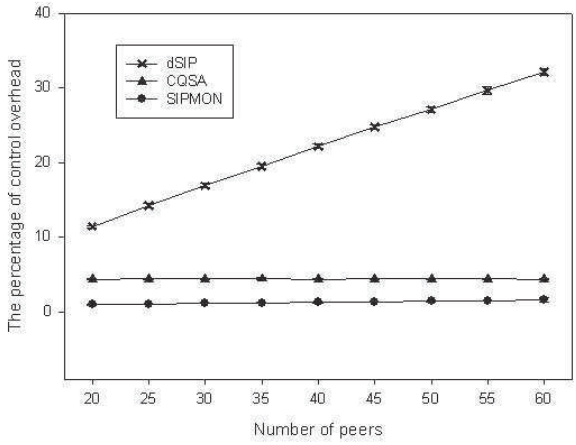
We use the following metrics to evaluate the performance of SIPMON. The first metric is the percentage of the increment of control overhead by dSIP, CQSA, or SIPMON messages on OLSR routing messages. This metric shows how many percentage of the control overhead generated by dSIP, CQSA, or SIPMON is introduced into the network in addition to OLSR routing overhead. The call-setup success ratio is another metric that describes the percentage of session setup success. The last metric is a post dialing delay, which is the difference between the times that a caller uses in sending INVITE to callee and that of receiving back SIP 180 RINGING. Fig. 4.6 illustrates the post dialing delay in SIPMON.

4.5.3 Simulation results

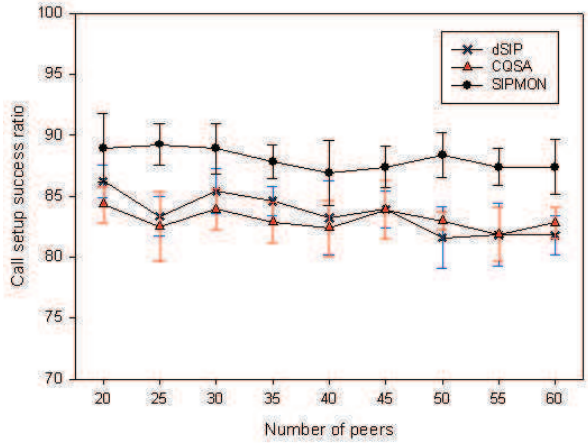
In Fig. 4.7(a), as the number of peers increases, the percentage of control overhead of dSIP rapidly increases. Increasing the number of peers has an effect on the overhead because each peer periodically advertises a SIP REGISTER request. The percentage of dSIP control overhead is 32.15% on average, where a number of peers is 60. In the CQSA scheme, the percentage of control overhead increases according to the number of calls. The number of INVITE request is 240; hence, the percentage of control overhead is approximately 4% regardless of the number of peers. In contrast to dSIP and the CQSA scheme, the number of SIP users and INVITE requests do not increase in SIPMON. The percentage of SIPMON control overhead is 1.64% on average, where the number of peers is 60. We measure call-setup success by calculating the number of successful SIP 180 RINGING replies over the number of the INVITE requests issued. According to the SIP specification, a caller sends duplicated INVITE requests for a call to a callee via UDP packets to gain a chance of a successful packet received at the destination under unreliable transportation. The destination can detect the duplicated INVITE requests by using Call-ID, unique identification for each call. In our simulation, a caller sends only one INVITE request for each call over UDP. Hence, the success or failure of call setup is collected by using only one INVITE request for each call. The call-setup success ratios of all protocols are quite similar, where SIPMON slightly outperforms those of other mechanisms as shown in Fig. 4.7(b). Notice that the call-setup success ratios decrease when the number of peers increases for all protocols.

A peer in dSIP periodically advertises its SIP URI every 10 s. Then, a caller must wait around the average value of this interval in order to get a target's IP address before constructing and sending an INVITE request. Hence, the post dialing delay from the simulations is about 5 s as shown in Fig. 4.7(c). In the CQSA scheme, a new type of OLSR message is used for a query and a reply. Consequently, the delay depends on the jitter value. As a basic OLSR implementation requirement, synchronization of control messages should be avoided. As a consequence, OLSR control messages should be emitted such that they avoid synchronization. To avoid such synchronization, a node should add an amount of jitter to the interval at which messages are generated and forwarded (Clausen & Jacquet, 2003). Whenever a node has to forward an OLSR message, it should keep the message for at least the jitter time, which is a random value between 0 to HELLO $interval/4$. On the contrary, the post dialing delay of SIPMON is the lowest because a SIP INVITE request can be immediately sent without knowing target's IP address.

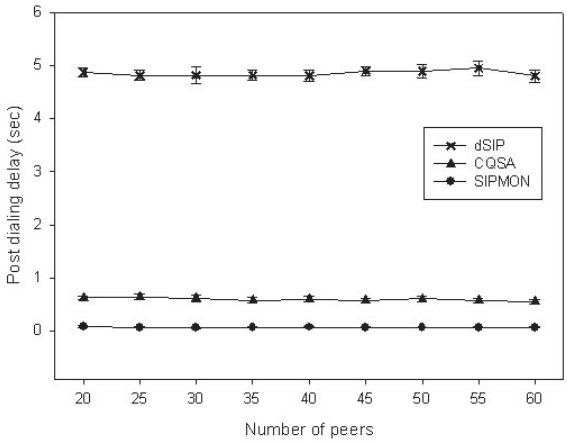
We are also interested in how the efficient SIPMON performs at various speeds of movement. Thus, we measure the performance of our scheme by varying the node velocities between 2 m/s and 20 m/s. In Fig. 4.8, the x-axis represents the node velocities. The explanation for control overhead and post dialing delay in Fig. 4.8(a) and 4.8(c) are similar to the previous simulation results because the broadcast interval time does not change regardless of the changes in speed. However, high node velocities can increase packet losses. In Fig. 4.8(b), the call-setup success ratio gradually reduces when the maximum speed is increased. The reason is that an OLSR node takes a certain amount of time to detect neighbor link failures and to update the routing table. During this transient period, data packets that are forwarded along the failed path will be dropped. In the simulations, many SIP requests are dropped during this time. The call-setup success ratios for all protocols are approximately 60% at the speed of 20 m/s due to high packet losses;



(a) The percentage of control overhead



(b) Call-setup success ratio



(c) Average SIP post dialing delay

Figure 4.7: Call setup results with node's maximum speed of 2 m/s with 95% confidence level

however, call-setup success ratios are still acceptable.

In Fig. 4.7, the broadcast interval of all protocols is 10 s. The different intervals give different outcomes. For example, small intervals can cause more control overheads. However, SIP post dialing delay will be smaller. Then, we perform more simulation again with broadcast interval of 5 s or $t = 5$, for all systems. Fig. 4.9 shows that control overhead of dSIP and SIPMON with a broadcast interval of 5 s increase about twice that of broadcast interval of 10 s since these two protocols use periodic broadcast mechanism. In the CQSA scheme, the control overhead percentages of $t = 5$ and $t = 10$ are similar because the number of call setup has not changed.

Call-setup success ratio of dSIP with $t = 5$ is increased 7.32% from the success ratio of dSIP with $t = 10$. Frequently broadcasting SIP REGISTER requests results in the high chance of successful call setup. In the CQSA scheme and SIPMON, the broadcast interval has an insignificant effect on the number of call-setup session ratio.

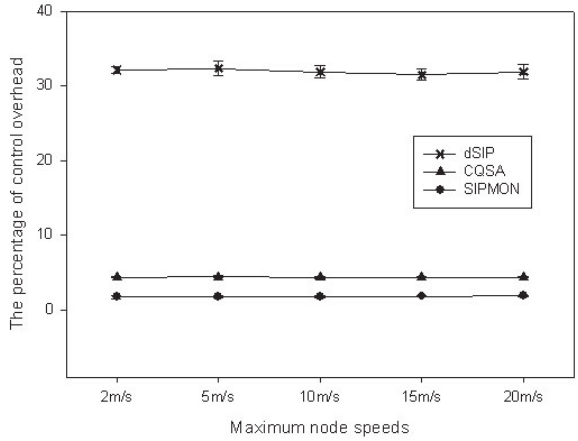
When $t = 5$ is used, post dialing delay of dSIP is reduced to 2.5 s as shown in Fig. 4.10. According to the simulation results, the post dialing delay of dSIP is around $t/2$, whereas the different broadcast interval values have no effect on the post dialing delay of the CQSA scheme and SIPMON.

4.5.4 Testbed implementation and results for stationary scenario

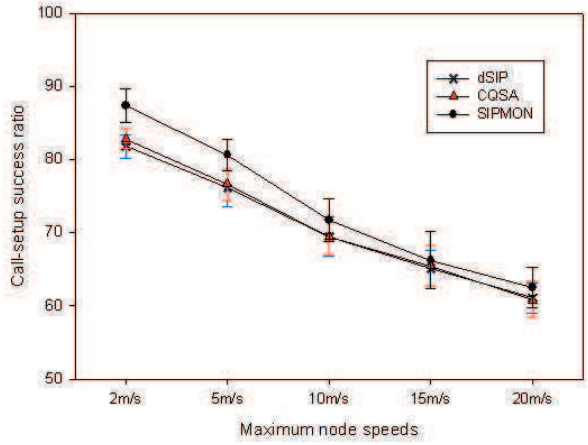
We develop a testbed to verify the correctness of the simulation results of SMON operations. SIPMON was developed and tested where a testbed was setup with twelve computers running Ubuntu kernel version 2.6.24. We used different kinds of machines for the testbed as shown in Fig. 4.11. Five of them were UltraClient equipped with IEEE 802.11g. There was one PC with a wireless card installed. The others were six Eee PC 901 laptops. All of the computers were connected in the Ad Hoc mode. We created a SMON plug-in by using the OLSR implementation (Tϕnnesen et al., 2004). We used MjSip (mjsip.org, 2006) to develop the java-based P2P SIP server. Twinkle version 1.1 (Boer, 2008), SIP softphone for Linux, was used for the testbed.

The testbed was deployed at the Internet Education and Research Laboratory (intERLab), Asian Institute of Technology, as shown in Fig. 4.12. We did not use any packet filtering software to create a multi-hop network. Even though all nodes were static, the route from node-11 to node-12 kept changing throughout the experiment because some activities occurred inside and outside the building. For example, when a car or a person passed by, it was possible that this movement blocked the wireless signal, causing route changes. The lines in the diagram represent possible routes between node-11 and node-12. Except for node-11 and node-12, all other nodes were fully connected in one hop. We collected data by using tcpdump (MG, 2009) For each run, all 40 SIP calls were made from node-11 to node-12 by varying the number of peers between 3 and 12. Results were averaged over a set of three runs, each of which was 12 minutes long.

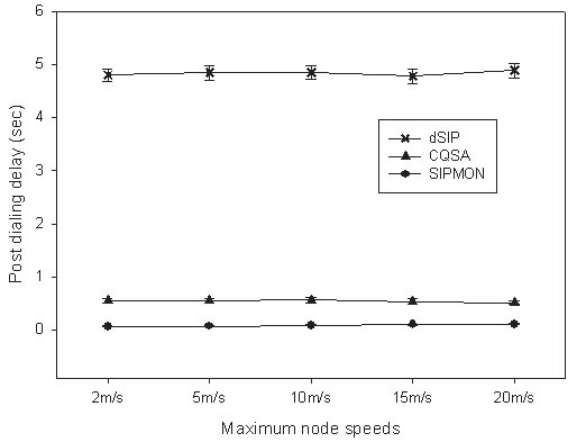
In the testbed, a peer in dSIP periodically advertises its SIP URI and its IP address every 5 s. In SIPMON, the primary peer principally advertises a LIST OF ALL MEMBERS



(a) The percentage of control overhead



(b) Call-setup success ratio



(c) Average SIP post dialing delay

Figure 4.8: Call setup results with node's maximum speed varied between 2 m/s and 20 m/s at peer of 60 with 95% confidence level

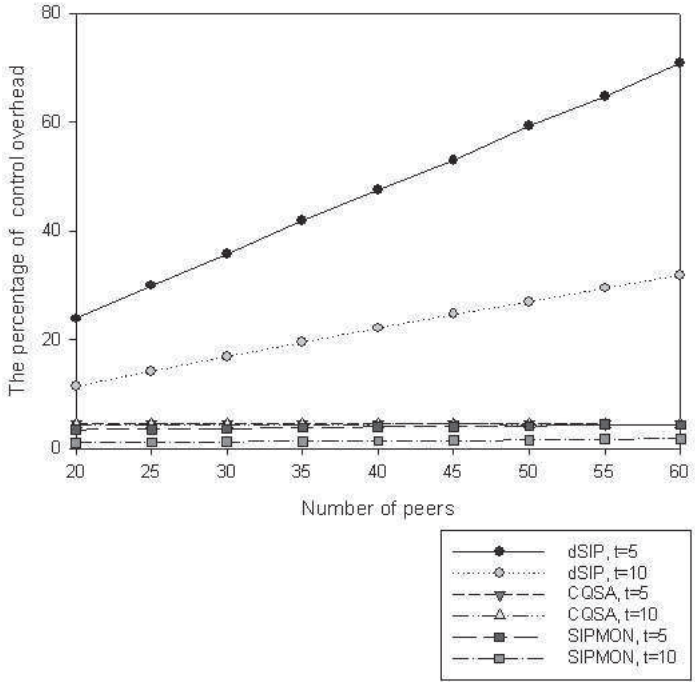


Figure 4.9: Control overhead comparison of all approaches by using different broadcast interval of 5 s and 10 s

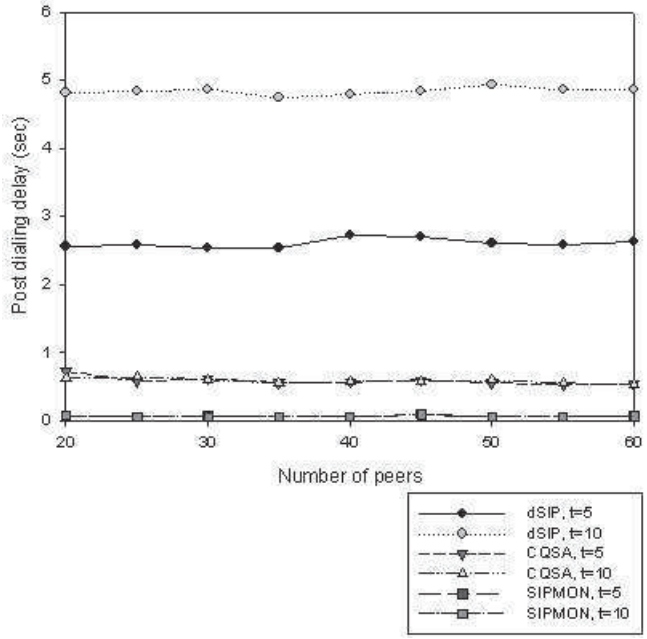


Figure 4.10: Average post dialing delay comparison of all approaches by using different broadcast interval of 5 s and 10 s

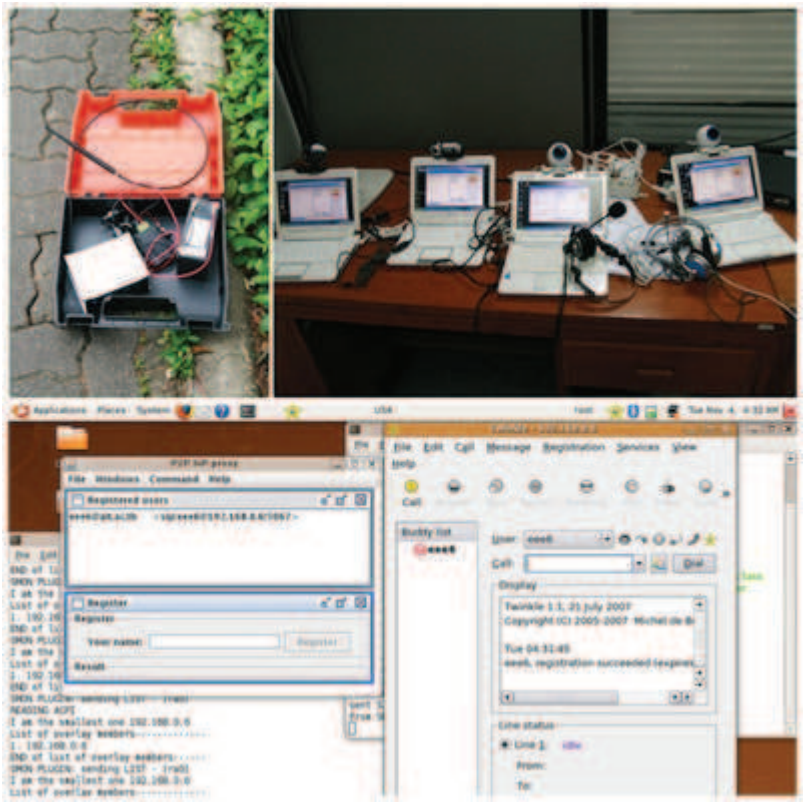


Figure 4.11: Testbed computers and screenshot

Figure 4.12: Testbed of twelve stationary nodes at intERLab

message in 5 s too. In Fig. 4.13(b), the x-axis refers to the number of peers between 3 and 12, and the y-axis represents the percentage of control overhead in addition to the OLSR control overhead. As the number of peers increases, the control overhead of dSIP chiefly goes up to 82% because all peers broadcast their SIP URIs. In CQSA scheme, a caller wishing to make a call must find the callee'a address by broadcasting a SLE message containing an INVITE request addressed to the callee's SIP URI. After receiving the INVITE request, the callee responds to the caller with its IP address and the caller can give a call to the callee using this IP address. Therefore, the CQSA scheme control overhead depends on the number of calls or broadcast SLE messages which add extra overhead to the normal OLSR routing messages. In the testbed results, the control overhead is 6% in addition to the OLSR control overhead, where the number of call is set at 40. We perform experiments to measure only the control overhead between the CQSA scheme and SIPMON by using 80 calls. The CQSA control overhead of 80 calls is 12.45%. If the number of calls increases, we claim that the number of control overhead will increase linearly for the CQSA scheme. On the other hand, the control overhead of SIPMON is 8.57% (40 calls) and 8.73% (80 calls). We claim that the SIPMON control overhead is steady regardless of the number of calls.

An observation from Fig. 4.14(b) is that the post dialing delay of SIPMON is the lowest since SIP requests can be sent without a need to wait for the target's IP address to be discovered. With the CQSA scheme, major delays come from jitter time, while dSIP spends most of the time waiting to discover the target's IP address. We do not show the call-setup ratios because no calls are dropped during the experiments for all the three protocols.

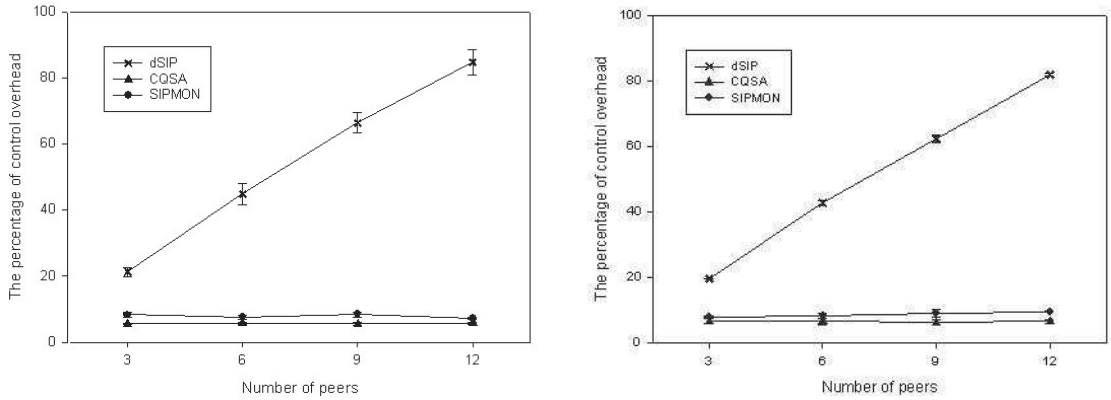
We perform simulations of twelve nodes with the same testbed parameters to compare simulation results with testbed results.

We observe the testbed results(Fig. 4.13(b) and Fig. 4.14(b)) and the simulation results (Fig. 4.13(a) and Fig. 4.14(a)) and discover that the chart patterns of both the testbed results and the simulation results are quite similar. This confirms the correctness of SIPMON simulation results.

4.5.5 Testbed implementation and results for moving scenario

The next testbed was made at Karon beach, Phuket, Thailand, at the 8th International Conference on ITS Telecommunications (ITST 2008). We used four Eee PC 901 laptops that together form SMON, running in the moving scenario. Two nodes were stationary along the beach, while the other two nodes were moving on the road as depicted in Fig 4.15. The dash lines were links before the nodes moved. Two cars moved the same direction and at a speed of 20 km/h,in the same direction but at different speeds, and in opposite directions and at the same speed. There were seven calls made from EEE-3 to EEE-4 during the movement. Results were averages over a set of three runs. Due to movement scenarios, we configured HELLO and TC intervals to 0.5 s and 3 s respectively to make OLSR nodes detect neighbors faster in the highly mobile environment.

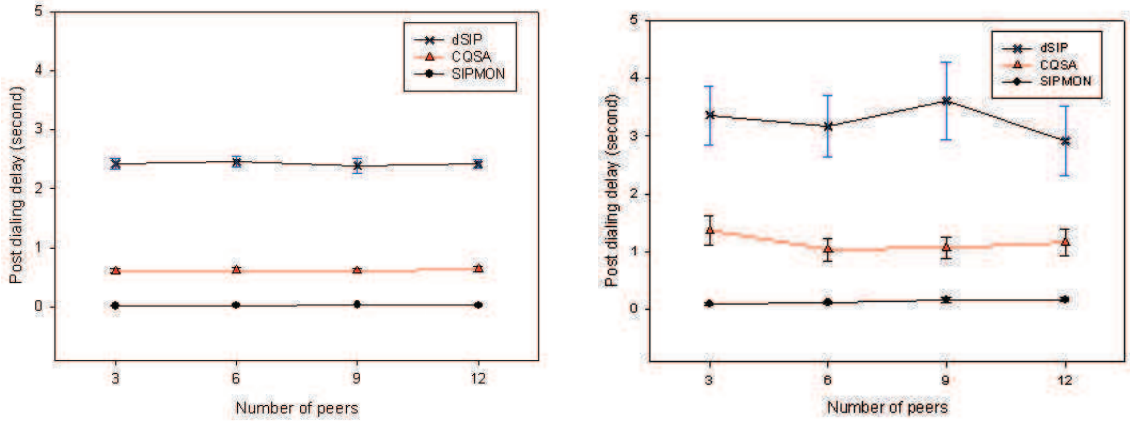
Fig. 4.16 shows the post dialing delay of SIPMON in the moving scenario in Phuket. The



(a) The percentage of control overhead (Simulation)

(b) The percentage of control overhead (Testbed)

Figure 4.13: Comparison of the percentage of control overhead between simulation results and testbed results



(a) Average SIP post dialing delay (Simulation)

(b) Average SIP post dialing delay (Testbed)

Figure 4.14: Comparison of average SIP post dialing delay between simulation results and testbed results

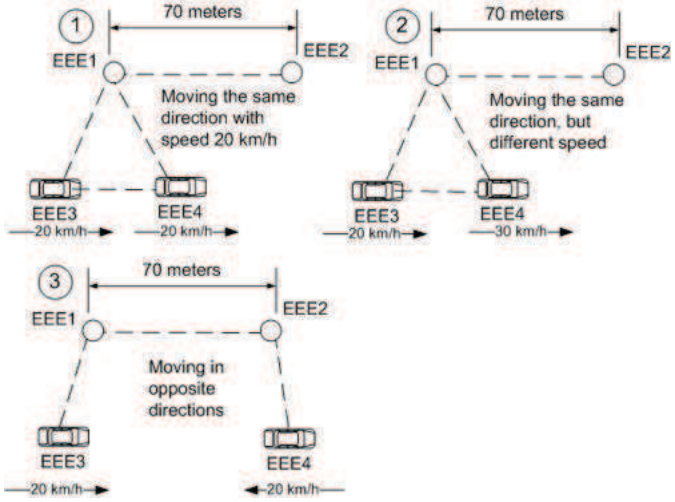


Figure 4.15: Testbed movement scenarios in Phuket, Thailand

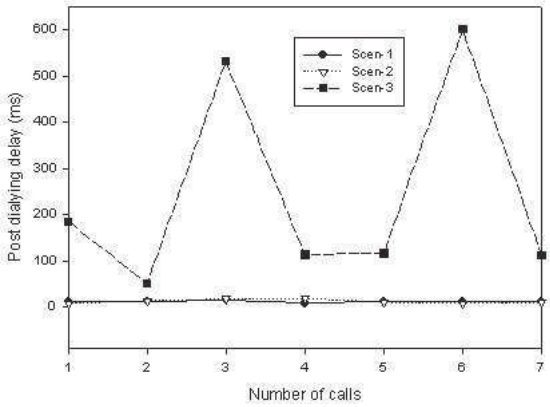


Figure 4.16: Call setup results of the moving testbed scenarios. Scenario 1, two nodes move in the same direction and speed. Scenario 2, two nodes move in the same direction, but at different speeds. Scenario 3, two nodes move in opposite directions. Phuket, Thailand

x-axis is the sequence of calls during the movement, and y-axis is the post dialing delay. In scenario 1, EEE-3 and EEE-4 were neighbors to each other throughout the movement because the two nodes moved in the same direction and at the same speed. EEE-3, as a 1-hop neighbor, could directly send an INVITE request to EEE-4. Hence, the post dialing delay was around 11 ms. In scenario 2, EEE-3 and EEE-4 moved the same direction at different speeds. Even though EEE-4 moved faster than EEE-3 by about 5-10 km/h, we observed that it was not fast enough to break the one-hop neighbor link between them. Therefore, the post dialing delay of SIPMON was almost the same as scenario 1. In the last scenario, EEE-3 moved in opposite direction to EEE-4. From our observation, there were two times that the post dialing delay was high due to route changes. It was confirmed by Fig. 4.16 that call numbers 3 and 6 were the points where the path between EEE-3 and EEE-4 changed. When the route changed, the post dialing delay was between 530 and 600 ms. The delay mainly came from the time to detect the neighbor, which was set at 500 ms (configured HELLO interval).

4.6 Discussions

We have presented SIPMON on OLSR. The overlay network is composed of P2P SIP and SMON. P2P SIP functions as a small traditional SIP registrar and proxy server, which typically accepts and processes SIP requests from an existing SIP phone application. To send a SIP request, if no target's address is found in its location service, P2P SIP will use SMON to find the target's IP address based on a DHT search. The design allows normal OLSR nodes to operate seamlessly with the P2P SIP nodes without modifying the OLSR standard.

Different MANET's routing protocols have distinctive performances. For example, proactive routing protocols have a lower delay in setting up a connection between two end nodes as compared to reactive routing protocols. However, the proactive routing protocols produce high overhead when routing information is exchanged. We use OLSR as the underlying routing protocol for SMON. For a fair comparison, we therefore compare SIPMON with other SIP on MANET approaches using OLSR: CQSA scheme (L. Li & Lamont, 2004) and dSIP (Leggio et al., 2005), and MANETSip (Fudickar et al., 2009).

CQSA scheme uses SLE messages to broadcast SIP requests within the OLSR network via MPRs. Both the CQSA scheme and SIPMON introduce low control overhead added to routing overhead. However, the average post dialing delay of SIPMON is five times lower than those of the CQSA scheme according to our testbed results (twelve static nodes). Next, we compare our approach with dSIP. Both simulation and testbed results show that SIPMON network reduces the number of control overhead significantly by 90% when compared to dSIP. The post dialing delay of dSIP is about half of advertised interval. In Fig. 4.10, dSIP with $t = 5s$ gives the post dialing delay of around 2.5 s and 5 s when $t = 10$. However, if t is small, the number of control overhead will increase exponentially. On the other hand, the post dialing delay of SIPMON is less than 100 ms on average according to both simulation and testbed results. Although simulation results confirm the SIPMON performance when the number of nodes is large, due to resource limitation, the testbed results are only valid for a small size of MANET. Next, we compare our post

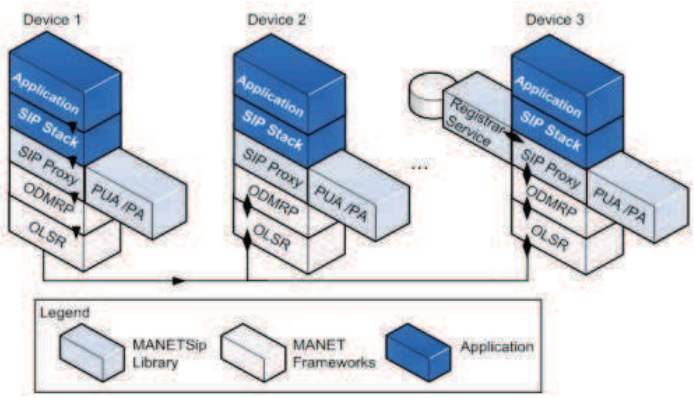


Figure 4.17: MANETSip testbed topology (source: MANETSip (Fudickar et al., 2009))

dialing delay with that of MANETSip (Fudickar et al., 2009). MANETSip uses a On-Demand Multicast Routing Protocol (ODMRP) (Lee et al., 2001) over OLSR to distribute a SIP request. Every node on a multicast network is a SIP registrar, referred as a SIP multicast network. A node periodically sends a SIP REGISTER request to all nodes on the multicast network in order to register itself on the SIP multicast network. In Fig. 4.17, the registration delay was given in their MANETSip testbed setup with a static linear network topology of three computers. To register a node at another node with a two-hop distance delay, it takes 164 ms on average on MANETSip. On of our testbeds as shown in Fig 4.15 based on a linear network topology of four computers (three-hop end-to-end nodes), we could obtain a delay of 14 ms which is significantly lower than the delay in MANETSip.

In the next chapter, we will extend SMON to work on the Internet and address the problem of terminal mobility, which is important for IP telephony.

Chapter 5

SMON over MANET and the Internet (SMON+)

In the previous chapter, we presented the concept of P2P SIP on an isolated MANET running OLSR protocol. In this chapter, we extend P2P SIP on SMON to cover a fixed IP network or the Internet with terminal mobility. P2P SIP supports terminal mobility for real-time multimedia communications for seamless communication between MANET and the Internet users. When MANET is connected to the Internet, a novel Overlay OLSR Network (OON) extends OLSR network to cover some of the Internet nodes thus on OON. On top of this OON, we extend SMON to SMON+ by stretching SMON on MANET to cover OON as well; P2P SIP is then applied on top of extended SMON+. The advantage of using OON is that we do not need to modify any P2P SIP and SMON internal operations.

Section 5.1 explains our use of case scenarios and design criteria for an emergency network. Section 5.2 shows how OON can be built on the Internet, followed by SIPMON+ in section 5.3. Section 5.4 describes how SIPMON+ can provide SIP terminal mobility in more details, while section 5.5 shows the evaluation of SIPMON+. We shows the scalability analysis of SIPMON+ in the last section of this chapter.

5.1 Scenario description and design criteria

In disaster-struck fields where traditional communication services such as fixed or mobile telephone and local internet access are completely inoperable, a fast-deploying multimedia communication system that a number of emergency rescue teams can rely on and collaborate with a distant command headquarter will prove very useful in saving the lives of victims (Kanchanasut et al., 2007). Our target scenario is where we have several separated disaster areas as shown in Fig. 5.1. There are remote command headquarters that coordinate and advice on rescue operations. The majority of mobile nodes are rescuers located in MANETs. The minority of mobile nodes are headquarters located on the Internet. Each rescuer carries a WiFi capable mobile device that can provide multimedia communication for search and rescue operations. Fig 5.1 illustrates three categories of communications commonly required by most emergency rescue operations: (A) intra-site; (B) site-to-site; and (C) site-to-HQ communications. A rescuer can also move to another affected area. During the movement, an ongoing multimedia communication should be maintained as long as there is some networking coverage nearby.

Therefore, the design of emergency networks for post-disaster rescue communication must fulfill the following criteria.

Fast network deployment One of the most important requirements is to quickly establish communication systems used for rescue operations. Network setup procedures should be as simplest as possible. The devices to be used in emergency network

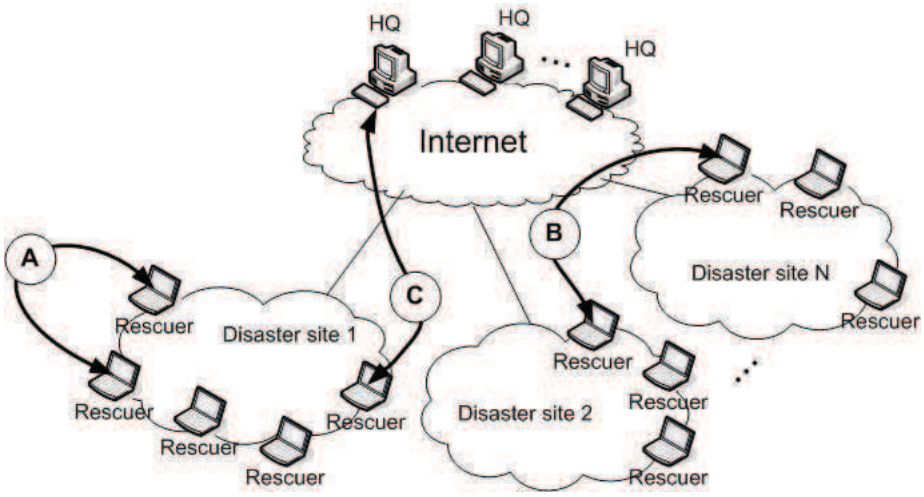


Figure 5.1: Our post-disaster scenario and three types of multimedia communications commonly required during disaster emergency response operations: (A) intra-site; (B) site-to-site; (C) site-to-HQ

may come from locally available sources and are likely commodity devices. Examples of such devices are: Laptop computers having WiFi and multimedia features, WiFi-capable mobile phones, and personal digital assistants (PDAs). However, setting up long-range network connectivity may require specialized equipments such as satellite terminals or point-to-point WiFi.

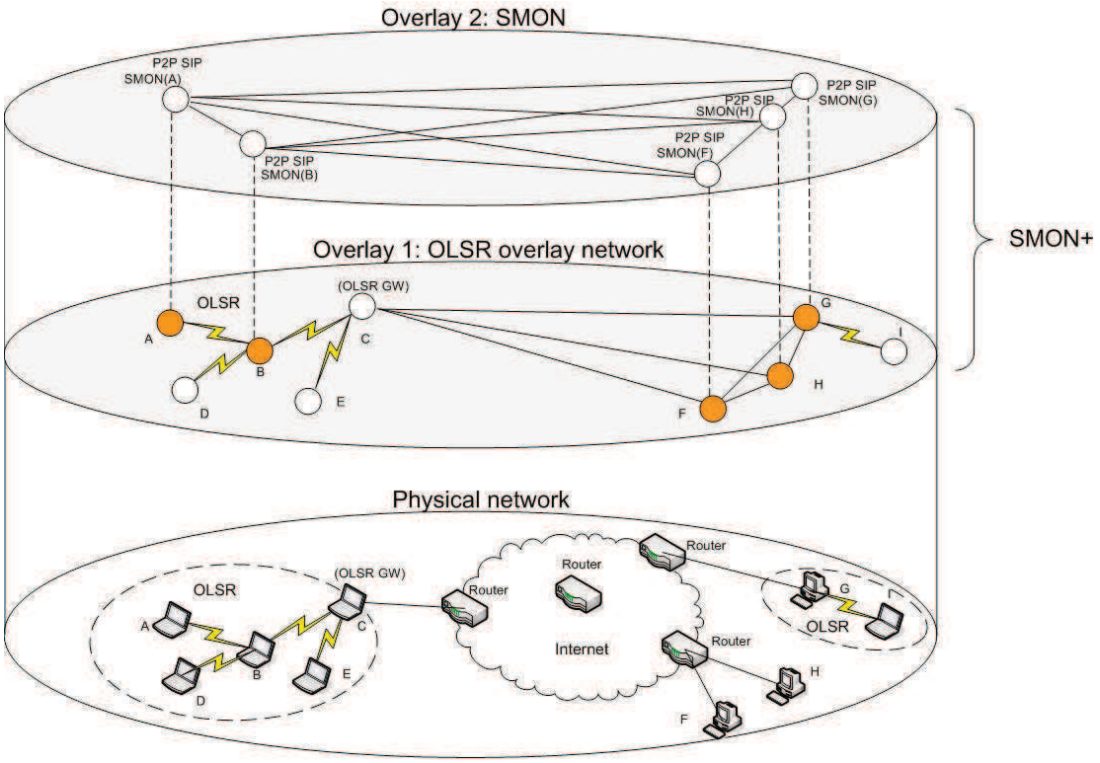
Multi-hop technology Emergency network should cover the entire operation area. It is very likely that multi-hop ad hoc network must be used to provide the coverage.

Terminal mobility support Mobile devices may have to move between the different coverage areas of two or more emergency networks. Therefore providing terminal mobility is a very important issue.

SIP-based communication support Session Initiation Protocol (SIP) is a very flexible and standardized communication protocol for multimedia communications (e.g. voice, video, text). Multimedia applications running in emergency networks should support SIP. Therefore, emergency networks should have features that facilitate SIP.

Tolerance to failures It is common that mobile nodes may break down during the rescue operation. A mobile node may also move at anytime resulting in disrupted communication, either to itself or to other mobile nodes relying on it. The design of emergency networks must assume that node and route failures are common. Applications running in the emergency networks must also tolerate failures and automatically recover from one or more of such failures.

Communication across the public Internet When there are several affected disaster areas and headquarters, the most practical way to interconnect them is through the public Internet. In the design of the emergency networks, we should support and allow communication across the Internet whenever possible. The routing protocol instance in one emergency network should be made aware of potential connectivity to the Internet. The instances of the emergency network routing protocol (running at different disaster-struck sites) should be able to automatically discover and communication with each other when they are connected to the public Internet.



DHT table of A	
NodeID	IP
SMON(A)	A
SMON(B)	B
SMON(F)	F
SMON(G)	G
SMON(H)	H

Routing table of A	
Dest. (IP)	Next hop (IP)
B	B
C	B
D	B
E	B
F	B
G	B
H	B
I	B

Figure 5.2: SMON+ (SMON over OON)

5.2 SMON+ and OLSR Overlay Network (OON)

MANET can have gateways that provide access to the Internet. These gateways have at least two logical network interfaces; one is a wireless interface to interact with nodes in MANET and the other interface that has connectivity to the Internet. A fixed gateway is preferred to a moving gateway because it increases the Internet access stability for MANET nodes. Fig. 5.2 shows OLSR node C acting as an OLSR gateway whose wired interface is connected to the LAN. Any traffic coming in and out of the MANET must go through the gateway. The OLSR network is assigned a valid subnet prefix, which can be routed on the Internet. In this kind of network, the MANET is viewed as a stub network, which does not allow outside traffic to use the network as a transit network.

To expand SMON from MANET to cover the Internet, we need to turn normal IP nodes of the Internet to communicate using OLSR protocol. These OLSR-compatible nodes on the Internet and those SMON nodes on MANETs form the first layer of overlay called the

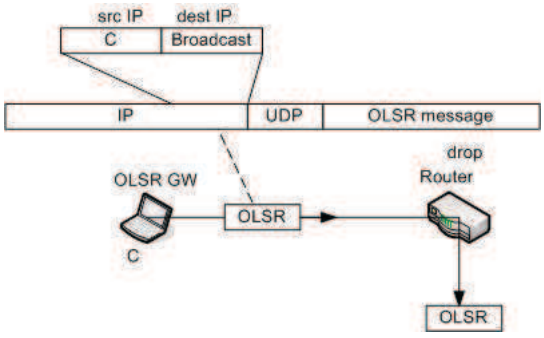


Figure 5.3: OLSR message cannot get through a router

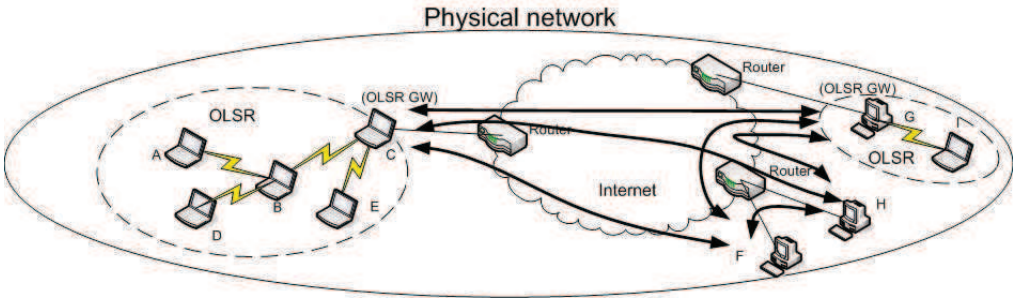


Figure 5.4: Unicast communications among fixed IP nodes

OLSR Overlay Network (OON) and those nodes from OON that are SMON nodes form the second layer of overlay called SMON+.

Fig. 5.2 shows SMON+, where each peer has complete knowledge of all other peers on SMON+. For example, let each node on OLSR network and the Internet be assigned IP addresses A, B, C, D, E, F, G, H, and I as shown in Fig. 5.2. A node identified by *SMON(A)* maintains links to all other overlay members, B on MANET and F, G, H on the Internet where each node is represented by its unique ID which is a result of an application of a hash function, called *SMON*, on its IP address. Thus, in *SMON(A)*, there is a hash table containing *SMON(B)*, *SMON(F)*, *SMON(G)*, and *SMON(H)* as its overlay neighbors. If the node whose IP address is B, which has now been assigned a unique identifier *SMON(B)*, gets disconnected from MANET, OLSR will detect that and inform*SMON(A)* to delete *SMON(B)* from its DHT table. The same node may reenter to MANET and get a new IP address and have a new ID.

In order to put SMON on fixed IP nodes on the infrastructure, we must make normal TCP/IP nodes on the Internet to understand OLSR and exchange OLSR messages. How-

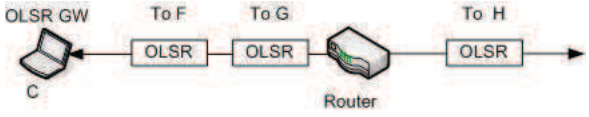


Figure 5.5: Sending and receiving OLSR messages via unicast communications at node C of Fig. 5.4

ever, OLSR is designed to work on MANET, where an IP packet carrying OLSR messages is broadcasted to all nodes within a sub-network. The main obstacle of making an OLSR overlay on the Internet is that OLSR messages are dropped at physical routers as shown in Fig. 5.3. The OLSR messages are encapsulated in a UDP/IP packet whose destination MAC address is configured as a broadcast address (FF:FF:FF:FF:FF:FF). This type of packet cannot be forwarded by the routers thus the fixed IP nodes cannot exchange OLSR messages to other OLSR nodes. We propose to use a broadcast-like concept to enable OLSR messages be exchange with fixed IP nodes. We do not assume broadcast capability for the general Internet. Therefore, we propose to mimic broadcast communication by using several unicast messages to carry OLSR routing messages. A fixed IP node uses a unicast address destined for every other fixed IP nodes. For example in Fig. 5.4 and Fig. 5.5, node C sends OLSR messages to node F, G, and H by using three unicast communications. However, before each fixed node begins exchanging OLSR messages, each fixed node must know the addresses of other nodes that are on the OON.

We propose to store the addresses of the peers that are on the OON in a centralized server on the Internet. These OON peers are required to keep a centralized server informed of their addresses. In order to join the OON, a joining peer needs to perform a bootstrap process by retrieving all addresses of these peers from the centralized server. For example according to in Fig. 5.4, after node C gets addresses of nodes F, G, and H by executing the bootstrap process, it uses these addresses as the destinations to send OLSR unicast packets. They exchange OLSR messages so that they can form one-hop OLSR overlay neighbors shown in Fig. 5.2. Nodes, located on MANET, can receive the one-hop information of OLSR nodes on the Internet via its OLSR GW (node C on 5.2). Other OLSR nodes on the Internet can have information of MANET nodes from OLSR messages exchanged with node C. For example, the MANET node B routing table contains nodes F, G, and H as two-hop neighbors. Exchanging OLSR messages on the Internet creates control overhead. However, we will show later that the control overhead of this network setup is at an acceptable level.

An OLSR node maintains information about its neighbors by using HELLO messages to periodically discover its one-hop neighbors. This neighbor detection can be used for maintaining the OLSR nodes on OON as well. Whenever an OLSR peer detects that a neighbor is lost due to either graceful shutdown or accident disconnection, it can stop exchanging OLSR messages with this neighbor. SMON+ is responsible for managing the overlay network between the MANET and the Internet, while P2P SIP offers P2P SIP user registration and location discovery service based on DHT table provided by SMON+. Once OON is established, MANET nodes and OLSR nodes on the fixed network can be treated as if they belong to a single OLSR network as shown in Fig. 5.6. Moreover, the single OLSR network allows SMON+ nodes to move to other networks while SIPMON+ handles terminal mobility efficiently, which will be explained in section 5.4.

5.3 SIPMON+: P2P SIP on SMON+

Similar to SIPMON, P2P SIP is applied on top of SMON+, which we refer to as SIPMON+, to support SIP registration, call setup, and terminal mobility. Messages ex-

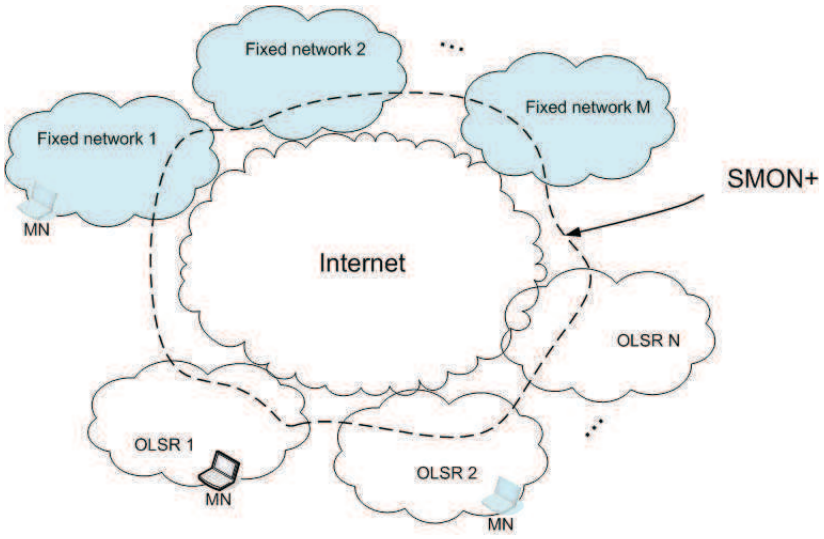


Figure 5.6: SMON+ on a single OLSR network on both MANETs and fixed networks

changed among P2P SIP proxies are regular SIP messages. Furthermore, the P2P SIP is a point, where an existing SIP based application can seamlessly operate with SMON+. On SMON+, P2P SIP performs registration for its SIP user by distributing SIP object identification over other nodes on SIPMON+.

Since we expand SMON to SMON+ by using OON, P2P SIP functionalities are the same for both SMON and SMON+. The procedures of P2P SIP registration, call setup, and binding update are previously explained in Chapter 4. In this section, we only show P2P SIP registration for SIPMON+ by using the same example as in section 4.2. The call setup and binding update procedures are similar to P2P SIP on SMON.

In Fig. 5.7 illustrates SMON+, where P2P SIP is deployed on top of it. The physical topology can be referred to Fig. 5.2. John is a SIP user at *SMON(A)* with SIP URI "John@abc.com". John's SIP phone sends SIP REGISTER to its P2P SIP using a local address. Next, P2P SIP of *SMON(A)* determines the object ID from *P2P_SIP ("John@abc.com")*, denoted as Obj_{John}. The P2P SIP of *SMON(A)* calls *SMON.query(Obj_{John})* to find an address of a peer whose ID is the closest toObj_{John}, which is the address of *SMON(F)*. Therefore, the P2P SIP of *SMON(A)* forwards the REGISTER request to P2P SIP of *SMON(F)*. After receiving the REGISTER request, P2P SIP at *SMON(F)* adds the binding of SIP URI and IP address of John to its location database and replies SIP 200 OK to P2P SIP of *SMON(A)*. Call setup and binding update processes are similar to P2P SIP on SMON as described in Chapter 4.

The reasons that we do not propose to use SIPMON on MANET with existing SIP on the Internet are as follows:

1. SIPMON and legacy SIP must communicate through a SIP gateway. There must be a SIP gateway that translates SIPMON protocol into SIP protocol on the Internet, and vice versa, in order to support SIP-based communication between SIP nodes on both MANETs and the Internet. Considering that there are several MANETs, a SIP gateway on each MANET needs to be configured to support each MANET

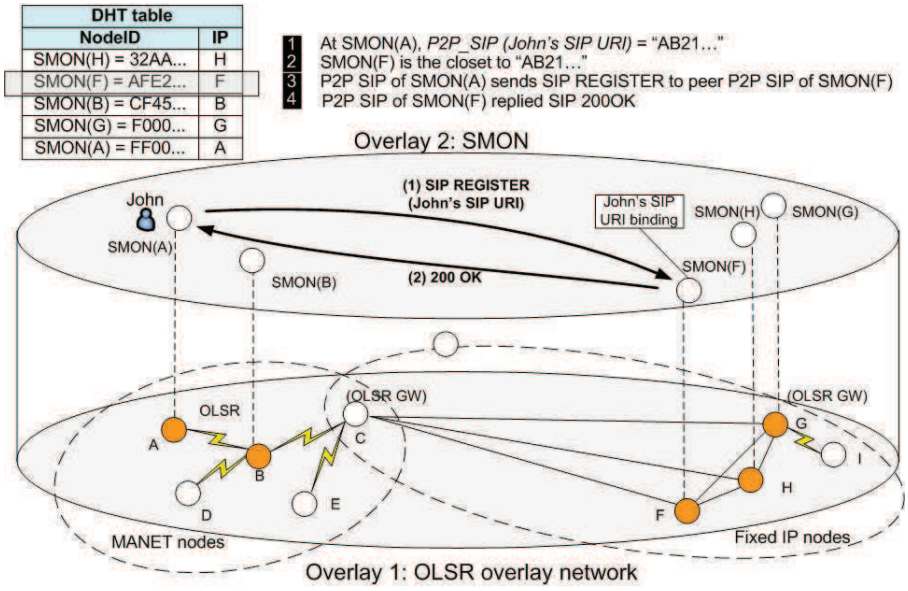


Figure 5.7: SIP user registration example in SIPMON+

for their users to communicate with the Internet users only. Configurations must be done on both the Internet SIP gateway and MANET SIP gateway statically. This will add deployment complexity of setting up during emergency. Moreover, MANET may attach or detach itself to/from the Internet causing its point of attachment to the Internet to change. Again, its SIP gateway needs to be reconfigured to enable SIP to work on the new network environment.

2. Using gateway leads to a single point of failure problem. If the SIP gateway fails to work, which could be due to mobility, the entire SIP support would fail. On SIPMON+, however, the overlay will recover automatically when one of its members fails.

3. Multiple MANETs can move and change their points of attachment to the Internet. They can sometimes swap their points of attachment. For example, MANET A and MANET B swap their point of attachments. Both SIP gateways of these MANETs need to be reconfigured statically. Considering the case that several MANETs swap their points of attachment, this is time-consuming process which we need to reconfigure all SIP gateways belonging to these swapping MANETs during which the entire MANET A will lose communication with MANET B. It is clear that the reconfiguration of this case will certainly add complexity to network deployment during emergency situation.

Therefore, we propose to extend SIPMON to run across the Internet forming a single overlay network. The single overlay network allows number of mobile nodes to move to any network while providing seamless mobility support. In this SIPMON+ architecture, there is no need for any special SIP gateway. Due to distributed SIP based on SIPMON+, it avoids a single point of failure. As a result, this scheme reduces the complexity of setting up of the emergency network and addresses a single point of failure problem.

5.4 SIP terminal mobility support on SIPMON+

Seamless mobility support is one of the main requirements when connecting a mobile ad hoc network (MANET) to a fixed infrastructure network allowing a mobile node to roam from a mobile network to a fixed infrastructure network, such as the Internet, and vice versa, without interrupting ongoing sessions. Such mobility with session continuity is what we refer to as terminal mobility where a terminal can change its locations by moving to a new network, while maintaining the running sessions. Terminal mobility can be handled at the network layer by Mobile IP, RFC 2002 (Perkins, 1996), or at the application layer by the Session Initiation Protocol (SIP), RFC 2543 (Handley et al., 1999). Mobile IP and its variants support TCP/IP fully and is transparent to the transport or application layers, however it is known to have long handoff delay due to triangular routing for the case of mobile IP without route optimization, packet encapsulation overhead and the need for home addresses. Terminal mobility provided by SIP, on the other hand, is not suitable for TCP-based applications as it is difficult for an application to maintain TCP connections when moving across subnets but SIP has been demonstrated to be appropriate for Voice over IP (VoIP) communications with Real-time Transport Protocol (RTP/UDP) with low handoff delay when compared to mobile IP (Yeh et al., 2006; Zeadally & Siddiqui, 2007).

On top of SMON+, we apply a distributed SIP, or a P2P SIP as opposed to a centralized SIP, to offer SIP terminal mobility support for heterogeneous networks in order to deliver seamless interoperability to both users on MANET and the Internet. Unlike Mobile IP, SMON+ provides terminal mobility without any home agent (HA) or foreign agent (FA). These agents are necessary for mobility management in Mobile IP where a mobile terminal keeps the same IP address when it moves to a foreign networks and communications between HA and FA rely on establishing tunnels to/from mobile nodes, hence delays in call set up. With SIP, a mobile terminal in SMON+ can freely change to a new IP address according to the network to which it attaches itself. Changing of IP address certainly terminates the ongoing sessions, but SIP on SMON+ will reconnect the previous communication. This reconnection is handled at the application layer and is thus transparent to the terminal as well as being applicable to both IPv4 and IPv6.

In fixed IP networks, SIP proxies are setup based on centralized architecture that may not be suitable for MANET, since node mobility can cause disconnection from the centralized SIP proxies. Although terminal mobility supported by SIP is provided for infrastructured networks, SIP can be applied to support terminal mobility for MANETs as well because SIP is the network independent application layer. Several SIP on MANET solutions (Fu et al., 2005; Leggio et al., 2005; Khlifi et al., 2003; Banerjee & Acharya, 2004; L. Li & Lamont, 2004; Yu & Agarwal, 2005; Castro & Kassler, 2006; Zhang et al., 2006; Stuedi et al., 2007; Fudickar et al., 2009) based on P2P have been proposed, but they still do not explicitly focus on the terminal mobility problem.

5.4.1 Types of terminal mobility

The following mobile types cause terminal mobility between MANETs and fixed networks as shown in Fig. 5.8. First, a mobile node moves between OLSR networks with different

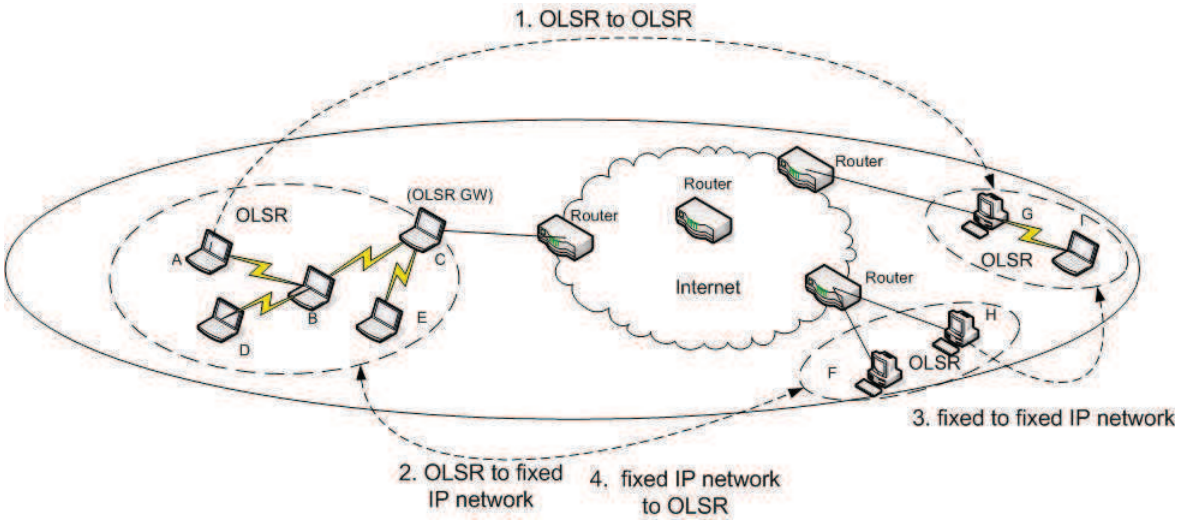


Figure 5.8: Mobile types

network prefixes. For example, an OLSR node with prefix N moves to another OLSR network whose prefix is not N. Second, an OLSR node moves to connect to a fixed IP network. Third, a mobile node on a fixed IP network moves to another fixed IP network. Fourth, a mobile node on a fixed IP network moves to an OLSR network. The P2P SIP on SMON+ supports all mobile types. After moving to a new subnet, the node gains a new IP address. The process of getting a new IP address on the fixed network and OLSR is different. On the fixed IP network, the node can simply lease an IP address via a DHCP server. In OLSR, obtaining an IP address is more complicated than that in the fixed network due to MANET characteristics, RFC 2501. We assume that the OLSR node can acquire an IP address through address autoconfiguration (Boudjit et al., 2005; Clausen & Baccelli, 2005; Weniger, 2006; Mase & Adjih, 2006). We assume that address the autoconfiguration is available and focus our attention on the sequence of the handoff once the MN receives a new IP address at a new network.

SIP supports terminal mobility, which can be either pre-call or mid-call (Schulzrinne & Wedlund, 2000; Yeh et al., 2006). The pre-call mobility is the binding update process, which ensures that Correspondent Node (CN) can reach Mobile Node (MN) after MN moves to a new subnet. The mid-call mobility handles handoff between CN and MN that allows them to resume an ongoing session.

5.4.2 Pre-call mobility

The pre-call mobility is transparent to a SIP user because P2P SIP deals with this mobility. After a SMON+ node changes subnet and gets a new address, it has to update the new SIP binding with this new address by sending a SIP REGISTER request according to the P2P SIP binding update as previously mentioned. However, before sending the request, the SMON+ node must join the OLSR overlay network and SMON respectively. Fig. 5.9 shows an example of pre-call mobility when *SMON(E)* moves from OLSR 1 to OLSR 2. In this example, we assume that every node including CN joins SMON+, and *SMON(B)*

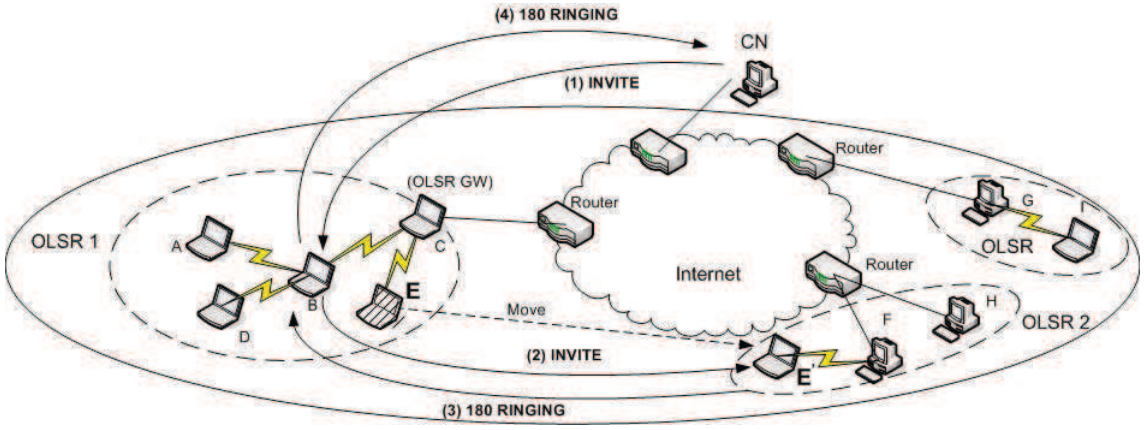


Figure 5.9: An example of the call setup flows after pre-call mobility on the SIPMON+

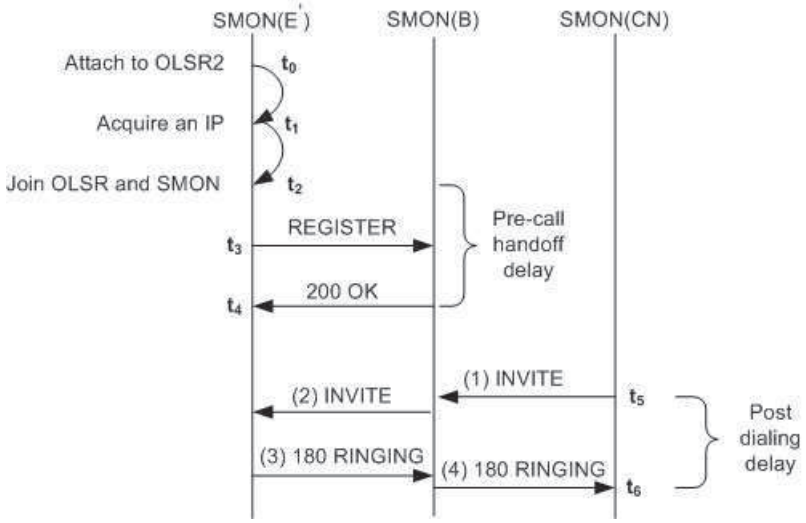


Figure 5.10: The sequence of times for SIP binding update after the node acquires a new address in pre-call mobility

is a new P2P registrar for *SMON(E)*. After *SMON(E)* at t_0 moves to attach OLSR 2, it receives a new IP address, $\acute{E}$, at time t_1 as shown in Fig. 5.10. *SMON(É)* begins joining SMON+ at t_2. The interval t_1 and t_2 is small because P2P SIP recognizes the change of address via SMON+. Since SMON+ messages are OLSR messages, *SMON(É)* can starts sending SMON+ messages along with normal OLSR messages. Once OLSR routing convergence is complete, the SMON+ is also fully constructed. At t_3 and t_4, P2P SIP of *SMON(É)* sends a new SIP REGISTER request to P2P SIP of *SMON(B)* and receives SIP 200 OK. From this point forward, the binding update is successful, and *SMON(CN)* can give a call to *SMON(É)* by using P2P SIP.

We define the interval t_2 and t_4 as the handoff delay of pre-call mobility. The delay of call setup or post-dialing delay is measured by using the interval t_5 and t_6, which is a difference between the times of sending INVITE and receiving SIP 180 RINGING. However, we consider the interval t_2 and t_3 as the main delay for the binding update process.

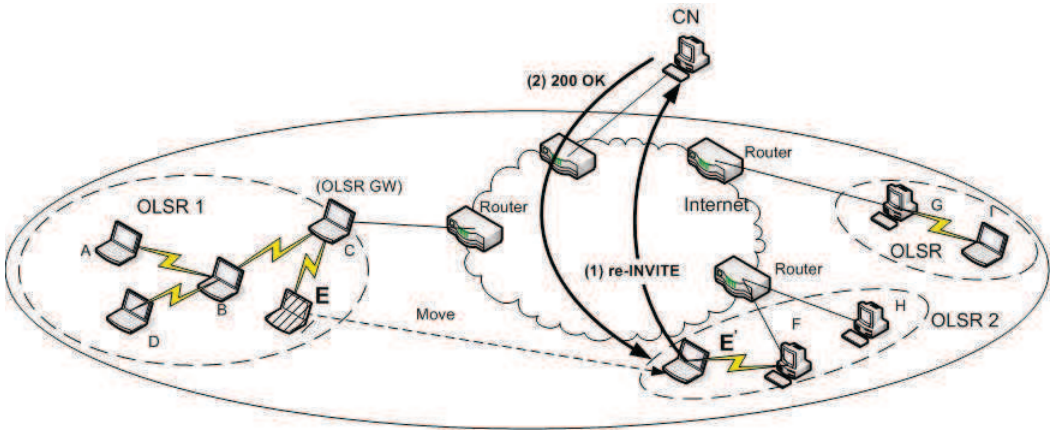


Figure 5.11: An example of the mid-call mobility

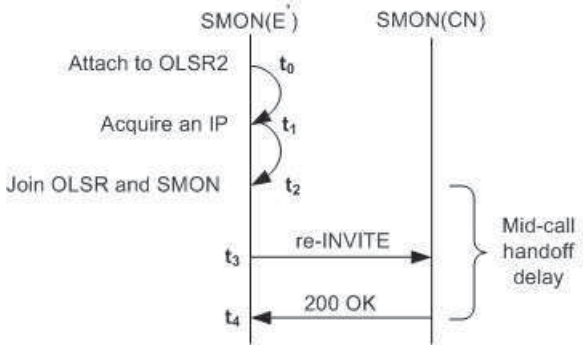


Figure 5.12: Handoff delay in the mid-call mobility

5.4.3 Mid-call mobility

Mid-call mobility allows the SIP node to maintain running RTP sessions after it changes to a new IP address. Mobile users may change their points of attachment during their ongoing session. For example, in a post-disaster emergency network scenario, a vehicle representing a mobile user moves from one disaster site to another site. P2P SIP and SMON+ do not become involved in the mid-call mobility. A SIP stack of the phone application must be able to detect a change of address and directly handles the mid-call mobility. In Fig. 5.11 and Fig. 5.12, when a SIP phone application at *SMON(E)* detects a change in address, $\acute{E}$, and there is a live session going on, it directly sends a re-INVITE request to *SMON(CN)* with the same call-ID as the previous call setup with a new IP address contained in the Contact header of the request. After *SMON(CN)* replies SIP 200 OK to *SMON(É)*, they can resume the opened RTP sessions. For mobile type 2 and 3, *SMON(É)* can immediately send re-INVITE once it finishes joining SMON+. However, *SMON(É)* has to wait until routing convergence is finished before it sends a re-INVITE request in case of mobile type 1 and 4, in which it joins a multi-hop OLSR network.

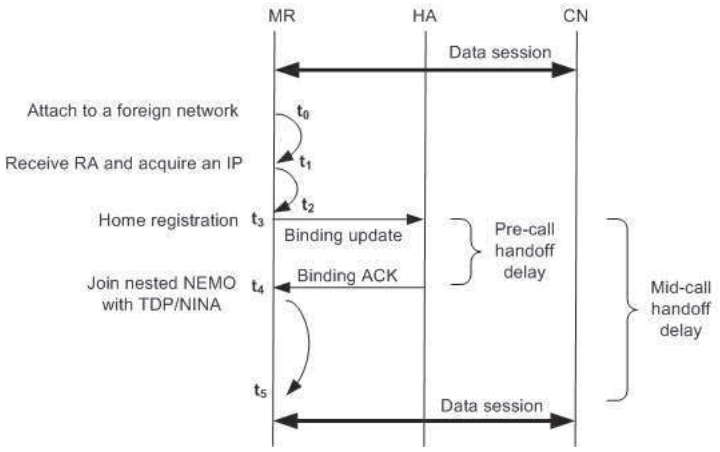


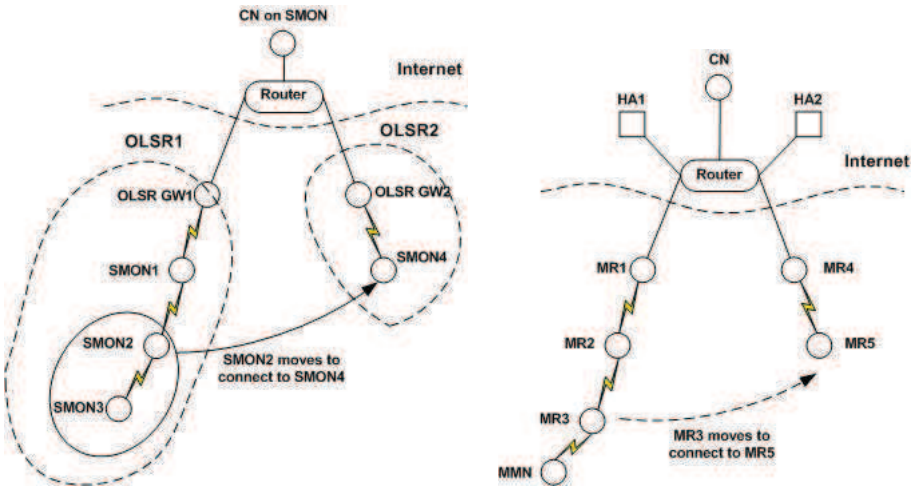
Figure 5.13: Pre-call and mid-call handoff delay in MANEMO

5.5 Evaluation of SIPMON+

In this section, we describe how the testbed environment was setup to evaluate the performance of SIPMON+ as compared with MANEMO (Wakikawa et al., 2007), MIP6-MANET (Y. S. Chen et al., 2006), and SIPHoc (Stuedi et al., 2007). A comparison of SMON+ with a network layer mobile-IP based MANEMO in term of handoff delay was made for a specific linear network topology. MANEMO provides network mobility support with route optimization among Mobile Routers (MRs) in a nested multi-hop network based on MANET routing protocol. An MR acts as a MANET node while providing access point capability to a group of nodes that do not have mobility function. We compared the hand-off performance on SMON+ with those on MANEMO with Tree Discovery protocol (TDP (Thubert et al., 2007) and Network In Node Advertisement (NINA) (Thubert et al., 2008). The TDP is a distance vector protocol which provides a loop-free topology and NINA supplies route optimization by preventing unnecessarily forwarding along a path to multiple HA within the nested MRs. When the MR moves to a new location, it makes the network mobility transparent to all the nodes under itself. Terminal mobility in MANEMO requires the use of Home Agent (HA) with the purpose of exchanging handoff signaling and binding updates. MANEMO uses mobile IPv6 to support terminal mobility. Fig. 5.13 shows handoff delay of terminal mobility in MANEMO with TDP/NINA.

MIP6-MANET uses mobile IPv6 to provide a session continuity between CN and MN. We compare a handoff delay between SIPMON+ and MIP6-MANET based on the results given by Y. S. Chen et al. (2006).

SIPHoc does not provide the measurement on terminal mobility handoff. We cannot compare the post dialing delay of SIPMON+ with SIPHoc. However, SIPHoc shows the post dialing delay between SIP users on MANET and the Internet on static networks. Then, we compare the post dialing delay of SIPMON+ and SIPHoc with the same network topology in static environment.



(a) SIPMON+ topology (b) MANEMO topology

Figure 5.14: SIPMON+ and MANEMO testbed topology

5.5.1 Testbed implementation

Our testbed is setup for post-disaster scenario that consists of two separated disaster areas. Rescuers may move between these areas by using vehicles. We assume that the disaster does not completely destroy an existing network infrastructure. These two areas can be linked via the infrastructure. There is a remote command headquarter which resides in the infrastructure. The testbed represents for this post-disaster scenario. We compare our approach that does not using mobility agents, e.g. HA or FA, with MANEMO based on mobility agents.

We deployed two kinds of testbed scenarios, which were stationary and moving scenarios as shown in Fig. 5.14. P2P SIP on SMON+ was developed and tested by using six Eee PC 901 laptops running Ubuntu kernel version 2.6.24-21 and one router. Fig. 5.14(a) illustrated the physical network topology of SMON+. OLSR GW 1, SMON 1, SMON 2, and SMON 3 were in the first OLSR network, and OLSR GW 2 and SMON4 were in the second OLSR network. All nodes on SMON 1, SMON 2, SMON 3, SMON 4, GW1, and GW2 including CN on the Internet formed SMON+. The router was used to simulate the Internet connectivity. An SMON+ plug-in was created by using the OLSR implementation (Tϕnnesen et al., 2004). We used MjSip (mjsip.org, 2006), SIP stack, to develop the java-based P2P SIP. Linphone 2.0.1 (linphone.org, 2007), an open-source SIP phone, was used to measure call setup delays or post dialing delays between SMON 3 and CN in the static scenario. In the moving scenario, where SMON 2 and SMON 3 moved to connect under SMON 4, we did not use Linphone to measure handoff delays since Linphone did not support SIP terminal mobility. We developed our own small SIP client built-in SIP SIP terminal mobility, which can detect a change of it IP address via SMON+, for mid-call mobility measurement.

In MANEMO, two home agents were set up including SHISA with NEMO basic support (moblieip.jp, 2004). We used five portable computers running as MRs with NetBSD 4.99.54, each of which was installed our implementation TDP/NINA, an extension of Ze-

bra routing software (zebra.org, 2003).. An Eee PC functioned as the Mobile Network Node (MNN) attaching to MR3. In Fig. 5.14(b), MR3 along with MMN disconnected from MR 2 and moved to connect to MR 5. MR 1, MR 2, and MR 3 registered to HA 1, while MR 4 and MR 5 registered to HA 2. We used the same SIP phone application, Linphone, to make a communication and measure post-dialing delay between MMN and CN. We used IEEE 802.11g for wireless communication and Ethernet 100 Mpbs for wired communication for both SMON+ and MANEMO networks.

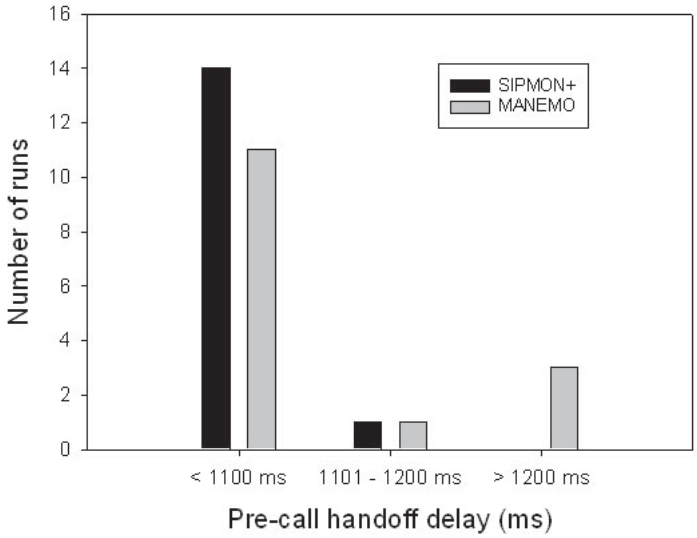
5.5.2 Testbed results

We evaluated the performance of SIPMON+ and MANEMO by measuring pre-call and mid-call mobility handoff delay, post-dialing delay, and control overhead. Since these delays involve the delay at the data link layer where different data-link technologies and connection modes (e.g. access point mode or ad hoc mode) give different delays when a mobile node changes its data-link attachment. Therefore, for our comparison, we start taking handoff delay measurement as soon as a mobile node for the case of SMON+ or an MR for MANEMO obtain a new IP address after reattaching itself with a new network only. For testbed result analysis, we used tcpdump (MG, 2009), packet sniffer software, to collect experimental testbed data. We repeated our experiments 15 times for pre-call and mid-call mobility handoff delays while for post-dialing delay we took averages over 200 SIP calls as it does not involve physical movement.

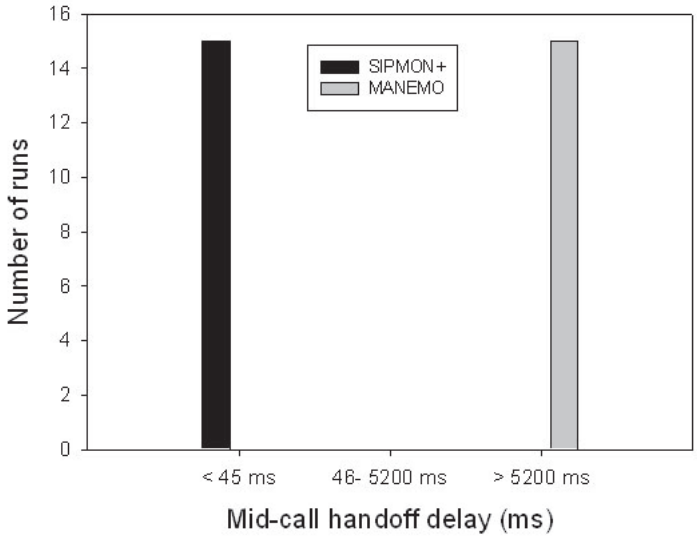
Pre-call and mid-call mobility handoff delay

We compare the pre-call and mid-call mobility handoff delay between P2P SIP on SMON+ and MANEMO. The pre-call mobility handoff delay of P2P SIP on SMON+ is the interval between t_2 and t_4, as shown in Fig. 5.10. When SMON 3 moves to OLSR 2 as shown in Fig. 5.14(a), SMON 3 receives a new IP address. SMON 3 then rejoins SMON+ followed by sending one of the P2P SIP on SMON+ nodes its new SIP binding based on the P2P registration as mentioned in section 4.2. In Fig. 5.15, the pre-call mobility handoff delay of SMON 3 was 1058 ± 13.34 ms on average, which 14 out of 15 runs had delays less than 1100 ms. The pre-call mobility handoff delay of MANEMO is the interval between t_3 and t_4 as defined in Fig. 5.13. When MR3 moves to connect to MR5 as shown in Fig. 5.14(b), MR3 receives a new IP address, and after that MR3 sends HA1 its new binding update. In the testbeds, the pre-call mobility handoff delay of MR3 was 1247 ± 235.79 ms on average.

We measured the mid-call mobility handoff delay of P2P SIP on SMON+ using the mid-call handoff delay period as displayed in the diagram in Fig. 5.12. In Fig. 5.15, all runs of P2P SIP on SMON+ had mid-call handoff delays less than 45 ms, while all runs of MANEMO had mid-call handoff delays higher than 5200 ms. The mid-call handoff delay of SMON 3 was 32 ± 5.88 ms on average. Due to SIP terminal mobility at the SIP client, SMON 3 could immediately send a re-Invite request to CN after it moved to OLSR 2. On the other hand, routing convergence of TDP/NINA in MANEMO ($t_4 - t_5$), as shown in Fig. 5.13, was several seconds. After the convergence, an ongoing session was resumed. Hence, the mid-call mobility handoff delay of P2P SIP on SMON+ was significantly lower as opposed to TDP/NINA.

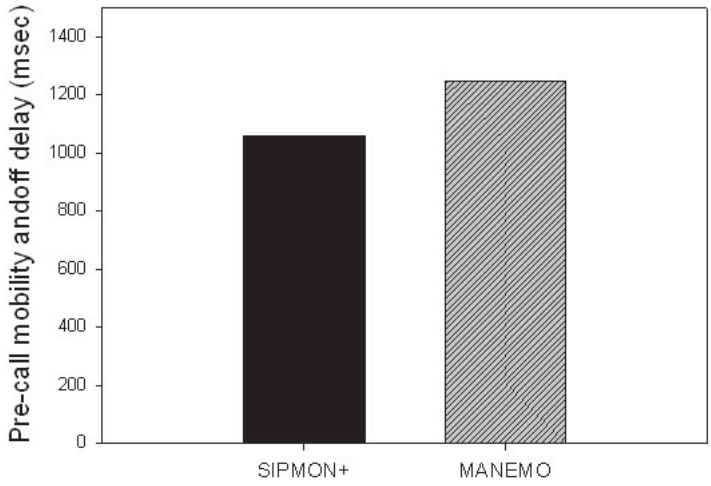


(a) Distribution of runs in pre-call mobility handoff delay

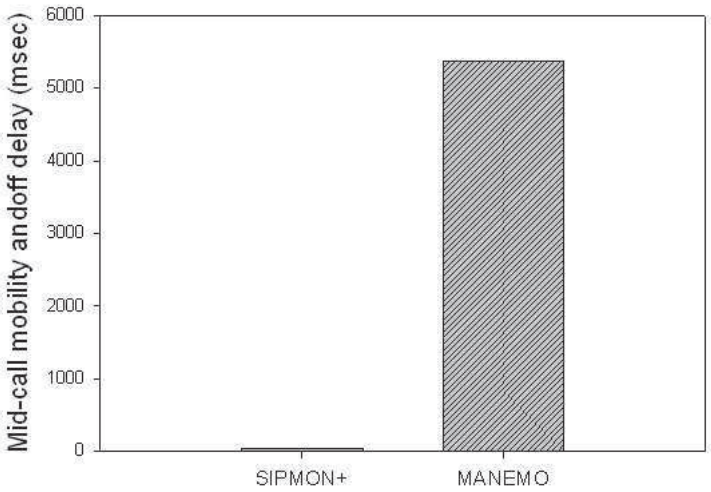


(b) Distribution of runs in mid-call mobility handoff delay

Figure 5.15: Pre-call and mid-call mobility handoff delay comparison between P2P SIP on SMON+ and MANEMO



(c) Average pre-call mobility handoff delay



(d) Average mid-call mobility handoff delay

Figure 5.15: Pre-call and mid-call mobility handoff delay comparison between P2P SIP on SMON+ and MANEMO (cont.)

Post dialing delay

The post dialing delay is the difference between the times that a caller sends the INVITE request to the callee and that of receiving back 180 RINGING as shown in Fig. 5.10. In P2P SIP on SMON+, 200 calls were made from CN to SMON3 before SMON2 and SMON3 moved. After they moved to OLSR2, CN gave the other 200 calls to SMON3. In MANEMO, similarly, 200 calls were made from CN to MNN, and the other 200 calls were set up after MR3 moved to connect to MR5.

In P2P SIP on SMON+, all 200 calls had post dialing delays below 200 ms as shown in Fig. 5.16(a). The number of calls with post-dialing delays below 100 ms was 164 or 84% of total calls. The average post-dialing delay was 73.95 ± 3.91 ms, which was considered as very small delay. In MANEMO, a call setup signaling between CN and MNN went to a tunnel established between HA1 and MR3, which added a delay to the call setup. The post-dialing delay in MANEMO was 115.40 ± 13.13 ms on average.

In Fig. 5.16(b), when SMON 2 and SMON 3 moved to OLSR 2, the post dialing delay of 200 calls was 72.45 ± 3.67 ms, which was similar to the result of the previous case, before moving, whereas the post dialing delay of MANEMO was 128.30 ± 12.79 ms, which was higher than the previous case because the communication path between CN and MNN was changed from CN → HA1 → MR1 → MR2 → MR3 → MNN to CN → HA1 → HA2 → MR4 → MR5 → MR3 → MNN, 1 hop added.

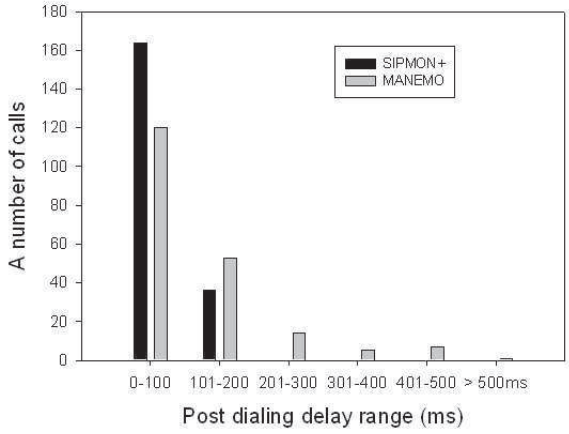
Control overhead

The control overhead was measured by using bytes generated per second. In SIPMON+, we measured the control overhead by counting all OLSR routing and SMON messages. For MANEMO, the control overhead was calculated by using Router Advertisement (RA) with TIO messages and NINA messages. The control overhead of SIPMON+ was higher than the control overhead of TDP/NINA, as shown in Fig. 5.17. We used HELLO and TC interval of 2 s and 5 s respectively, as suggested in the OLSR [RFC 3626]. In MANEMO, the RA interval was every 2 s. The average of control overhead in bytes per second was 1038 in MANEMO and 1486 in SIPMON+. Even though the control overhead of SMON+ is higher, the next section will explain how it can be scalable.

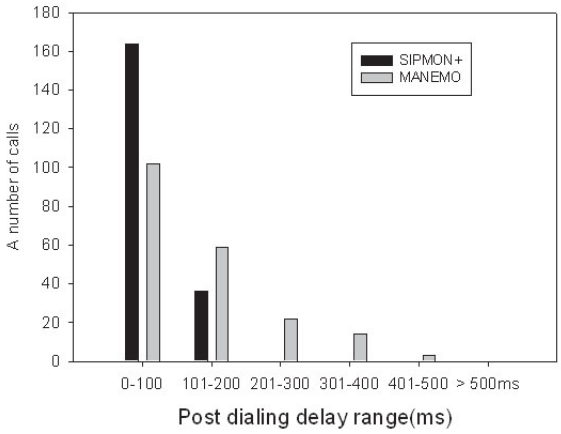
Based on our testbeds, setting up MANENO is time-consuming process. Installing hardware and software of HAs and MRs is a specialist job that requires expertise. On the other hand, SIPMON+ mostly operated on any computers with a standard Wi-Fi without implementing centralized servers. Hence, we can turn commercial laptops into SIPMON+ nodes in order to establish the post-disaster recovery network within an hour or two.

5.5.2.1 Comparison with MIP6-MANET

We compare MIP6-MANET (Y. S. Chen et al., 2006) with SIPMON+ handoff performance. Similar to MANEMO, MIP6-MANET uses mobile IPv6 mechanism to handle terminal mobility. Fig. 5.18 shows the network topology of MIP6-MANET testbed, which is similar to the topology of our testbed. In MIP6-MANET, the authors define handoff



(a) Before SMON 3 and MR 3 moved



(b) After SMON 3 and MR 3 moved

Figure 5.16: Post dialing delay (ms)

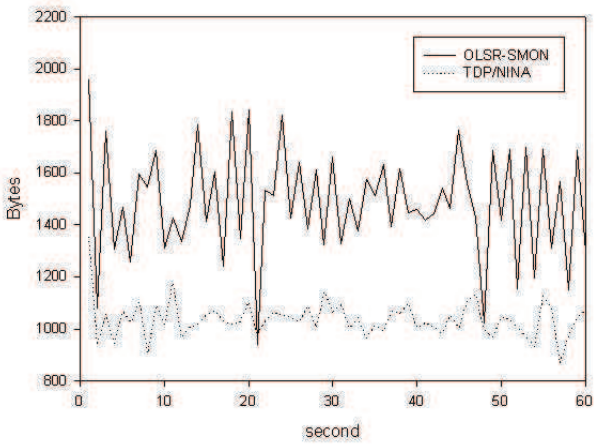


Figure 5.17: The average control overhead of SIPMON+ compared with TDP/NINA MANEMO

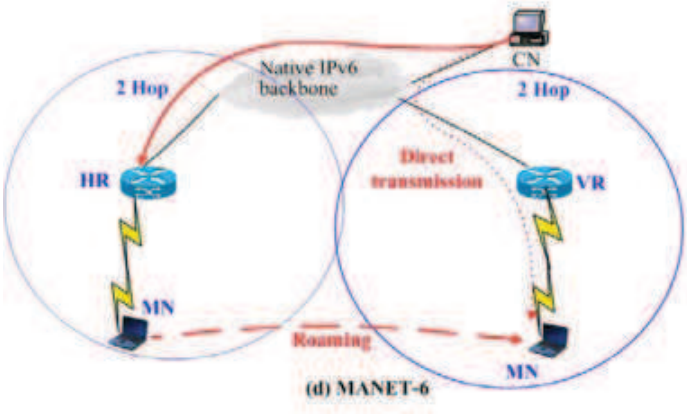


Figure 5.18: MIP6-MANET testbed topology (Source: MIP6-MANET (Y. S. Chen et al., 2006)

jitter as a summation of the handoff delays which include both pre-calls and mid-calls. The handoff jitter when the MN moves to the other network is 5000ms whereas SIPMON+ uses 1055ms and 32ms to complete the handoffs of the pre-call and the mid call mobility respectively. Thus, SIPMON outperforms MIP6-MANET in terms of handoff jitter.

5.5.2.2 Comparison with SIPHoc

Besides MANEMO and MIP6-MANET, we compare the post dialing delay with SIPHoc (Stuedi et al., 2007), which provides SIP interoperability between MANET and Internet users. Because SIPHoc provides post dialing delay measurement on static networks, we cannot compare the handoff delay with SIPHoc. One of testbed results in SIPHoc shows that the post dialing delay between an OLSR node and a fixed IP node is 250 ms. The physical distance between the OLSR node to its gateway is four hops. With the same network topology in static environment, the post dialing delay of SIPMON+ is lower than that of SIPHoc 3.5 times.

5.6 Scalability analysis

We use the number of bytes transmitted per second to describe the control overhead of both SMON+ and OLSR. In MANET, mobile nodes exchange OLSR messages by using a broadcast mechanism, but on the Internet, fixed nodes use unicast communication to exchange OLSR messages. Hence, we divide the control overhead calculation into two parts: within MANET and over the Internet according to OON architecture.

Control overhead within MANET.

We show control overhead in bytes/second between SMON (including OLSR) and MANEMO in a linear topology according to our testbed scenario.

SMON and OLSR control overhead. OLSR nodes periodically advertise HELLO messages to discover its one-hop neighbors. The number of bytes transmitted per second of HELLO messages is:

$$B_h = (P_h/\tau_h) \times N,$$

where P_h is the HELLO message size in bytes, τ_h is the HELLO interval, and N is the total number of OLSR nodes. MPRs periodically advertise TC messages in order to exchange information of network topology. The number of bytes transmitted per second of TC messages is:

$$B_{tc} = (P_{tc}/\tau_{tc}) \times N_{mpr} \times (N_{mpr} - 1),$$

where P_{tc} is the TC message size in bytes, τ_{tc} is the TC interval, and N_{mpr} is the number of MPRs. N_{mpr} in a liner network topology is $N - 2$. Hence, the OLSR control overhead is:

$$B_{olsr} = B_h + B_{tc}.$$

The sizes of HELLO (P_h) and TC (P_{tc}) are calculated according to HELLO and TC message format.

$$P_h = P_{hrd} + 8 + (16 \times N_{nbrs})$$

$$P_{tc} = P_{hrd} + 4 + (16 \times N_{nbrs}),$$

where P_{hrd} (Ethernet header + IPv6 header + UDP header + OLSR header) is 90 bytes. N_{nbrs} the number of neighbor nodes in a linear topology. N_{nbrs} is equal to 2 in this case. SMON uses two types of messages: JOIN and LIST OF ALL MEMBERS. Only a primary node periodically advertises a LIST of ALL MEMBERS message. The number of bytes transmitted per second of LIST of ALL MEMBERS messages is:

$$B_{list} = (P_{list}/\tau_{SMON}) \times N_{mpr},$$

where P_{list} is the LIST of ALL MEMBERS message size and τ_{SMON} is the LIST of ALL MEMBERS broadcast interval. The number of bytes transmitted per second of JOIN messages is:

$$B_{join} = (N \times P_{join} \times N_{mpr})/\delta T,$$

where P_{join} is the JOIN message size. We assume that the time interval between t_0 and t_1 or δT is observed period for which we are interested in calculating the control overhead. Hence, the number of the SMON control overhead is:

$$B_{smon} = B_{list} + B_{join}.$$

The sizes of LIST OF ALL MEMBERS (P_{list}) and JOIN (P_{join}) are calculated based on SMON message format:

$$P_{list} = P_{hrd} + 4 + (16 \times N)$$

$$P_{join} = P_{hrd} + 20$$

The control overhead of SMON and OLSR is:

$$B_{smon\_olsr} = B_{smon} + B_{olsr}. \tag{5.1}$$

MANEMO control overhead. MANEMO uses RA and NINA messages (Tazaki et al., 2009). Hence, the control overhead is:

$$B_{manemo} = P_{ra}/I_t \times N + \sum_{i=1}^{n}(P_{nina}(i)/I_t), \tag{5.2}$$

where P_{ra} is the packet size of RA, $P_{nina}(i)$ is the packet size of NINA at each $MR(i)$, and I_t is the periodic interval time of RA and NINA packets. P_{ra} = Ethernet header + IPv6 header +ICMP header + RA header + link-layer address option + prefix option + tree information option = 142 bytes. $P_{nina}(i)$ = Ethernet header + IPv6 header +ICMP header + Neighbor advertisement header + link-layer address option + Network In Node option $\times(N_c(i) + 1)$, where $N_c(i)$ is the number of nodes in child MRs of node i (i = 1 is the root node of the tree).

From Eq. 5.1, we assume that all OLSR nodes join SMON, and the value of δT is very high, so B_{join} approaches to 0. We show that the control overhead of SMON (including OLSR) and MANEMO in Fig. 5.19 base on Eq. 5.1 and Eq. 5.2 where $\tau_h = 2$, $\tau_{tc} = 5$, $\tau_{SMON} = 5$, and $I_t = 2$. When the number of nodes increases, the control overhead of both approaches increases. However, the control overhead of SMON (including OLSR) is higher than

Control overhead of OLSR nodes on fixed network or OON.

In MANEMO, there is no control overhead added on the fixed network since MANEMO does not form any overlay network across the Internet. We consider control overhead induced by SMON (including OLSR) on the Internet only because fixed nodes on the

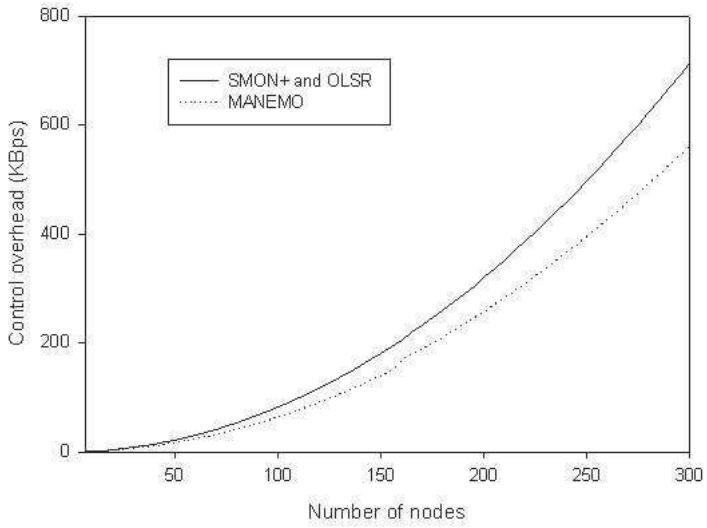


Figure 5.19: Control overhead of SMON (including OLSR) compared with TDP/NINA MANEMO: theoretical calculation.

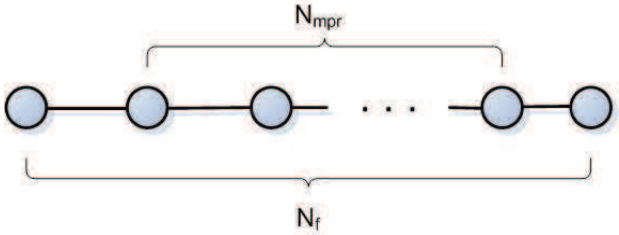


Figure 5.20: A linear topology of OLSR nodes on the fixed networks

OON exchange OLSR messages. The SMON control overhead can be neglected because the predominant overhead is due to OLSR routing messages. The number of HELLO messages generated at each OLSR node on the fixed networks in bytes per second is:

$$B_{h\_gen} = P_{fh}/\tau_{fh} \times N_{nbrs},$$

where P_{fh} is the size of HELLO messages, τ_{fh} is the HELLO interval, and N_{nbrs} is the number of neighbors. This is due to the fact that each OLSR node on OON performs N_nbrs unicast connections instead of a single broadcast. Similarly, the number of HELLO messages received at each OLSR node on the fixed networks in bytes per second is

$$B_{h\_rev} = P_{fh}/\tau_{fh} \times N_{nbrs}.$$

The size of a HELLO message is:

$$P_{fh} = 8 + (16 \times N_{nbrs}).$$

The number of TC messages generated at each OLSR node on the fixed networks in bytes per second is:

$$B_{tc\_gen} = P_{ftc}/\tau_{ftc} \times N_{nbrs},$$

where P_{ftc} is a size of TC message and τ_{ftc} is the TC interval. Each node receives TC messages from all MPRs. Hence, the number of TC messages received at *each* OLSR node on the fixed networks in bytes per second is:

$$B_{tc\_rev} = P_{ftc}/\tau_{ftc} \times N_{mpr}.$$

The size of TC message is:

$$P_{ftc} = 4 + (16 \times N_{nbrs}).$$

The number of the OLSR routing messages generated and received at *each* OLSR node is:

$$B_{olsr\_gen\_rev} = B_{h\_gen} + B_{tc\_gen} + B_{h\_rev} + B_{tc\_rev}. \tag{5.3}$$

We consider two scenarios: one is a linear topology as shown in Fig. 5.20 and another is a mesh topology, where all nodes are fully connected. In the liner topology, each node has

2 neighbors except for the far left and far right nodes. Therefore, the number of MPRs (N_{mpr})is $N_f - 2$, where N_f is the number of OLSR nodes on the fixed networks. The number of neighbor (N_{nbrs}) is 2. In the mesh topology, all nodes are connected as one-hop neighbors, which do not generate TC messages. These nodes only exchange HELLO messages. Hence, the number of MPRs (N_{mpr}) is 0. The number of neighbors (N_{nbrs}) in the mesh topology is N_f.

From Eq. 5.3, we vary the number of nodes (N_f), the HELLO interval (τ_{fh}), and the TC interval (τ_{ftc}) to plot the graph of the control overhead generated and received per each node as depicted in Fig. 5.21. The y-axis represents the number of OLSR routing messages generated on OON in kilobyte per second. The x-axis displays the number of OLSR nodes on the Internet ranging from 10 to 500. The linear topology consumes very low network bandwidth as compared with the mesh topology. In Fig. 5.21, the control overhead can be reduced by increasing the HELLO and TC intervals. However, these intervals have an effect on time spent for neighbor discovery. The larger interval results in a higher delay of neighbor discovery. It is reasonable to use the higher HELLO and TC intervals for OLSR nodes on the fixed networks because these nodes do not move frequently.

If we consider attaching one OLSR network to the Internet with a typical size of upto 100 nodes (Laouiti, Mhlethaler, et al., 2002), while each node on OLSR can become an SMON+ node, the overhead as shown in graph (Fig. 5.19) is only 82 KBps on OLSR and is about 215 KBps (Fig. 5.21) for a typical 100 nodes on OON on the Internet. If there are 500 OON nodes on the Internet, the control overhead is about 24 MBps. Given that a single SMON+ node can support upto 100 unintelligent SIP phones, if we have 200 nodes (or 600 nodes) on SMON+, it means 20,000 (or 60,000) SIP users can be supported.

5.7 Discussions

We have presented SIP mobility support for an emergency network based on SIPMON+, which is the P2P overlay network on top of OLSR providing service lookups without relying on any centralized directory servers. We propose am overlay-over-overlay architecture that extends SIPMON from MANET to cover fixed nodes on the Internet. This overlay-over-overlay network is composed of OON turning fixed IP nodes into OLSR nodes at the first overlay and SMON at the second overlay network or SMON+. The SMON+ allows overlay nodes on both MANET and the Internet to operate seamlessly without the need of HA and IP tunneling.

We compare the performance of SIPMON+ and MANEMO by measuring post-dialing delay, pre-call and mid-call mobility handoff delay, and control overhead. The performance comparison is based on our defined post-disaster scenario consisting of two MANETs and one command headquarter node located on the infrastructure. The post-dialing delay of SIPMON+ is 1.6 times lower than that of MANEMO because a call setup signaling between a MMN and CN must go through a tunnel between a MR and HA, resulting in a delay added to the call setup time. SIPMON+ outperforms MANEMO in terms of mid-call mobility handoff delays. However, the control overhead of SIPMON+ is higher than that of MANEMO. In section 5.6, we show that the control overhead of SIPMON+

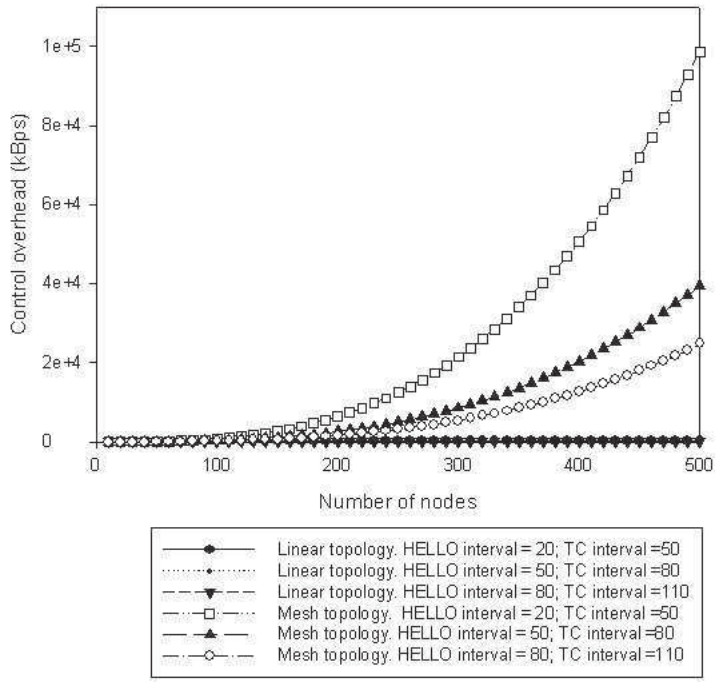


Figure 5.21: The number of OLSR routing messages generated and received on OON

is considered as an acceptable value for MANET and infrastructured network. Based on our testbed scenario, we do not insist that our results can be applied to general cases.

SIPMON+ provides terminal mobility support for a node in a flat network, while MANEMO as a two-tier architecture provides terminal mobility support for a larger group of nodes. Hence, MANEMO can support a higher number of users than SIPMON+. However, SIPMON+ provides better handoff than MANEMO. The next chapter explains a combination of SIPMON+ and MANEMO used to serve a large group of users and provide efficient terminal mobility.

In the next chapter, we will apply P2P SIP on SMON and SMON+ to a real demonstration. We develop an emergency communication application, called Easy Disaster Communication (EasyDC), which provides rich multimedia communications on MANETs.

Chapter 6

Multimedia communication application for an emergency network

This chapter describes a prototype of a multimedia communication for a disaster emergency network based on SIPMON and SIPMON+. The prototype was the demonstrated Digital Ubiquitous Mobile Broadband OLSR (DUMBO)[1] a project of the Internet Education and Research Laboratory (intERLab), Asian Institute of Technology (AIT) to prove the concept. Apart from the demonstration, we conducted performance comparisons between the SIPMON+ and the MANENO approaches for post dialing delay, control overhead, RTP packet loss, scalability, and network deployment issues. We used different scenarios such as static, one moving, and multiple moving nodes for this comparison.

Section 6.1 provides details of Easy Disaster Communication (EasyDC). In section 6.2, we compare different Vehicle to Infrastructure (V2I) testbeds designed for an emergency network.

6.1 Easy Disaster Communication (EasyDC)

In the first version of EasyDC, a broadcast mechanism based on the concept of dSIP was adopted to support a user location discovery. A node periodically advertised its username and IP address to other nodes in the network so that the username could be reached by others users using this IP address. However, the OLSR implementation (Tϕnnesen et al., 2004) did not allow broadcast traffic coming from applications to be forwarded except for OLSR routing messages. Therefore, the first version of EasyDC was implemented by representing a broadcast message through several unicast messages. As a result, to flood one message to N participants in OLSR, a node would use N unicast communications to send the same message to all participants defined in network_list.txt. EasyDC uses Java Media Framework API (JMF) (mjsip.org, 2006) to send audio and video across the network.

Fig. 6.1 shows the user interface components of EasyDC. Before starting any communications, a user has to register oneself to the system by typing a username in the registration window. Each OLSR node running EasyDC exchanges register messages to each other in order to update the list of online users with IP addresses as shown in P2P tools window. The emergency button allows users to send a help message to all users on the list in order to request emergency attention. The incoming message window is used to display any messages received by the application. Many users can join the same chat channel to have a shared conversion. To make a video or voice call, the user selects a target user in the user list panel and then clicks on the video or a call button.

[1] http://www.interlab.ait.ac.th/dumbo/

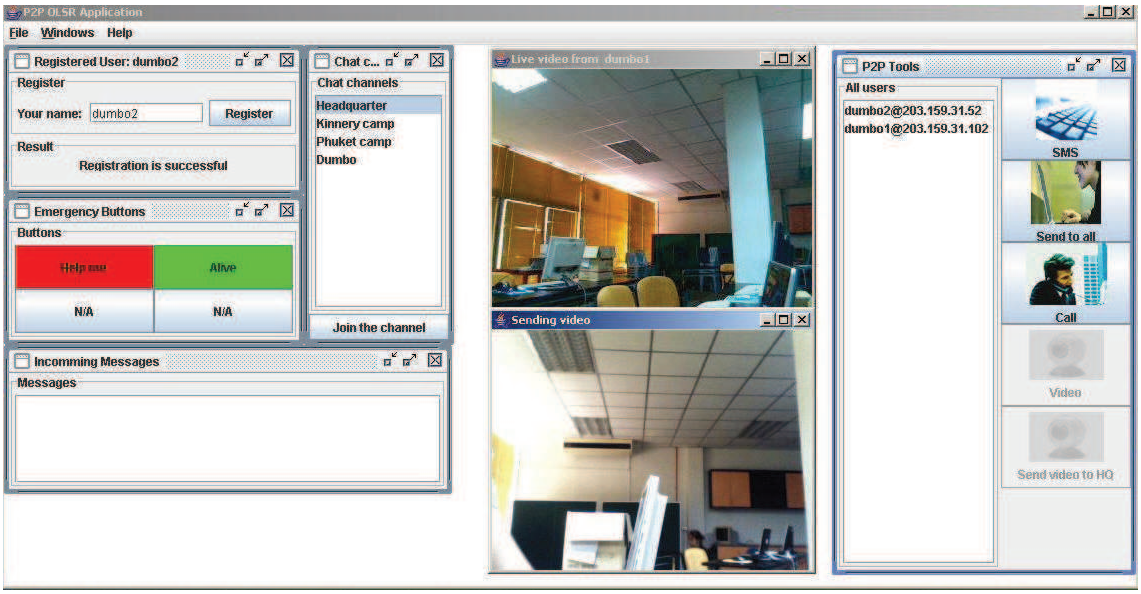


Figure 6.1: EasyDC screenshot

A pre-configured file containing the list of nodes is the drawback of this first version of EasyDC. The content of file needed to be manually synchronized when a new MANET joined the network. Maintaining the same network list.txt on every node added extra time in setting up the network. Moreover, the different content of network list.txt on each node could cause inconsistency for EasyDC. In the broad- cast mechanism, EasyDC created higher control overhead due to the exchanges of registered messages. Moreover, it used the proprietary signaling protocol to handle the setting up of a multimedia session.

The next version of EasyDC was fully implemented with automatic configuration and resource discovery by using the cross-layer design. The overlay messages are encapsulated in OLSR packets, which can be efficiently retransmitted by MPRs. The use of MPR forwarding functionality significantly reduces duplicate retransmissions during the flooding procedure. The goal of implementing the second version of EasyDC was to evaluate the performance of SMON+ and to provide mobility support for V2I and Vehicle to Vehicle (V2V) network in the DUMBO project, where some or all parts of network infrastructure are assumed to have been destroyed by a disaster. MANET nodes were placed in vehicles and long tail boats, while making communications to wired nodes on the infrastructure as a V2I network. Call setup signaling defined in the first version was replaced by SIP due to SIPMON+. Many laptops, running SIPMON+ on OLSR, formed MANET covering the disaster area. This MANET was connected to HQ machine on the infrastructure via an OLSR gateway.

Apart from the demonstration, intensive experimental data collection showed the performance of SIPMON+ compared to MANEMO by using the same physical topology.

6.2 V2I testbeds for emergency network

In an emergency network, the terminal mobility with session continuity within a large network coverage without relying on any existing network infrastructure is mandatory as the rescue worker may need to move randomly in a disaster area or the rescued victim may need emergency medical assistance while being transported in an ambulance. As a consequence, intercommunication among rescue workers (V2V) as well as with the central command center (V2I) is necessary. A stand alone network like MANET can solve the problem of network coverage as it works in a multi-hop fashion. But to support the mobility and uninterrupted communication with the central command center, situated anywhere in the Internet, this stand alone MANET is not enough. MANET needs some mechanisms to enable it to be attached to the internet and mobility support for individual nodes, a group of nodes or group mobility.

In (Arefin et al., 2009), we consider three approaches to connecting mobile ad hoc networks to an infrastructure for a post disaster rescue team communication system, namely, simply connecting the flat OLSR network to the Internet, and two MANEMO set ups: MANEMO-A where NEMO is enhanced by Tree Discovery and Network In Node Advertisement (TDP/NINA) (Thubert et al., 2007, 2008) and MANEMO-B with OLSR (Clausen & Jacquet, 2003) on mobile routers. We developed these three functional prototypes to prove our concept and to compare their performances from the point of view of terminal mobility for individual nodes and for groups of nodes. As the outcome of our experiments, we propose the adoption of MANEMO with OLSR without the use of home agents due to the ease of deployment with terminal mobility to provide session continuity and minimal handover period when group mobility is handled by SIPMON+. We evaluate our proposed scheme based on its performances in our simulation for multimedia communication application with post dialing delay, end-to-end packet loss and the number of control overheads.

Terminal Mobility with SIPMON+

SMON (Wongsaardsakul & Kanchanasut, 2007) works on a standalone MANET, but it can be expanded to work on Internet-Connected MANET by building the OLSR Overlay over the Internet (OON) and applying SMON on both MANET and OON. Thus, terminal mobility with session continuity can be handled by distributing SIP proxies over SMON+ or SIPMON+. Once SIPMON+ node moves to a new network, it acquires a new IP address. It informs a correspondent node (CN) about the new address by using the SIP re-INVITE request to resume an ongoing session. Any MANET node running SIPMON+ software can interact with any other node on the Internet which joins the SIPMON+ overlay network without the need to install any centralized server or home agents. These ubiquitous mobile nodes can be installed in vehicles for V2V and V2I communication setting up an emergency network for a post disaster scenario.

Terminal Mobility with Home Agents

Mobile IP (Perkins, 1996) and Mobile IPv6 (Johnson et al., 2004) enable the end host to move in different access networks without disrupting the ongoing session. But this technique becomes inefficient when a group of users move together; every end host must

be aware of this mobility and the Home Agent must take care of this movement although the mobile users are relatively static in relation to one another (McCarthy et al., 2006). Moreover Mobile IPv6 supports mobility when the mobile nodes are one hop away from the Access Router (AR). To communicate with the infrastructure, NEMO Basic Support protocol (NEMO BS)(Ellison et al., 1999) was designed to support the most basic scenario of network mobility where the mobile nodes' mobility are aggregated and handled as one mobile network with the use of a Mobile Router (MR). The movement is transparent to the mobile nodes which means the mobile nodes do not need to be aware of their mobility and can run any mobility support protocol. Whenever any MR changes its point of attachment, it sends a Binding Update to the Home Agent and receives Binding Acknowledgement from the Home Agent; a bidirectional tunnel is established between the Home Agent and the respective MR. For our prototypes, we implemented our MR's using the Zebra protocol (zebra.org, 2003) while our Home Agents are running the SHISA protocol (moblieip.jp, 2004).

To handle nested NEMO scenarios where one mobile network is connected to another mobile network creating a multi-hop scenario, NEMO BS counteracts some inefficiency; during the end-to-end communication all the packets are intercepted by the Home Agent although they communicate within the same network. This causes multiple IP-in-IP encapsulation and redundant paths between nested networks. MANEMO (Wakikawa et al., 2007) (MANET for NEMO) where Mobile Router (MR) acts as a MANET node, solves these kinds of problems in the nested NEMO scenario. In MANEMO, MANET routing protocol is used to exchange information optimally between local mobile networks. We have developed two MANEMO scenarios (Kanchanasut et al., 2008) as follows:

MANEMO-A We use the Tree Discovery Protocol (TDP) (Thubert et al., 2007) and the Network In Node Advertisement (NINA) (Thubert et al., 2008) protocol for local communications among MANET nodes. TDP is a distance vector protocol which runs inside the Mobile Router (MR) to construct the directed acyclic graph. TDP selects the next hop from the multiple candidate next hops to reach the external network. The route optimization protocol NINA runs on top of the tree established by the TDP. Both TDP and NINA perform the routing functionality in the nested NEMO scenario. In the nested NEMO, the upper level MR provides the internet gateway service to the underlying MR. TDP with NINA, a 2-pass routing protocol where TDP builds a loop-less structure; a tree and NINA exposes the Mobile Network Prefixes (MNPs) up the tree in order to make sure that local traffic remains in the same subnet.

MANEMO-B We deploy the OLSR protocol on each MR's egress interface. In order to provide the capability of injecting external routing information into an OLSR MANET, a node with a non-MANET interface (MR) periodically issues a Host and Network Association (HNA) message, containing sufficient information (MNP of its underlying network) for the recipients to construct an appropriate routing table.

We conducted a couple of testbeds and movements to evaluate the three V2I schemes. From our earlier emergency network experiments (Kanchanasut et al., 2008), it was found

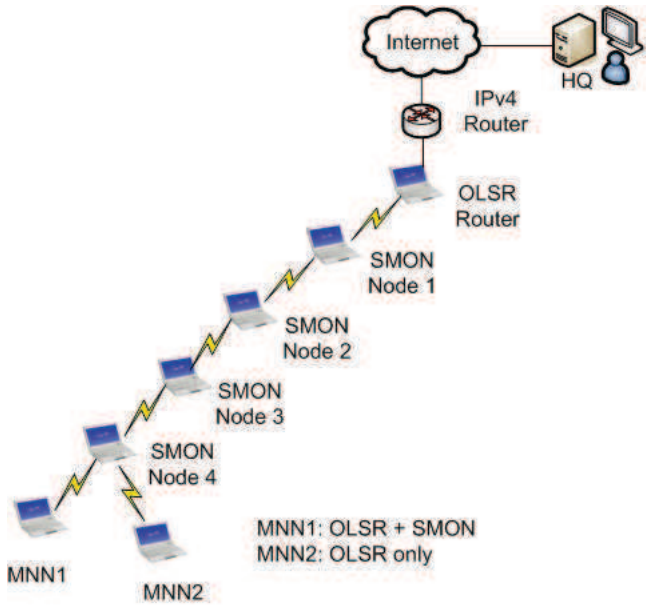


Figure 6.2: Testbed topology of SIPMON+

that with emergency scenarios, extending the network with multi-hops linear topology can stretch the network and enable it to reach far ends of coverage areas with ease of deployment and management as compared to other topologies. We thus adopt a linear topology of mobile routers in our testbed where we create four hops linear connection from the end mobile node to the gateway router as for the case of flat OLSR as shown in Fig. 6.2 for MANEMO-B, or to the access point for MANEMO-A as shown in Fig. 6.3. From this set up, we start moving nodes according to two different scenarios: single node/group movement and multiple nodes/group movement. Our multimedia application is running at the HQ machine and the last multimedia application terminal. We used the same SIP soft-phone (linphone.org, 2007) for multimedia communication for all our measurements.

In this SIPMON+ scenario, each SIPMON+ node is running SIPMON+ software on top of OLSR to create an overlay network and performs both OLSR node and router functionalities. The OLSR gateway is an OLSR node with two interfaces: the Wi-Fi interface runs on an Ad hoc mode to join in the overlay network, and the other interface is the LAN interface connected to the internet.

Fig. 6.3 shows a scenario using MANEMO. In our testbed, we used two different routing protocols (OLSR and TDP/NINA) to perform the MANET function on NEMO. While we used the OLSR protocol, the egress interface of the MR is in ad hoc mode and the interconnection between MRs are egress-to-egress. But for TDP/NINA, the egress interface is configured in the managed mode to enable it to be attached to the upper layer MR's ingress interface working as an Access Point. Our experimental movement scenario is described below:

- For static cases we do not move to any direction. We communicate from the last multimedia application terminal (MNN1) to HQ and measure data on different parameters.

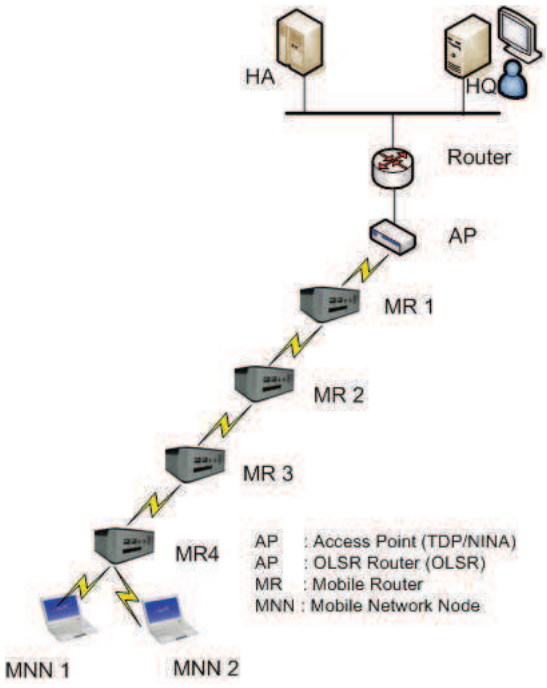
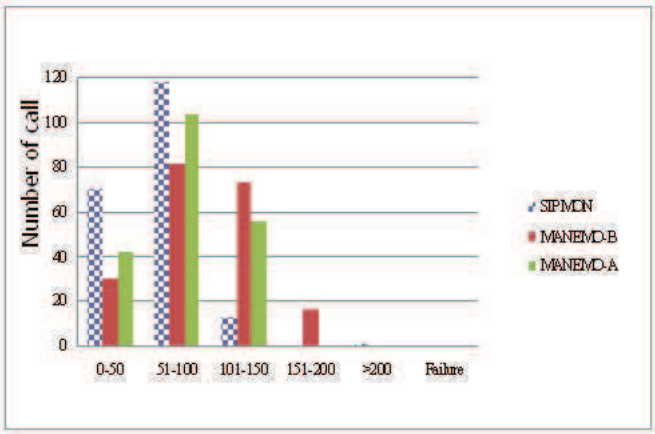


Figure 6.3: Testbed topology of TDP/NINA and OLSR MANEMO

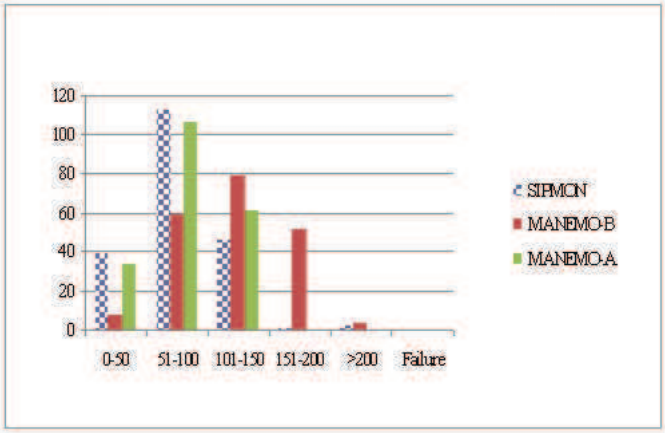
- For single node/group movements we start from a static scenario. We communicate from the last node (MNN1) to HQ and start moving toward the HQ. In this case we have two levels of movements. In the first level, we stop for a while when our mobile nodes (SIPMON+ Node/ MR) change its point of attachment to the upper level node (SIPMON+ Node 2/ MR2). We move further one level up to attach to SIPMON+ Node 1/ MR1.
- For multiple node/group movements, again we start with the static scenario and communicate from the last node (MNN1) to HQ. We start moving with the mobile node with its parent node/router together, maintaining the connection within the group (MR3 and MR4 with MNN1 and MNN2). The parent node changes its association to MR1 instead of MR2, and MNN1 still connects to the parent node. The number of hops from the last node to HQ is one less than the starting number of hops from HQ.
- We repeat the above mentioned experiment (static, single movement and multiple movements) but this time we initiate the session from HQ's machine to MNN1.

a) Post dialing delay

Post dialing delay is an important indicator for a communication system. In the testbed, we initiated the session from different terminals (MNN and HQ) and observed the impact of terminal mobility on each session's setup time. We measured the post dialing delay as the time difference between the sending of the SIP INVITE request and the reception of 180 RINGING requests. During movement, we experienced variant post dialing delay; Fig. 6.4, Fig. 6.5, and Fig. 6.6 show the distribution of these different post dialing delays.



(a) Distribution of post dialing delay for static case (MNN calls HQ)



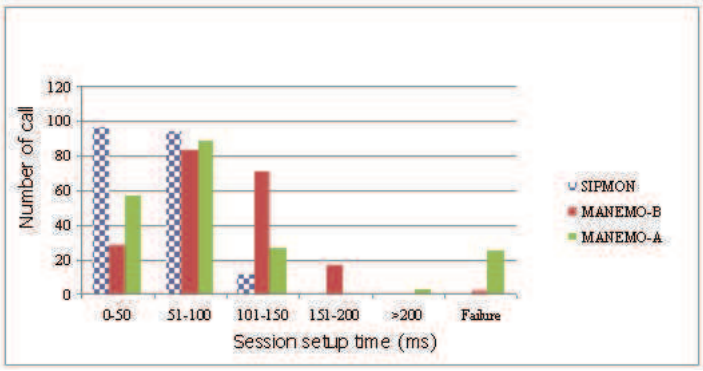
(b) Distribution of post dialing delay for static case (HQ calls MNN)

Figure 6.4: Post dialing delay for static case

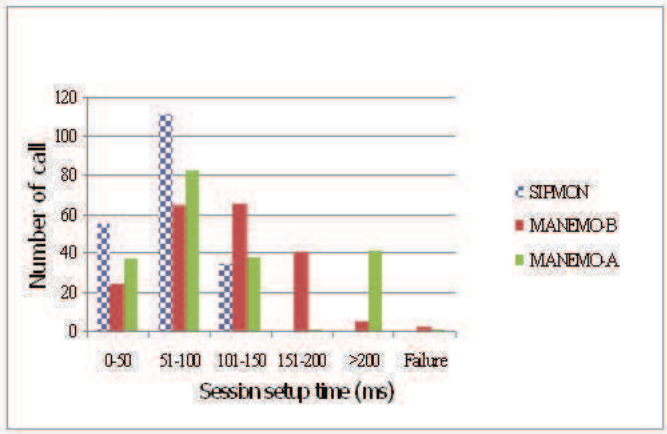
Fig. 6.4(a) and Fig. 6.4(b) show the post dialing distribution for static cases among the three systems. It can be clearly observed that SIPMON+ on OLSR takes the least time to establish the session. The most number of successful session establishment times for SIPMON+ are within 50-100 ms where MANEMO-A and MANEMO-B took 50-150 ms. There is no call failure in static cases for all three systems.

For single node/group mobility, the end terminal changes its point of attachment twice. While MNN1 sends an INVITE request to HQ and moves towards HQ, MNN1 can not send the INVITE request to HQ due to the absence of a route to the host at MR4. Consequently, the call drop is higher in this case for MANEMO systems. SIPMON+ has no call drop in this single movement as shown in Fig. 6.5(a) and Fig. 6.5(b).

For multiple movements, the communicating terminal changes its point of attachment only one time. MANEMO-A drops the call that MNN1 initiates the session to HQ, but for the opposite direction there is no call drop as shown in Fig. 6.6(a) and Fig. 6.6(b).



(a) Distribution of post dialing delay for single movement case (MNN calls HQ)

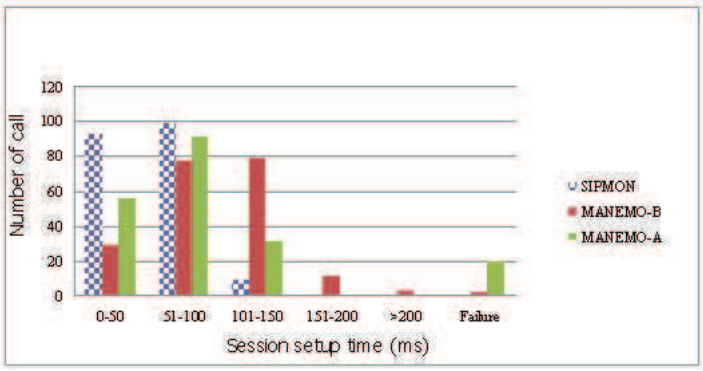


(b) Distribution of post dialing delay for single movement case (HQ calls MNN)

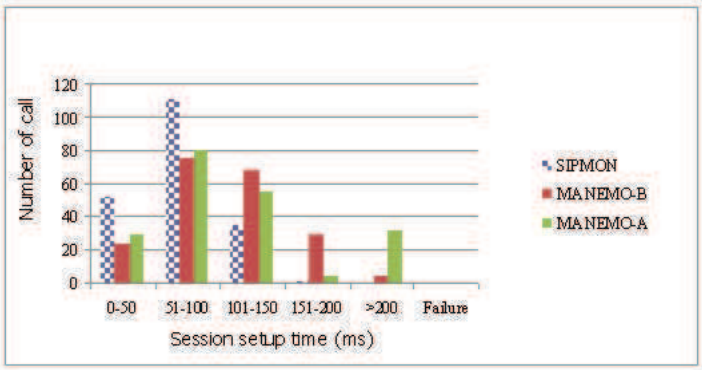
Figure 6.5: Post dialing delay for single movement case

There is no call drop in SIPMON+ for all moving patterns. On the other hand, in MANEMO-A and MANEMO-B, we can see from Fig. 6.7(a) and Fig. 6.7(b) that most of the calls are dropped during the movement as the node changes its point of attachment. The number of call drops is higher while the call is made from MNN to HQ in the case of MANEMO-A as shown in Fig. 6.7(a). This is because during the handover, there is no route to HQ. Hence, all calls made during this time are immediately dropped without a SIP INVITE retries due to the reason that there is no route to the host while the number of call failures (HQ to MNN) is fewer than MNN to HQ because HQ still does not receive binding updates from the MR during the movement. Therefore, it uses the previous care-of-address to communicate with the MR. All SIP INVITE requests are dropped during this handover. However, once the handover is finished, SIP INVITE requests can reach the MNN without dropping the call.

Fig. 6.8(a) and Fig. 6.8(b) show the comparison of average post dialing delay among the three systems. We have made 200 calls for each system for each type of movement patterns. The drop calls are considered for the duration of 32 s as specified in RFC 3261 (Rosenberg et al., 2002). It is observed from Fig. 6.8(a) and Fig. 6.8(b) that the



(a) Distribution of post dialing delay for multiple movement case (MNN calls HQ)



(b) Distribution of post dialing delay for multiple movement case (HQ calls MNN)

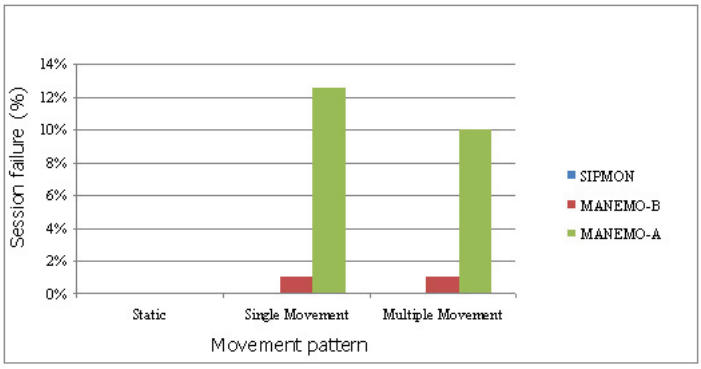
Figure 6.6: Post dialing delay for multiple movement case

post dialing delay for SIPMON+ is steady during the whole period of experiment. The average time for post dialing is much higher in MANEMO-A due to call drop and longer handover time.

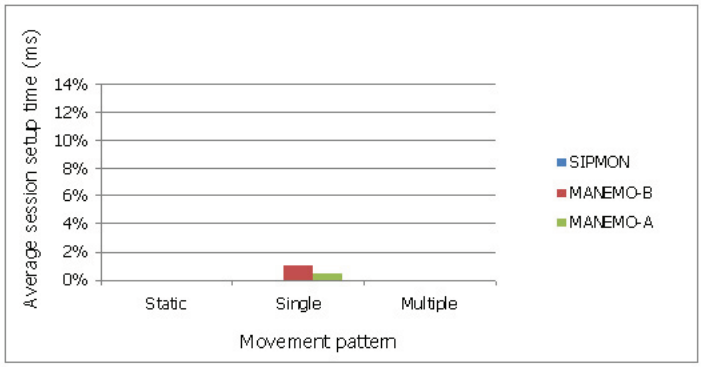
b) Control overhead

Control overhead is a major comparison issue between different protocols. In our systems both OLSR and TDP/NINA are proactive routing protocols but OLSR is a link-state routing protocol while TDP/NINA is a distance vector routing protocol and their bandwidth consumption is not equal. In SIPMON+, OLSR control messages are TC, HELLO and the overlay messages, MANEMO-B uses TC, HELLO and HNA messages as the control messages and MANEMO-A uses RA (Router Advertisement) with Tree Information Option (TIO) and NA (Neighbor Advertisement) with NINA as the control messages.

SIPMON+ on OLSR uses IPv4 while the other two systems use IPv6. But we have compensated these by adding additional bytes to the control packets of SIPMON+ accordingly to make it comparable to IPv6.



(a) Session failure when MNN call HQ



(b) Session failure when HQ call MNN

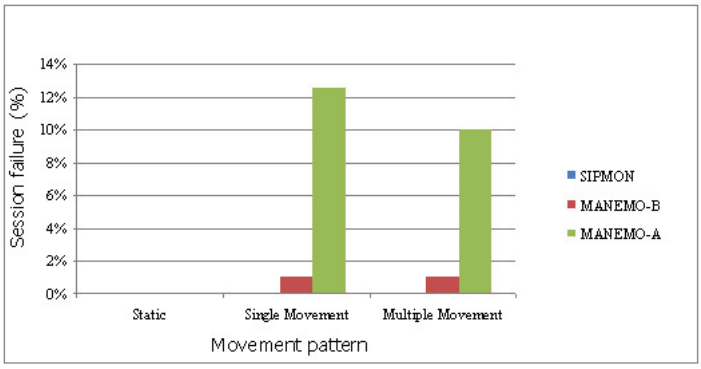
Figure 6.7: Session failure (%) for static and movement patterns

From the experimental results as shown in Fig. 6.9, it is observed that there is not much variation in control overhead with respect to different movement patterns as all these routing protocols are proactive routing protocols. The difference in sending control packets between the three systems is negligible (less than 200 bytes/sec). In terms of bandwidth consumption (bytes/sec), SIPMON+ consumes more bandwidth than the other two systems.

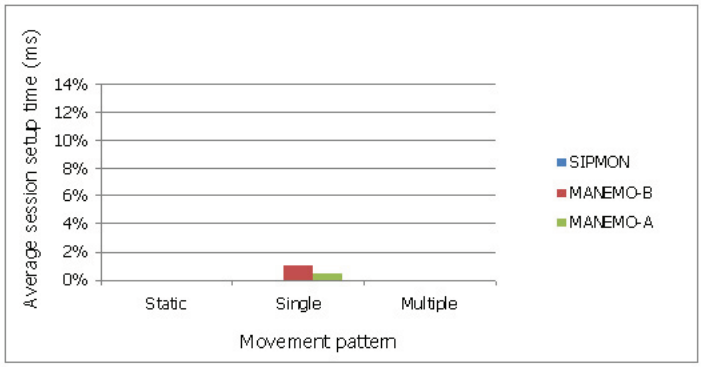
c) Packet loss

RTP packet loss is the number of data packets that are lost during an ongoing session. We capture the data using the packet sniffer (MG, 2009) at both ends (MNN and HQ) during an ongoing session and measure the packet loss in percentages.

From the observed data as shown in Fig. 6.10, it is found that there is no packet loss in SIPMON+ regardless of moving patterns. But for both MANEMO-A and MANEMO-B, in static cases, they perform their best as there is no movement. The highest packets are dropped for MANEMO-A as during movement it first de-associates from the previous current AP and associates with the new AP. During this time, large numbers of RTP packets are dropped.



(a) Average post dialing delay when MNN call HQ



(b) Average post dialing delay when HQ call MNN

Figure 6.8: Average post dialing delay for static and movement patterns

d) Mobility pattern

According to our current prototype of SIPMON+, each node has to be running SIPMON+ software and move individually. For group mobility, a number of nodes can move individually and may make multi-hops among them, which is an inefficient group movement. But MANEMO group mobility is supported by MR and the underlying nodes do not need to be aware of this movement.

Each MR has to registrar its Care of Address (CoA) with its Home Agent. Whenever any MR changes its point of attachment, it sends a Binding Update to and receives Binding Acknowledgement from the Home Agent. Thus a bidirectional tunnel is established between the Home Agent and the respective MR. During such movements, this binding process takes more than 2 sec with respect to SIPMON+ (less than 2 sec).

e) Scalability

Both SIPMON+ and MANEMO are able to scale to a large network. In SIPMON+, each node acts as an additional hop to the destination, so adding more nodes increases the

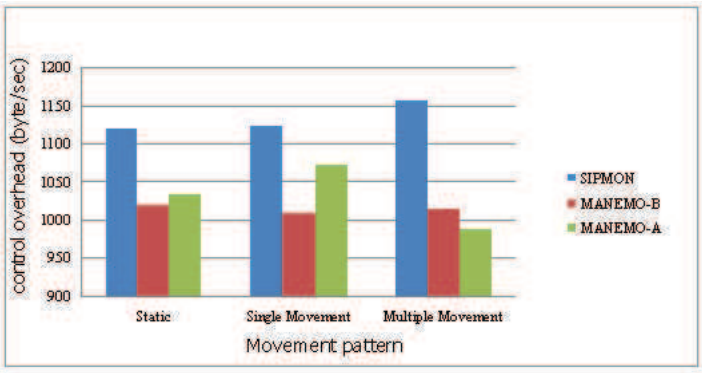


Figure 6.9: Control overhead comparison

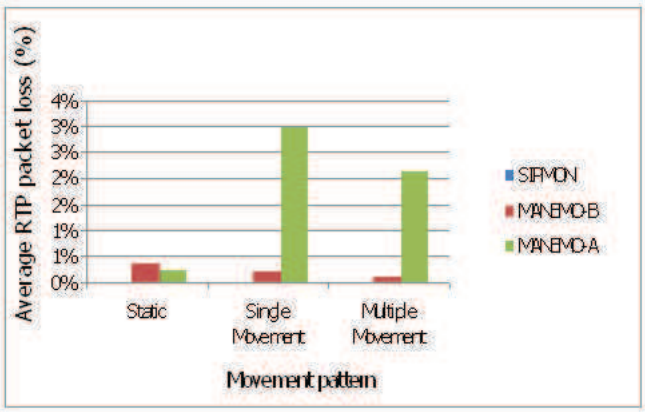


Figure 6.10: RTP packet loss comparison

Table 6.1: Simulation outputs for SIPMON+ on OLSR

Moving pattern	PDD MNN2HQ/HQ2MNN (ms)	Control overhead (byte/sec)	RTP packet loss (%)
Static	36 / 49	1013	0
Single	25 / 46	867	0
Five	33 / 42	912	0

probability to add more hops to reach the destination. But in MANEMO, adding MR increases the number of hops, and one MR can support a large number of nodes without increasing the hop number. In that respect, MANEMO is more scalable than SIPMON+.

f) Deployment issue

In an emergency situation, the communication network deployment time is a vital issue. The more time it takes, the more delay it causes for the rescue worker to start the rescue operation and the more causalities occur. In MANEMO, the MR must be installed in the vehicle first and the Home Agent (HA) must setup on the Internet. As a result, it takes a long period of time with respect to SIPMON+ as it does not need to setup any other equipment other than the SIPMON+ nodes. Any node that runs SIPMON+ software can join the network without informing any other nodes. But in MANEMO, the Home Agent must be configured with the MR's network prefix prior to using it in the network.

g) Simulation output

To confirm our testbed output, we perform a simulation of SIPMON+ on OLSR with the same testbed topology with 6 SIPMON+ nodes. The average call setup delay is around 1.5 to 2 times lower than that of our testbed because Linphone adds extra delays due to its internal process. The number of control overheads of the simulation results is a little bit lower than the testbed control overhead. For packet loss, the simulation results show no packet loss similar to the testbed results. Due to a lack of simulation source code availability, we could not simulate MANEMO-A and MANEMO-B.

6.2.1 An integration of SIPMON+ and MANEMO

In the testbed, SIPMON+ performs best in terms of session setup time and deployment, but it does not support efficient group mobility. For this reason, we propose the implementation of SIPMON+ on MRs of MANEMO OLSR to overcome the shortcomings of group mobility of SMON on flat OLSR.

We show the performance of SIPMON+ on each MR of OLSR MANENO in a large network of 30 nodes through simulation as shown in Fig. 6.11. We used CBR traffic of packet size 512 bytes and 20 Kbyte rate from the last MNN1 to HQ and HQ to MNN1. For the

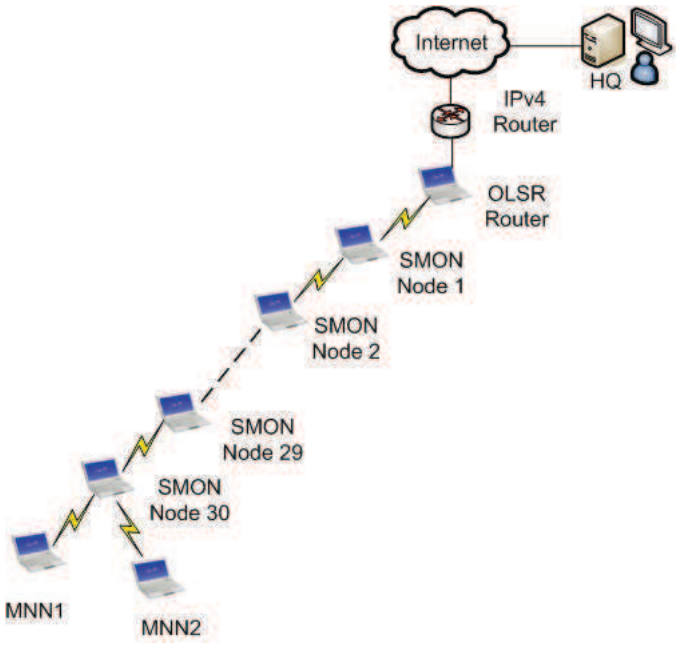


Figure 6.11: SIPMON+ on OLSR in a linear of 30 nodes

post dialing delay, we considered about 15 ms in additional delay for communications between MNN to SIPMON+ Node 30 and the Internet part. We ran the simulation for 1300 s with 220 session establishment and calculated the average of five sets of simulation output. The control messages are HELLO, TC, HNA and the overlay messages.

Movement Pattern

a) Static: For the static cases, we strictly maintained 30 linear hop distance from the source to the destination during the test.

b) Single Group: For single movement cases, we start communication from MNN1 to HQ and move SIPMON+ node 30 toward HQ while maintaining the connection between MNN1 and SIPMON+ node 30. We stop further movement when SIPMON+ node 30 connects to SIPMON+ node 1, and then we analyzed the trace file for measurements.

c) Group of Five: In this case, we start from the static case and communicate from MNN1 to HQ. We start moving lower 5 SIPMON+ nodes toward HQ while they are connected linearly. When SIPMON+ node 26 connects with SIPMON+ node 1, we stop moving and measure the data.

d) We did the same experiment as described above with Group of 10, 15, 20, 22, 25 and 28 SIPMON+ nodes.

Table 6.2: Simulation outputs (Call setup between MNN to HQ)

Moving pattern	PDD MNN2HQ/HQ2MNN (ms)	Control overhead (byte/sec)	RTP packet loss (%)
Static	854 / 901	14863	5.76
Single	168 / 416	14187	1.55
Five	722 / 302	14204	1.64
Ten	372 / 429	14201	1.77
Fifteen	555 / 429	14163	2.06
Twenty	392 / 556	14206	2.65
Twenty-two	685 / 473	14159	2.71
Twenty-five	609 / 569	14180	3.80
Twenty-eight	849 / 781	14160	4.23

e) All the above mentioned movement patterns are repeated while we start communications from HQ to MNN1.

From the simulation output in Table 6.2, it is observed there is no fixed pattern in performance. There are two impacts during movement. As for movement, the node change its point of attachment, and at the same time the hop count to the destination decreases. Due to the change of the point of attachment, some packets are dropped and post dialing is delayed, but after handover the post dialing delay decreases as the hop count to the destination decreases. Most call drops originate from a routing instability, causing no path to the destination. The instability results from a frame collision at the data link layer. This routing instability occurs even in static environments. When an OLSR node does not receive control messages, such as HELLO or TC, from its neighbors within a time specified in the holding timer, it removes these neighbors from the routing table which causes routing changes. All calls are dropped during this period.

6.3 Discussions

Based on the demonstration, EasyDC can be easily set up and quickly deployed for emergency networks in a simulated post-disaster recovery scenario (Kanchanasut et al., 2007, 2008). In addition, we perform performance comparisons between SIPMON+ and two MANEMOs based on OLSR and TDP/NINA in terms of post dial delay, control overhead, and RTP packet loss.

The linear network of six-hop topology was used in the testbed scenarios as shown in Fig. 6.2 and Fig. 6.3. The measurements were taken from calling results between MMN1 on MANET and HQ located in the infrastructured network. SIPMON+ had the lowest post dialing delay on average for the static and moving scenarios as shown in Table 6.3. The average post dialing delay of MANEMO-A and MANEMO-B was higher, especially in moving scenarios due to the higher handoff delay.

SIPMON+ control overhead was the highest among the protocols because of its overlay

Table 6.3: The average post dialing delay comparison of three systems for all testbed scenarios according to Fig. 6.8

Moving pattern	SIPMON+	MANEMO-A	MANEMO-B
Static	61 ms	78 ms	95 ms
Single movement	55 ms	4067 ms	413 ms
Multiple movement	55 ms	3267 ms	440 ms

messages added to the network. MANEMO-A had the lowest control overhead because its routing messages were exchanged between hop-by-hop neighbors whereas OLSR routing messages were forwarded to the entire network. Tazaki et al. (2009) compared the control overhead of OLSR and TDP/NINA MANEMO in field experiments (disaster situation). Their results confirmed that TDP/NINA outperformed OLSR MANEMO in the case of spare networks in terms of control overhead, but OLSR MANEMO's performance was better in the case of dense networks.

The average RTP packet loss was the percentage of data packets that was lost during calls between MMN1 and HQ. The highest packet loss was found in MANEMO-A in the case of the moving scenarios. A large number of packet drops occurred during deassociation and association from a current access point to a new access point while MR was moving. The average RTP packet losses of MANEMO-A, MANEMO-B, and SIPMON+ were 3.41%, 0.38% and 0% respectively. The results of RTP packet losses could imply the number of call failures as calls was made during this time. As a result, a number of call-setup failure ratios of MANEMO-A, MANEMO-B, and SIPMON+ are 3.83%, 0.5%, and 0% respectively.

The testbed results have confirmed that SIPMON+ is suitable for emergency networks, where an IP telephony is used for communication purposes. If IP telephony is the main application in such a network, SIPMON+ on OLSR MANEMO outperforms pure MANEMO; however, with applications where quality of data transmission is of importance, then pure MANEMO provides less packet loss (Tazaki et al., 2009).

Chapter 7

Conclusions and Discussions

We have proposed the SIP framework for MANET, which can tolerate single-point and multiple-point node failures and can support terminal mobility. Our framework demonstrates an efficient VoIP infrastructure for emergency networks. This chapter provides a summary of the four main topics of this dissertation, namely SMON: a structured overlay network, P2P SIP on SMON, an extension of SMON for heterogeneous network with the Internet and MANET, and a prototype multimedia application on P2P SIP on SMON+. Results of each topic are included followed by a discussion on future work.

7.1 Conclusions

This dissertation focuses on how to deploy SIP on MANET is such a way that applications such as IP telephony can be effectively provided. SIP-based applications on MANET are potentially useful as an alternative means of communication where a traditional communication infrastructure is not present such as in rural areas and especially in post-disaster scenarios. This is because mobile ad hoc networks can be set up rapidly and its topology can be adapted to function even in difficult terrains. In order to provide effective means of communication among mobile communities, it is necessary to ensure that terminal mobility is supported because nodes on the network can be vehicles such as ambulances, cars, motor cycles or just rescuers on the move. Terminal mobility ensures that an ongoing communication is not disrupted while these nodes are moving, particularly in post-disaster recovery operations. Once the Internet connectivity becomes available to these temporary networks, these mobile nodes should be able to seamlessly communicate with nodes in the infrastructured networks thus our proposed framework will have to accommodate stand alone MANET as well as an integration of MANETs and the Internet.

In order to address IP telephony application on MANET, the effectiveness of terminal mobility handling is our main focus; hence, we adopted post dialing delay, call success and failure ratios, handoff delay, RTP packet loss ratio, and control overhead as the basis for comparing our work against other schemes.

Our work can be summarized as follows:

1) Structured Overlay SMON on MANET

Traditionally, SIP is a client and server application protocol which would not be able to sustain the dynamic linkages on MANET. We propose a new framework called SIPMON or P2P SIP on MANET whose main focus is on providing terminal mobility for IP telephony applications in emergency networks. A novel structured overlay network, SMON, a mesh network, based on DHT provides an underlying structure for user discovery for the upper layer SIPMON. SMON is an improvement on an earlier work called CrossROAD

by Delmastro (2005). Objects can be stored and searched on CrossROAD using DHT. To optimize its overlay network, CrossROAD uses a cross-layer approach by embedding its overlay messages inside OLSR routing messages and directly accessing OLSR routing table for topological information in order to keep the overlay network up-to-date. Like CrossROAD, SMON is a cross-layer overlay network, which updates its overlay structure whenever there are any topological changes detectable from the underlying OLSR routing protocol. In order to avoid backward incompatibility with normal OLSR, SMON creates its own messages but it uses OLSR optimized information dissemination through MPRs. Having used the DHT, CrossROAD can look up a desired item using a constant time. However, CrossROAD uses the pure flooding technique for node discovery where each node sends a PublishService to discover others, which creates a very high control overhead as the number of CrossROAD nodes increases. In SMON, pure flooding for node discovery has been replaced by allowing only primary peers to advertise a LIST OF ALL MEMBERS message which helps reduce the control overhead significantly as shown in Chapter 3. Node discovery is used for the formation and maintenance of an overlay network with mobility. This represents a reduction of 95% in control overhead with the same query success ratio and query delay as CrossROAD. This is achievable as shown through simulations with 60 nodes. We also provide merge and split operations of two or more SMONs which could occur when the network gets partitioned. In addition, SMON seamlessly operates with normal OLSR nodes because we use OLSR message extension, which offers backward compatibility with the OLSR standard.

2)SIPMON: P2P SIP on SMON

On top of SMON, a P2P SIP, referred to as SIPMON, is implemented in order to provide SIP registration and user discovery in a non-centralized manner. On each member of SMON, SIPMON provides a small traditional SIP registrar and proxy server, which can accept and process SIP requests from other SMON nodes for a SIP based application. Like in CrossROAD, each node is assigned a unique identification and each node is responsible for maintaining the application objects whose object ID is the numerically closest to it node ID; SIPMON stores SIP binding record as objects in SMON nodes. SMON is responsible for maintaining node IDs of peers on the overlay network, while P2P SIP stores and maintains SIP objects, (SIP URIs, IP addresses), on SMON node. Each P2P SIP keeps the SIP objects whose IDs or hashed value of this SIP URIs, are the numeric closest to its nodes ID. We define SIP registration, call setup, and binding update procedures for SIPMON.

In Chapter 4, we compare SIPMON with previous works with SIP on OLSR which include the CQSA scheme (L. Li & Lamont, 2004), dSIP (Leggio et al., 2005), and MANET-Sip (Fudickar et al., 2009).

> ***Comparison with CQSA.*** The CQSA (L. Li & Lamont, 2004) scheme uses the Service Location Extension (SLE) message, an OLSR message extension, to broadcast SIP requests via MPRs. When a SIP user, a caller, wishes to make a call, it must perform user location discovery by broadcasting SLE containing an INVITE request addressed to a target SIP URI. Upon receiving the request, every node compares its SIP URI and the one in the INVITE request. Only the target SIP endpoint responds to the caller with its IP address. When the number of calls increases, the

number of control overhead in CQSA scheme also increases. On the other hand, the number of control overhead messages in SIPMON is relatively steady regardless of the number of calls. In our testbed, the CQSA control overhead is 6.10% (40 calls) and 12.45% (80 calls) compared with OLSR control overhead. The control overhead of SIPMON is between 8.57% (40 calls) and 8.73% (80 calls). In our experimental testbed, we compared the post dialing delay between our approach and the CQSA scheme, and it was found that the post dialing delay of SIPMON is five times lower than that of the CQSA scheme.

Comparison with dSIP. dSIP (Leggio et al., 2005) uses the pure broadcast technique to distribute SIP REGISTER requests to the whole network. A node in dSIP periodically advertises SIP REGISTER requests containing its SIP URI along with its IP address or a binding. Upon receiving the REGISTER request, the rest of the network keeps this binding with its time-to-live in its cache for possible use in the future until the valid period expires. When the binding is invalid in the cache, the caller has to wait for the next binding advertisement in order to extract the callee's up-to-date IP address, which can be used to initiate a call. We show by testbed that our SIPMON reduces the number of control overhead significantly to 90% as compared with dSIP's results. The post dialing delay of SIPMON is less than 30 ms, while the post dialing delay of dSIP is 2.5 s on average when the advertised interval of REGISTER request is set to 5 s.

Comparison with MANETSip. MANETSip (Fudickar et al., 2009) uses a multicast routing protocol over OLSR to distribute a SIP request. Every node on a multicast network is a SIP registrar, referred as a SIP multicast network. A node periodically sends a SIP REGISTER request to all nodes on the multicast network in order to register itself on the SIP multicast network. The authors claim that using the multicast network can reduce the number of exchanged SIP requests, but they do not provide any control overhead measurement. However, the registration delay was given in their MANETSip testbed setup with a static linear network topology of three computers. To register a node at another node with a two-hop distance delay, it takes 164 ms on average on MANETSip. Our testbeds based on a linear network topology of four computers (three-hop end-to-end nodes) showed that the delay was 14 ms, significant lower as compared with that of MANETSip.

3) P2P SIP on SMON+

MANET can be attached to an infrastructure; thus, SIPMON has to be able to operate within heterogeneous environment. We extend SMON to SMON+ to cover both MANETs and infrastructured networks to provide seamless SIP service through a single SIPMON. The single overlay architecture allows a number of mobile nodes to move between MANETs and the Internet while maintaining ongoing call sessions, or session continuity. We do not propose the integration of SIPMON with existing SIP on the Internet because it will require a SIP gateway to translate SIPMON protocol into the Internet SIP, and vice versa. Configuring a SIP gateway for each MANET adds complexity to the setting up of an emergency network and introduces a single point of failure problem. In SIPMON+, there is no need for a special SIP gateway. This single overlay architecture

reduces the complexity of setting up an emergency network and prevents a single point of failure problem.

To extend SMON to cover the Internet, eligible members on the Internet must understand OLSR. We introduce a novel Overlay OLSR Network (OON) turning normal IP nodes of the Internet to OLSR nodes that can communicate with other OLSR nodes on MANETs. Once done, SMON can be applied on top of OON, called SMON+. We do not assume broadcast capability for the Internet. Therefore, we use several unicast communications to carry OLSR messages exchanged among nodes on the Internet, e.g. desired headquarter nodes and MANET gateways. However, as our scenario described in Chapter 5, we expect only a few headquarter nodes and a few MANET gateways to present on the Internet, resulting in an acceptable volume of unicast traffic.

To the best of our knowledge, SIPMON on SMON+, or SIPMON+, is the first SIP overlay network designed to provide terminal mobility support for both MANET and the Internet users. We consider two SIP terminal mobilities: pre-call and mid-call mobility. The pre-call mobility is the binding update process, which allows MN to be reachable while moving to another subnet. The mid-call mobility handles handoff between CN and MN to resume an ongoing session after MN moves to a new subnet.

SIP terminal mobility has been demonstrated to be appropriate for VoIP communications with Real-time Transport Protocol (RTP/UDP) with low handoff delay when compared to mobile IP (Zeadally & Siddiqui, 2007). However, terminal mobility provided by SIP is not suitable for TCP-based applications as it is difficult for a SIP application to maintain TCP connections when moving across subnets. This is also the limitation of SIPMON+ since SIPMON+ is a SIP-based protocol.

We target the emergency network application where the linear network topology is focused. The linear network topology represents an extreme case where rescuers line up to sweep the affected-disaster area while maintaining communication from the farthest end to the command headquarters. SIPMON+ performs better in terms of mid-call handoff delay. However, we do not claim that the results can be applied to general cases.

SIPHoc (Stuedi et al., 2007) provides SIP interoperability between MANET and Internet but only for users in a static network environment. Therefore, to evaluate SIPMON+, we have to compare its performance with those network with network mobility where mobile ad hoc networks get connected to the Internet such as MANEMO which is an integration of MANET and Network Mobility (Wakikawa et al., 2007) using IPv6, with MIP6-MANET (Y. S. Chen et al., 2006) using mobile IPv6 and finally with SIPHoc for static case.

Comparisons with other MANET and infrastructure approaches are summarized below.

Comparison with MANEMO. SIPMON+ outperforms MANEMO in terms of mid-call mobility. The control overhead of SMON+ is, however, higher than that of MANEMO's. However, we later show by analysis in Chapter 5 that the control overhead of SIPMON+ is acceptable for fixed IP networks.

Comparison with Mobile Ipv6-MANET. Besides MANEMO, we compare

handoff performance between our approach with MIP6-MANET. Similar to MANEMO, MIP6-MANET uses mobile IPv6 mechanism to handle terminal mobility. In MIP6-MANET, the authors define handoff jitter as a summation of the handoff delay which includes both pre-calls and mid-calls. The handoff jitter when the MN moves from one network to another is 5000ms whereas SIPMON+ uses 1055 ms and 32 ms to complete the handoffs of the pre-call and the mid call mobility respectively. Thus SIPMON outperforms MIP6-MANET in terms of handoff jitter.

Comparison with SIPHoc. Although SIPHoc does not explicitly address the terminal mobility problem, it provides SIP interoperability between MANET and Internet. Since SIPHoc provides SIP functions for users on static networks, we cannot compare the handoff delay between our approach and SIPHoc's. However, with the same physical network topology in static environments, the post dialing delay of SIPMON+ is 3.5 times lower than those of SIPHoc.

4) SIPMON+ Prototypes and Testbeds As a proof of concept, we developed a prototype multimedia application called Easy Disaster Communication (EasyDC) which runs on SIPMON+. We successfully demonstrated that our framework could be easily set up and deployed using a simulated post-disaster recovery scenario (Kanchanasut et al., 2007, 2008).

In (Arefin et al., 2009), we conducted performance comparison between our approach and MANENO's for post dial delay, control overhead, and RTP packet loss. Two types of MANEMO networks for our comparison were MANEMO with OLSR only where each mobile router of MR is running OLSR and MANEMO with TD/NINA Tazaki et al. (2009). We concentrated on group mobility performance between SIPMON+ and MANEMO by using different testbed scenarios such as static, single node movement, and multiple nodes movement. We considered post dialing delay, RTP packet loss, and percentage of call-setup failures were the most important aspects for our systems using six-hop network topology. Measurements were taken from calling results between a mobile node (MN) on MANET and another fixed node located (HQ) on the infrastructured network.

As discussed in Chapter 6, it was found that SIPMON had the lowest post dialing delay on average for all the scenarios though SIPMON control overhead was the highest among the protocols because of its overlay messages added to the network. TDP/NINA MANEMO had the lowest control overhead because its routing messages were exchanged between hop-by-hop neighbors, whereas OLSR routing messages were forwarded to the entire network. Tazaki et al. (2009) compared the control overhead of OLSR and TDP/NINA MANEMO in field experiments with simulated disaster situation. The experiments confirmed that TDP/NINA could outperform OLSR MANEMO in the cases of spare networks in terms of control overhead, but MANEMO with OLSR performance was better in the case of dense networks.

The average RTP packet loss was the percentage of data packets that were lost during calls between MN and HQ. The highest packet loss was found in TDP/NINA MANEMO in the case of moving scenarios. Large numbers of packet drops occurred during de-association and association from a current access point to a new access point while a mobile router (MR) was moving. The average RTP packet loss of TDP/NINA MANEMO,

OLSR MANEMO, and SIPMON were 3.41%, 0.38% and 0% respectively. The results of RTP packet loss could imply that the number of call failures as calls were made during this time. As a result, the percentage of call failures of TDP/NINA MANEMO, OLSR MANEMO, and SIPMON+ are 3.83%, 0.5%, and 0% respectively.

SIPMON+ outperforms TDP/NINA and OLSR MANEMO in terms of the post dialing delay, the RTP packet loss, and the percentage of call failures, but OLSR and TDP/NINA MANEMO can serve a higher number of mobile network nodes. This is because one MR in MANENO can act as a router for a group of nodes. Therefore, we propose the implementation of SIPMON+ on each MR of OLSR MANENO in order to provide support for a group of nodes without using a home agent. In Chapter 6, we evaluate the performance of 31 SIPMON+ nodes in different moving scenarios using simulation. The post dialing delay is 559 ms, regarded as an agreeable value for a call setup delay. The percentage of RTP packet loss is only 2.9%. The control overhead is 14258 bytes/second, which is not too high for 31 SIPMON+ nodes. From this simulation result, we confirm that SIPMON+ on OLSR MANEMO with a group mobility support is suitable for emergency networks.

7.2 Discussions

Though SIPMON effectively provides users with location lookup services, it does not provide a presence service. Presence is used to indicate the status of SIP users, such as online or offline in real-time similar to a buddy list of Instant Messaging (IM) applications. Since the presence service uses SIP requests to maintain users' status, we propose that SIPMON be extended to handle SIP presence for MANET users as well by adding the presence server functionalities on SIPMON.

Though our primary interest was to provide session continuity for mobile users of an IP telephony service in emergency networks, our proposed framework can be readily applied to day-to-day expansion of the Internet connectivity. OLSR can be used as edge networks for such purpose and SIPMON+ can be a platform for telephony for mobile users. In order to provide TCP-oriented applications, future work can address how TCP-based applications are provided on SIPMON+ as well as other mobility issues, such as session mobility.

References

Arefin, K. R., Wongsaardsakul, T., & Kanchanasut, K. (2009). Vehicle-to-infrastructure manet with group mobility for emergency multimedia communication. In *Aintec 2009: Proceedings of the 5th asian conference on internet engineering.* ACM.

Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., & Stoica, I. (2003). Looking up data in p2p systems. *Communications of the ACM*, *46*(2), 43–48.

Banerjee, N., & Acharya, A. (2004). Peer-to-peer sip-based services over wireless ad hoc networks. In *The 1st annual international conference on broadband networks.*

Boer, M. (2008). *Twinkle - sip softphone for linux.* (available online at http://www.twinklephone.com)

Boudjit, S., Laouiti, A., Muhlethaler, P., & Adjih, C. (2005). Duplicate address detection and autoconfiguration in olsr. In *Snpd-sawn '05: Proceedings of the sixth international conference on software engineering, artificial intelligence, networking and parallel/distributed computing and first acis international workshop on self-assembling wireless networks* (pp. 403–410). Washington, DC, USA: IEEE Computer Society.

Castro, M. C., & Kassler, A. J. (2006). Optimizing sip service provisioning in internet connected manets. In *International conference on software in telecommunications and computer networks* (pp. 86–90).

Cerf, V., Dalal, Y., & Sunshine, C.(1974). *Specification of Internet Transmission Control Program* (No. 675). RFC 675. IETF.

Chakraborty, D., Joshi, A., Yesha, Y., & Finin, T. (2006). Gsd: A novel group-based service discovery protocol for manets. In *Proceedings of the 20th international conference on advanced information networking and applications.*

Chen, T.-W., & Gerla, M.(1998). Global state routing: A new routing scheme for ad-hoc wireless networks. In *Ieee international communications conference* (pp. 171–175). IEEE.

Chen, Y. S., Yang, Y. H., & Hwang, R. H. (2006). Sip-based mip6-manet: Design and implementation of mobile ipv6 and sip-based mobile ad hoc networks. *Computer Communications*, *29*(8), 1226–1240.

Clausen, T., & Baccelli, E. (2005). A simple address autoconfiguration mechanism for olsr. In *Ieee international symposium on circuits and systems, 2005. iscas 2005* (pp. 2971–2974).

Clausen, T., & Jacquet, P. (2003). *Optimized Link State Routing Protocol (OLSR)* (No. 3626). RFC 3626 (Experimental). IETF.

Corson, M. S., & Ephremides, A. (1995). A distributed routing algorithm for mobile wireless networks. In *Wireless networks* (Vol. 1, pp. 61–81). Kluwer Academic

Publishers.

Corson, M. S., & Macker, J. (1999). *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations* (No. 2501). RFC 2501 (Informational). IETF.

Delmastro, F. (2005). From pastry to crossroad: Cross-layer ring overlay for ad hoc networks. In *Third ieee international conference on pervasive computing and communications workshops 2005* (pp. 60–64).

drscholl. (2000). *Napster protocl specification – opennap: Open source napster server.* (available online at `http://opennap.sourceforge.net/napster.txt`)

Dube, R., Rais, C. D., Wang, K.-Y., & Tripathi, S. K. (1997). Signal stability-based adaptive routing (ssa) for ad hoc mobile networks. In *Ieee personal communications magazine* (pp. 36–45).

Dutta, A., Madhani, S., Chen, W., Altintas, O., & Schulzrinne, H. (2004). Fast-handoff schemes for application layer mobility management. In *Pimrc 2004.*

Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., & Ylonen, T. (1999). *SPKI Certificate Theory* (No. 2693). RFC 2693 (Experimental). IETF.

Fu, C., Glitho, R. H., & Dssouli, R. (2005). A novel signaling system for multiparty sessions in peer-to-peer ad hoc networks. In *Ieee wireless communications and networking conference.*

Fu, C., Glitho, R. H., & Khendek, F. (2006). Signaling for conferencing in integrated 3g/mobile ad hoc networks. In *Proceeding of the 11th ieee symposium on computers and communications.*

Fudickar, S., Rebensburg, K., & Schnor, B. (2009). Manetsip – a dependable sip overlay network for manet including presentity service. In *Fifth international conference on networking and services* (pp. 314–319).

Gafni, E. M., Member, S., Dimitri, Bertsekas, P., & Member, S. (1981). Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, *29*, 11–18.

Ghodsi, A. (2006). *Distributed k–ary system: Algorithms for distributed hash tables.* Unpublished doctoral dissertation, School of Information and Communication Technology, Department of Electronic, Computer, and Software Systems, Stockholm, Sweden.

Haas, Z. J., Pearlman, M. R., & Samar, P. (2002). *The zone routing protocol (zrp) for ad hoc networks.* draft-ietf-manet-zone-zrp-04.txt (work in progress).

Handley, M., & Jacobson, V. (1998). *SDP: Session Description Protocol* (No. 2327). RFC 2327 (Proposed Standard). IETF.

Handley, M., Schulzrinne, H., Schooler, E., & Rosenberg, J. (1999). *SIP: Session Initiation*

Protocol (No. 2543). RFC 2543 (Proposed Standard). IETF.

Helal, S., Desai, N., verma, V., & Lee, C.(2003). Konark – a service discovery and delivery protocol for ad-hoc networks. In *Proceedings of the third ieee conference on wireless communication networks.*

Jiang, M., Li, J., & Tay, Y.(1999). *Cluster based routing protocol(cbrp).* draft-ietf-manet-cbrp-spec-01.txt.

Jodra, J. L., Vara, M., Ma Cabero, J., & Bagazgoitia, J.(2006). Service discovery mechanism over olsr for mobile ad-hoc networks. In *Aina '06: Proceedings of the 20th international conference on advanced information networking and applications* (pp. 534–542). Washington, DC, USA: IEEE Computer Society.

Johnson, D., Hu, Y., & Maltz, D.(2007). *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4* (No. 4728). RFC 4728 (Experimental). IETF.

Johnson, D., Perkins, C., & Arkko, J.(2004). *Mobility Support in IPv6* (No. 3775). RFC 3775 (Proposed Standard). IETF.

Jung, J.-W., Mudumbai, R., Montgomery, D., & Kahng, H.-K.(2003). Performance evaluation of two layered mobility management using mobile ip and session initiation protocol. In *Ieee globecom.*

Kanchanasut, K., Tunpan, A., Awal, M. A., Das, D. K., Wongsaardsakul, T., & Tsuchimoto, Y.(2007). Dumbonet: a multimedia communication system for collaborative emergency response operations in disaster-affected areas. *International Journal of Emergency Management*, *4*(4), 670–681.

Kanchanasut, K., Tunpan, A., Awal, M. A., Das, D. K., Wongsaardsakul, T., & Tsuchimoto, Y.(2007). *A multimedia communication system for collaborative emergency response operation in disaster-affected area* (Tech. Rep. Nos. TR_2007–1). intERLab, Asian Institute of Technolog.

Kanchanasut, K., Wongsaardsakul, T., Chansutthirangkool, M., Laouiti, A., Tazaki, H., & Arefin, K. R. (2008). Dumbo ii: a v-2-i emergency network. In *Aintec 2008: Proceedings of the 4th asian conference on internet engineering* (pp. 37–38). New York, NY, USA: ACM.

Karger, D., Sherman, A., Berkheimer, A., Bogstad, B., Dhanidina, R., Iwamoto, K., et al.(1999). Web caching with consistent hashing. *Computer Networks*, *31*(11-16), 1203–1213.

Khlifi, H., Agarwal, A., & Grégoire, J.-C. (2003). A framework to use sip in ad-hoc networks. In *Canadian conference on electrical and computer engineering.*

Klein, M., Konig-Ries, B., & Obreiter, P. (2003a). *Lanes – a lightweight overlay for service discovery in mobile ad hoc networks* (Tech. Rep. No. 2003-6). University of Karlsruhe.

Klein, M., Konig-Ries, B., & Obreiter, P. (2003b). Service rings – a semantic overlay

for service discovery in ad hoc networks. In *Proceedings of the 14th international workshop on database and expert systems applications.*

Klingberg, T., & Manfredi, R.(2002). *Gnutella protocol development.* (available online at http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html)

Laouiti, A., Mhlethaler, P., Najid, A., & Plakoo, E. (2002). Simulation results of the olsr routing protocol for wireless network. In *1st mediterranean ad-hoc networks workshop (med-hoc-net).*

Laouiti, A., Qayyum, A., , & Viennot, L. (2002). Multipoint relaying: An effcient technique for flooding in mobile wireless networks. In *Proceeding of the hawaii intl.conf. on system sciences.*

Lau, G., Jaseemuddin, M., & Ravindran, G. (2005). Raon: A p2p network for manet. In *Second ifip international conference on wireless and optical communications networks.*

Lee, S. J., Su, W., & Gerla, M.(2001). On-demand multicast routing protocol in multihop wireless mobile networks. In *Acm/kluwer mobile networks and applications* (pp. 1298–1302).

Leggio, S., Manner, J., Hulkkonen, A., & Raatikainen, K. (2005). Session initiation protocol deployment in ad-hoc networks: a decentralized approach. In *Proceedings of the international workshop on wireless ad-hoc networks.*

Li, L., & Lamont, L. (2004). Service discovery for support of real-time multimedia sip applications over olsr manets. In *Olsr interop & workshop.*

Li, L., & Lamont, L. (2005). A lightweight service discovery mechanism for mobile ad hoc pervasive environment using cross-layer design. In *Proceedings of ieee percom workshops.*

Li, Z., Yin, X., Yao, P., & Huang, J.(2006). Implementation of p2p computing in design of manet routing protocol. In *Proceedings of the first international multi-symposiums on computer and computational sciences - volume 2 (imsccs'06)* (pp. 594–602). Washington, DC, USA: IEEE Computer Society.

Liang, J., Kumar, R., & Ross, K. W. (2004). The kazaa overlay: A measurement study. In *Proceedings of the 19th ieee annual computer communications workshop.*

linphone.org. (2007). *Linphone, an open-source sip video-phone for linux and windows.* (available online at http://www.linphone.org)

Manner, J., Leggio, S., & Raatikainen, K. (2006). An internet sip gateway for ad-hoc networks. In *The 3rd annual ieee communications society on sensor and ad hoc communications and networks* (Vol. 3, pp. 740–745).

Mase, K., & Adjih, C.(2006). *No Overhead Autoconfiguration OLSR.* draft-mase-manet-autoconf-noaolsr-01.txt (work in progress).

McCarthy, B., Edwards, C., & Dunmore, M. (2006). The integration of ad-hoc (manet) and mobile networking (nemo): Principles to support rescue team communication. In *Third international conference on mobile computing and ubiquitous networking.*

MG, L. (2009). *Tcpdump/libpcap.* (available online at `http://www.tcpdump.org`)

mjsip.org. (2006). *Mjsip.* (Dpt. of Information Engineering at University of Parma, available online at `http://www.mjsip.org`)

moblieip.jp. (2004). *Shisa.* (available online at `http://www.mobilip.jp`)

Mohanty, S., & Akyildiz, I. F. (2007). Performance analysis of handoff techniques based on mobile ip, tcp-migrate, and sip. *IEEE Transactions on Mobile Computing*, *6*(7), 731–747.

Morris, R., Jannotti, J., Kaashoek, F., Li, J., & Decouto, D. (2000). Carnet: a scalable ad hoc wireless network system. In *Ew 9: Proceedings of the 9th workshop on acm sigops european workshop* (pp. 61–65). New York, NY, USA: ACM.

Murthy, S., & Garcia-Luna-Aceves, J. J. (1995). A routing protocol for packet radio networks. In *The first international conference on mobile computing and networking* (pp. 86–95). ACM.

Nakajima, N., Dutta, A., Das, S., & Schulzrinne, H. (2003). Handoff delay analysis and measurement for sip based mobility in ipv6. In *Communications, 2003. icc '03. ieee international conference on* (Vol. 2, pp. 1085–1089).

Nguyen, D., & Minet, P. (2007). Analysis of mpr selection in the olsr protocol. In *Ainaw '07: Proceedings of the 21st international conference on advanced information networking and applications workshops* (pp. 887–892). Washington, DC, USA: IEEE Computer Society.

Nidd, M. (2001). Discovery in deapspace. In *Pieee personal communiations ieee personal communications* (pp. 39–45).

Ogier, R., Templin, F., & Lewis, M. (2004). *Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)* (No. 3684). RFC 3684 (Experimental). IETF.

Park, V. D., & Corson, M. S. (1997). A highly adaptive distributed routing algorithm for mobile wireless networks. In *Ieee conference on computer communications* (Vol. 3, pp. 1405–1413). IEEE.

Pei, G., Gerla, M., & Chen, T.-W. (2000). Fisheye state routing in mobile ad hoc networks. In *Icdcs workshop on wireless networks and mobile computing* (pp. 71–78).

Perkins, C. (1996). *IP Mobility Support* (No. 2002). RFC 2002 (Proposed Standard). IETF.

Perkins, C., Belding-Royer, E., & Das, S. (2003). *Ad hoc On-Demand Distance Vector (AODV) Routing* (No. 3561). RFC 3561 (Experimental). IETF.

Perkins, C., & Bhagwat, P. (1994). Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Acm conference on communications architectures, protocols and applications* (pp. 234–244). ACM.

Postel, J. (1981). *Internet Protocol* (No. 791). RFC 791 (Standard). IETF. (Updated by RFC 1349)

Pucha, H., Das, S., & Hu, Y. (2004). Ekta: An efficient dht substrate for distributed applications in mobile ad hoc networks. In *Proceedings of the 6th ieee workshop on mobile computing systems and applications (wmcsa 2004)* (pp. 163–173).

Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). A scalable content addressable network. In *Proceedings of acm sigcomm 2001.*

Ros, F. J. (2007). *Masimum um-olsr.* (available online at `http://masimum.inf.um.es/?Software:UM-OLSR`)

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., et al. (2002). *SIP: Session Initiation Protocol* (No. 3261). RFC 3261 (Proposed Standard). IETF.

Rowstron, A., & Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Ifip/acm international conference on distributed systems platforms (middleware)* (pp. 329–350).

Sailhan, F., & Issarny, V. (2005). Scalable service discovery for manet. In *Percom '05: Proceedings of the third ieee international conference on pervasive computing and communications* (pp. 235–244). Washington, DC, USA: IEEE Computer Society.

Schulzrinne, H., & Wedlund, E. (2000). Application-layer mobility using sip. *SIGMOBILE Mob. Comput. Commun. Rev.*, *4*(3), 47–57.

Sesay, S., Yang, Z., & He, J. (2004). A survey on mobile ad hoc wireless network. *Information Technology Journal 3*, 168–175.

Sparks, R. (2003). *The Session Initiation Protocol (SIP) Refer Method* (No. 3515). RFC 3515 (Proposed Standard). IETF.

Stoica, I., Morris, R., Karger, D., Kaashoek, F., & Balakrishnan, H. (2001). Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 acm sigcomm conference* (pp. 149–160).

Stuedi, P., Bihr, M., Remund, A., & Alonso, G. (2007). Siphoc: Efficient sip middleware for ad hoc networks. In *Proceedings of the acm/ifip/usenix 8th international middleware conference conference.*

Stuedi, P., Riva, O., , & Alonso, G. (2008). Demo abstract - ad hoc social networking using mand. In *Mobicom 2008.*

Tazaki, H., Meter, R. V., Wakikawa, R., Wongsaardsakul, T., Kanchanasut, K., Amorim, M. D. de, et al. (2009). Selecting an appropriate routing protocol for in-field

manemo experiments. In *Pe-wasun '09: Acm international symposium on performance evaluation of wireless ad hoc, sensor, and ubiquitous networks.*

Thubert, P., Bontoux, C., & Montavont, N. (2007). *Nested nemo tree discovery.* thubert-tree-discovery-06.txt (work in progress).

Thubert, P., Wakikawa, R., Bernardos, C., Baldessari, R., & Lorchat, J. (2008). *Network in node advertisement.* draft-thubert-nina-02.txt (work in progress).

Toh, C.-K. (1997). Associativity-based routing for ad-hoc mobile networks. In *Wireless personal communications journal, special issue on mobile networking and computing systems* (Vol. 4, pp. 103–139). Kluwer Academic Publishers.

Tϕnnesen, A., Lopatic, T., Gredler, H., Petrovitsch, B., Kaplan, A., Tücke, S.-O., et al. (2004). *Olsrd, an adhoc wireless mesh routing daemon.* (available online at http://www.olsr.org)

Wakikawa, R., Thubert, P., Boot, T., Bound, J., & McCarthy, B. (2007). *Problem statement and requirements for manemo.* draft-wakikawa-manemo-problem-statement-01.

Wang, H.-P., Zhang, T.-J., & Yang, X.-Z. (2006). Enabled sip-based multimedia services with mpls technology in ad hoc wireless networks. In *Proceedings of the international conference on intelligent infomation hiding and multimedia signal processing* (pp. 189–192).

Weniger, K. (2006). *PDAD-OLSR: Passive Duplicate Address Detection for OLSR.* draft-weniger-autoconf-pdad-olsr-01.txt (work in progress).

Wongsaardsakul, T., & Kanchanasut, K. (2007). A structured mesh overlay network for p2p applications on mobile ad hoc networks. In *The 4th international conference on distributed computing and internet technology* (Vol. 4882, pp. 67–72). Springer.

Yeh, C.-H., Wu, Q., & Lin, Y.-B. (2006). Sip terminal mobility for both ipv4 and ipv6. In *Icdcsw '06: Proceedings of the 26th ieee international conferenceworkshops on distributed computing systems* (p. 53). Washington, DC, USA: IEEE Computer Society.

Yu, Y., & Agarwal, A. (2005). A sip-based multicast framework in manet. In *Ieee international conference on wireless and mobile computing, networking and communications.*

Zahn, T., & Schiller, J. (2005a). Madpastry: A dht substrate for practicably sized manets. In *Proc. of 5th workshop on applications and services in wireless networks.*

Zahn, T., & Schiller, J. (2005b). Mapnas: A lightweight, locality-aware peer-to-peer based name service for manets. In *Lcn '05: Proceedings of the the ieee conference on local computer networks 30th anniversary* (pp. 499–500). IEEE Computer Society.

Zeadally, S., & Siddiqui, F. (2007). An empirical analysis of handoff performance for sip,

mobile ip, and sctp protocols. *Wireless Personal Communications*, *43*(2), 589–603.

zebra.org. (2003). *Gnu zebra – routing software.* (available online at http://www.zebra.jp)

Zhang, X., Du, X., & Haas, Z. (2006). Performance evaluation of sip-based session establishment over dsr-routed manets. *MILCOM*, *0*, 1–7.

Zhao, B. Y., Kubiatowicz, J. D., & Joseph, A. D. (2001). *Tapestry: An infrastructure for fault-tolerant wide-area location and routing* (Tech. Rep. No. UCB/CSD-01-1141). UC Berkeley.

Zhu, F., Mutka, M. W., & Ni, L. M. (2003). Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services. In *Percom* (pp. 235–242).