An Extensible Information Dissemination Scheme over the Optimized Link State Routing Protocol for Mobile Ad Hoc Networks

Kaveh Mehrjoo

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Dr. Scott F. Midkiff, Chair

Dr. Luiz A. DaSilva

Dr. Mohamed Eltoweissy

September 5, 2007

Arlington, Virginia

Keywords: MANET, OLSR, Service Discovery, Information Dissemination, Extensible

An Extensible Information Dissemination Scheme over the Optimized Link State Routing Protocol for Mobile Ad Hoc Networks

Kaveh Mehrjoo

(Abstract)

A mobile ad hoc network (MANET) is formed by a collection of self-organizing nodes. Such networks are being deployed in a variety of environments, for example to provide mission-critical services in times of crises. Nodes participating in a MANET tend to have limited energy and computing resources and depend on various network-based resources to operate as a cohesive system. The same features such as dynamic and adaptive network topologies that make MANETs powerful also make the discovery and operation of network services a challenge.

This thesis presents the design and implementation of an extensible information dissemination scheme that is integrated with the Optimized Link State Routing (OLSR) protocol to address the challenges of service discovery in mobile ad hoc networks. The thesis presents a detailed design of the information dissemination scheme based on the Naval Research Laboratory's (NRL) ProtoLib network protocol programming framework.

In the proposed scheme, a solution that separates the routing process from the NRL OLSR routing protocol was designed, thus making OLSR a topology discovery protocol. This can further facilitate the implementation of various routing algorithms based on other metrics, such as the signal-to-noise ratio (SNR) of wireless links or the nodes' level of cooperation, when forwarding messages in the network. Additionally, a reusable event-driven programming interface to the NRL OLSR routing protocol was designed

and implemented in this research. Events are triggered based on changes in the network topology. This programming interface can be used by other management and monitoring clients on the network for receiving real-time updates about link and topology changes as seen by OLSR. A priority message delivery scheme was developed that provides different quality of service (QoS) levels for information dissemination in mobile ad hoc networks. To ensure a fair use of the transport media and support various message sizes, a message fragmentation solution was implemented.

The proposed information dissemination solution was then deployed in a real wireless ad-hoc environment for further validation and testing. Using experiments with six nodes and various test scenarios, this research verified the functionality and characterized the performance of the proposed system. It was observed that the link-state nature of information dissemination solution helped it to adapt to topology changes. It was also realized that service discovery latency after information convergence in the network was independent of the number of nodes between the service providers and clients. The experiments also confirmed that the immediate message delivery scheme provides superior quality of service to registered users in presence of radio interference and other delays caused by Multipoint Relay Nodes (MPR) message forwarding in OLSR.

Acknowledgements

First and foremost, I would like to thank my parents. You have been more supportive over the course of my life than I could ever imagine. I have no clue where I would be without your continuous support.

Next, I would like to thank my advisor, Dr. Midkiff. I have learned a lot both professionally and personally from this opportunity. Dr. Midkiff, you have been a mentor and a role model. Thank you for your patience and guidance throughout this research.

I am very thankful to Dr. DaSilva and Dr. Eltoweissy for serving in my committee and for taking your time to review my work.

I would also like to thank my friend Hanif Kazemi. I truly value your friendship. Thanks for all the technical advice throughout undergraduate and graduate coursework .

Next, I would like to thank LTC Michael I. Brownfield for introducing me to the IEEE 802.11 MAC layer and for providing me with an excellent research opportunity working with wireless sensor networks and OPNET network simulation at Virginia Tech.

I am also thankful to my friends and colleagues at GE Fanuc Automation, Vince Epperson and Ben Branham. Thanks for introducing me to network packet programming and industrial automation technologies.

I am also thankful to my colleagues and mentor at Sprint Nextel, particularly Carl J. Gray, and Neil R. Harrington. Thanks for providing me with various learning opportunities and introducing me to network security services and solutions.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Mobile ad hoc networks (MANETs) are formed by a collection of self-organizing nodes. In MANETs, every node is responsible for routing and forwarding packets through the network. Mobile ad hoc networks are being deployed in a variety of environments ranging from on-demand networks in response to sudden events, such as emergency response communication and radio networks, to planned strategic networks in military applications. These networks are used for providing mission-critical services and applications in times of crises. Nodes participating in a MANET tend to have limited energy and computing resources and depend on various network-based resources to operate as a cohesive system.

The same features, such as dynamic and adaptive network topologies, that make MANETs powerful also make the discovery and operation of network services a challenge. Protocols designed for service discovery in fixed infrastructure networks mostly rely on centralized approaches that assume reliable network communication. However, connections between nodes in wireless ad hoc networks are dynamic and often change due to channel impairments and node mobility. These challenges make traditional service discovery protocols impractical for MANET applications.

1.2 Motivation

These limitations have motivated recent research targeting service discovery for mobile ad hoc networks. Service discovery solutions for MANETs that are implemented independent of the routing layer require support from a routing protocol to operate. This approach creates additional control message overhead since both the service discovery and the underlying routing protocol utilize flooding techniques to discover other nodes and resources in the network.

The number of messages needed for service discovery can be reduced by coupling service discovery schemes with the underlying MANET routing protocol. As discussed in Chapter 2, previous research has focused on integrating service discovery with a reactive routing protocol in MANETs. Solutions based on reactive routing protocols suffer from extended service discovery delays and discovery failures due to changes in the network topology. Additionally, good service selection based on the link state routing protocol's topology table can group clients with nearby servers, resulting in localized communication which, in turn, reduces interference and increases opportunities for multiple concurrent transmissions in different parts of the network. Optimized service selection based on the topology table can significantly increase the overall network throughput.

Previous solutions that integrate service discovery with a MANET link state routing protocol result in message size requirements that limit their use to a selected few services in the network. In particular, an extensible information dissemination scheme based on a link state routing protocol that can accommodate various network services with different quality of service (QoS) levels remains a challenge.

1.3 Contributions

In this work, I introduce a new extensible information dissemination scheme that is integrated with the Optimized Link State Routing (OLSR) protocol to address the

challenges of service discovery in mobile ad hoc networks. I present a detailed design of the information dissemination scheme based on the Naval Research Laboratory's (NRL) ProtoLib network protocol programming framework.

The major contributions of this research are as follows.

- I developed an extensible information dissemination scheme for mobile ad hoc networks. The proposed scheme supports various network services with different message sizes and service discovery requirements.

- I developed a reusable event-driven programming interface to the NRL OLSR routing protocol. Events are triggered based on changes in the network topology. This programming interface can be used by other management and monitoring clients on the network for receiving real-time updates about link and topology changes as seen by OLSR.

- I developed a solution that separates the routing process from the NRL OLSR routing protocol, thus making OLSR a topology discovery protocol. This can further facilitate the implementation of various routing algorithms based on other metrics, such as the signal-to-noise ratio (SNR) of wireless links or the nodes' level of cooperation, when forwarding messages in the network.

- I developed a priority message delivery scheme that provides different levels of quality of service for information dissemination in mobile ad hoc networks.

- I developed a message fragmentation solution for the proposed information dissemination scheme. The fragmentation solution ensures fair use of the transport media for processes with various message sizes.

- I implemented the proposed information dissemination scheme based on the NRL OLSR routing protocol.

- I evaluated system performance metrics using real-life scenarios.

1.4 Thesis Outline

The remainder of this thesis is arranged as follows. The next chapter reviews background and previous research addressing service discovery in mobile ad hoc networks. In Chapter 3, the solution approach and methodology is introduced together with basic assumptions and requirements. Chapter 4 presents the design and implementation of the proposed information dissemination scheme and integration techniques with the underlying routing protocol. I present our experimental results and observations in Chapter 5. Finally, Chapter 6 summarizes findings and briefly outlines directions for future research.

Chapter 2

Previous Work

Providing efficient service discovery in a mobile ad hoc network is a complex task. Various solutions have been proposed for service discovery in MANETs. These solutions can be categorized by the level of integration between the service discovery techniques and the underlying routing protocol. Previous research suggests that integrating service discovery with a MANET routing protocol can increase the efficiency and improve the performance of the service discovery process. This chapter presents a brief overview of routing protocols for mobile ad hoc networks. Service discovery techniques are then reviewed with respect to how they are integrated with MANET routing protocols. In Section 2.1, a general overview of mobile ad hoc network routing is presented. Section 2.2 reviews the most popular service discovery solutions. These solutions are analyzed based on their level of integration with the underlying routing protocols in Sections 2.3 and 2.4.

2.1 Mobile Ad Hoc Network Routing

Different approaches have been taken to address the problem of routing in a self-organizing and dynamic mobile ad-hoc network. In reactive routing approaches a routing protocol does not take the initiative for finding a route to a destination until it is required. These protocols only attempt to discover routes on demand and by flooding queries in the network. Reactive routing protocols can often reduce control messaging overhead at the cost of increased latency in finding routes to the destination. The Ad-hoc On-demand

Distance Vector (AODV) [1], Dynamic Source Routing (DSR) [2], and Temporally-Ordered Routing Algorithm (TORA) [3] routing protocols lie in this category of reactive routing protocols. Proactive routing protocols are based on periodic exchange of control messages. The route to the destination is provided immediately when needed, but often at the cost of higher control overhead. Optimized Link State Routing (OLSR) [4] is a proactive routing protocol that reduces control overhead through its optimized flooding techniques and compact control messages. Zone Routing Protocol (ZRP) [5] is an example of a hybrid routing protocol for mobile ad hoc networks. ZRP divides the network into overlapping routing zones. The intra-zone protocol (IARP) proactively generates a zone topology. Hence, nodes within a zone hold a topology of the neighbors in that zone. The reactive inter-zone protocol (IERP) determines inter-zone routes for packets by sending route request (RREQ) messages to all border nodes in a reactive manner.

OLSR is an example of an optimized link-state routing protocol. It is discussed in more detail here since it is the basis of the system developed in this research. Due to its proactive nature, it has the advantage of having the routes immediately available when needed. It also inherits the stability of a link-state routing algorithm. In a pure link-state protocol, all the links with neighbors are declared and flooded to the entire network. To improve performance in an ad-hoc mobile environment, OLSR reduces the size of control packets. Instead of all links, it declares only a subset of links with its neighbors for which it serves as a multipoint relay selector (MPRS). The protocol is particularly suitable for large and dense networks, as it minimizes the flooding of its control traffic by using only the selected nodes, called multipoint relays (MPR) to diffuse its messages through the network. Only the multipoint relays of a node retransmit its broadcast messages. Apart from normal periodic control messages, the protocol does not generate extra control traffic in response to link failures and additions. Participating nodes in the topology keep an updated topology table with information regarding links between nodes in the network.

2.2 Overview of Service Discovery Protocols

Protocols designed for discovery of services in fixed infrastructure networks mostly rely on centralized approaches that assume reliable communication in the network. Previous solutions for service discovery include the following.

- JINI [6]
- Salutation [7]
- Universal Plug and Play (UPnP) [8]
- Universal Description, Discovery and Integration (UDDI) [9]
- Service Discovery Protocol (SDP) [10]
- Service Location Protocol (SLP) [11]

These protocols utilize special nodes acting as central service repositories, where service providers can register their service offerings. The clients then broadcast service discovery messages to locate service coordinators in the network. Direct messages are then sent to the service coordinators for service binding and registrations. In typical mobile ad hoc network environments, the topology is not stable and wireless link conditions are continuously changing. Hence, service discovery protocols that are intended for fixed wired networks are not efficient and practical for wireless MANET applications. This has motivated recent research targeting service discovery in MANETs, including the following.

- Alliance-based Service Discovery for Ad-Hoc Environments (Allia) [12]
- Grid Service Discovery (GSD) [13]
- DEAPspace [14]
- Konark [15]
- Service Awareness and Discovery in Mobile Ad Hoc Network (SANDMAN) [16]

These protocols facilitate discovery of various network services independent of the underlying routing protocol. Hence, these protocols rely on a MANET routing protocol for providing end-to-end routing of service discovery messages creating additional overhead to support service discovery functions in addition to routing functions.

2.3 Service Discovery Independent of Routing

All service discovery protocols implemented independent of the routing layer require the support of a routing protocol. This creates additional control message overhead. These protocols can be further categorized based on the service discovery architecture, which can be centralized, direct, or hybrid in nature.

2.3.1 Centralized Approach to Service Discovery

In the centralized approach dedicated service coordinator nodes are responsible for providing information regarding the available services in the network. Using such a centralized method is not suitable for a mobile ad hoc environment. Link conditions and interference in the wireless network can cause topology partitioning and make the central specialized service directory nodes unreachable. This can further limit service discovery and lookup, hence preventing nodes from locating available services in the network. The Universal Description, Discovery and Integration (UDDI) service discovery scheme [9] is an example of a centralized approach.

2.3.2 Direct Approach to Service Discovery

As an alternative to the centralized methods, direct approaches define a decentralized architecture without relying on particular designated nodes. A client floods the network with service discovery messages. Each service provider then responds with a service reply message if it finds a match to the service request. This direct approach can be expensive in ad hoc networks that have limited wireless communications resources since it may cause a large number of broadcast messages to be transmitted throughout the network whenever a client attempts to locate a particular service. This creates significant overhead in the case of on-demand routing protocols. In proactive routing protocols, the

service discovery messages do not affect the number of routing messages. Additionally, the direct approach has the potential drawback of limiting the service availability in presence of topology partitioning. The service discovery schemes that follow a direct approach include. Universal Plug and Play (UPnP) [8], Service location Protocol (SLP) [11] and DEAPspace [14].

2.3.3 Hybrid Approach to Service Discovery

The hybrid approach combines aspects of both direct and centralized solutions. Using the hybrid approach can further limit periodic exchanges of service discovery message broadcasts to a node's neighborhood. These schemes may rely on hierarchies where particular nodes proactively broadcast service discovery messages in the network to announce availability of new resources. Protocols that utilize a hybrid service discovery architecture are Salutation, Konark, Jini.

2.4 Service Discovery Integrated with Routing

The proposed protocols for MANET service discovery that are independent of the routing protocol can lead to a high level of messaging overhead. It is important to realize that both the service discovery and the underlying routing protocol utilize flooding techniques to discover other nodes and resources in the network. Hence, integrating service discovery with the routing protocol can significantly reduce additional control overhead and significantly increase service availability in mobile ad hoc environments.

Additionally, good service selection based on a routing protocol's route and topology information can group clients with nearby servers, resulting in localized communication that can reduce interference and allow for multiple concurrent transmissions in different parts of the network. The cross-layer service discovery approach, which integrates service discovery and routing, leverages existing routing control messages for service discovery and allows clients to switch to closer servers as the network topology changes. As a result, the number of control messages needed for service discovery can be reduced

by coupling service discovery with the underlying MANET routing protocol. In Section 2.4.1 and 2.4.2 the integration strategies are further categorized based on the underlying routing protocol used for disseminating service discovery messages throughout the network.

2.4.1 Service Discovery using Reactive Routing Protocols

Koodli and Perkins [17] propose integration of service discovery with an underlying reactive routing protocol. In their approach Service Discovery Requests (SREQs) are piggybacked on Routing Request (RREQ) packets and Service Discovery Replies (SREP) are added on Routing Reply (RREP) messages. This solution mainly relies on a reactive routing protocol, such as AODV, DSR, or TORA, and, therefore, suffers from increased latency of service discovery as network size and node mobility increase. Broadcast service discovery traffic triggers additional control messaging in the underlying reactive routing protocol and significantly increases the routing message overhead in the network.

2.4.2 Service Discovery using Proactive Routing

In proactive link state routing protocols, every node maintains a topology table with regularly updated information on the links between the participating nodes. Integrating service discovery and binding with network topology can result in immediate service discovery after information convergence in the network. Using an integrated service and topology table, nodes can choose closer service providers and effectively enhance service selection in the network. Effective service selection can improve network throughput by up to 400 percent [18]. The availability of explicit routing information such as route breaks or updates allows clients to efficiently detect changes in the network topology and switch to closer servers without additional delays.

2.4.2.1 Service Discovery using OLSR

Jodra and Vara [19] propose adding a new message type, called a Service Discovery Message (SDM), to the OLSR routing protocol. SDM messages are eight bytes long and are used by service providers to announce new services available in the network. New

services are advertised once, with a specific lifetime, and cached at each node. Client nodes send broadcast query messages when unable to locate a service in the local cache. These SDM messages are then forwarded by MPR nodes through the network.

A similar approach is taken by Li and Lemont [20] by coupling support for the Session Initiation Protocol (SIP) for real-time session service location and discovery with the OLSR routing protocol. In both the proxy-based and proxy-less SIP networking architectures, a key issue for providing support for SIP in a MANET is service location and discovery, i.e., determining the location of the SIP server node where SIP signaling messages can be sent. In their approach, a new Service Location Extension (SLE) message type is added to OLSR to provide service location support for SIP service discovery. In the SLE message, a service is advertised with a service-type field and one or more Uniform Resource Locator (URL) fields denoting SIP server location(s). MPR nodes forward the SLE messages through the network. SIP servers periodically advertise their location using the SLE messages. The SLE messages can also be used by clients as a query message to request the location of a SIP server.

These solutions enjoy improved discovery performance by coupling service location discovery with the OLSR proactive routing protocol. SIP server discovery over OLSR can reduce call-setup latency and voice data transmission delay, factors which are critical to the performance of voice or video calls [20]. Also, the MPR forwarding mechanism leveraged when launching queries for service locations reduces additional service discovery overhead and significantly improves the system's overall performance [20].

Although the two proposed protocols have enjoyed more optimized service discovery in the network by integration with OLSR, they are not extensible enough and cannot provide information dissemination for other in a generalized manner, i.e. to support other applications. In Li and Lemont [20], the SLP extension to OLSR can only locate SIP servers in the network. The proposed SLE messages cannot be used by other service providers in the network when disseminating information. Hence, this limits the applicability of the proposed protocol to support other applications. Similarly the SDM

messages proposed in Jodra and Vara [19] can only hold eight bytes of data. This further limits the application of the proposed protocol to only simple service descriptions for service discovery. Other applications that require larger message sizes for service discovery and information dissemination cannot be implemented over the scheme proposed by Jodra and Vara.

2.4.2.2 Service Discovery using OSPF-MCDS

DaSilva, *et al.*, [21] introduce the integration of various network-level service functions with the underlying modified Open Shortest Path First with Minimal Connected Dominating Sets (OSPF-MCDS) routing protocol [22]. OSPF-MCDS is an application of the Open Shortest Path First (OSPF) routing protocol that uses the minimum connected dominating set (MCDS) of nodes to propagate route updates. OSPF-MCDS is a link-state and proactive routing protocol optimized for mobile ad hoc network environments. It is realized that the integration of network services, including certificate authority (CA) and other security functions, network management, and Domain Name Service DNS, with the routing information is beneficial for efficient operation of a network as a cohesive system in a dynamic mobile ad hoc environment. However, the specific integration of services is done in an application-specific manner and is not sufficiently general to support discovery of a new service without modification.

2.4.3 Service Discovery using Hybrid Routing Protocols

In Extended ZRP [23], Ververidis and Polyzos proposed adding service discovery capabilities to the ZRP hybrid routing protocol. An extra field is added to the Neighbor Discovery Protocol (NDP) HELLO messages for storing service identifiers (IDs). Unique Universal Identifiers (UUIDs) are added to routing messages to hold service descriptions. The IARP routing messages are also modified to accommodate service information.

In HAID [24], a hybrid adaptive approach based on the AODV routing protocol is introduced to reduce latencies caused by subnet mobility in mobile ad hoc networks. HAID is designed as a hybrid scheme that utilizes a proactive model for service advertisements by providers and a reactive model for service queries conducted by

clients. In HAID, special Service Provider Advertisement Messages (SADV) messages are used during the advertisement phase. A service provider periodically generates an advertising message that describes the service characteristics and sends it to all of its one-hop neighbors. Any node can use the service request messages (SREQ) to query for a service during the service discovery phase. A service reply message (SREP) message is sent by the service providers back to the requesting node when the desired information is found.

SPIZ [25] presents an adaptive service discovery protocol integrated with an efficient hybrid routing protocol based on ZRP. This scheme allows nodes to adapt their own zone radii dynamically based on changes in the network. These changes include conditions, such as mobility and popularity levels. Based on a variable zone radius, SPIZ combines proactive service advertisement together with a reactive strategy. Integration with the network layer routing protocol is used to achieve in a lightweight scheme that reduces the amount of unnecessary network flooding.

Finally, in CARD [26] a hybrid service discovery scheme is developed using a zone-based protocol for service discovery. CARD mainly focuses on short-term transactions and adopts a hybrid approach in which a node uses periodic updates to reach a neighborhood within a predefined number of hops. Reactive querying beyond the neighborhood radius is used when the service is not available in the local neighborhood. Nodes maintain special contacts beyond the local neighborhood. These contacts are polled periodically to update routes and other service presence. During the discovery phase queries are sent to the contacts directly.

Table 2.1. Summary of Mobile Ad Hoc Network Service Discovery Protocols

MANET Service Discovery Independent of Routing		
DEAPspace [14]	Konark [15]	
SANDMAN [16]	Allia [12]	GSD [13]
MANET Service Discovery Integrated with Routing		
Reactive	***Proactive***	***Hybrid***
AODV [1]	OLSR-SDM [19]	Extended-ZRP [23]
DSR [2]	OLSR-SLP [20]	HAID [24]
TORA [3]	OSPF-MCDS [21]	SPIZ [25]
		CARD

2.5 Summary

Table 2.1 summarizes previous solutions addressing the problem of service discovery in mobile ad hoc networks. In this chapter, various service discovery schemes were reviewed that are designed to address challenges posed by MANET environments. These solutions were categorized based on their level of integration with the underlying routing protocol. Additionally, integrated service discovery schemes were analyzed based on the characteristics of the underlying routing and, specifically, based on the routing protocol used. Routing protocols used as part of these service discovery schemes are AODV, OLSR, OSFP-MCDS, and ZRP. Previous work suggests that integrating the service discovery messaging with the underlying routing protocol can significantly reduce additional control overhead and significantly increase service availability in mobile ad hoc environments. In this research, we examined an approach to create a general scheme that integrates service discovery with routing, in particular that is based on OLSR. This goal is described in Chapter 3.

Chapter 3

Problem Statement and Methodology

This chapter presents the objectives and methodology used throughout this research. The purpose of this research is to develop an efficient scheme for service discovery in mobile ad hoc networks that improves service availability and selection. The expected key contributions of this research are the design and implementation of an integrated information dissemination technique that uses Optimized Link State Routing (OLSR) protocol control messages for propagating updates through the network. An extensible event-based programming interface is also provided for topology extraction from the OLSR routing protocol and information dissemination. This work also introduces and validates an innovative technique for separating the route selection process from the OLSR protocol for topology dissemination based on the provided programming interface.

3.1 Problem Statement

Service discovery protocols designed for fixed and infrastructure-based networks are not suitable for mobile ad hoc network environments because they mainly rely on central directories for locating available network services, and broadcast service request message to bind with available resources in the network. In MANETs, however, the network topology is very dynamic and nodes have limited energy and bandwidth resources. Additionally, the central service directory approach introduces single point of failure. In recently years, various protocols have been proposed for providing service discovery for MANET applications. These solutions, however, mainly rely on some MANET routing protocols for supporting request and service delivery from clients to service providers.

This assumption makes these solutions inefficient in wireless mobile ad hoc networks due to the redundant control packet flooding and extremely high network overhead. Additionally, these service discovery protocols that are middleware-oriented solutions and are implemented independent of the network routing layer are unable to provide optimized service selection and binding between clients and service providers. Network partitioning can reduce the overall network throughput and service availability in mobile ad hoc networks.

Motivated by these limitations I focused my research on the design and implementation of an optimized, adaptive and extensible service discovery and information dissemination scheme based on the Optimized Link State Routing protocol. This new proposed information dissemination scheme is aimed at improving service discovery for mobile nodes using an event-driven programming interface that is triggered based on real-time changes in the network topology.

3.2 Goals and System Definition

This section presents the methodology used to design and implement the proposed information dissemination scheme using the OLSR routing protocol. First, the protocol design approach is outlined, followed by a discussion of the implementation approach.

3.2.1 Design Approach and Methodology

Chapter 2 presented the necessary background to understand the challenges faced in mobile ad hoc networks and introduced both network layer-independent and integrated techniques designed to improve service discovery in such networks. In this research the OLSR routing protocol was selected as the underlying network routing protocol due to its proactive and link state characteristics which can significantly improve service availability for mobile nodes [27]. The overall service discovery latency can also be reduced by using an optimized proactive routing protocol for transporting service request

and reply messages. Additionally, OLSR has proven to be a scalable and optimized routing protocol for mobile ad hoc network environments [28]

The primary goal of this research is to develop a new, optimized and extensible information dissemination scheme using the routing control message and topology repositories in the OLSR protocol for transporting information messages. By encapsulating information-carrying Bucket Data Units (BDU) in the HELLO and TC messages of the OLSR protocol I was able to utilize the routing control messages for delivering information throughout the wireless ad hoc network. This technique utilizes the link state and proactive characteristics of the underlying routing protocol for efficient information dissemination and service discovery. An event-based application program interface (API) is designed and implemented for service binding using the OLSR network topology repository for locating available resources in the network.

To validate the proposed framework the routing process is separated from the OLSR routing protocol. The separated routing client then uses the event-based topology programming interface and other network service repositories available with the new scheme when populating the routing table. This allows for extensible routing algorithm implementations based on metrics like, signal-to-noise Ratio (SNR) of the wireless links between nodes participating in the OLSR topology.

3.2.2 Implementation

The proposed information dissemination scheme is implemented using the Fedora Core 5 distribution of the Linux operating system [29]. I selected the Naval Research Laboratory's implementation of OLSR, also known as NRL OLSR [30], as the routing protocol for our implementation. The NRL implementation is a version of OLSR protocol based on the OLSR specifications [4] with a few enhancements. The NRL code base has additional features including:

- a "willingness" attribute for localized Mulitpoint Relay Node (MPR) activation;
- the full link state topology can be distributed including non-MPR cross links;

- support for several MPR selection protocols, including classical flooding, Non Source Specific Multipoint Relay (NS-MPR), Source Based Multipoint Relay (S-MPR), Multipoint Relay Connected Dominating Set (MPR-CDS), and Essential Connected Dominating Set (E-CDS;)
- neighbor link quality assessed by a smoothed hysteresis function;
- many run-time parameters available, including HELLO interval, link state update interval, timeout factors, link quality assessment parameters, MPR willingness, and message TOS;
- configurable debugging verboseness;
- experimental features such as fuzzy-sighted routing and support for Simplified Multicast Forwarding;
- IPv6 Support
- operational in Windows, MacOS, Linux, and some embedded personal digital assistant (PDA) systems, such as Zaurus and PocketPC.

NRL OLSR is implemented using the Protolib programming toolkit [31], which allows for cross-platform support including building code into different network simulation environments such as NS-2 and OPNET. My information dissemination scheme is integrated with NRL OLSR code based on the Protolib programming toolkit.

The NRL OLSR code base is used for my implementation, since it is known to be more stable and suitable for large scalable wireless network deployments [32]. Chapter 4 provides details about the design of the proposed information dissemination scheme and the event-driven API for OLSR.

The Protolib programming toolkit also provides encapsulation of the Linux operating systems' inter-process communication sockets (IPC) [33]. These IPC raw sockets are mainly used in our OLSR topology API for event notification and information encapsulation and delivery between local processes.

3.3 Performance Metrics

The performance of the new service discovery scheme over OLSR routing protocol is evaluated based on additional network control message overhead and round-trip delay for information dissemination. These performance metrics are defined below.

3.3.1 Information Dissemination Round-Trip Delay

The information dissemination round-trip delay is defined as the time required for a single information Bucket Data Unit to travel from the service provider to the registered client and back again. This is a measure of the information dissemination delay in the mobile ad hoc network.

3.3.2 Message Overhead

The message overhead is defined as the overhead caused by encapsulating the information Bucket Data Units in OLSR control messages. This metric depends on the information propagation radius and information update period. All information dissemination control overhead transmitted by a participating node in the network is considered to be message overhead. The total message overhead is the sum of overhead introduced by Bucket Data Unit encapsulations in both HELLO and TC messages.

3.4 Validation and Tradeoffs

The new information dissemination scheme is deployed in a small wireless ad hoc network test bed for further validation and testing. As indicated above, the metrics of interest are control message overhead and information propagation round-trip delay. The following test bed setup is used for validating the implementation.

3.4.1 Test Bed Used for Validation

Wireless nodes are Dell Latitude C640 notebook computers with 520 megabytes (MB) of random access memory (RAM) and Intel® Pentium 4 processors. The nodes use Xircom® IEEE 802.11b wireless cards for wireless ad hoc network connectivity between

the participating nodes. The Fedora Core 5 distribution of the Linux operating system is used on all the notebook computers. The NRL implementation of OLSR is the code base for routing and implementation of the information dissemination scheme. The NRL Protolib programming toolkit is used for implementing various clients and service providers in the network.

3.4.2 Test Scenarios

The following two usage scenarios are considered to demonstrate the capabilities of the system in a real-life pervasive computing application environment.

- **Small Apartment:** This scenario represents mobile nodes participating in a wireless ad hoc network located in a two-bedroom apartment. Links are relatively stable since nodes experience less channel interference and are located in close proximity.

- **Office Building:** This test scenario considers performance in a typical small to medium size office complex, with high wireless interference caused by infrastructure-based IEEE 802.11b access points operating on the same radio frequency (RF) as the mobile ad hoc networks. Wireless nodes are more dispersed and experience a relatively high degree of RF interference.

3.4.4 Methodology Tradeoffs

Much of the previous research on service discovery in mobile ad hoc networks has focused on simulation studies for validating their approach. Due to the complex nature of wireless mobile ad hoc networks, it is not certain whether the simulation results can provide a practical understanding of performance of the service discovery protocols in a large-scale MANET deployment. In this research, our proposed information dissemination scheme was implemented in the Linux operating system using the NRL OLSR code. This allows for more practical experiments to study the system in a real mobile ad hoc network environment. However, the method does have limits in that a large scale deployment and experimentation is more difficult due to the hardware and

resource limitations. Hence, experimentation is based on a limited number of nodes within close proximity and with limited node mobility.

3.5 Summary

This chapter presented the problem of service discovery in mobile ad hoc network environments that is addressed in this research. The proposed extensible and event-driven information dissemination extension to NRL OLSR was then introduced. The methodology used to validate and characterize the new scheme was further discussed. The next chapter describes the modules and different components designed in this research to provide an extensible and efficient event-based information dissemination platform for MANET applications.

Chapter 4

MANET Information Dissemination: Design and Integration with NRL OLSR

4.1 Overview

This chapter outlines the design of the proposed extensible event-driven information dissemination scheme for mobile ad hoc networks. First, an overview of system components, data structures, and information repositories is presented followed by an example scenario of a Domain Name Service (DNS) and Certificate Authority (CA) in the network. In the following sections each component is described in more detail.

4.1.2 Process Service IDs

Users of the new information dissemination Topology Server (TP) are identified using a unique Service Identifier (SID). Different SIDs can be used for providing different level of quality of service for information dissemination in the network. Local users are processes on one machine interfacing with the Topology Server locally using operating system-supported inter-process communication (IPC) sockets, also known as IPC pipes. Remote processes are processes located on a remote machine that register with the Topology Server and provide information updates using Bucket Data Units (BDU) in the network.

4.1.3 Events API

The Topology Server (TP) process provides an extensible event-driven programming interface for information dissemination through the network. This programming interface allows for internal processes to receive notification and data updates based on various OLSR topology events. Additionally, internal user processes can subscribe to the topology events and provide information to be disseminated in the network. Table 4.1 contains a list of defined OLSR events and their descriptions. These events are used to trigger alerts based on changes in the network or when a new service becomes available. ***HELLOTimeout*** and ***TCTimeout*** events are triggered when a new HELLO or TC message is generated. The ***SocketEvent*** flag is set on reception of an OLSR control message including other OLSR nodes HELLO and TC messages. Other ***SocketEvent*** are defined for receiving a ***Duplicate*** packet, a change in the link status between neighbors, or a change in the MPR-selector set. Extra TC message generation can also flag a TP event. There are topology related events defined for times that nodes turn into an MPR and start generating TC message to disseminate and update the network topology. An MPR node that changes its role in the topology and becomes a regular node would also trigger a topology change event. An update can be sent to users subscribed to the list of published events.

Table 4.1. List of Defined Events for Information Dissemination and Network-based Alerts

Event Type			Code
Send Event			
	Hello Timeout	SEND HELLO	A
	TC Timeout	SEND TC	B
Receive Event			C
RECEIVED :			
	Duplicate Message		D
	HELLO Message		E
	Neighbor Stable for HELLO		F
	TC Slow Down		G
	Neighbor Link Status change To:		
		ASYM LINK	H
		SYM LINK	I
		MPR LINK	J
		LOST LINK	K
	TC Message		L
	MSSN NOT CURRENT		M
	Extra TC		N
	TC FORWARDING		O
Neighbors Stable For TC			P
Set Link State			
	LINK_DOWN		Q
	LINK_DEFAULT		R
	LINK_UP		S
Node Type Change			
Change State to an MPR Node			T
Change State to a Regular Node			U
Change In the MPR-Selector Set			V

4.1.4 IPC Pipes

Special inter-process communication sockets are defined to facilitate the communication with the Topology Server for information dissemination and event notifications. Table 4.2 contains a list of defined IPC sockets.

Table 4.2. IPC Pipes Defined for Communication with the Topology Server

IPC Pipe Name	Description
RECV1	Used for Receiving Data Updates from Local Process
MYREG	Used for new Process Registration
SERVICE ID	Open Pipe for Sending Bucket Data to Registered users

The IPC pipes are used by local process for the following.

- Registration with the Topology Server
- Providing updates for network dissemination
- Receiving updates from the network

4.1.5 Data Codes and Data Direction

Data codes are used to differentiate between information flows through the network. The Topology Server on each node is responsible for formatting the Bucket Data Units for each user and adding the appropriate Service IDs and data codes. New service providers in the network can use data codes to allow for service lookup and binding.

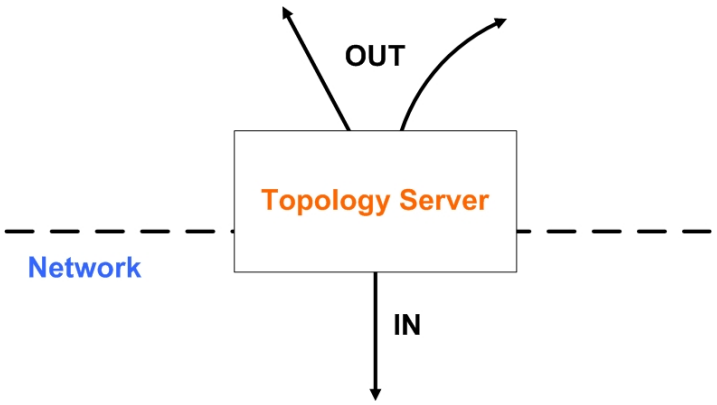

Figure 4.1. Topology Server and data directions.

Each BDU is then encapsulated into a HELLO or TC message and disseminated through the network by the OLSR Multi Point Relay (MPR) nodes. Data updates can have two

general directions. Figure 4.1 illustrates the two defined directions for data through the Topology Server process. An outward flow indicates the flow of information from the Topology Server to the local process registered on the same machine. An inward flow is used to indicate a network update. The data directions are stored in user's Data Record Card for each event that the user is subscribed to. The format of user Data Record Cards is presented in Section 4.1.7.

4.1.6 New User Subscription Handshake

New users of the system first go though the subscription handshake with the Topology Server. This allows for providing and receiving updates using the (TP) process with the established inter process communication pipes. The internal processes use the ***MYREG*** pipe to subscribe with the TP process. After successfully registering with the Topology Server, local users receive event notifications and data updates over the IPC pipes. The ***RECV1*** socket is used for providing data updates to the Topology Server for network dissemination. Figure 4.2 illustrates the communication handshake between the TP processes and two examples clients.

In this example, two processes first register and subscribe to OLSR topology events through the programming interface using the MYREG IPC pipe. After successfully registering with the Topology Server, a new IPC pipe is established using the process SID when providing updates. The RECV1 pipe is then used by the processes for providing updates for network dissemination. The data direction parameters are used to indicate the direction of update messages.

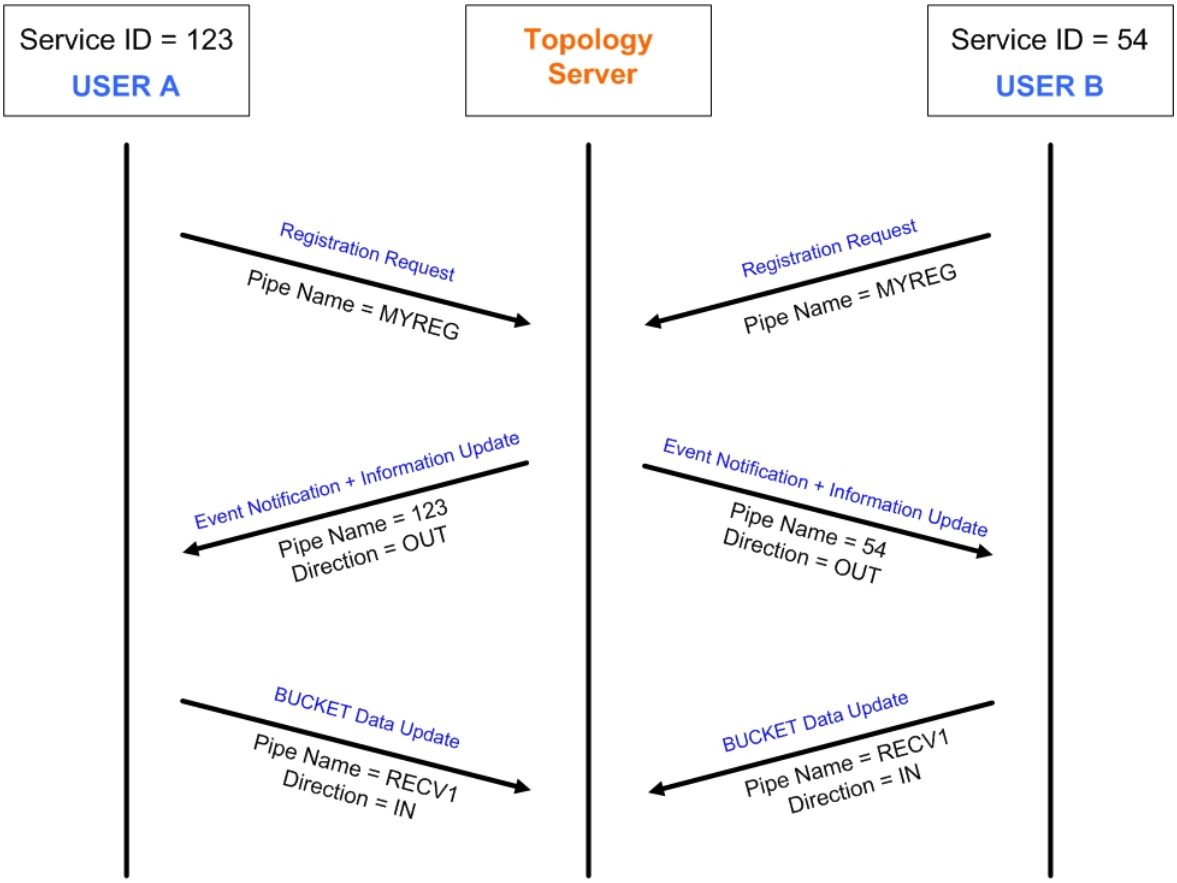

Figure 4.2. Subscription handshake over IPC pipes.

4.1.7 User Data Record Card

A new Data Record Card (DRC) is created for each user after successfully registering with the topology Server. User Data Record Cards use the format shown in Figure 4.3 for storing event subscriptions, event update frequencies, and data pointer to local data storages in the Topology Server process.

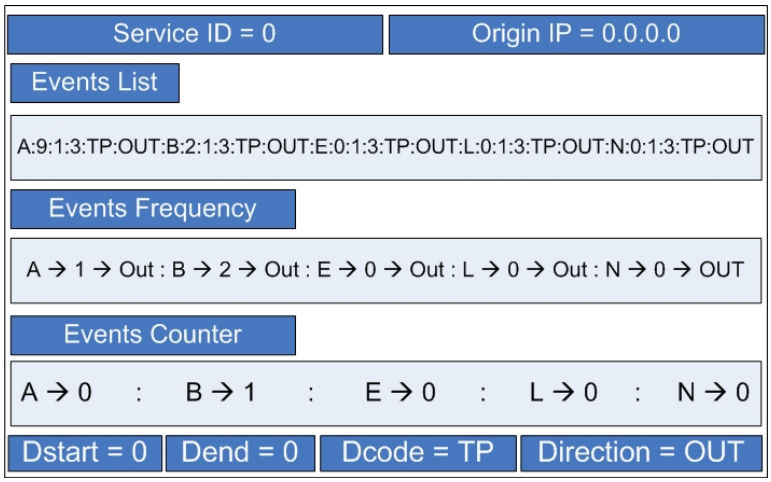

Figure 4.3. Data Record Card used for storing user registration.

The ***ServiceID*** field in the user data record card contains user's unique identifier. The same SID is used for establishing IPC socket with the Topology Server process during user registration. ***OriginIP*** of the registered user is stored and used when encapsulating updates in Bucket Data Units. An IP address of 0.0.0.0 is used for local processes that do not have any outgoing network updates. The ***EventList*** contains a detailed list of topology-based events that users can subscribe to for sending or receiving updates. This list also contains the data update direction, and data codes for each event. The event frequencies are later used when providing updates. An independent ***EventCounter*** is kept for each event. Two pointers are stored in ***DStart*** and ***DEnd*** fields indicate user's data location in the ***BIGDATABUFFER*** data repository. The various data codes are used for differentiating between different data flows in the network. Finally, direction of information update is also stored for each event for different events.

4.1.8 Local and Remote Users

Local processes residing on the same machine registered with the TP process with an IP address of 0.0.0.0 which is stored in the corresponding Data Record Card. A remote process is a process using the topology server from a remote machine. The IP address of the primary network interface of the remote machine is stored in remote user's Data Record Card.

4.1.9 Information Repositories

The following data repositories are defined in the Topology Server for storing user Data Record Cards, and Data updates.

- RegisteredUsersDB
- BIGDATABUFFER

After successfully registering with the Topology Server, a new Data Record Card is created and added to the ***RegisteredUsersDB*** information repository as shown in Figure 4.4.

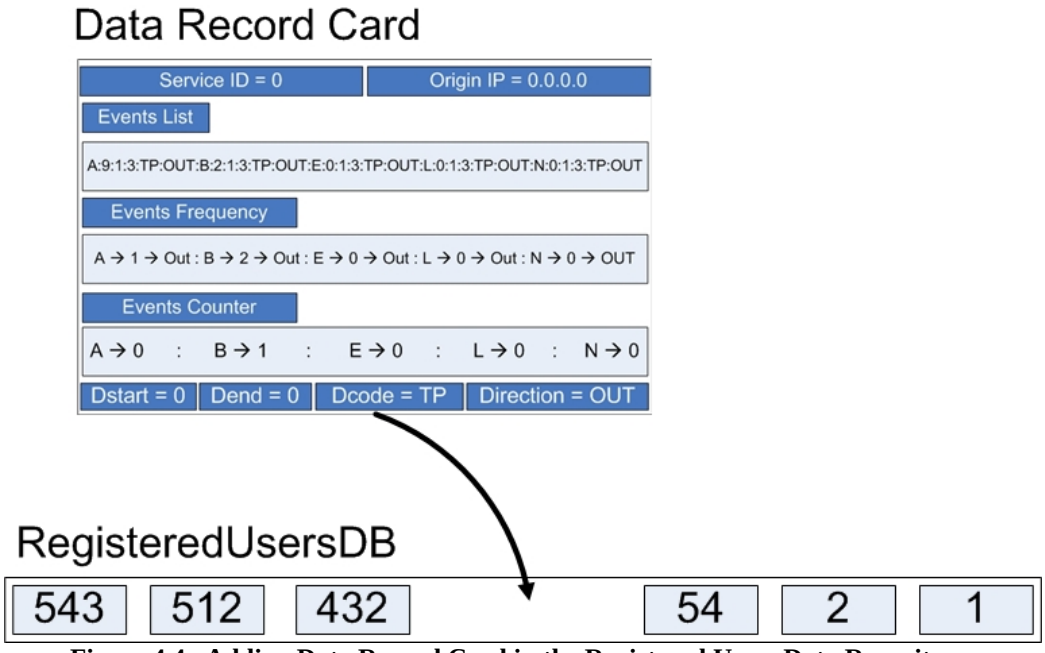

Figure 4.4. Adding Data Record Card in the Registered Users Data Repository.

When data updates are received for registered users, the data is stored in the BIGDATABUFFER data repository. A pointer to user's data start and data end is added to each user's Data Record Card.

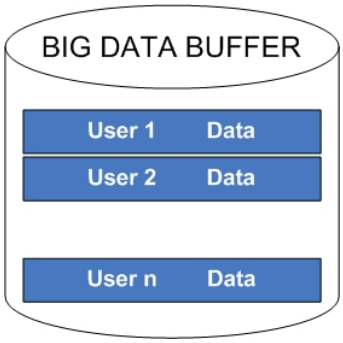

Figure 4.5. Data repository used for storing user data.

User data is stored in contiguous block in the BIGDATABUFFER information repository. This data is later used by the Topology Server process when disseminating user updates through the network. BIGDATABUFFER is a dynamic and extensible information repository for storing user updates with support for variable data sizes.

4.1.10 Information Bucket Data Units

Topology Server is responsible for disseminated data updates through the network. Bucket Data Units are used for propagating user updates. The Bucket Data Units are encapsulated in the outgoing HELLO and TC messages. These buckets contain user data updates and are used for disseminating information update through the network by the MPR nodes.

Buckets are designed as flexible data units that can store user data up to a system defined fragmentation threshold. User data exceeding this threshold will be fragmented by the Topology Server. Non-MPR nodes can provide their updates by encapsulating Bucket Data Units in HELLO messages. MPR nodes are then responsible for processing and forwarding the Bucket Data Units received from nodes in their MPR selector set. These buckets are aggregated at the MPR node and then encapsulated into outgoing TC messages. Figure 4.6 illustrates HELLO and TC bucket encapsulation.

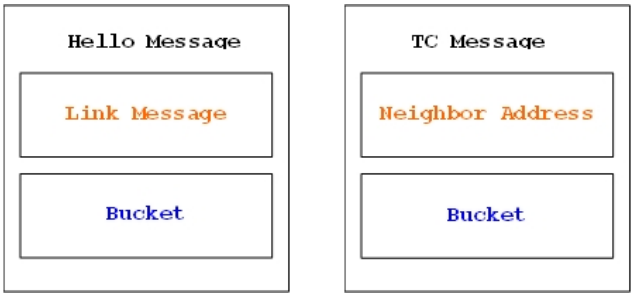

Figure 4.6. Bucket Data Unit encapsulation in the HELLO and TC messages.

4.1.11 Finite State Machine Design

The Topology Server process was designed using a top-down approach to decompose the process into four distinct phases and to link them together using a finite state machine (FSM). Figure 4-7 illustrates the highest level of abstraction for the TP process FSM. The ***RegisterNewUser*** state is first initiated to register a local process with the Topology Server. New users register with the TP process using the handshake outlined in Section 4.6. After successfully registering with the TP process processes can receive updates.

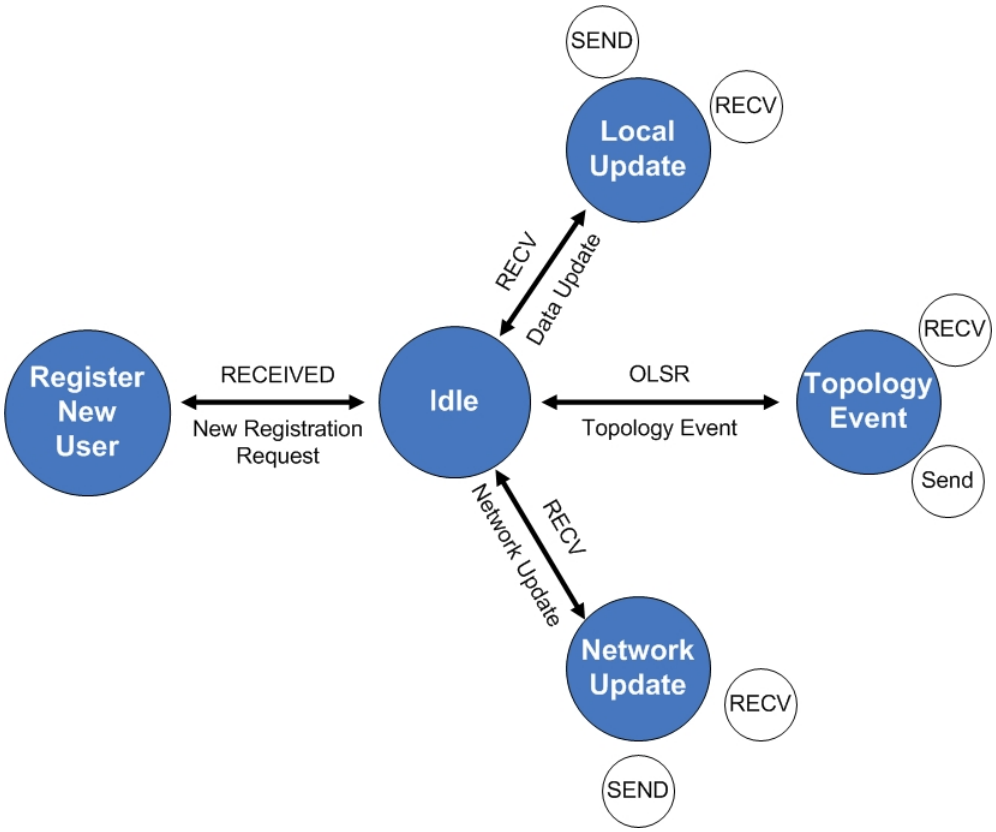

Figure 4.7. Finite State Machine design of the Topology Server process.

Three general events are expected and processed by the event handlers. These events are further categorized as:

- Network update events
- Topology change events
- Local update event

The TP process enters the ***NetworkUpdate*** state when a new OLSR message containing a Bucket Data Unit is received. A network update event is also generated when local user updates need to be encapsulated in the outgoing HELLO and TC messages. Topology change events are used to notify registered users of any changes in the network topology. The ***TopologyEvent*** processing state is used to service events related to changes in the MPR-selector set and neighbor link updates. The TP server enters the ***LocalUpdate*** state when local users send data updated to the process for network dissemination. Section 4.3 covers local data updates state in more detail.

4.1.12 Example Information Dissemination

A brief overview of the information dissemination scheme using the proposed Topology Server process is presented in this section. Figure 4.8 shows an example information dissemination scenario in a network of seven nodes. The MPR nodes are responsible for disseminating information updates using the Bucket Data Units through the network. Nodes 1, 7, and 6 are using the Topology Server to send and receive data updates for network services they use. Each node can act as both a server and a client to a particular service at the same time. In this example, node 1 is a hosting a certificate authority (CA) and is providing signed certificates through the network using Bucket Data Units. Since node 1 is not an MPR node in the topology it first uses a HELLO message to send the certificate updates to node 3 which is an MPR node in the topology. Node 3 then uses TC messages and encapsulates node 1's updates in Bucket Data Units formatted for network dissemination.

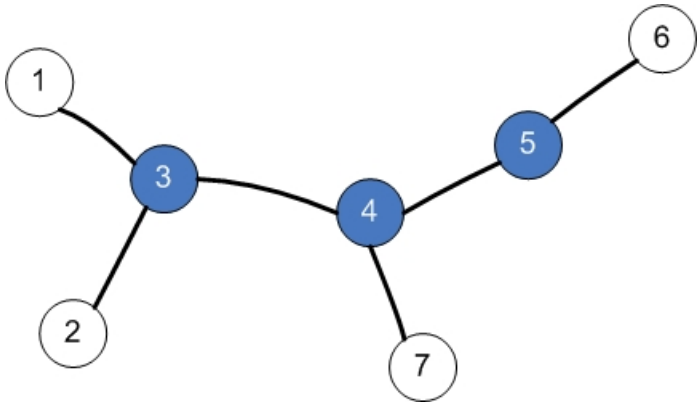

Figure 4.8. Example of a 7-node topology utilizing the proposed information dissemination scheme.

As can be seen in Figure 4.9, the Topology Server process at node 1 has two records in the RegisteredUsersDB data repository. The first Data Record Card is used for the local routing process. The IP address is set to 0.0.0.0 in the Data Record Card since the routing process is not disseminating information through the network and is only receiving network topology updates such as changes in the MPR selector set, or link state changes between neighbors. These updates are received based on frequencies and events used during the registration with the Topology Server. A more detailed discussion of routing process separation is presented in Section 4.5. The data field in routing process's Data Record Card is set to "TP" to indicate Topology updates, and the data direction is set to OUT. As can be seen in Figure 4.9 the Data Record card also contains a list of subscribed events for the routing process.

In this example a service ID of "5674" is used for the certificate authority (CA) process on node 1 when registering with the Topology Server. Data type is set to CA and data direction is "IN" to indicate network dissemination. When providing network updates, a Bucket Data Unit is created and used for propagating certificates in the network for the certificate server process in node 1. Figure 4.9 shows Bucket Data Units in OLSR HELLO messages generating by node 1. The IP address, Data Code, and Service ID, of the certificate authority is added to the Bucket Data Unit. This allows for other processes in the network to successfully locate and subscribe to the certificate authority process when receiving updates. Section 4.6 contains a detailed outline of Bucket Data Unit messages and encapsulation schemes.

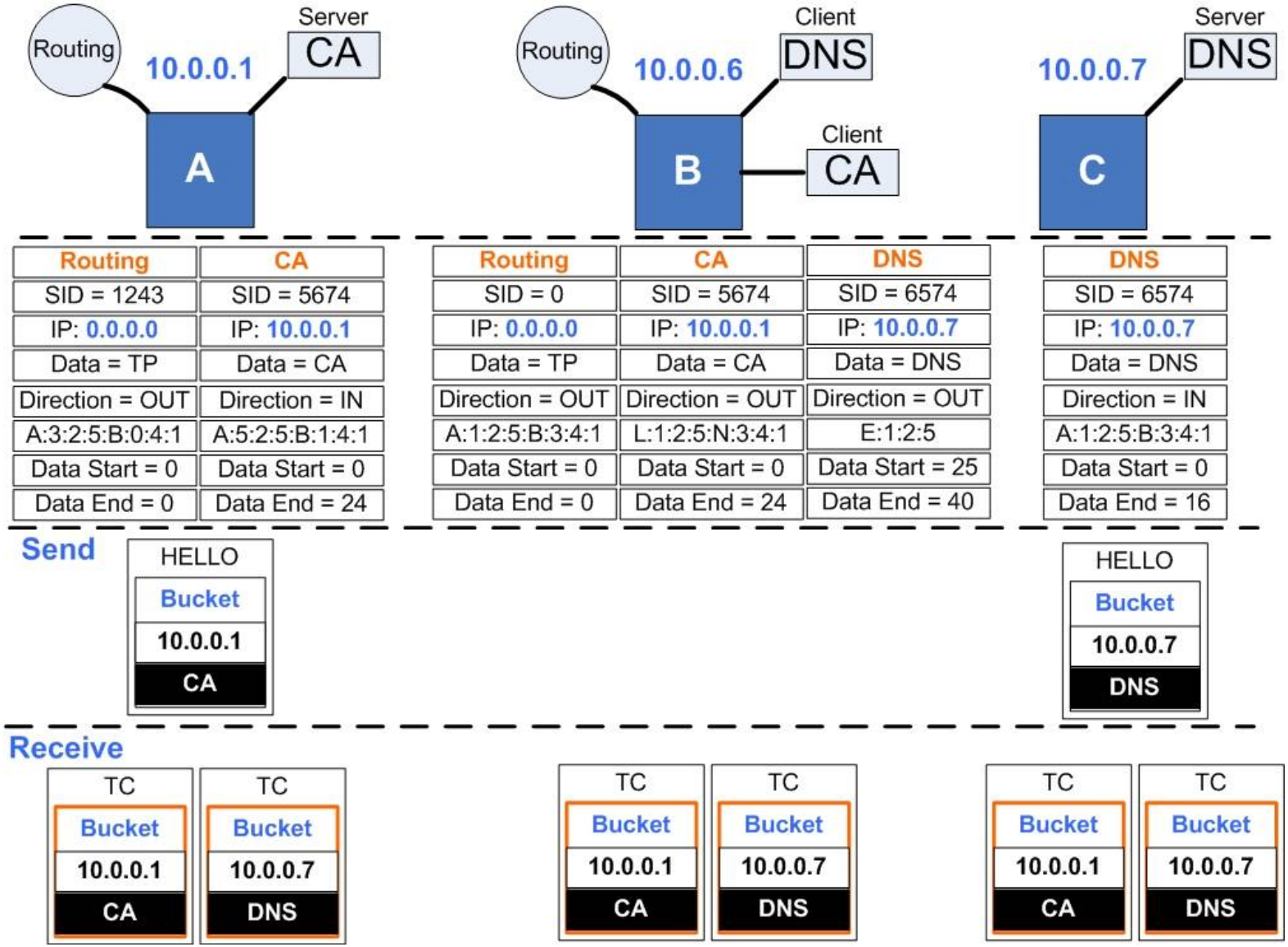

Figure 4.9. Example information dissemination.

Node 7 is responsible for providing Domain Name Service (DNS) updates in the network using the Topology Server process. The ***Data_Start*** and ***Data_End*** pointers are used to locate the data updates in the BIGDATABUFFER repository. DNS information is provided to the Topology Server using local IPC sockets that are established during the registration process. Other nodes in the network can register to the DNS service by using the DNS process's unique Service ID and IP address.

Node 6 contains two client processes, and a separate routing process registered with the Topology Server. Node 6 is using the Topology Server process to receive DNS updates from node 7 and security certificates from node 1. As can be seen in Figure 4.9, the RegistereUsersDB data repository contains Data Record Cards for the two clients. The Data direction is set to OUT in the Data Record Cards, since the DNS and certificate clients are receiving updates from the network and providing these updates to the local clients. Node 6 receives information updates from its MPR, node 5. MPR nodes in the network are responsible for processing and forwarding TC messages with Bucket Data Units for their MPR-selector set. Section 4.11 covers more a detailed discussion of bucket aggregation and dissemination by MPR nodes.

4.2 Client Registration Scheme

New users of the system, register with the Topology Server by sending special Registration Request Messages (RQM) over the **MYREG** IPC socket. The registration request message format is shown in Figure 4.10.

SID : Size : Event : Frequency : Priority : TTL : DataCode : DataDirection

Figure 4.10. Client Registration Request message.

The registration request message starts with a user Service ID that uniquely identifies the process in the network. The ***Size*** is used to allocate data buffers for storing data updates received for the user. Registered ***Events*** are then listed after the size field. Each event has a ***Frequency*** that is used to identify the interval of updates to and from the user. The ***Priority*** field is used to provide different quality of service using the immediate delivery feature of the scheme presented in Section 4.9 of this document.

The ***TTL*** filed indicated the time to live for the update and number of MPR nodes it can propagate in the network. Since each process can act as a client and a server at the same time, the ***DataCode*** field is necessary to identify various traffic flows. The ***DataDirection*** specifies whether this update is going to the internal process or is destined for the network. Different values of SIDs can be used for different quality of service levels provided for message delivery and information dissemination in the network. Quality of service and details of priority based message delivery are explained in Section 4.9.

The TP process continuously listens on the MYREG IPC pipe for new user registration requests (Figure 4.11). Once a RECEIVE event is realized on the MYREG pipe. The ***OnRegPipeEvent*** handler will receive the registration request and pass the data to the ***ProcessRegRequest()*** subroutine.

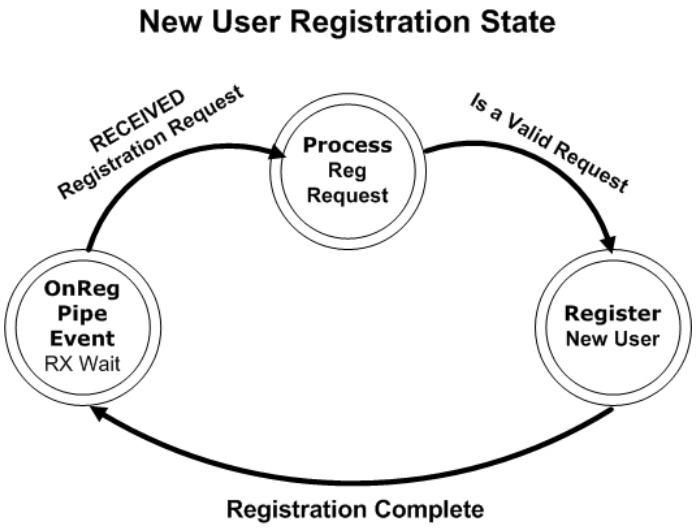

Figure 4.11. New user registration state.

The registration request is further processed based on events and frequencies in the registration request message (RQM). If the user registration is valid and the user does not already have a record in RegisteredUsersDB, the request will be forwarded to the ***RegisterNewUser()*** subroutine that will then create and add a new record in the RegisteredUsersDB. Format of a new user registration record is outlined in Figure 4.12. Service ID field will be populated with the new users SID, and an IP address of 0.0.0.0 is used for users that locally register to the Topology Server using the MYREG pipe.

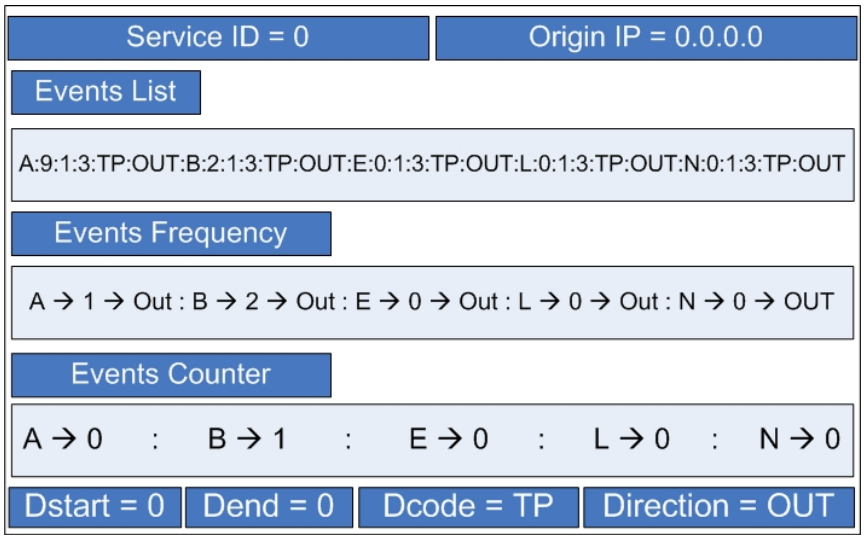

Figure 4.12. Example of a local user's Data Record Card after registration with the TP process

After creating a new record the ***RegisterNewUser()*** subroutine allocates a data unit with the size originally requested by the user in the BIGDATABUFFER data repository. A data pointer is added to the user's data storage record in the Data Record Card. DStart and DEnd fields point to the data storage location for the registered user in the BIGDATABUFFER data repository. The data direction information is used to indicate the direction of the update. If the direction is set to "IN," the update is encapsulated into the Bucket Data Unit for network dissemination. However, if the Data direction is set to "OUT," the data is sent to the locally registered processes based on the appropriate event and with the frequency specified by the user during the registration process. Finally, the total number of registered users is incremented and kept in the Topology Server by the ***ReisterNewUser()*** subroutine.

Each process that is registering with TP process can subscribe to the various events defined in Section 4.1. A list of events to which a user is subscribed and the corresponding frequencies are stored in the user's Data Record Card. An event counter is used to keep track of the occurring events for each registered user. The registration process is then responsible for creating an IPC socket for receiving updates from the Topology Server. The IPC pipe is named after the user's service ID. The overall TP event processing scheme is further explains in Section 4.3.

4.3 Receiving and Processing Updates From Local Users

The Topology Server uses a dynamic data structure for storing information updates. This dynamic buffer holds the data for both local and remote registered processes. Data units are packed in contiguous segments as updates are received. Each data unit's starting location is stored in the appropriate data record for the registered process in the RegisteredUsersDB data structure. As shown in Figure 4.13, each user adds its SID to the beginning of the data segment that it is sending to the TP process. This allows for differentiation between incoming updates received on the IPC socket (the RECV1 pipe) from different users.

SID : DATA

Figure 4.13. Message format used for providing updates to the TP server.

The maximum data segment size is already decided from the registration process. Users are not allowed to send updates to the TP processes with sizes exceeding the maximum data size which they registered.

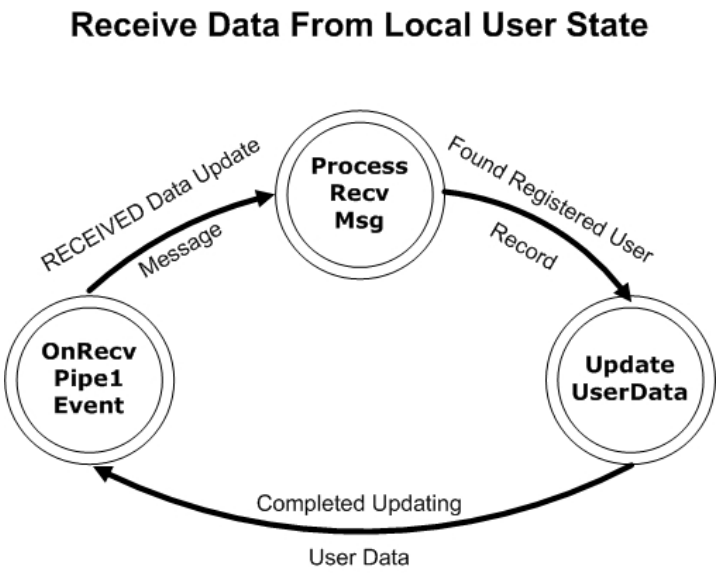

Figure 4.14. Receive data from local user state.

On receiving a new update from the RECV1 socket, the TP process enters the ***ReceivingData*** state. In this state, the data update message is passed to ***ProcessRecvMsg()*** for further processing**.** This subroutine searches through the

RegisteredUsersDB data repository for a matching SID. When a registered user record is located, a pointer to this record is passed to the ***UpdateUserData()*** subroutine.

UpdateUserData() is responsible for updating the BIGDATABUFFER data structure with the received data units from registered users. The DStart pointer is retrieved from user's data record and is used to locate the starting location of the data segment in the in the BIGDATABUFFER. Using the data start location in the BIGDATABUFFER and the data unit size, the newly received message is successfully updated for the registered user. The data stored in the BIGGDATABUFER is later used for Bucket Data Unit encapsulation when disseminating the information in the network.

4.4 Topology Server Event Processing Scheme

The TP process enters the topology event processing state when a change in the OLSR network is detected by the node. The ***ProcessEvent()*** handler searches through the registered users Data Record Cards for possible subscribed users. Each user record contains a list of events together with update intervals, and an event counter. An update flag is set if the event's counter is same as the event frequency for the user.

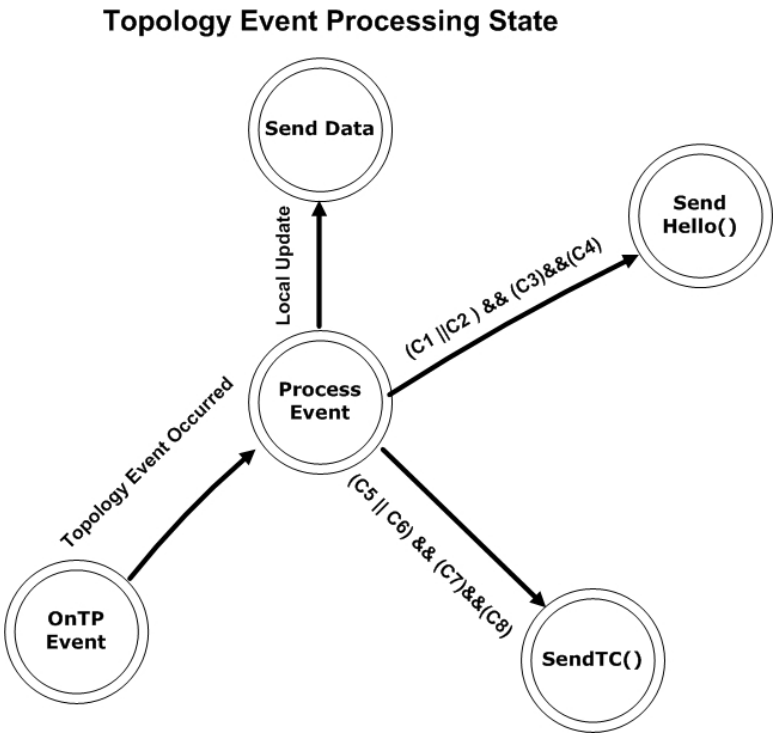

Figure 4.15. Topology Event Processing state triggered based on changes in the network.

The data direction in the user's record card is then used to signal the direction of the update. If the data direction is set to "OUT," the ***SendData()*** subroutine is called. The user's data is kept in the BIGDATABUFFER and a pointer is stored in the user's record card. Data start and Data end pointers stored in user's record card are then used by **SendData()** for retrieving user's data unit. The service ID field in user's data record is used as the IPC pipe name for interfacing with local processes in providing the update. Additionally, the direction field in the user's record card is checked for a network update. If the data direction is set to "IN" to indicate a network update, the **ProcessEvent()** handler then checks the event type for HELLO or TC Events.

If the condition for sending a HELLO Message is set, the TP process would then call ***SendHello()*** subroutine to send a HELLO message with a Data Bucket Unit encapsulating user's update. For condition **CA** to be true the following conditional statements (C4.1) must be true

CA: [(C1 OR C2) AND (C3) AND (C4)] **(C4.1)**

C1: Hello Timeout Event

C2: User with a high priority Service ID has update to send

C3: Regular Node

C4: Data Direction = IN

The ***ProcessEvent()*** handler would transmit a TC message instead if it is running on an MPR node. If the MPR selector set is not empty for the node indicating that the node is an MPR, a TC message is used for Data Bucket encapsulation when propagating user's updating in the network. For condition **CB** to be met, all of the following conditional statements (C4.2) must also be true:

CB: [(C3 OR C4) AND (C5) AND (C6)] **(C4.2)**

C3: TC Timeout Event

C4: User with a high priority Service ID has update to send

C5: Regular Node

C6: Data Direction → IN

4.5 Routing Process Separation

By separating the route selection process from the OLSR protocol, other routing algorithms can be easily implemented using this modular design. The new routing processes can subscribe to the TP process and receive topology updates together with other metrics such as signal-to-noise ratio (SNR) of the wireless links or the power and processing resources of each node.

4.5.1 Implementation

The routing function of the NRL OLSR routing process is separated from the topology discovery function of OLSR in this proposed implementation (see Figure 4.16). The ProtoLib protocol library is used for implementing the new separate routing process in Linux. Both OLSR and the new routing module inherit their basic network and IPC socket communication functionalities from the NRL ProtoLib programming toolkit.

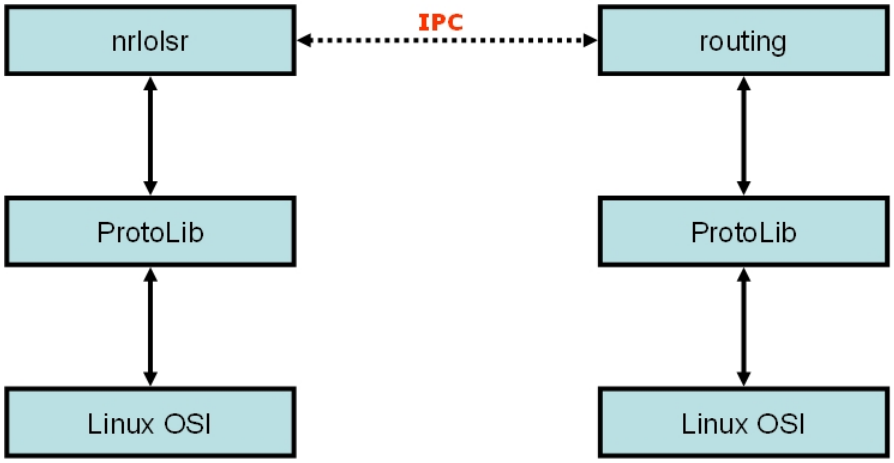

Figure 4.16. ProtoLib programming toolkit used for separating the routing process from OLSR

Figure 4.17 further illustrates a sample registration handshake used for event subscription with the Topology Server. This registration allows the independent routing process to receive topology updates triggered based on OLSR topology change events.

The Routing process starts by first registering with the TP process. The SID used for the routing process is set to 0 in this implementation.

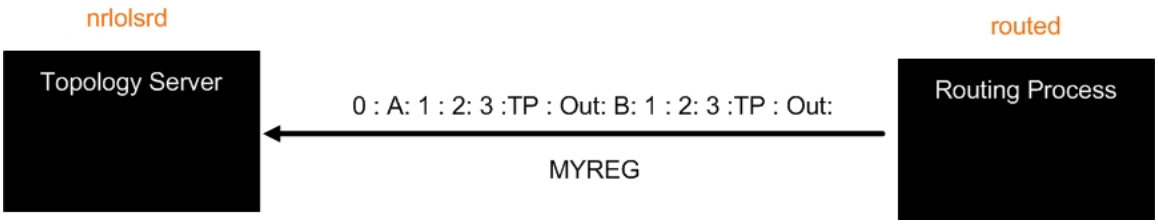

Figure 4.17. Routing process registration handshake with the Topology Server process.

4.5.2 Events Subscription Request

The Routing process registers for OLSR protocol events that indicate a change in the MPR selector set, neighbor table, or link state changes in the topology. These events are made available through the proposed information dissemination programming interface. Table 4.3 contains a more detailed list of events used by the routing process.

Table 4.3. Events used by the Separated Routing Process

Event Code	Event Type	Frequency	Data Code	Direction
A	HELLO Timeout	3	TP	OUT
B	TC Timeout	1	TP	OUT
E	Received HELLO	1	TP	OUT
L	Received Current TC	1	TP	OUT
N	Received Extra TC	1	TP	OUT

4.5.3 Data Update Frequency Settings

The frequency of requested updates from the Topology Server is also defined during the registration state. In this example, the frequency of topology update is set to send updates after every three HELLO messages.

4.5.4 Topology Server Registering of the New Routing User

A new Data Record Card is created for the routing process after a successful registration. The routing process's Record Card is added to the RegisteredUsersDB data repository to be used by the Topology Server when providing event based updates. An example of a routing process Data Record Card is illustrated in Figure 4.18.

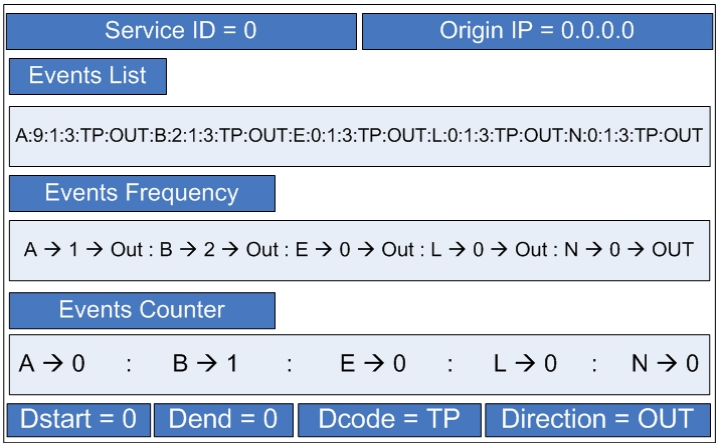

Figure 4.18. Data Record Card used for the separated routing process.

The global Data Direction flag is set to OUT indicating a local process update over the established IPC pipe. The Topology Server uses the SID of the routing process for sending the updates. When an OLSR event to which the routing process has subscribed occurs, the counter is incremented by one unit. When the event counter reaches the event frequency set during the subscription, updates are sent to routing processes using the IPC pipe. Every field in the Data Record Card created for the routing user can be used for data lookup when searching for events in the TP process. The SID field, however, is mainly used by the Topology Server Event handler when sending updates to the routing process.

4.5.5 Sending Topology Updates To The Routing Process

The TP server updates to the routing process are triggered by changes in the topology. Figure 4.19 shows the topology update sent to the routing process.

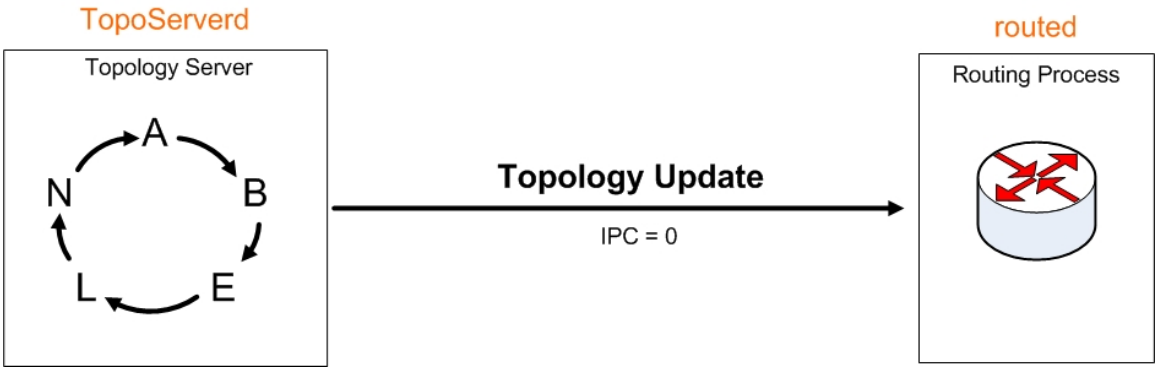

Figure 4.19. Topology updates sent to the separated routing process.

As shown in Figure 4.20, the ***OnTPEvent()*** handler passes the particular OLSR topology event to the ***ProcessEvent()*** for further processing. Since the routing process's Data Record Card indicates a request for a topology updates on the registered Events, the ***SendNBRTopology()*** subroutine is invoked for updating the local routing process.

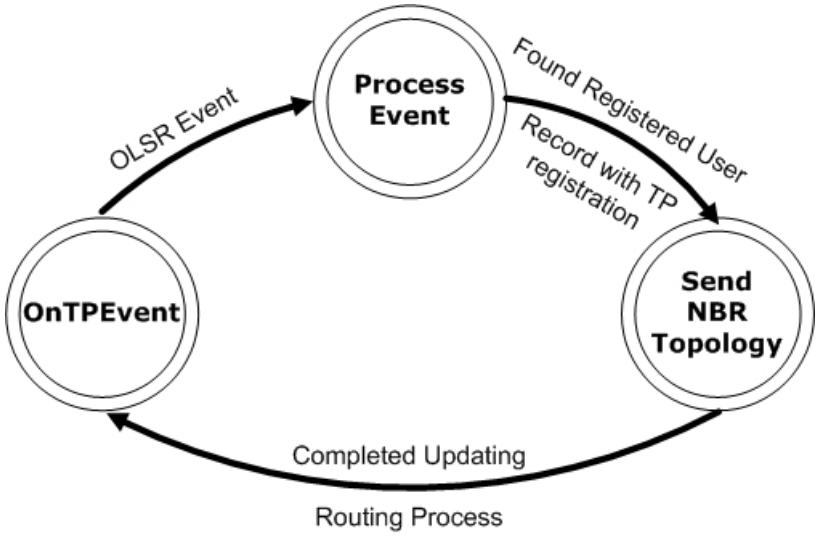

Figure 4.20. Updating the Routing process .

4.5.6 Receiving Updates in the Routing Process

The routing process first starts by parsing the Topology update received from the TP process. The ***OnRoutePipeMessage()*** handler passes the received string to ***RoutePipeProcess()*** subroutine for further parsing. The *RoutePipeProcess()* validates the received topology update and calls the ***Builder()*** subroutine. The *Builder()* subroutine is responsible for rebuilding the topology table and the neighbor information repository based on the received topology message received from the TP process. This process is shown in Figure 4.21.

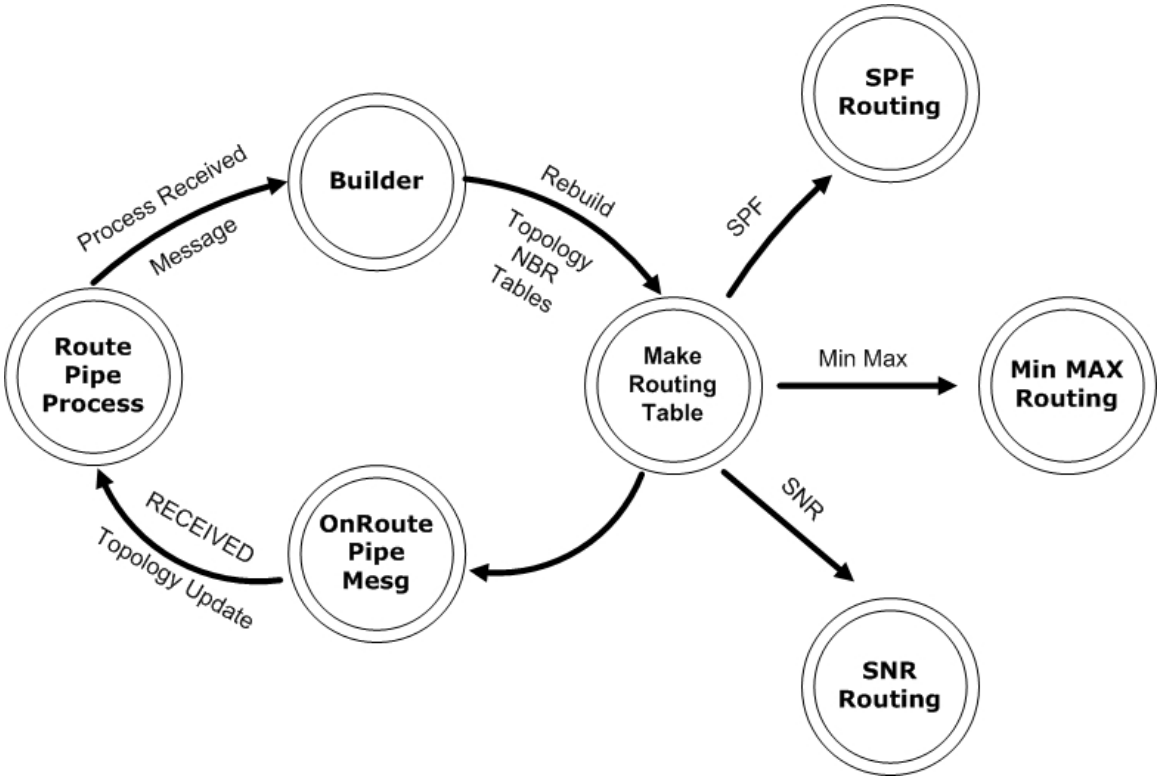

Figure 4.21. Routing process finite state machine.

After populating the topology and neighbor repositories in the routing process the ***makeRoutingTable()*** subroutine is called. The *makeRoutingTalbe()* subroutine is responsible for selecting an appropriate routing algorithm based on predefined settings. Different routing algorithms, such as Shortest Path First (SPF), Min-Max Routing (MMR), and routing based on SNR metrics for the wireless links can be used.

4.6 Bucket Data Unit Message Format

This section presents the proposed Bucket Data Unit layout. The Bucket Data Units are used to disseminate information in the topology using HELLO and TC messages sent by OLSR routing protocol. First, a brief discussion of OLSR HELLO and TC message formats is presented.

4.6.1 OLSR Packet Format

All OLSR traffic is sent in OLSR packets. These packets consist of an OLSR header and a body as displayed in Figure 4.22. An OLSR packet body contains of one or more OLSR messages.

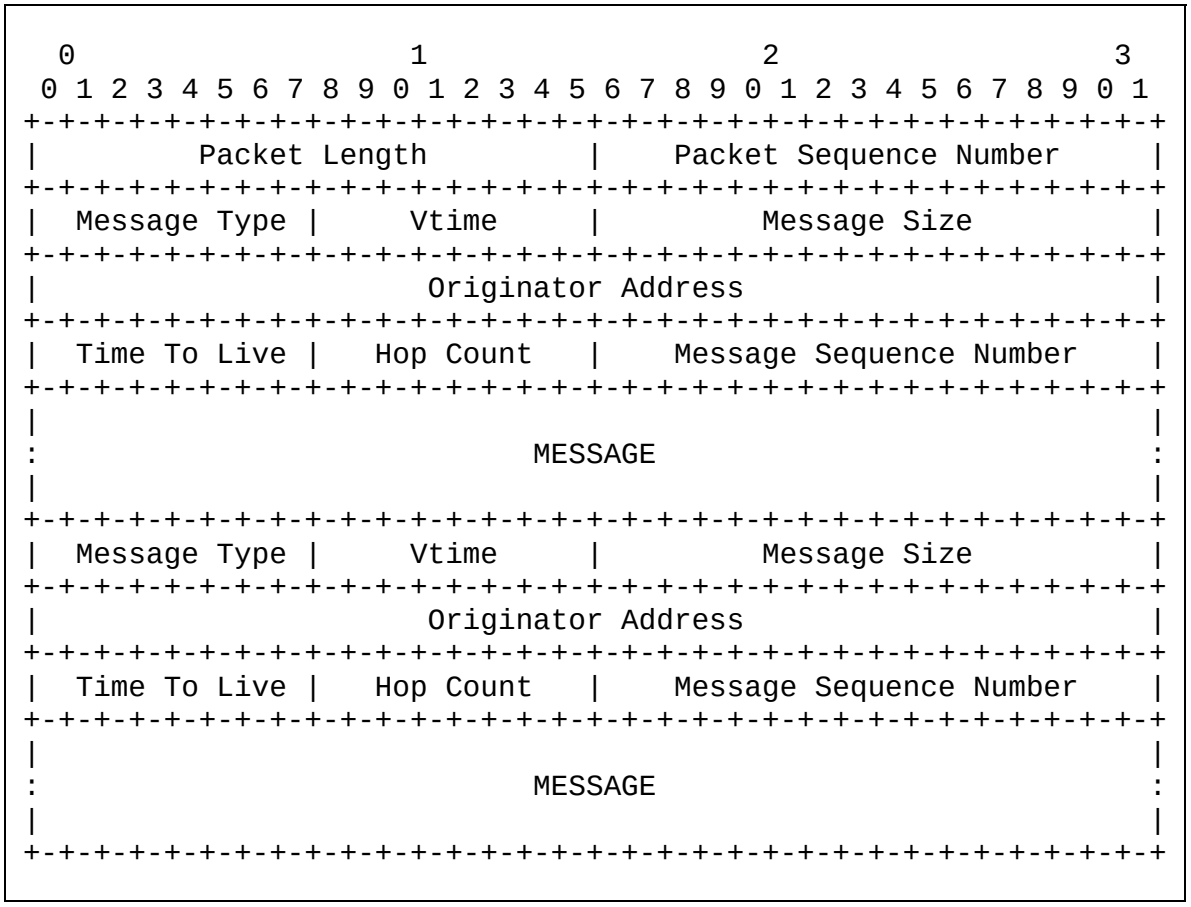

Figure 4.22. OLSR packet format and OLSR message encapsulation.

HELLO and TC messages are then encapsulated into the message field of an OLSR packet to be forwarded in the network. Participating nodes use the ***Message Type*** field to determine how to process received messages. Messages can be flooded onto the entire network or flooding can be limited to nodes within a diameter (in terms of the number of hops) from the originator of the message. The nodes in the network can examine the header field of a message to obtain information on the distance and the originator of the message. Periodic HELLO messages are sent by nodes to announce and propagate local links and neighbors. MPR nodes are responsible for processing and forwarding TC

messages in the network. Using periodic TC messages, network topology changes are disseminated by the MPR nodes.

4.6.2 HELLO Message Format

The layout of an OLSR HELLO message is shown in Figure 4.23. OLSR HELLO messages are used for link state propagation, neighbor detection, and MPR selection signaling.

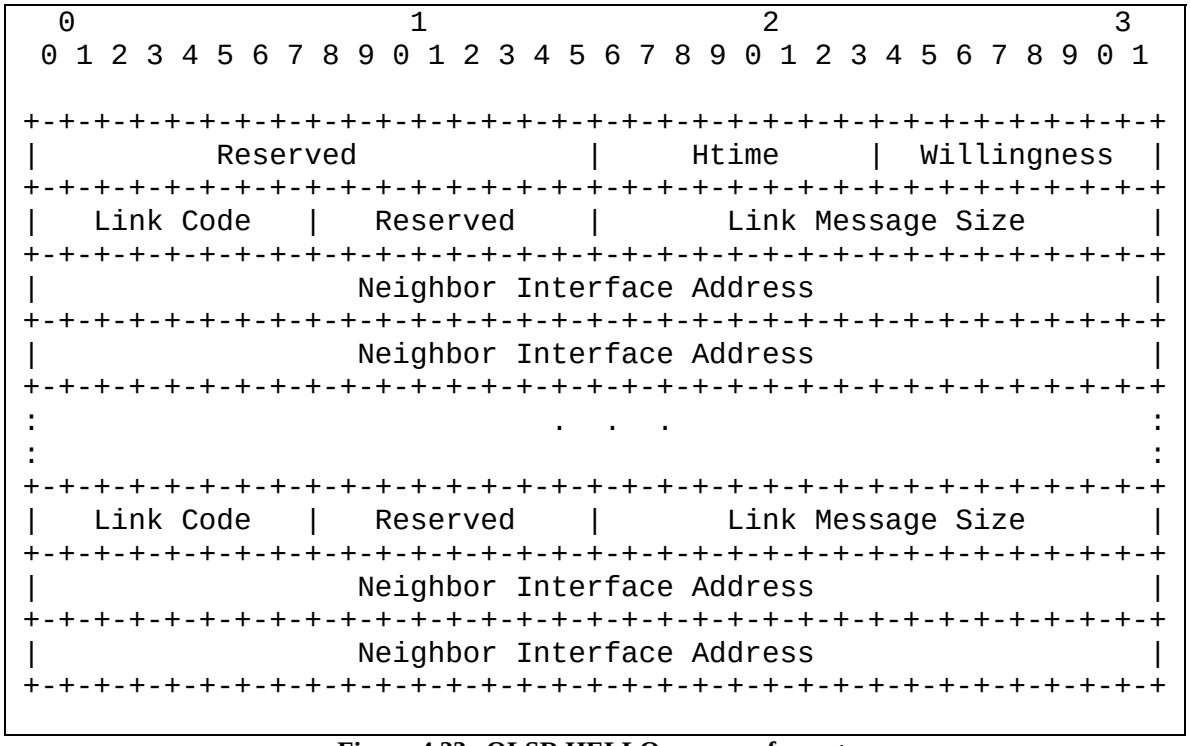

Figure 4.23. OLSR HELLO message format.

To facilitate information dissemination over OLSR, we propose encapsulating special Bucket Data Units in outgoing HELLO messages. Local processes can reserve Bucket Data Unit's during the registration process with the Topology Server process. The TP process then uses the RegisteredUsersDB information repository to retrieve a user's data updates. Information updates are formatted into Data Bucket Units and are encapsulated into outgoing HELLO messages.

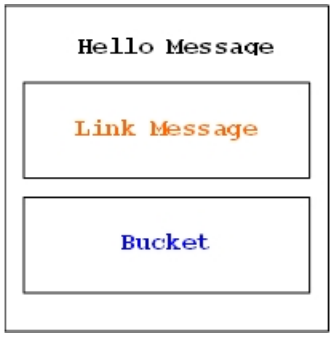

Figure 4.24. OLSR HELLO message Bucket Data Unit encapsulation.

This provides an extensible framework for local processes to send information updates using the existing HELLO messages provided by OLSR. A format of a HELLO message encapsulating Data Bucket Unit is as shown in Figure 4.25.

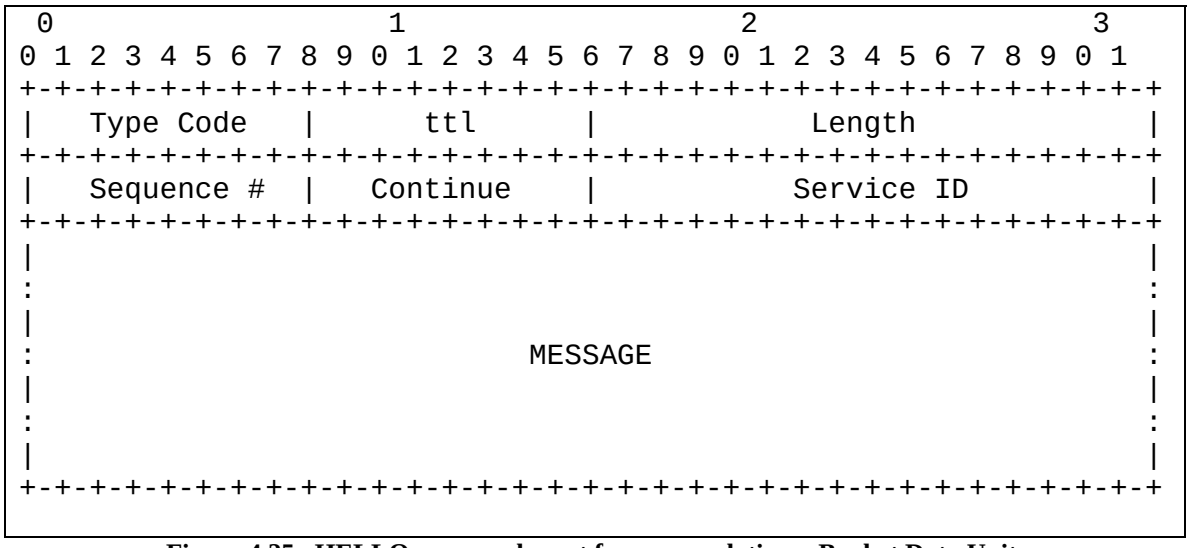

Figure 4.25. HELLO message layout for encapsulating a Bucket Data Unit.

The ***Type Code*** is used to indicate the start of an information BUCKET encapsulated in an OLSR HELLO message. The ***TTL*** field is used to specify the number of MPR nodes an information BUCKET is allowed to travel. The ***Length*** field contains the length of the information update. A ***Sequence*** field is used to store a unique sequence number for each update. The ***Continue*** is used together with the sequence number field to mark fragmented information buckets. Section 4.8 covers the proposed Bucket Data Unit fragmentation scheme. To further relate a Bucket Data Unit to a service provider in the topology, a unique ***Service ID*** field is used. The raw data update is encapsulated in the

MESSGAGE field of the Bucket Data Unit. Overall, each data bucket unit is adding 8 bytes of overhead. A more detailed discussion of performance and overhead based on the experimental evaluation is provided in Section 5.3.

4.6.3 TC Bucket Format

MPR nodes receive periodic HELLO messages from nodes in their MPR selector set and generate TC messages based on changes noticed in the neighborhood. Figure 4.26 shows format of a OLSR TC message.

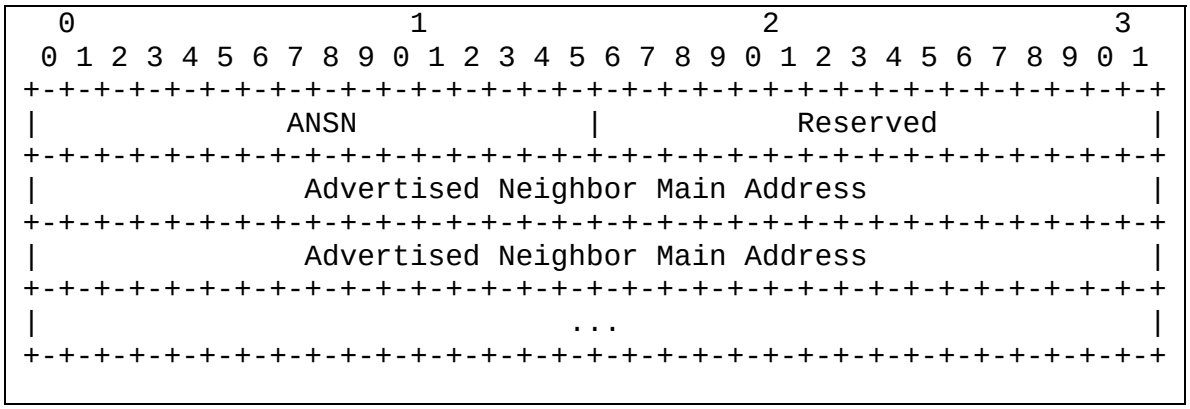

Figure 4.26. OLSR TC message format.

As outlined in Section 4.11, MPR nodes are responsible for information dissemination and forwarding the Data Bucket Units received from nodes in their MPR selector set. As also outlined in Section 4.11, MPR nodes extract the data in HELLO Buckets received from nodes in their MPR-selector set, and generate special TC Bucket Data Units. The TC Bucket Data Units are then encapsulated into the outgoing TC messages generated by MPR nodes and forwarded in the network by other MPR nodes. Figure 4.27 shows a high-level view of Bucket Data Unit encapsulation in a TC message.

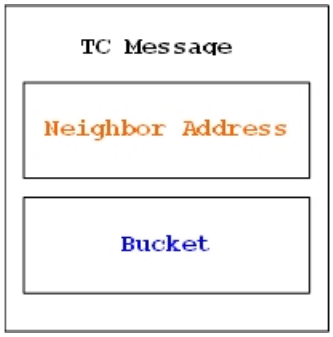

Figure 4.27. TC message encapsulation of a Bucket Data Unit.

The information Bucket Data Units are encapsulated into outgoing OLSR TC messages after the last advertised neighbor address. The IP address of the originating process is added to the Bucket header. This allows for multiple information servers to use the Topology Server for information disseminate over OLSR in the network. By encapsulating special Bucket Data Units into outgoing TC messages information updates are disseminated in the network using the existing topology dissemination infrastructure provided by OLSR. Each TC Bucket Data Unit adds 12 bytes of overhead. The format of a TC Bucket Data unit is shown in Figure 4.28.

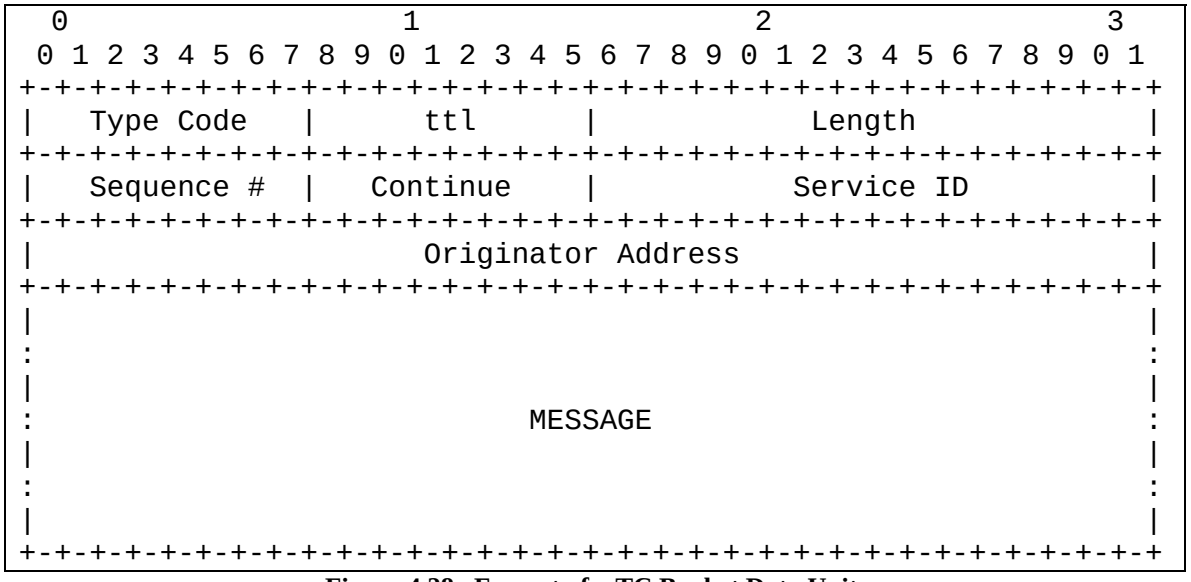

Figure 4.28. Format of a TC Bucket Data Unit.

The Message size in each Bucket Data Unit is reserved during the registration process with the Topology Server. The scheme allows for extensible MESSAGE sizes with a maximum size defined at compile time.

4.7 Information Dissemination Using Buckets

A Topology Server generates Bucket Data Units and encapsulates the bucket into outgoing HELLO and TC messages. Each registered user that has data to send is allocated a unique Bucket Data Unit in the outgoing HELLO or TC messages. As outlined in Section 4.4, a registered User's data record, shown in Figure 4.29, is checked before a Bucket Data Unit is generated for TC or HELLO encapsulation.

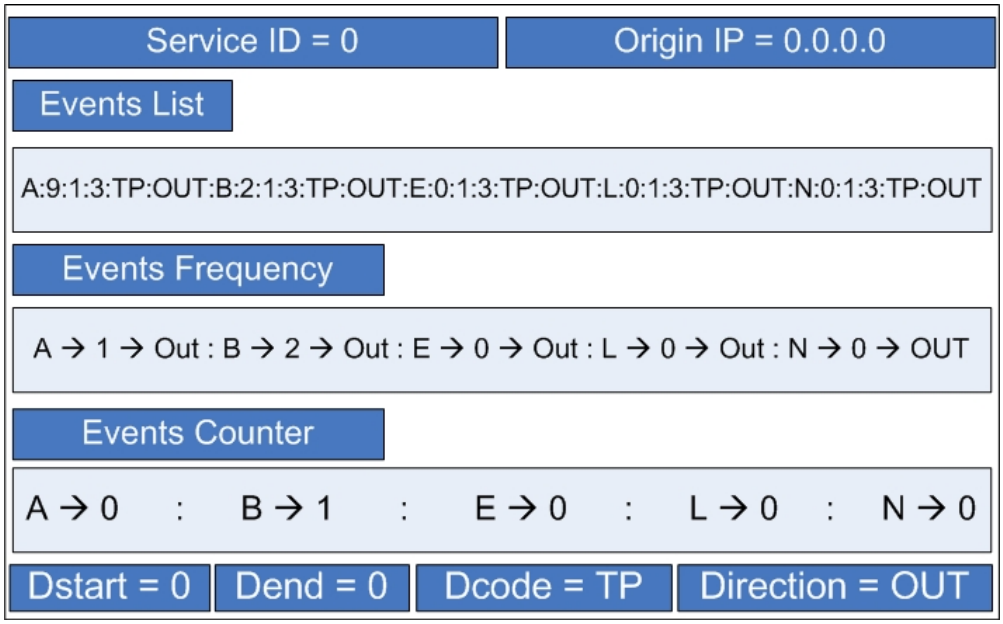

Figure 4.29. User's Data Record Card.

4.7.1 TC Bucket Data Unit Information Encapsulation

The Topology Server process in an MPR node first retrieves a list of registered users from the locally stored RegisteredUsersDB data repository. This list contains a complete record of both local and remote processes with valid registration to the ***TC_SEND*** event as described in Table 4.1. These records are checked for data direction and frequency. If the data direction filed in user's data record card is set to "IN" and the user's event counter is same as the frequency of update, a new Bucket is generated for network information dissemination. Figure 4.30 contains the logic added to the *SendTC()* subroutine in the NRL implementation of OLSR protocol to accommodate the new Bucket Data Unit encapsulation scheme.

```
-------------------------------------------------------------------------------
      Send TC
      {
        For ( Items in Service DB )
          {
                If ( Direction == IN ) and ( Frequency == Counter )
                {
                        TC.addBucket( BigBuffer, Data_start, Data_End
      )
                      // This encapsulates a new Bucket Message into
                      a TC Message
                }
         }
        socket.SetTTL(1);
        socket.SendTo(buffer, len, broadAddr);
      }
-------------------------------------------------------------------
```

Figure 4.30. Algorithm used for Bucket encapsulation in outgoing TC messages.

The ***addBucket()*** retrieves the data pointer of the user's data stored in the *BIGDATABUFFER* repository. This data is added into a Bucket Data unit. The new Bucket Data Unit is then encapsulated into a TC message. The Originator_Address field in a TC Bucket Data Unit is set to the registered process's IP address. For a local process, an IP address of 0.0.0.0 is used. The Service ID field is also obtained from a user's Data Record Card and populated accordingly. Using different Service IDs, Servers and Clients of a particular service can easily locate each other in the network. This provides for more extensible service discovery in ad hoc networks using OLSR for topology and information dissemination. The TypeCode is set to ***TC_BUCKET_MSG = 0xff***. This type also indicates the start of a new TC Bucket Data Unit. A global sequence number is used for Bucket generation and is incremented after each successful bucket encapsulation. The same sequence number is also used to set the Sequence number field of a Bucket Data Unit.

4.7.2 HELLO Bucket Data Unit Information Encapsulation

For non-MPR nodes, a HELLO Bucket message is generated for information dissemination. Using HELLO Bucket messages, nodes can notify their MPR of new information updates. MPR nodes receive HELLO Bucket Data Units from nodes in their MPR selector set and generate TC messages with TC Bucket Data Units. The TC Bucket Data Units contain updates received from the MPR-selector set is then forwarded by the MPR nodes through the network. Similar logic is added to the ***SendHello()*** subroutine to accommodate the new HELLO Bucket Data Unit encapsulation. Figure 4.31 shows the algorithm used for encapsulating buckets into outgoing HELLO messages.

```
-------------------------------------------------------------------------
      Send HELLO
      {
        For ( Records in Service DB )
                    {
                If ( Direction == IN ) and ( Frequency == Counter )
                {
                    HELLO.addBucket( BigBuffer, Data_start,
      Data_End )
                      // This encapsulates a new Bucket Message into
                      a Hello Message
                }
           }
        socket.SetTTL(1);
        socket.SendTo(buffer, len, broadAddr);
      }
-------------------------------------------------------------------------
```

Figure 4.31. Send Hello algorithm used for bucket encapsulation into OLSR HELLO messages.

The ***addBucket()*** subroutine retrieves a list of registered users and uses their data record for populating the HELLO Bucket Data Unit. Type codes are used to indicate start of a new HELLO Bucket Data Unit. The Data pointer to user's data starting location in the BIGDATABUFFER repository is used to retrieve the user's information update. This information update is then added to the MESSAGE field of the newly generated HELLO Bucket Data Unit. The sequence number for the Bucket is also incremented after each successful transmission. The Buckets are encapsulated into the outgoing HELLO

messages and then broadcasted through the network. The Bucket encapsulation process is shown in Figure 4.32.

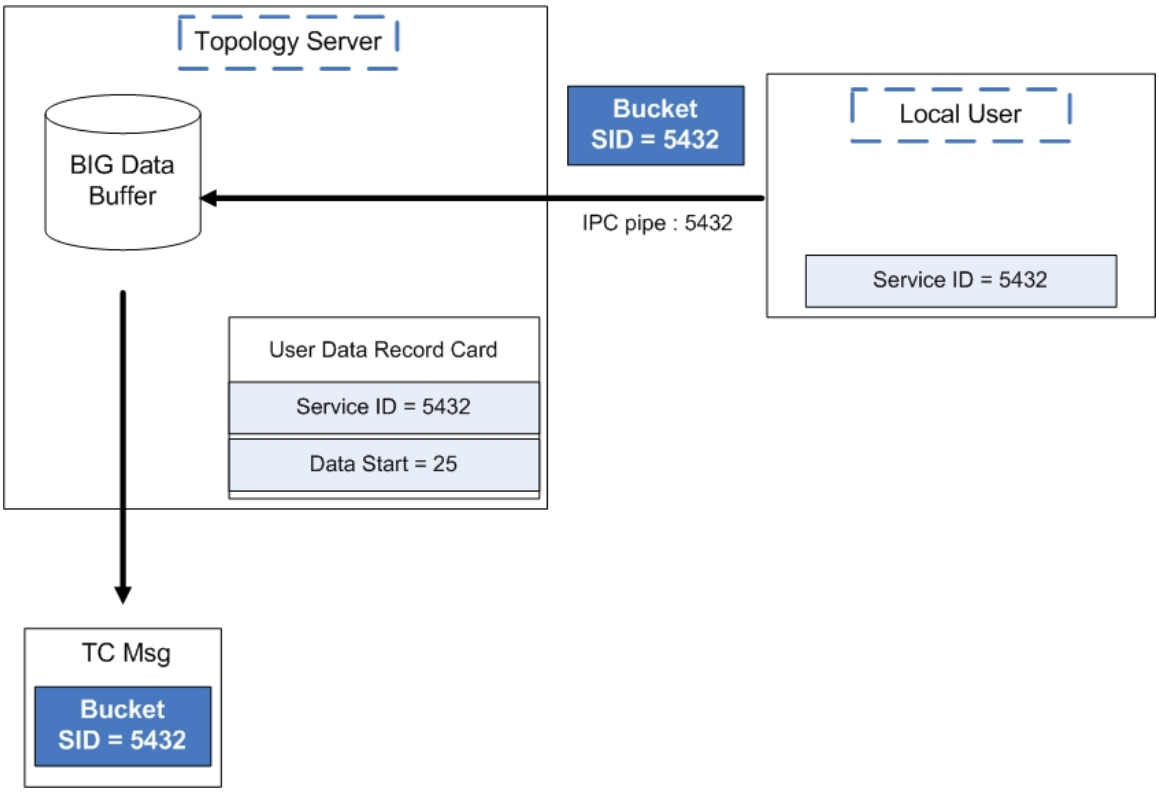

Figure 4.32. OLSR message encapsulation process based on received updates in the TP process.

4.7.3 Information Processing and Data Extraction

The processing logic used when receiving HELLO and TC message depends on the node type. MPR nodes follow a different processing flow than non-MPR nodes. If the TC message is received by an MPR node, it first creates a local copy of the received packet for processing and forwards the original TC message. By using a local copy and immediately forwarding the original copy, faster network convergence can be achieved when disseminating information updates in the network over OLSR. In processing TC messages with Bucket Data Units, MPR nodes first check for registered users in the RegisteredUsersDB data repository. If a registered user is found the corresponding Data Record Card is used for storing the user's update in the BIFDATABUFFER local repository. A pointer to the user's data location in the repository is retrieved from the user's Data Record Card. This data pointer is usually set when a new user registers with

the Topology Server. If a record is not retrieved by the MPR for the ServiceID and IP address of the Bucket Data Unit, the Bucket Data Unit will not be processed locally, but is still forwarded to reach other nodes through the network. Figure 4.33 outlines the algorithm used for processing the received Bucket Data Units encapsulated in OLSR TC messages. When non-MPR nodes receive a TC messages, they go through the buckets in the message and follow the same processing algorithm locally. In this case there is no need for forwarding the message in the network, since only MPR nodes are in charge of topology and information dissemination.

```
TC Received()
{
   If(MPR Node)
   {
     Copy_TC ()
     Forward_TC()
     Process(Copy_TC);
         If (New Bucket)
         {
                If(Found Registered User)
                { / / Existing Service ID
              Update(User Data);
             }
         }
   }
 Else
   { / / Not an MPR node
         If (New Bucket)
         {
                If(Found Registered User)
                { / / Existing Service ID
              Update(UserData);
                }
     }
   }
}
```

Figure 4.33. Algorithm used for processing a bucket encapsulated in OLSR TC message.

The ***SendData()*** subroutine is called to provide the update to the appropriate registered user over the IPC communication pipes. The local user's IPC socket name is same as user's Service ID (SID), which is kept in each user's Data Record Card. If it is determined that the data update is destined to a remote process, the information is

updated in the user's data storage in the BIGDATABUFFER data repository. The update starting location in the BICDATABUFFER is stored in the user's Data Record Card.

An MPR node would create a new Data Record Card and register a new user if it is unable to locate a record that matches the Service ID of the Bucket Data Unit received in a new HELLO message. This allows MPR nodes to create a Bucket Data Unit for users and disseminate the update through the network on the next scheduled TC Message. A new user is registered to the Topology Server by simply adding a new Data Record Card with the Service ID field set to Data Bucket Unit's Service ID. Figure 4.34 outlines the population of a Data Record Card based on a Bucket Data Unit.

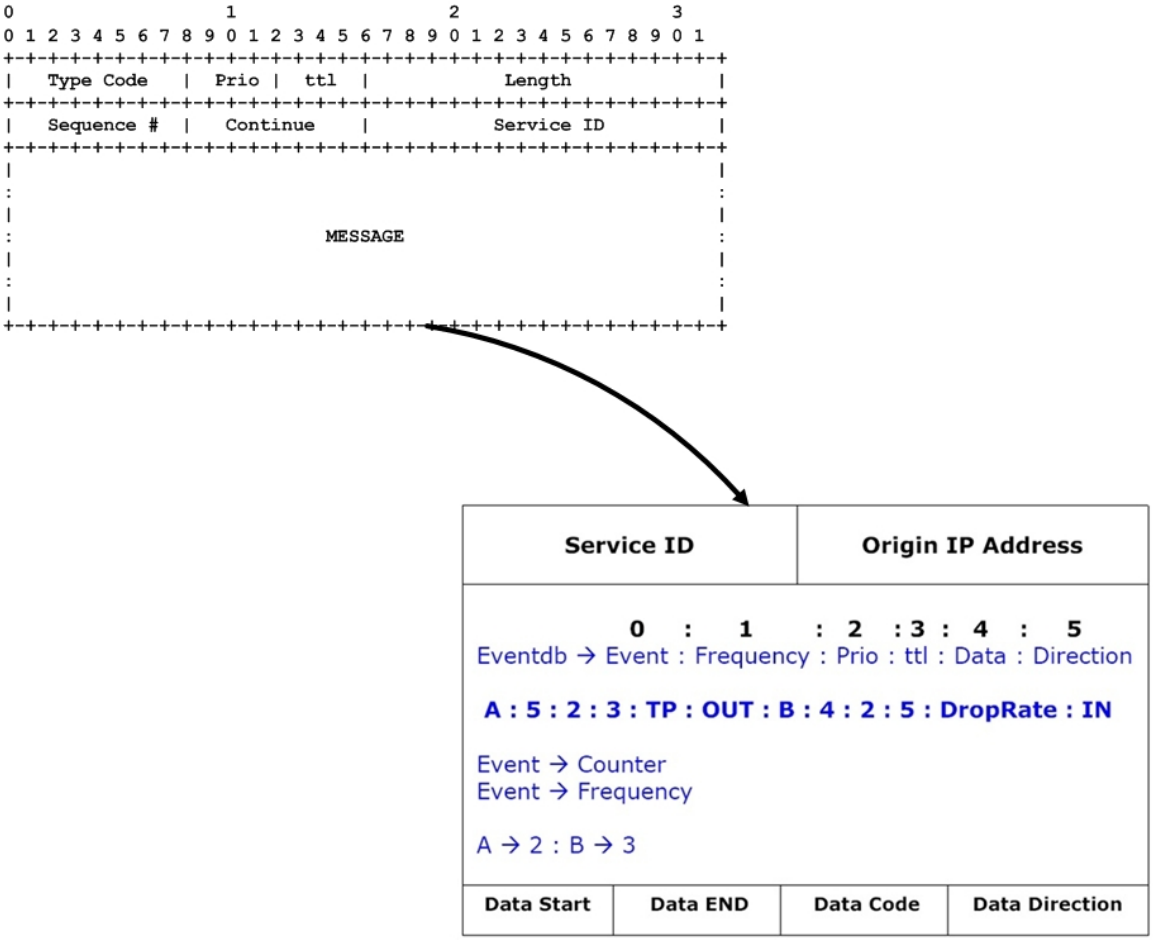

Figure 4.34. Data Record Card generation based on a received Bucket Data Unit.

Non-MPR nodes only check the list of locally registered users when they receive a new HELLO message. The Bucket Data Unit is ignored if nodes fail to locate a record of a

local user. However, when a record is retrieved for registered users the update is sent to the user using its open IPC pipe. This process is outlined in Figure 4.35.

```
Hello Received
{
 If( MPR )
    {
          If (New Bucket)
          {
               RegisterNewUser()
               Update (ServiceDB)
          }
          Else
          {
                If(Local User)
                   {
                     SendData(Update);
                   }
                Else
                   {
                    Update(BigDATABUFFER)
                   }
          }
     }
If ( Not MPR )
    {
          If (New Bucket)
          {
             Ignore();
          }
          Else
          {
                If(Local User)
                   {
                     SendData(Update);
                   }
          }
    }
}
```

Figure 4.35. Hello Bucket Data Unit extraction.

4.7.4 Providing Local Updates to Registered Users

The Topology Server sends data updates to local processes as they arrive in Bucket Data Units encapsulated in TC or HELLO messages. The data is first stored in the BIGDATABUFFER repository. The data update's starting location is stored in the user's data record card. The Bucket Data Unit is then forwarded to the local process using the IPC socket established during the registration process with the TP process. The user's SID is used for naming IPC pipes. Upon receiving a network update, the Topology Server retrieves the local user's Data Record Card from the RegisteredUsersDB repository and uses the Services ID field to open an IPC socket for sending updates to the local user. The information updates are sent as soon as the Bucket Data Units (BDUs) are processed. Figure 4.36 illustrates the process of forwarding the Bucket Data Unit to a local user

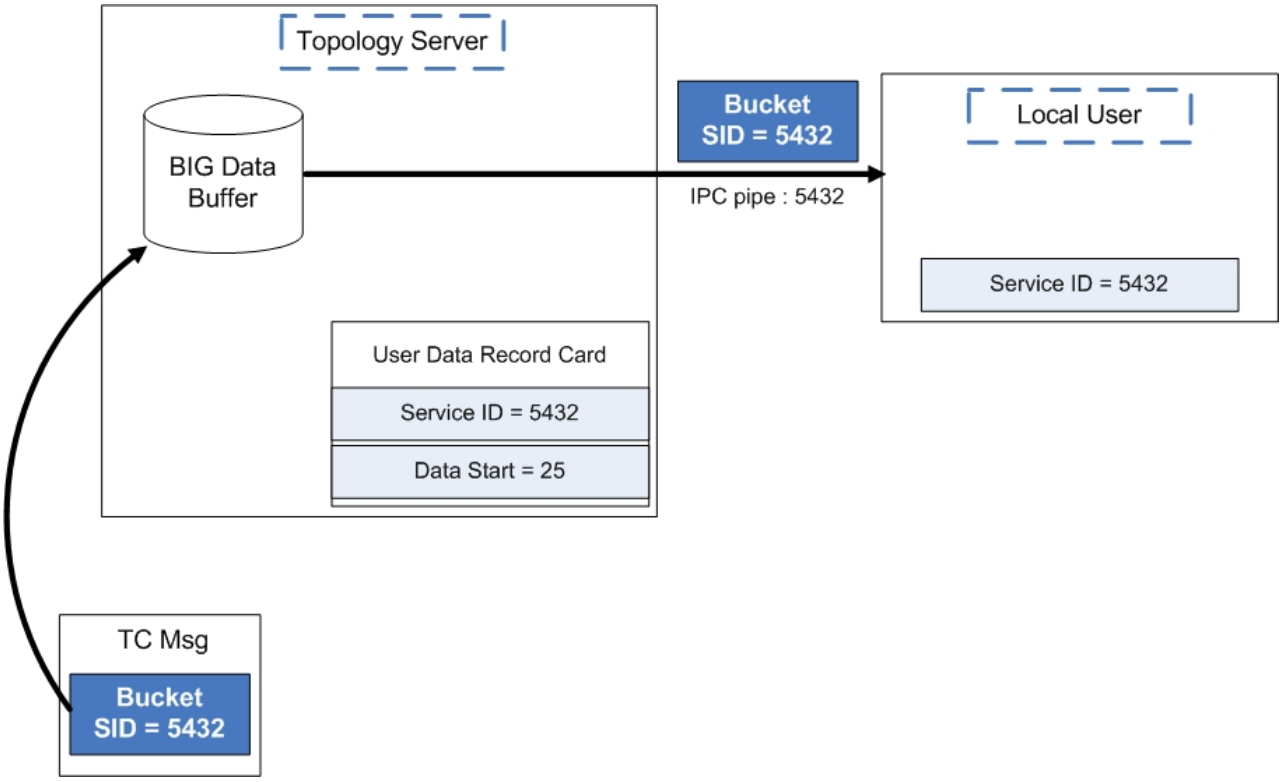

Figure 4.36. Updating local users based on Bucket Data Units received in HELLO and TC messages.

4.8 TC Bucket Fragmentation

For users with information updates exceeding the fragmentation threshold, multiple contiguous TC messages are sent using fragmented BDU segments. This provides an extensible delivery scheme for information dissemination in the network. The bucket fragmentation technique also ensures on time data delivery for users that require extended data sizes for service offering also improving throughput when there is a MAC layer fragmentation threshold limitation.

4.8.1 TC Bucket Fragmentation Scheme Overview

The Continue field in the TC Bucket Data Unit is used to indicate the start and end of a fragmentation flow. The Sequence number of the Bucket Data Unit is incremented for each fragment. The Continue field is used both as the fragment offset and as a flag to indicates that more fragments are to be transmitted. Figure 4.37 illustrates Bucket fragmentation at an MPR node. In this example, the information update received from the local user process is four times the fragmentation threshold defined in the system. Hence, the Topology Server generates four TC messages, each containing a Bucket Data Unit. The Continue field is incremented for each Bucket Data Unit. This field is also used when reassembling the information from the data fragments at the destination.

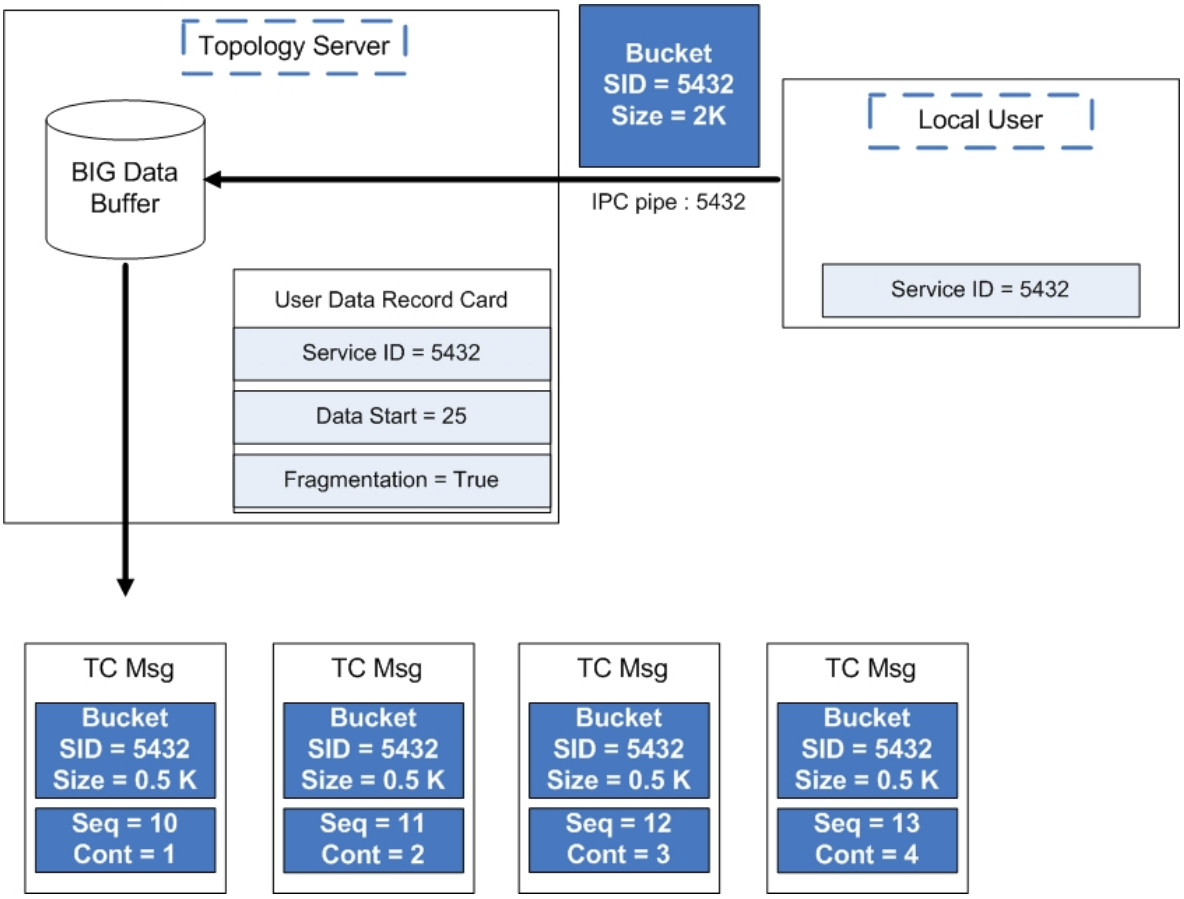

Figure 4.37. Bucket Data Unit fragmentation.

4.8.2 Reassembling Information from Received TC Bucket Fragments

As outlined in Figure 4.38, the fragmented Buckets are reassembled at the receiving node using the Continue field in each Bucket Data Unit. The user's Data Record Card is used to obtain the starting location of the data storage pointer in the BIG DATA BUFFER repository. Received fragments are then put together using their sequence number together with the Continue field in each Bucket.

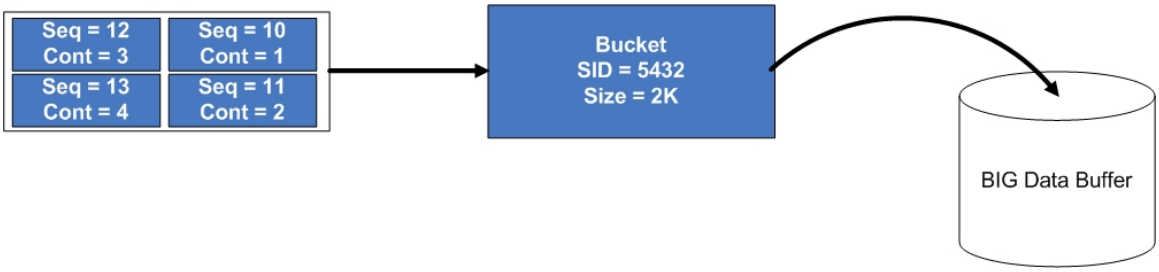

Figure 4.38. Reassembling information updates from fragmented buckets.

The fragment reassembly algorithm is shown in Figure 4.39.

```
-------------------------------------------------------------------------------
If(Bucket.Continue != 0)&&(First Bucket)
{//Fragmentation Case → First Bucket
        Current_Location = User_Data_Start_Pointer
        Update(BIGDATABUFFER, Current_Location,
        Bucket_Message,frag_threshold)
}
Else if (Bucket.Continue != 0)&&(During Fragmentation)
{// During Fragmentation
        Current_Location = Continue x Fragmentation_Threshold
        Update(BIGDATABUFFER, Current_Location,
        Bucket_Message,frag_threshold)
}
Else if (Bucket.Continue != 0)&&(Last Fragment)
{// Last Fragment
        Current_Location = Continue x Fragmentation_Threshold
        Update(BIGDATABUFFER, Current_Location,
        Bucket_Message,Data_end-Current)
```

Figure 4.39. Fragmentation reassembly algorithm.

For the first received fragment, the ***Current_Location*** variable is set to the data starting location pointer. The Message portion of the received Bucket Data Unit is copied into the BIGDATABUFFER repository. The copy size is set to fragmentation threshold for the first received fragment. The Continue field of the Bucket Data Unit is set to the *Current_Location* variable. The *Current_Location* is then updated to point to the start of the next following fragment location in the BIGDATABUFFER. Equation 4.1 illustrates the *Current_Location* calculation. Figure 4.40 shows the reassembly process and update of local users.

Current_Location = Continue x Fragmentation_Threshold (4.1)

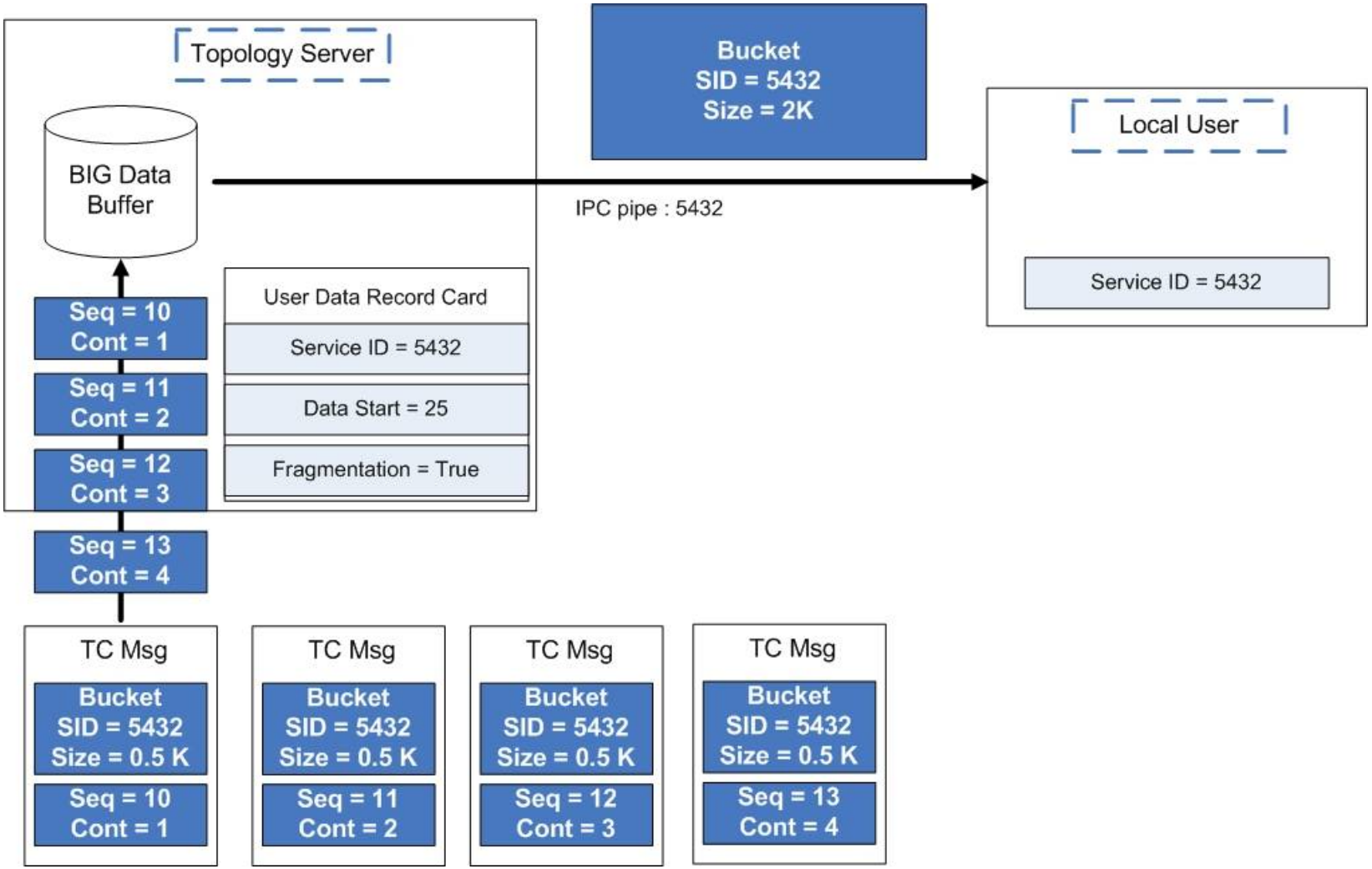

Figure 4.40. Example fragmentation reassembly and update.

4.9 User Priorities and Service IDs

Different levels of quality of service can be achieved when disseminating information updates through the network. Table 4.3 specifies quality of service levels used in this work, which are called “Bronze,” “Silver,” “Gold,” and “Platinum.” Bronze is lowest level and Platinum is the highest level service. The ***SendNow()*** subroutine is called from the *OnTPEvent()* handler when processing different OLSR events to better server users with priority registrations. The *SendNow()* subroutine can generate out of sequence HELLO and TC messages to accommodate faster information dissemination through the network.

Table 4.3. Quality of Service Levels Based on Registered Service IDs

Variable	Bronze	Silver	Gold	Platinum
TC Factor	0.5	0.25	0.1	0.05
HELLO Factor	0.5	0.25	0.1	0.05
Service IDs Range	5555 to 6666	6666 to 7777	7777 to 8888	8888 to 9999

Two new system variables ***Immediate_TC_Factor*** and ***Immediate_HELLO_Factor*** are defined and used by the ***SendNow()*** subroutine when adjusting the immediate delivery times. These variables have a direct impact on the delay noticed by clients and the overall information network convergence time. The time before the next priority TC and HELLO is set using the immediate factor variables using Equation 4.2.

$$\textbf{TC\_wait\_time} = (\textbf{immediate\_TC\_factor}) \times (\textbf{TC\_Interval}); \qquad (4.2)$$
$$\textbf{Hello\_wait\_time} = (\textbf{immediate\_Hello\_factor}) \times (\textbf{Hello\_Interval});$$

The *SendNow()* subroutine generates a new OLSR Message depending on the node type. If the Topology Server process is running on a MPR node a TC message is generated. For non-MPR nodes, a HELLO message is used to propagate the priority update in the network. The Bucket Data Units for the user process are encapsulated into the OLSR messages in a manner similar to that of a regular delivery case.

The *SendNow()* subroutine can be called from the ***ProcessEvent()*** handler when processing OLSR events for users with priority registration.

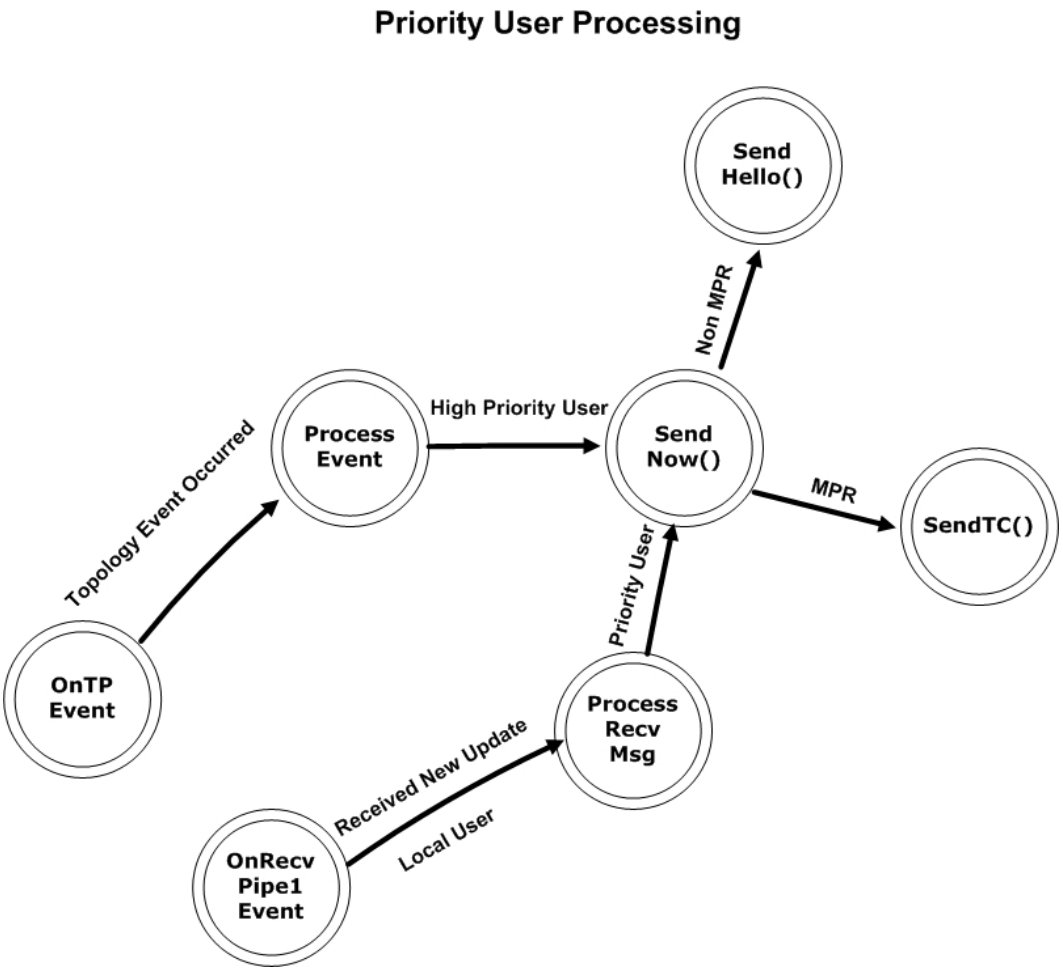

Figure 4.41. Finite state machine view of SendNow() handler.

Information updates received from local processes using priority Service IDs are also processed by *SendNow()*. Finally, an MPR node can call *SendNow()* when processing received HELLO messages with Bucket Data Units that have priority Service IDs. This can expedite the propagation of a new data received from a node in the MPR-selector set. Figure 4.41 shows the finite state diagram of the *SendNow()* process handler. The algorithm for *SendNow()* is shown in Figure 4.42.

```
---------------------------------------------------------------------------------------
        SendNow(int ServiceId)
        {
                time_left=0;

                TC_wait_time = (immediate_TC_factor)*(TC_Interval);
                Hello_wait_time = (immediate_Hello_factor)*(Hello_Interval);

                If(MPR)
                {
                        time_left= Next_TC_Time;

                        if (time_left>TC_wait_time)
                          {

                                SendTc();
                          }

                }
                Else
                {//Non MPR
                        time_left= Next_HELLO_Time;

                        if (time_left>Hello_wait_time)
                       {
                           SendHello();
                        }

                }
---------------------------------------------------------------------------------------
```

Figure 4.42. Algorithm used for immediate delivery of Buckets using TC and HELLO messages.

The ***time_left*** variable is set using OLSR protocol timers for the periodic TC and HELLO message generation. If the *TC_Wait_time* value is larger than the wait time before the next scheduled TC message, a new priority message is generated.

4.10 Information Dissemination Radius

The radius of the information dissemination through the network is determined from the Bucket Data Unit's time to live (*TTL)* field. The MPR nodes are responsible for forwarding the Bucket Data Units encapsulated in OLSR TC messages. By setting the

TTL field for each user, the Topology Server on the MPR node can limit the maximum number of MPR hops the Buckets are allowed to traverse. Figure 4.43 shows how the MPR nodes adjust the TTL field accordingly in the received buckets.

Before Forwarding the Bucket Data Units received for each user, the *TTL* is checked by the Topology Server of an MPR node. If the value in the *TTL* field of the Bucket Data Unit is set to one, the MPR node stops forwarding the Bucket Data Unit. This effectively limits the radius of a new information update through the network.

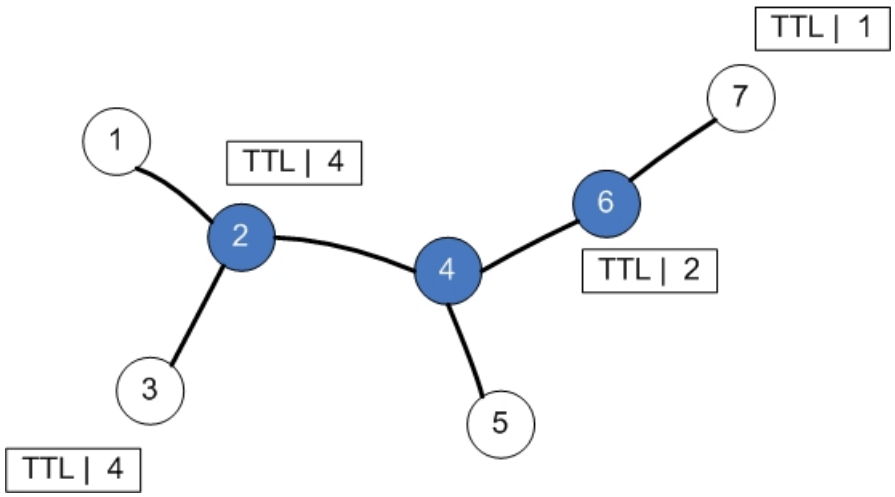

Figure 4.43. MPR view of the Information Dissemination scheme over OLSR.

Additionally, MPR nodes are responsible for setting the TTL field for the registered users in their MPR-selector. The *TTL* value is determined and set during the registration process with the Topology Server process. Upon receiving a new OLSR TC Message, MPR nodes will decrement the TTL field of the encapsulated Bucket Data Units and forward the TC message through the network. This effectively controls the number of MPR nodes a new Bucket Data Unit is going to traverse, hence defining its dissemination radius.

4.11 MPR Nodes and Bucket Aggregation

Under the new information dissemination scheme over OLSR, MPR nodes are responsible for processing and forwarding the Bucket Data Units through the network. The MPR nodes receive Bucket Data Units from the nodes in their MPR-selector set with different frequencies and data sizes. In this section the role of MPR nodes is discussed in more detail.

4.11.1 Bucket Data Unit Aggregation and Forwarding

MPR nodes are responsible for forwarding topology and information updates through the network. Information is received in form of HELLO Bucket Data Units from nodes in the MPR selector set. When a new HELLO Bucket Data Unit is received at the MPR node, the user is registered with Topology Server. In Figure 4.44, node 2 is sending a Bucket to its MPR node that contains a data update. In this case, node 3 registers node 2 as a user of the Topology Server by adding a new Data Record Card in the RegisteredUsersDB repository. The new data, which is encapsulated in the HELLO Bucket Data Unit received from node 2, is stored for the new registered user in the BIGDATABUFFER repository. Service ID and the IP address for node 2's CA process is stored in its Data Record Card. The Data Direction is set to "IN" to flag a network dissemination.

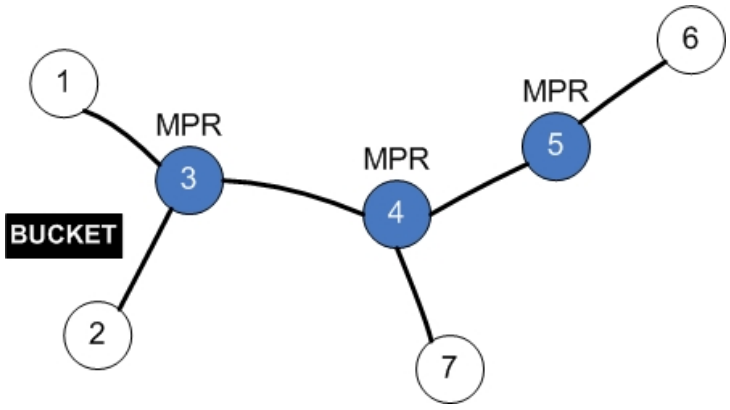

Figure 4.44. MPR node role in dissemination bucket through the network.

MPR Node 3 generates a TC message and encapsulates the data update received from node 2 in the TC Bucket Data Unit for network dissemination. Other MPR nodes in the topology forward the TC message through the network. The Service ID for the certificate authority and the IP address of Node 2 is added to the Bucket to allow other nodes in the network to locate and subscribe to this service. Figure 4.45 shows the detailed processing of this update.

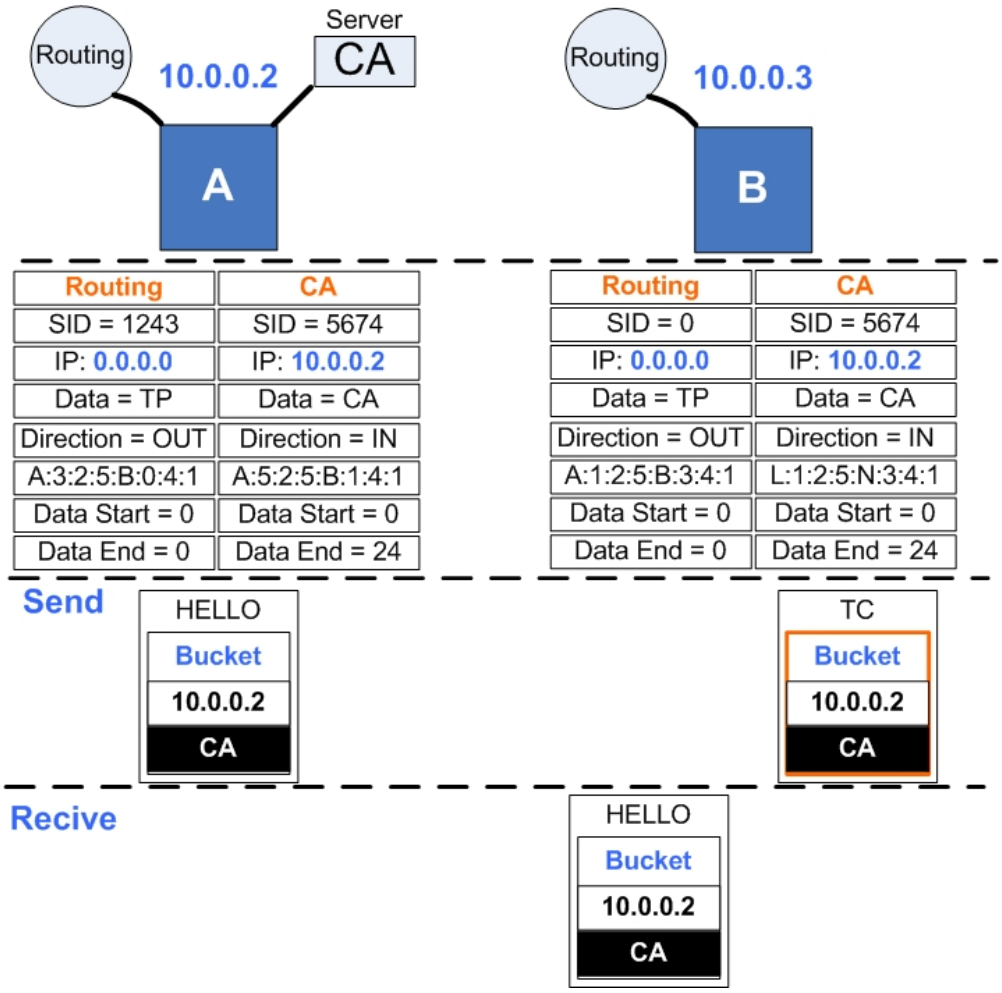

Figure 4.45. Detailed overview of information update using MPR nodes.

MPR nodes treat the received Bucket Data Units from the MPR-selector set as local processes using the topology server. Figure 4.45 shows the MPR nodes RegisteredUsersDB repository. A new Data Record Card is created for the processes in the MPR-selector set using the Topology Server for information dissemination.

4.11.2 Multiple Network Service Providers

Multiple Bucket Data Units can be encapsulated into one TC message at the MPR node. This accommodates disseminating updates for multiple nodes in the MPR selector set or multiple service provider processes using an MPR node. Figure 4.46 shows two nodes using their MPR for providing network updates.

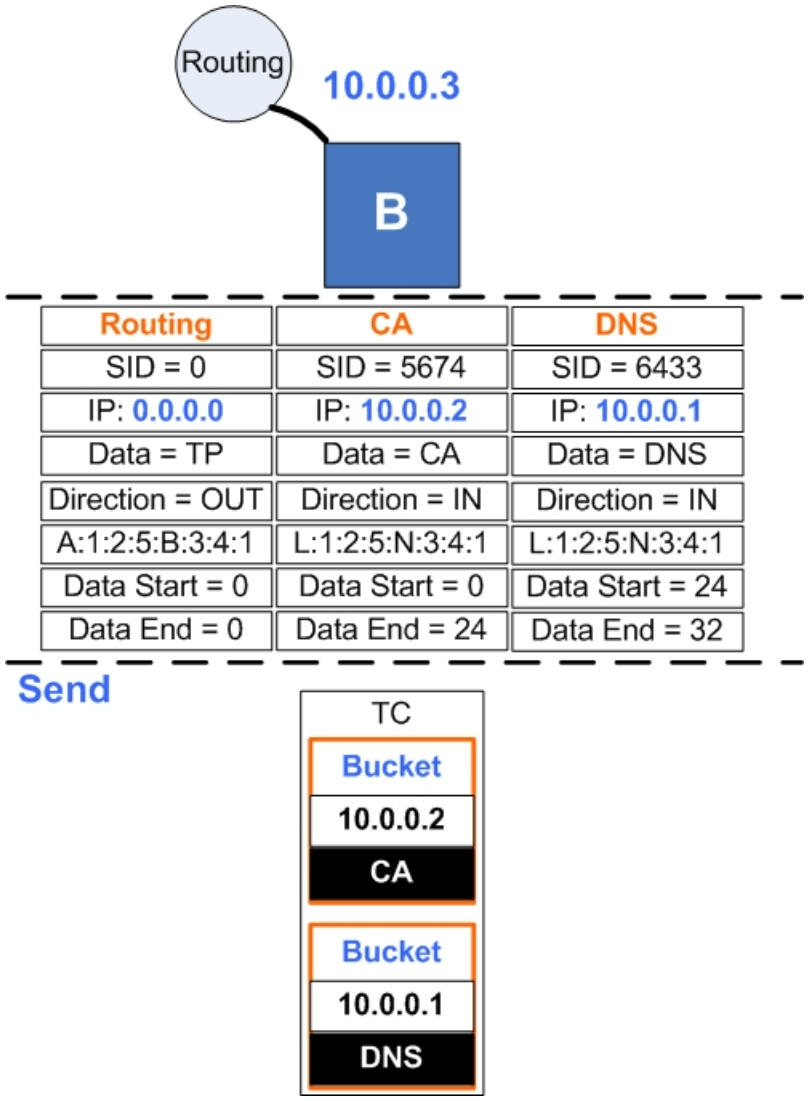

Figure 4.46. Two nodes using an MPR node when providing network updates.

4.12 Summary

This chapter outlined the high-level design and architecture of the proposed extensible event-driven information dissemination scheme for mobile ad hoc networks. Section 4.1 presented an introduction of system components and information repositories. The section also introduced an example service discovery and binding. Then, in the subsequent sections, more detailed descriptions of each component and methods used for interprocess communication and information encapsulation were described.

The next chapter presents the results of performance evaluation experiments and overall performance analysis of the information dissemination scheme developed in this research.

Chapter 5

Information Dissemination: Experimentation, Testing, and Results

5.1 Overview

In Chapter 4, I presented the detailed design and implementation of an extensible information dissemination solution and the integration strategies with NRL OLSR routing protocol. In this chapter, the results of testing and evaluation of the proposed scheme are presented. I examine the average information dissemination delay and added messaging overhead and observe the overall performance of the proposed scheme under different circumstances. This experimental approach uses more realistic scenarios compared to evaluations performed using simulation.

The rest of this chapter is structured as follows. In Section 5.2, the test configuration of our experimental evaluation is presented. In Section 5.3, I describe the system performance metrics that was considered during the experiments. The test platform and example clients implemented for the experimentations are outlined in Sections 5.4 and 5.5. Then, in Section 5.6, I introduce the system variables used when analyzing the round-trip delay and message overhead. Section 5.8 contains the results of several experiments, together with overall analysis and observations. In Section 5.9, a summary of the evaluation experiments and observations are summarized.

5.2 Test Configuration

Two test configurations were used: (1) a dense scenario in six nodes; and (2) a real office environment scenario in presence of other IEEE 802.11 infrastructure-based access points. The first scenario, the “small apartment scenario,” was a dense scenario with six nodes in close proximity of each other, as shown in Figure 5.1. In this configuration, nodes were placed in a small apartment covering an approximate area of 850 square feet in the following four locations.

- Master Bedroom
- Bathroom
- Living room
- Hallway

This setup further allowed for testing service discovery delay and overhead in a small mobile ad hoc network with a low level of radio interference caused by other IEEE 802.11 access points.

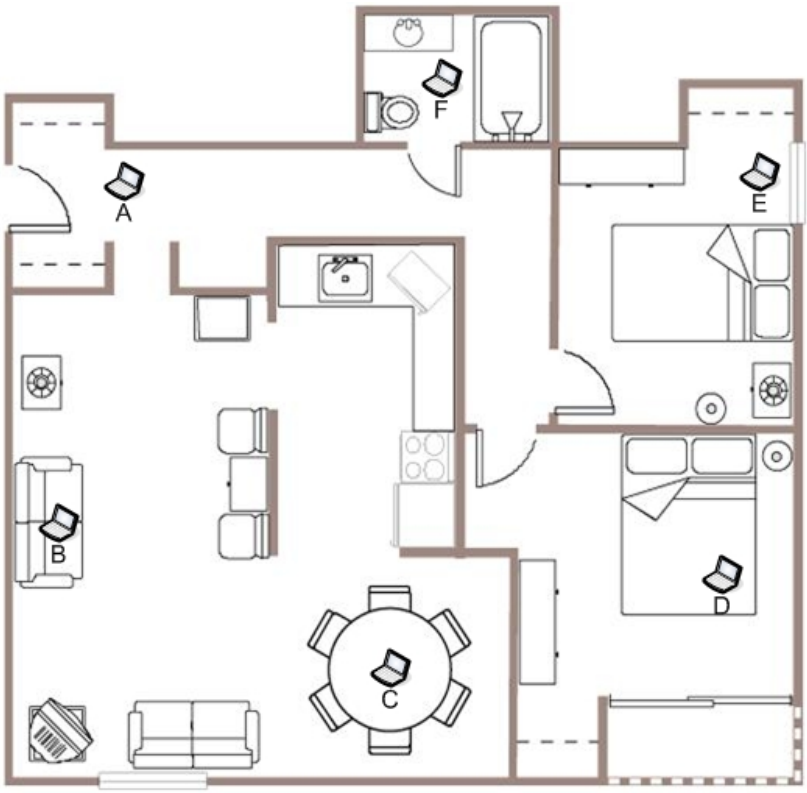

Figure 5.1. Small apartment configuration

All the the MANET nodes were running the Linux Fedora Core 5 operating system [29]. Using the IPTABLES [34] firewall available in the operating system, multi-hop network topologies were created to allow for different number of MPR nodes between source and destination.

The second test scenario was designed to measure the performance of the proposed scheme in a real office environment, covering a 20,000 square foot office space area with radio interference from other IEEE 802.11 infrastructure-based access points. Figure 5.2 shows the office test setup.

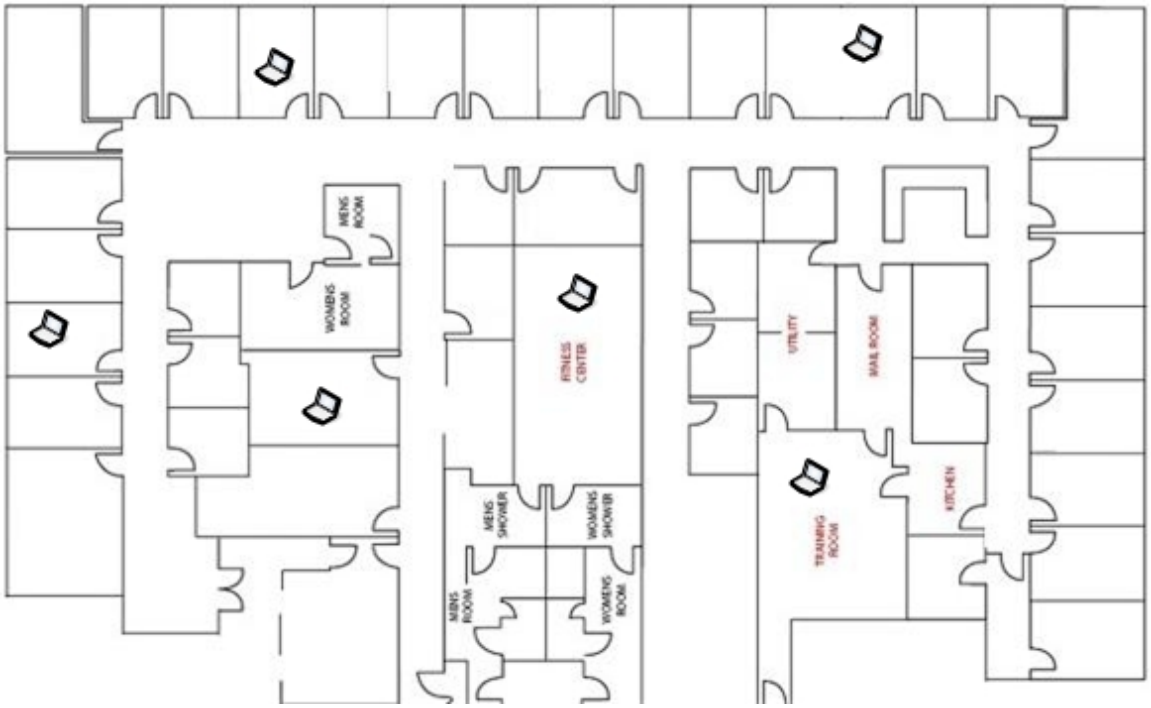

Figure 5.2 Office space test configuration with wireless radio interference from other existing IEEE 802.11 infrastructure based access points.

5.3 Performance Metrics

The performance of the new proposed service discovery scheme over the OLSR routing protocol was evaluated based on additional network control message overhead and information dissemination round-trip delay. Using the NRL Protolib programming toolkit, service provider and user clients were designed and implemented to facilitate testing these two attributes.

5.4 Test Platform

The wireless nodes were Dell Latitude C640 notebook computers with 520 MB of RAM and Intel ® Pentium 4 CPUs with 1.8-GHz processors. Xircom® IEEE 802.11b wireless cards were used to provide the wireless ad hoc network connectivity between the participating nodes. The Fedora Core 5 distribution of the Linux operating system was installed on all of the notebook computers. I used the NRL implementation of the OLSR routing protocol for routing and for implementing the information dissemination scheme. The NRL Protolib programming toolkit was then used for implementing various clients and service providers in the network.

5.5 Test Clients

The service provider process uses a range of different Service Identifiers to register to the Topology Server process internally. As described in Section 4.9, these Service Identifiers provide different quality of service levels using the TC and HELLO message immediate delivery thresholds. Figure 5.4 illustrates a high-level design of the test clients.

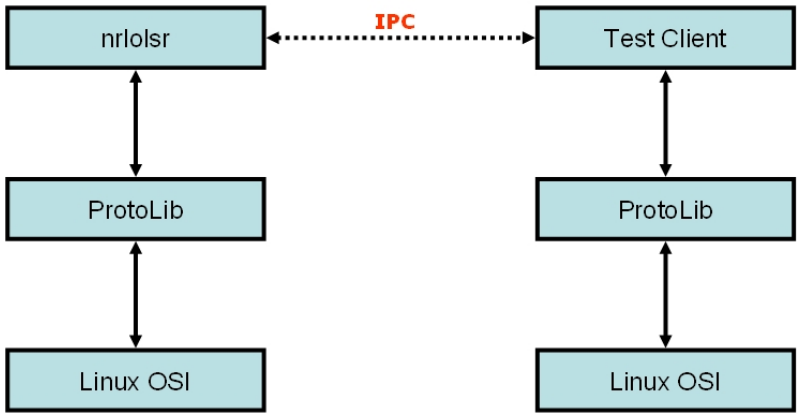

Figure 5.4. High-level view of the test client implementation.

Figure 5.5 illustrates a network topology with six nodes running the information dissemination scheme. There are three MPR nodes responsible for forwarding the information Bucket Data Units (BDU) to their MPR selector sets. In this example, node 2 is transmitting a bucket using a HELLO message and MPR node 3 then encapsulates the

received bucket into a TC message and disseminates the information through the network.

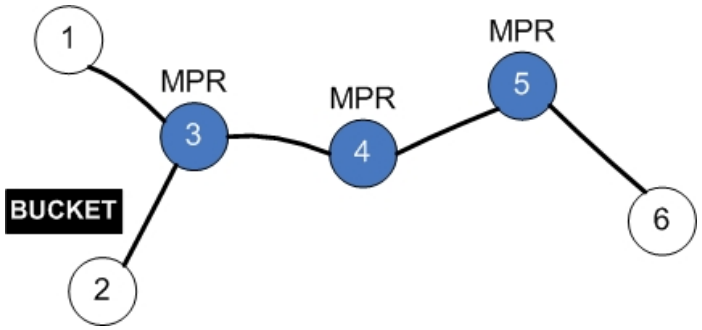

Figure 5.5. Network topology with seven nodes including three MPR nodes.

To measure the round-trip delay of an information update, a service provider process transmits an arbitrary number to the local Topology Server process at the source node for network dissemination. The TP process at the source node then encapsulates the locally received update in the Bucket Data Units in an outgoing TC or HELLO message. On receiving the packet at the destination, the Topology Server extracts the data. The client receives the information update at the destination node and increments the number to indicate successful receiving and processing of the information update. The time elapsed between generation of an arbitrary number at the source node and receiving the acknowledgement from the client process is measured and recorded as the round-trip delay. Figure 5.6 shows the interaction of an example service provider and a client.

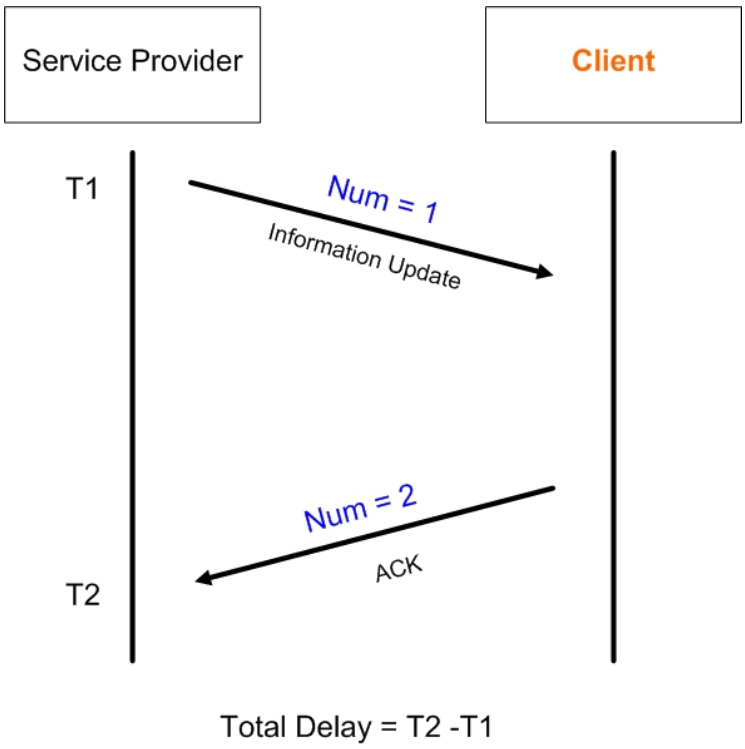

Figure 5.6. Round-trip delay for test client.

The information dissemination round-trip delay includes the new service registration delay, processing delay at each node, transmission delay of the wireless links between nodes, and forwarding delay caused by forwarding the TC messages that contain the information Bucket Data Units. The message overhead is calculated by measuring the number of Bucket Data Units encapsulated in the outgoing OLSR routing protocol TC and HELLO control messages.

Figure 5.7 illustrates a sample output of the client process used for testing the system round-trip delay. The round-trip delay is calculated by measuring the elapsed time between transmitting an update and receiving an acknowledgement from a registered client. The round-trip delay values vary based on wireless link conditions and topology changes in the network.

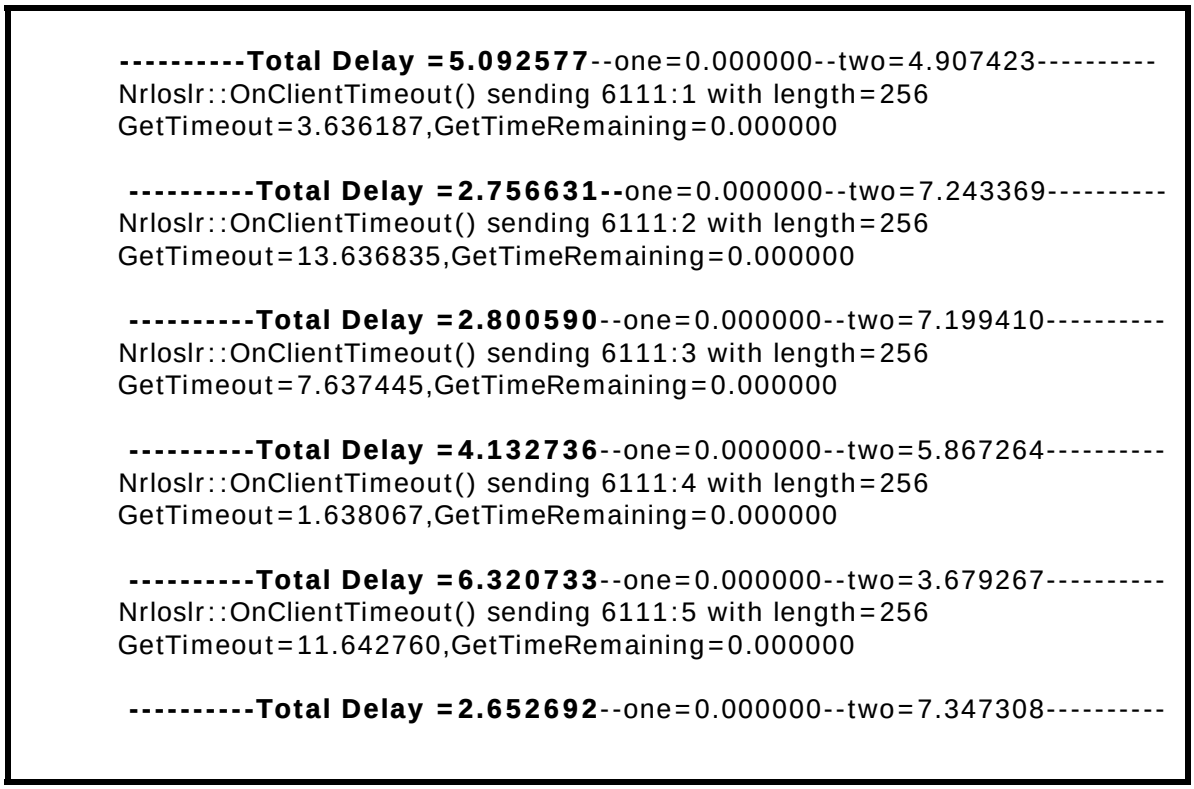

Figure 5.7. Sample output of the client used for testing information dissemination round-trip delay.

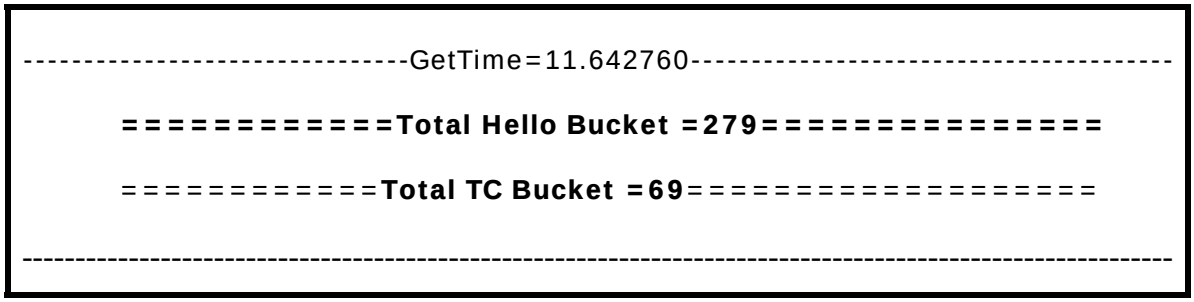

Figure 5.8. Sample output of the client used for testing messaging overhead.

The message overhead due to Bucket Data Unit control fields is calculated by counting the number of HELLO and TC Buckets processed at each participating node.

5.6 Test Variables

The following parameters of the information dissemination scheme over OLSR have been modified to further study the overall performance of the system under different environments with various amount of network traffic.

TC Interval

Various OLSR TC intervals are chosen to analyze the effect of topology updates on the information dissemination round-trip delay. Table 5.1 shows the TC intervals used for testing.

Table 5.1. OLSR TC Intervals

OLSR TC Intervals				
1 (sec)	2 (sec)	3 (sec)	4 (sec)	5 (sec)

Immediate TC and Hello Threshold

As outlined in Section 4.9, different priority factors can be used for providing various levels of quality of service. These variables control the amount of wait time between receiving an update from a local service provider and disseminating it through the network using TC and HELLO messages. The proposed immediate delivery factors are based on the Service Identifiers used during registering process. Table 5.2 lists all of the immediate delivery thresholds set based on the different service IDs.

Table 5.2. Immediate HELLO and TC Delivery Factors used for Providing QoS

Priority Factor Variables	Bronze	Silver	Gold	Platinum
Immediate TC Factor	0.5	0.25	0.1	0.05
Immediate HELLO Factor	0.5	0.25	0.1	0.05
Service IDs Range	5555 to 6666	6666 to 7777	7777 to 8888	8888 to 9999

Number of MPR Nodes

The number of MPR nodes between the service provider and the registered client is changed based on Table 5.3. A linear topology is considered to increase the number of hops between source and destination when testing for average round-trip delay.

Table 5.3. Number of MPR Nodes between Source and Destination

Number of MPR Nodes between Source and Destination				
2	3	4	5	6

Information Update Interval at the Service Provider

Interval of information update also controls the rate of new Bucket Data Unit generation at the service provider process. To test the responsiveness of the proposed scheme, the interval of information updates at the origin is changed based on Table 5.4.

Table 5.4. Information Dissemination Update Intervals

Interval of Network Information Update (s)				
1	5	10	20	30

Fragmentation Threshold

The information Bucket Data Unit fragmentation threshold is changed based on Table 5.5. This configuration is used to study the overall effect of fragmentation on control messaged overhead in the network.

Table 5.5. Fragmentation Threshold

Fragmentation Threshold (bytes)				
128	256	512	1024	2048

5.7 Test Runs

Each data set consisted of ten 1000-seconds tests with a network of six nodes. Clients on the nodes were started by hand, so the source nodes started generating packets before others. During the test setup time, the destination node was listening but not generating any packets. Information updates were generated by the source nodes based on the test variables outlined in Section 5.6. The overall control message overhead and information dissemination round trip delays were then calculated using the test clients with different test parameters. Section 5.8 presents the results and performance analysis of the information dissemination system based on the gathered data.

5.8 Results and Analysis

This Section presents the results of the experiments. First the round-trip information dissemination delay results for both test scenarios are presented followed by the control message overhead data. These results are further analyzed and evaluated in the following sections.

5.8.1 Round-Trip Information Dissemination Delay

To test the effect of the number of MPR nodes on the average round-trip information dissemination delay, nodes are configured with a TC interval of one second and the number of MPR nodes between the service provider and the user is changed. It can be seen from Figure 5.9 that the number of MPR nodes between the source and destination nodes has a direct impact on the round-trip delay of information dissemination in the network for the office space test scenario. However, the average delay is not affected in the small apartment test scenario where nodes are located in close radio range and links are experiencing a low level of wireless interference from other IEEE 802.11-based access points.

Table 5.6. Average Information Dissemination Round-Trip Delay for Two Test Scenarios

Number of MPR Nodes	2	3	4	5	6
Small Apartment Delay (s)	1.56	1.65	1.63	1.57	1.6
Office Space Delay (s)	2.8	3.2	3.2	3.4	3.9

The average information dissemination delay is calculated using the average round-trip delay specified in Equations 5.1.

Average Information Dissemination Delay = 1/2 x (Average Round Trip Delay) (5.1)

Average information dissemination delay results are presented in Table 5.7.

Table 5.7. Average Information Dissemination Delay for Two Test Scenarios

Number of MPR Nodes	2	3	4	5	6
Small Apartment Delay(s)	0.78	0.825	0.815	0.785	0.8
Office Space Delay (s)	1.4	1.6	1.6	1.7	1.95

It can be seen that the link interference present in the office space scenario can significantly increase the overall latency for information dissemination in a wireless mobile ad hoc network. Also, in a small network setup where nodes are located in close proximity, increasing the number of MPR nodes does not increase the average delay in the network.

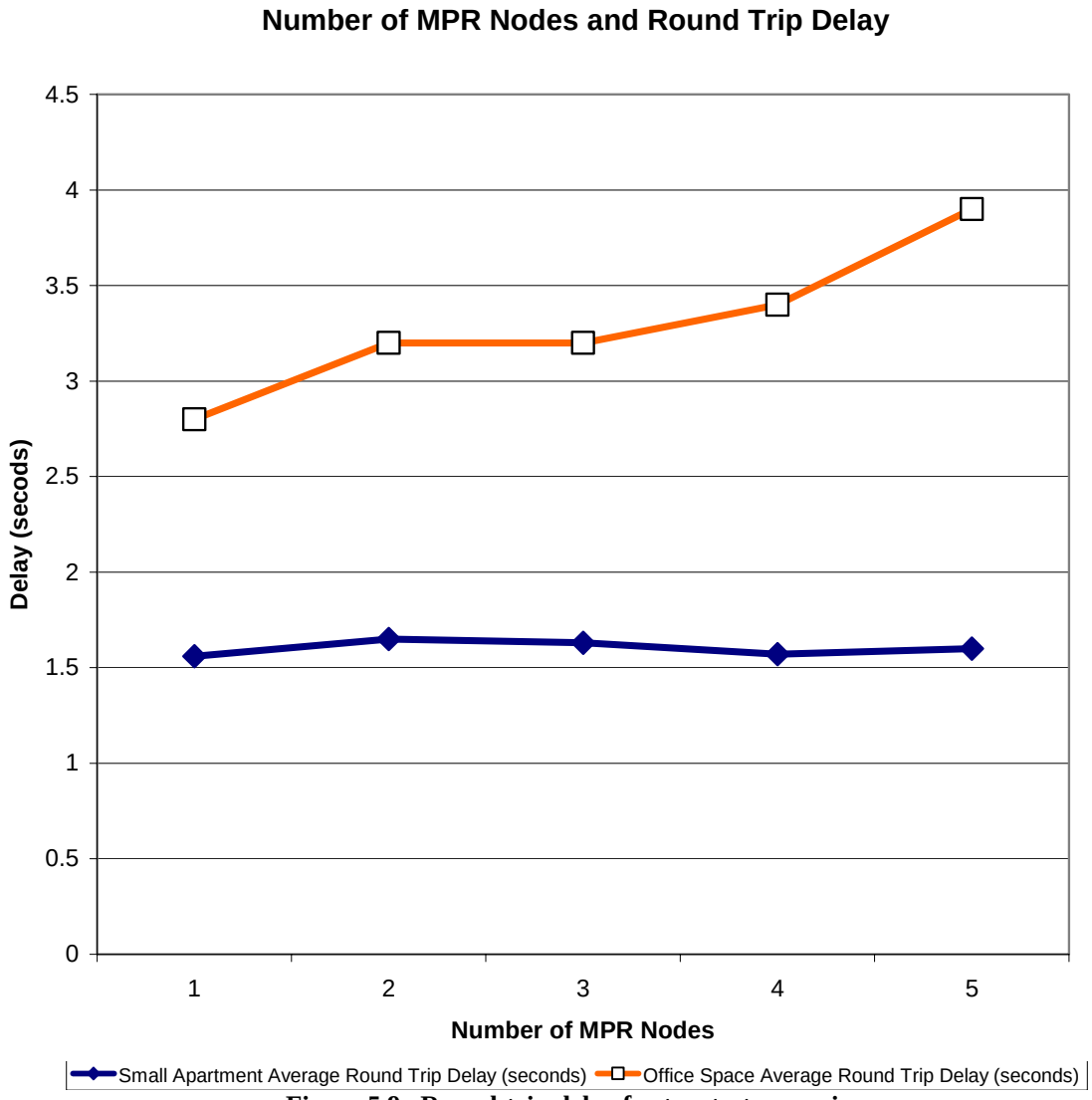

Figure 5.9. Round-trip delay for two test scenarios.

TC Interval and Round Trip Information Dissemination Delay

Table 5.9 summarizes the average information dissemination round-trip delay results measured for the office space test scenario. Figure 5.10 further illustrates the results for a

network with six MPR nodes using various TC intervals and immediate delivery thresholds.

Table 5.8. Office Information Dissemination Delay Test Results (s)

Immediate Threshold	TC Interval	2	3	4	5	6
1	**Normal**	2.13	3.8	3.5	3.6	3.8
0.5	**Bronze**	1.75	2.5	3	2.4	2.6
0.25	**Silver**	1.54	2.22	2.3	2.2	2.4
0.1	**Gold**	1.28	2.15	2	1.5	1.7
0.05	**Platinum**	1.16	1.19	1.2	1.3	1.2

The data further show that by decreasing the immediate delivery threshold shorter round-trip delay values can be achieved. It is interesting to note that the 0.05 immediate delivery factor can further reduce the average round-trip delay of the system to 1.3 seconds for all test scenarios.

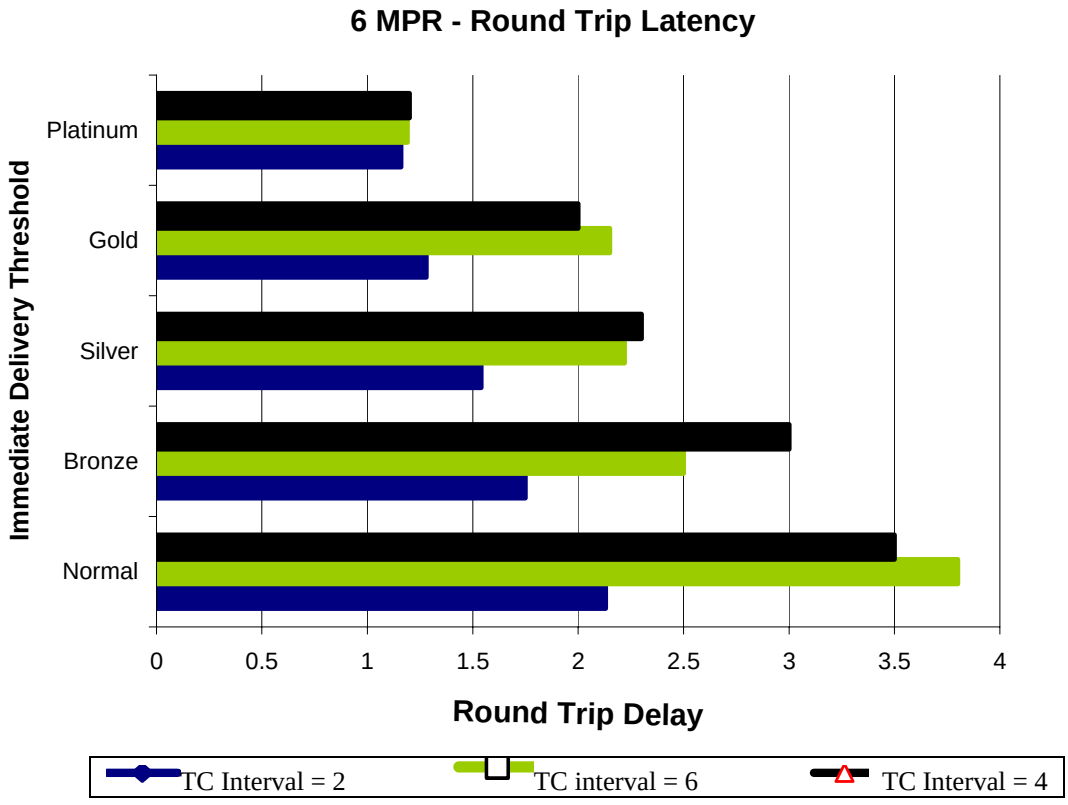

Figure 5.10. Office space information dissemination round-trip delay using six MPR nodes.

Time sensitive applications, such as multimedia traffic, can use higher priority service levels to achieve faster discovery and better overall performance. When the number of MPR nodes between the origin and the destination is increased, the immediate delivery factors are more effective in reducing the average round-trip delays. For instance, when there are only two MPR nodes between the service provider and the user, the average round-trip delay is not significantly reduced by the immediate delivery available in the proposed scheme. However, with six MPR nodes between the source and the destination, when using an immediate delivery factor of 0.05 the average delay is reduced to 1.2 from 3.8 seconds.

Service Discovery Time after Information Convergence in the Network

The service discovery latency after network convergence is measure by first starting the service provider process at the origin node and then starting the client process at the destination node after waiting for a total information convergence in the network. It can be realized from the results shown in Table 5.9 that the service discovery time after a full network convergence is independent of the number of MPR nodes.

Table 5.9. Service Discovery Time and Number of MPR Nodes

Number of MPRs	2	3	4	5	6
Service Discovery Time (s)	0.26	0.227	0.252	0.26	0.246

Additionally, adding more service providers to the network does not have a direct impact on the service discovery time of other available services in the network. Table 5.10 summarizes the service discovery time measurements for different of numbers of service providers.

Table 5.10. Service Discovery Time and Number of Service Providers

Number of Service Providers	1	2	5	10	20
Service Discovery Time (s)	0.235	0.265	0.257	0.26	0.246

Figures 5.11 and 5.12 further illustrate the service discovery time for various number of MPR nodes and service providers. Test results further show that the service discovery latency after network convergence for both scenarios is independent of the following factors.

- Number of MPR nodes
- Number of service providers
- Wireless link conditions in the network

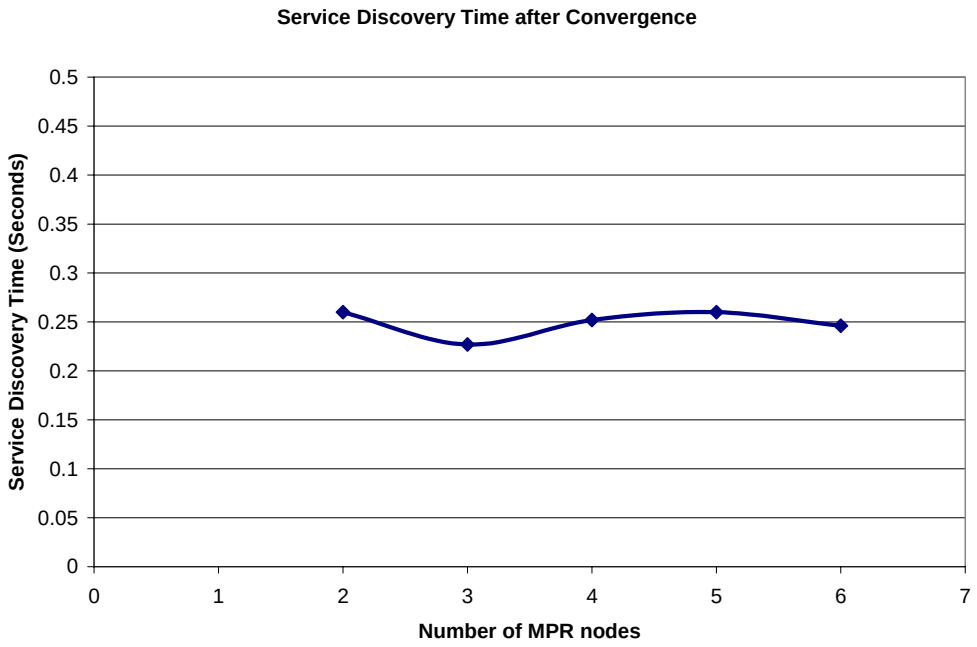

Figure 5.11 Service discovery time after information dissemination convergence for the office test scenario.

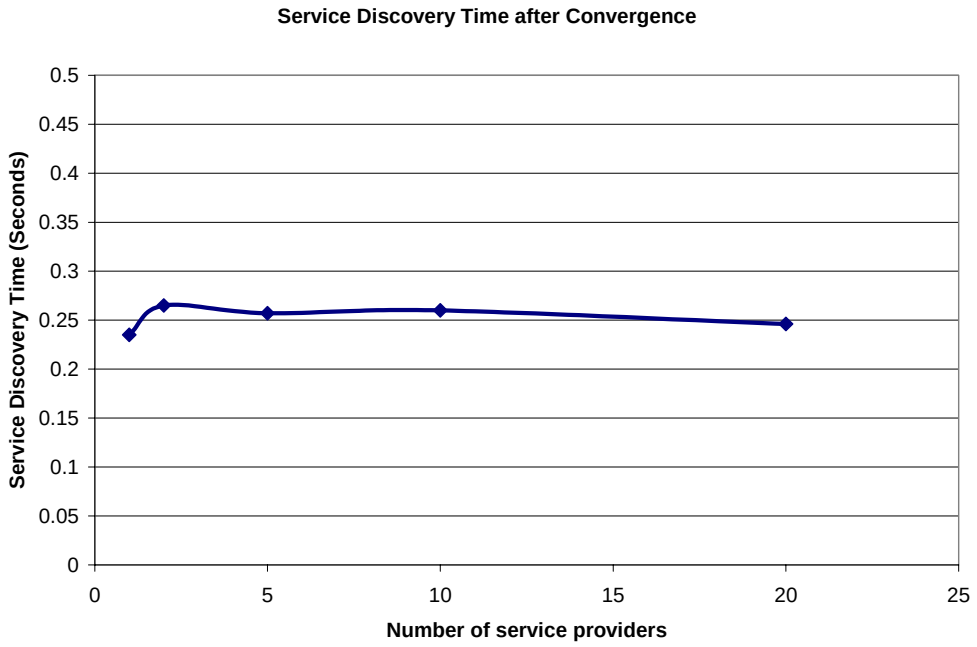

Figure 5.12 Service discovery time after information dissemination convergence for the small apartment scenario.

5.8.2 Control Overhead and Update Frequency

The amount of control overhead introduced by the proposed information dissemination scheme is proportional to the information update intervals at the service providers. Table 5.11 shows the results from testing in both the small apartment and office space scenarios. It can be seen that by increasing the update intervals at the service providers, the average control message overhead is significantly reduced in the network. This overhead is introduced by the control fields in the Bucket Data Units added to OLSR's outgoing HELLO and TC messages.

Table 5.11 Control Message Overhead Caused by Encapsulating Bucket Data Units in HELLO and TC Messages

Update Interval (s)	1	5	10	20	30
Overhead (bytes)	23664	4530	2468	1138	762

Figure 5.13 shows the same results for update intervals of 1 second to 30 seconds. This test scenario is conducted for one client over 1000 seconds.

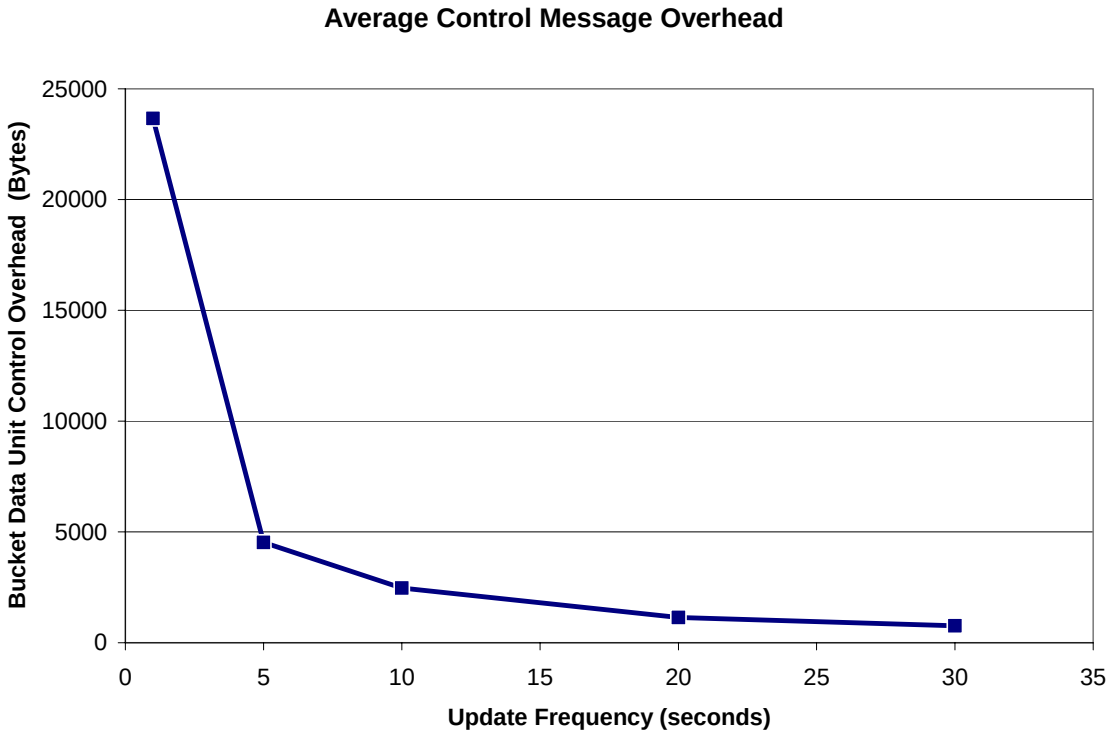

Figure 5.13. Average overhead caused by information dissemination and update intervals.

When more service providers are added in the network the amount of message overhead is increased as expected. Table 5.12 shows the data for 1 to 10 service providers in the network running for 1000 seconds and providing updates every 10 seconds.

Table 5.12. Measured Overhead for Different Numbers of Service Providers

Number of Service Providers	1	2	5	10
Overhead (bytes)	4527	9028	22579	44782

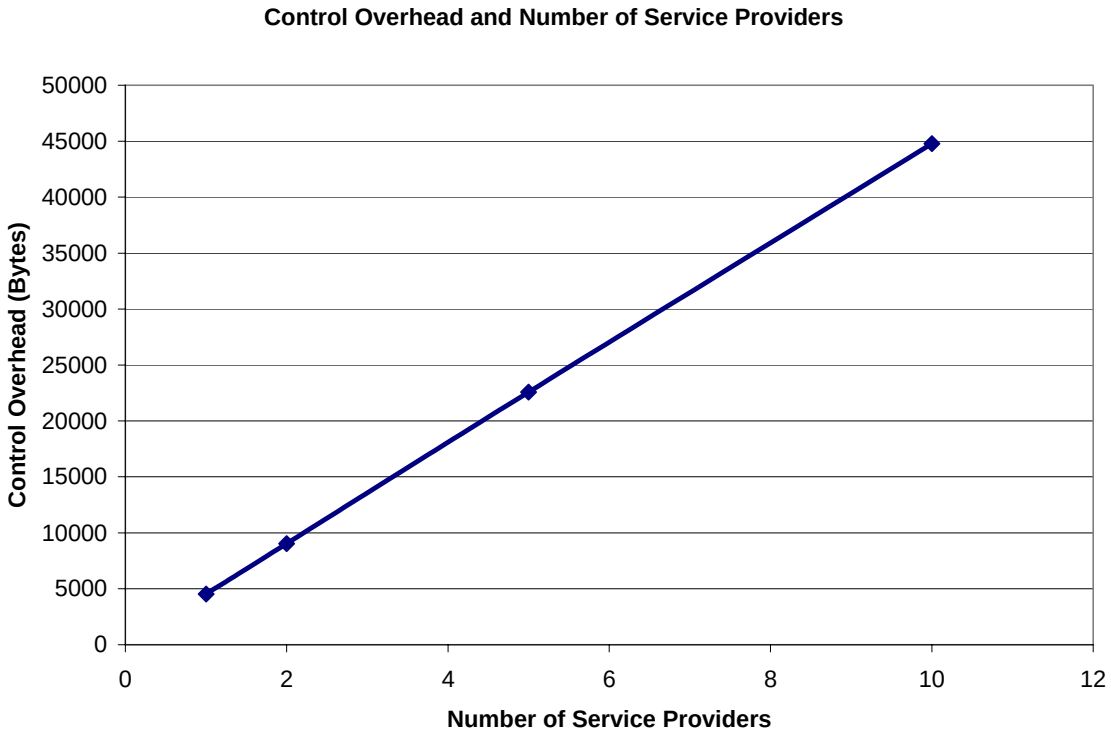

Figure 5.14. Control overhead caused by increasing the number of service providers in the network.

Bucket Data Unit fragmentation threshold can also have a direct impact on the control messaging overhead in the network. As can be seen from Table 5.14, the amount of control overhead is significantly increased when the fragmentation threshold is decreased for the outgoing Bucket Data Units in the HELLO and TC messages.

Table 5.13. Overhead Caused by Fragmentation

Fragmentation Threshold (bytes)	128	256	512	1024	2048
Control Overhead (bytes)	39530	19765	9876	4950	2468

For this test case, a service provider is transmitting data with the original size of 2 KB. The information is updated every 10 seconds. The test runtime is 1000 seconds and the fragmentation thresholds are changed based on Table 5.13. It can be seen from Table 5.14 and Figure 5.15 that when no fragmentation is used for dissemination, the added messaging overhead is 2468 bytes. As the fragmentation threshold is reduced to 128 seconds, the control message overhead is also increasing significantly to 39530 bytes.

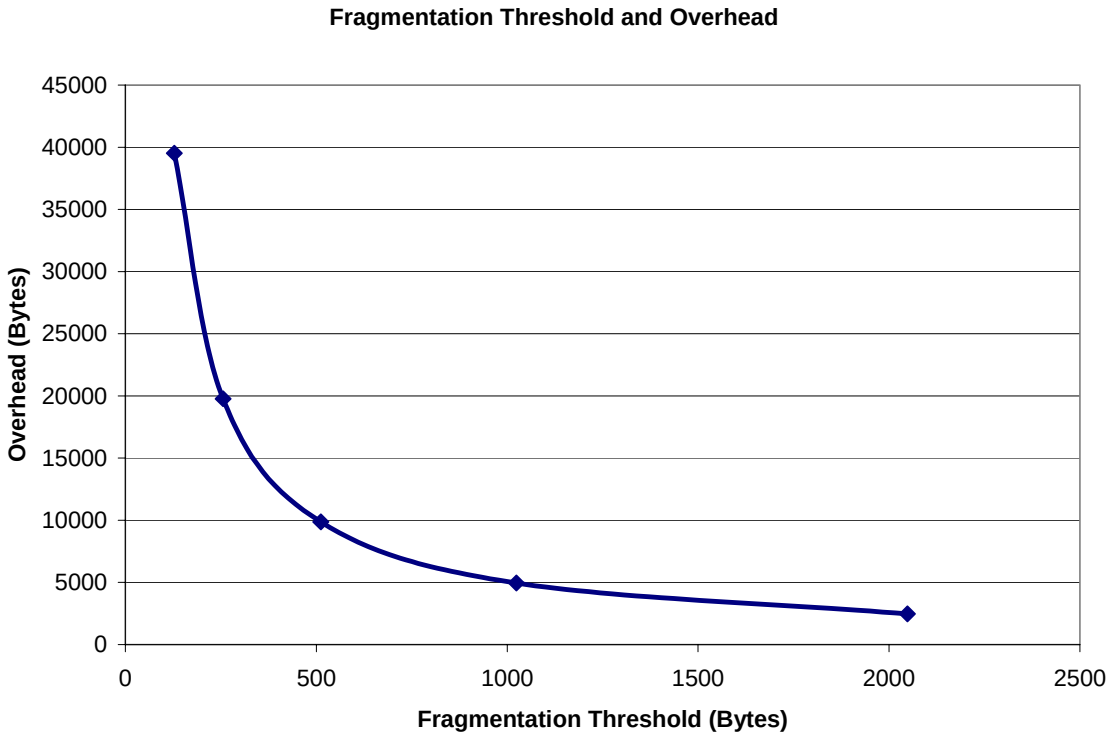

Figure 5.15 Control message overhead is increased when the fragmentation threshold is decreased for the Bucket Data Units

From Figure 5.15 it can be realized that by increasing the fragmentation threshold and avoiding Bucket Data Unit fragmentation, the overall messaging overhead can be reduced. Fragmentation is beneficial only when multiple service providers are using the Topology Server for information dissemination and some have large Bucket Data Units requirements. The fragmentation scheme can help to ensure the same level of access to all users regardless of the size of information updates. Fragmentation can also be used to improve the overall performance of the protocol when the MAC layer is enforcing a fragmentation size smaller than the data encapsulated in the Bucket Data Units.

5.9 Summary

In this chapter the overall experimentation approach for testing average information dissemination round trip delay and control message overhead metrics was presented. An example test client for testing these metrics was introduced. Multiple scenarios of priority delivery were tested and data was gathered on round-trip delivery delay and messaging

overhead. The next chapter concludes this work with a summary, a list of contributions, and closing remarks.

Chapter 6

Conclusion

This chapter summarizes the research, outlines contributions of this work, and presents directions for future research.

6.1 Summary

This research focused on the information dissemination component of service discovery, specifically to mobile ad hoc network environments based on integration of service discovery with the Optimized Link State Routing (OLSR) protocol.

In this thesis, we discussed the need for an extensible information dissemination solution for mobile ad hoc network environments. The inability of current service discovery schemes to effectively deal with unstable topology conditions due to rapid changes in links and node mobility in MANETs was discussed next.

An event-driven programming interface to the NRL implementation of the OLSR routing protocol was formulated. This programming interface was used to separate the routing process from NRL OLSR. The independent routing process was then tested and proved to be capable of receiving topology updates from the OLSR topology server process based on link and neighbor changes in the network.

A priority-based message delivery framework was then integrated with NRL OLSR to allow different levels of quality of service for information dissemination in the network for delay sensitive applications. To enhance the performance of the information dissemination scheme when operating in wireless networks with various medium access control (MAC) layer fragmentation threshold settings, a bucket data unit fragmentation mechanism was implemented.

Our proposed information dissemination scheme was then deployed in a real wireless ad hoc environment for further validation and testing. We presented and discussed the results of our experimental evaluation of the proposed information dissemination scheme. The metrics of interest were messaging overhead and information propagation round-trip delay. We deployed six nodes running the information dissemination scheme integrated with NRL OLSR in two test MANETs. One MANET was across an 850-square foot. residential apartment and the other MANET was spread across a typical office building with medium to high levels of link radio interference caused by other IEEE 802.11 infrastructure-based access points.

6.2 Observations

We observed that the link-state nature of our service discovery solution has helped it to be more adaptive to topology changes. The overall service discovery and selection was improved in the event of network partitioning and wireless link changes. It was also seen that service discovery times after information convergence in the network were independent of the number of MPR nodes between the service providers and clients. During the experimental evaluation of the proposed scheme it was observed that the service discovery latency after information convergence is independent of wireless radio interference and link conditions. We confirmed that our immediate message delivery scheme has provided superior quality of service to registered users in presence of radio interference and other delays caused by MPR nodes message forwarding.

6.3 Contributions

In this work, I introduced a new extensible information dissemination scheme that is integrated with the Optimized Link State Routing protocol to address the challenges of service discovery in mobile ad hoc networks. I presented a detailed design of the information dissemination scheme based on the Naval Research Laboratory's ProtoLib network protocol programming framework. The proposed scheme supported various network services with different message sizes and service discovery requirements. A reusable event-driven programming interface to the NRL OLSR routing protocol was designed and implemented. This programming interface can be used by other management and monitoring clients on the network for receiving real-time updates about link and topology changes as seen by OLSR. I developed a solution that separates the routing process from the NRL OLSR routing protocol, thus making OLSR a topology discovery protocol. This can further facilitate the implementation of various routing algorithms based on other metrics such as the signal-to-noise ratio (SNR) of wireless links or the nodes' level of cooperation when forwarding messages in the network. A priority message delivery scheme that provides different levels of quality of service for information dissemination in mobile ad hoc networks was designed and implemented. To ensure fair use of the transport media for users with various message size, a fragmentation solution was considered. Finally, the system performance metrics were further analyzed and evaluated using real-life scenarios.

6.4 Potential Future Work

The next step for this work is to make the source code for the information dissemination scheme based on NRL OLSR available. To accomplish, the source code needs to be made more robust and the proper accompanying documentation created. Using the ProtoLib programming toolkit, various other test clients can be implemented for discovering multimedia services such as Session Initiation Protocol (SIP) servers and video streaming

servers in mobile ad hoc networks. This can further prove the application of our proposed information dissemination scheme in more real-life scenarios.

A large scale deployment of the proposed information dissemination scheme can be modeled with network simulation tools. This would allow for further validating the scalability of the system using mobility models and scalability techniques.

Other MANET monitoring clients can be implemented using the event-driven programming interface to NRL OLSR that we have formulated as part of this research. These monitoring clients can be further used for receiving real-time updates of OLSR topology and link changes.

Based on the independent routing process and the event driven programming interface that was designed, other routing algorithms can be easily implemented. These new routing algorithms can easily use our information dissemination scheme together with a list of available resources in the network. Routing decisions can be based on other metrics such as the node's level of cooperation in the topology, the energy and computing resources available at participating nodes, and the signal-to-noise ratio (SNR) values of the wireless links.

This approach further allows the MANET to operate as a cohesive system when providing connectivity to mission critical applications, such as a communication network used in a disaster response situation.

References

[1] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On Demand Distance Vector (AODV) Routing," IETF RFC 3561, July 2003. Available at http://tools.ietf.org/html/rfc3561

[2] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*, T. Imielinski and H. Korth, Eds. Kluwer Academic Publishers, 1996, pp. 153–181.

[3] V. D. Park and S. Corson, "Temporally-ordered Routing Algorithm (TORA) Version 1 Functional Specification," draft-ietf-manet-tora-spec-04.txt IETF Internet Draft July 2001. Available at http://tools.ietf.org/html/draft-ietf-manet-tora-spec-04

[4] T. Clausen and P. Jacquet, Eds., "Optimized Link State Routing Protocol (OLSR)," IETF RFC 3626, Oct. 2003. Available at http://www.ietf.org/rfc/rfc3626.txt

[5] Z. J. Haas, M. R. Pearlman, and P. Samar. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. draft-ietf-manetzone-zrp-04.txt, IETF Internet Draft, Aug. 2002. Available at http://tools.ietf.org/id/draft-ietf-manet-zone-zrp-04.txt

[6] Sun Microsystems, "Jini(tm) Connection Technology." Available at http://www.sun.com/jini/

[7] Salutation Consortium, "Salutation Architecture Specification V2.0C, Parts 1-3," June 1999 Available at http://web.archive.org/web/20030622085538/www.salutation.org/about.htm

[8] UPnP Forum, "UPnP Device Architecture 1.0, 2003. Available At http://www.upnp.org/resources/documents.asp

[9] T. Bellwood, L. Clement, and C. von Riegen, Eds., "UDDI Version 3.0.1, OASIS Std.," Oct. 2003. Available at http://uddi.org/pubs/uddi-v3.0.1-20031014.htm.

[10] Bluetooth Consortium, "Specification of the Bluetooth System Core Version 1.0 b: Part e, Service Discovery Protocol (SDP)," November 1999

[11] E. Guttman, C. Perkins, and J. Veizades. "Service Location Protocol, Version 2," IETF RFC 2608, June 1999. Available at http://www.ietf.org/rfc/rfc2608.txt.

[12] O. V. Ratsimor, D. Chakraborty, S. Tolia, D. Khushraj, A. Kunjithapatham, A. Joshi, T. Finin, and Y. Yesha. "Allia: Alliance-based Service Discovery for Ad-hoc Environments." *ACM Mobile Commerce Workshop*, Sept. 2002, pp. 1-9.

[13] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha "GSD: A Novel Group-based Service Discovery Protocol for MANETs," *IEEE Conference on Mobile and Wireless Communications Networks*, Sept. 2002, pp. 140-144.

[14] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, and A. Schade, "DEAPspace: Transient Ad-Hoc Networking of Pervasive Devices," *Workshop on Mobile and Ad-Hoc Networking and Comptuing*, Aug. 2000, pp. 133–134.

[15] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark: A Service Discovery and Delivery Protocol for Ad-Hoc Networks," *IEEE Wireless Communications and Networking Conference*, March 2003, pp. 2107–2113.

[16] G. Schiele, C. Becker, and K. Rothermel, "Energy-Efficient Cluster-based Service Discovery for Ubiquitous Computing," *ACM SIGOPS European Workshop*, Sept. 2004, pp 19-22.

[17] C.-K. Toh, *Ad Hoc Mobile Wireless Networks. Protocols and Systems*, Prentice Hall PTR, Englewood Cliffs, NJ, 2002, pp. 231-242.

[18] A. Varshavsky, B. Reid, and E. de Lara, "The Need for Cross-layer Service Discovery in MANETs," Technical Report CSRG-492, Department of Computer Science, University of Toronto, 2004. Available at http://www.cs.toronto.edu/~walex/**cross-layer**.pdf

[19] J. L Jodra, M. Vara, J. M. Cabero, and J. Bagazgoitia, "Service Discovery Mechanism over OLSR for Mobile Ad-hoc Networks," Advanced Information Networking and Applications, vol. 2, 2006, pp. 534-542.

[20] L. Li and L. Lamont, "Support Real-time Interactive Session Applications over a Tactical Mobile Ad Hoc Network," *IEEE MILCOM*, Oct. 2005, pp 2910- 2916

[21] L. A. DaSilva, S. F. Midkiff, J. S. Park, G. C. Hadjichristofi, N. J. Davis, K. S. Phanse, and T. Lin, "Network Mobility and Protocol Interoperability in Ad Hoc Networks," *IEEE Communications Magazine*, Vol. 42, No. 11, pp. 88-96, Nov. 2004.

[22] U. Lee, S. F. Midkiff, and T. Lin, "OSPF-MCDS-MC: A Routing Protocol for Multi-channel Wireless Ad-hoc Networks," *IEEE Consumer Communications and Networking Conference*, January 2006, pp. 426-430.

[23] C. Ververidis and G. C. Polyzos, "Extended ZRP: Performance Evaluation of a Routing Layer Based Service Discovery Protocol for Mobile Ad Hoc Networks," *ACM International Conference on Mobile and Ubiquitous Systems: Networking and Services*, July 2005, pp. 114-123.

[24] C.-S. Oh, Y.-B. Ko and Y.-S. Roh, “An Integrated Approach for Efficient Routing and Service Discovery in Mobile Ad Hoc Networks,” *IEEE Consumer Communications and Networking Conference*, Jan. 2005, pp. 184- 189..

[25] D. Noh and H. Shin, “An Adaptive Hybrid Strategy of Integrated Service Discovery for Wireless Mobile Ad Hoc Networks,” International Conference on Distributed Computing Systems Workshops, Month 2006, p. 89.

[26] A. Helmy, S. Garg, N. Nahata, and P.Pamu, “CARD: A Contact-based Architecture for Resource Discovery in Wireless Ad Hoc Networks,” *ACM Mobile Networks and Applications*, Vol 10, October 2005, pp.99-113.

[27] E. Borgia, “Experimental Evaluation of Ad Hoc Routing Protocols,” Pervasive Wireless Networking *Workshop in PerCom*, March 2005, pp. 232 - 236.

[28] J. Hsu, S. Bhatia, M. Takai, R. Bagordia, and M. J. Acriche, “Performance of Mobile Ad Hoc Networking Routing Protocols in Realistic Scenarios,” *IEEE MILCOM*, October 2003, pp. 1268 - 1273.

[29] Fedora Core 5, “Fedora Project,” Available at http://fedoraproject.org/.

[30] Naval Research Laboratory, “The NRL OLSR Routing Protocol Implementation.” Available at http://cs.itd.nrl.navy.mil/work/olsr/index.php.

[31] Naval Research Laboratory, “The Protean Protocol Prototyping Library (ProtoLib).” Available at http://cs.itd.nrl.navy.mil/work/protolib/index.php.

[32] E. Nordstrom, P. Gunningberg, and H. Lundgren, “A Testbed and Methodology for Experimental Evaluation of Wireless Mobile Ad Hoc Networks, *TridentCom*, Febuary 2005, pp. 100 - 109

[33] The Linux Documentation Project, “Linux Interprocess Communications,” March 1996. Available at http://tldp.org/LDP/lpg/node7.html.

[34] Linux Guruz, “Linux iptables HOWTO,” Sept. 1999. Available at http://www.linuxguruz.com/iptables/howto/.

Vita

Kaveh Mehrjoo received the Bachelor of Science in Electrical Engineering from the Virginia Polytechnic Institute and State University (Virginia Tech) in December 2005. He worked at GE Fanuc Automation Americas in the Ethernet product acceptance team and Virginia Tech Air Transportation laboratories as a network and systems engineer. Upon his graduation, he joined Sprint Nextel as a telecommunications design engineer while he continued his studies in the Bradley Department of Electrical and Computer Engineering at Virginia Tech‘s Advanced Research Institute. Currently, Kaveh is working as a wireless systems support engineer at AirWave Wireless Inc.