MANET REFERENCE CONFIGURATIONS AND EVALUATION OF SERVICE LOCATION PROTOCOL FOR MANET

by

Mohamed M. Abou El Saoud

A thesis submitted to
The Faculty of Graduate Studies and Research
in partial fulfillment of the requirement for the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering
Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
Canada

April 2005

The undersigned recommend to the Faculty of Graduate Studies and Research acceptance of the thesis

MANET REFERENCE CONFIGURATIONS AND EVALUATION OF SERVICE LOCATION PROTOCOL FOR MANET

Submitted by **Mohamed M. Abou El Saoud**, B.Sc.
in partial fulfillment of the requirements for
the degree of M.A.Sc. in Electrical Engineering

Thesis Co-Supervisor
Professor Samy A. Mahmoud

Thesis Co-Supervisor
Professor Thomas Kunz

CHAIR, Department of Systems and Computer Engineering
Professor Rafik A. Goubran

Carleton University
April 2005

Abstract

Service Discovery Protocols (SDPs) allow users and applications to automatically locate services without prior configuration. Since Mobile Ad Hoc Networks (MANETs) constitute an essential part of the ubiquitous computing scenario, increasing efforts are geared toward providing service discovery in such environments. In this thesis, we implement Service Location Protocol for MANET (SLPManet), based on the widely employed Service Location Protocol (SLP) Version 2. We recognize the need for a legitimate framework for the qualitative and quantitative evaluation of SDPs. We hence develop BENCHManet, a benchmark composed of reference tests, each of which reflects the configuration found in various MANET applications. We examine the performance of SLPManet using BENCHManet, and highlight the protocol's strengths and weaknesses. We propose a caching modification that increases the discovery success by 38% for poorly performing scenarios. Moreover, the aggregate bandwidth consumption and average lookup latency decreased by up to 98%.

Acknowledgments

I would like to express my deep appreciation to my supervisor, Dean Samy Mahmoud, for his guidance and encouragement. His creativeness and inspiration for science would always serve as an enlivening and influential example for me.

I would like to express my gratitude to my supervisor, Professor Thomas Kunz, whose knowledge and numerous suggestions were extremely helpful and valuable. His promptness and continual advices were highly appreciated.

I am indebted to Noha, my fiancé, for her extreme patience and understanding at stressful times. Completion of this work would have been difficult without her sincere love and encouragement.

There certainly exists no words that could possibly express the extent of gratitude I owe my family. They have all been an ever lasting source of support and encouragement. I am deeply thankful to my father, Moaz, and beloved mother, Mona, for the uncountable sacrifices they made for me throughout the years. My love and respect to them are endless and immense.

Table of Contents

List of Tables

List of Figures

List of Acronyms

3G	3rd Generation
API	Application Program Interface
BCAST	efficient scalable BroadCAST routing protocol for MANET
BENCHManet	Benchmark for MANET
combat	Military Combat scenario
conf	Collaborative Conference scenario
DA	Directory Agent
DCF	Distributed Coordination Function
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSSS	Direct-Sequence Spread Spectrum
event	Event Coverage scenario
GENA	General Event Notification Architecture
HTTP	HyperText Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol
IP	Internet Protocol
LAN	Local Area Network
LDAPv3	Lightweight Directory Access Protocol Version 3
MAC	Media Access Control
MANET	Mobile Ad Hoc Network
march	Military March scenario
mesh	Residential Mesh Networks scenario

MN	Mobile Node
MTU	Maximum Transmission Unit
NS	Network Size
NS-2	Network Simulator Version 2
OLSR	Optimized Link State Routing
OSI	Open System Interconnection
OTcl	Object Tool Command Language
pan	Personal Area Networks scenario
PDA	Personal Digital Assistant
pursuit	Police Pursuit scenario
QoS	Quality of Service
rescue	Rescue Operation scenario
RFC	Request For Comments
RPGM	Reference Point Group Mobility
SA	Service Agent
SAAdvert	Service Advertisement
SD	Service Duplication ratio
SDP	Service Discovery Protocol
SIG	Special Interest Group
SL	Service Advertisement Lifetime
SLP	Service Location Protocol
SLPManet	Service Location Protocol for MANET
SM	Salutation Manager
SMB	Server Message Block
SOAP	Simple Object Access Protocol
SP	Services per Server Node

SQ	Simultaneous Requests
SR	Service Agent ratio
SrvRply	Service Reply
SrvRqst	Service Request
SSDP	Simple Service Discovery Protocol
TCP	Transmission Control Protocol
TM	Transport Manager
TTL	Time To Live
UA	User Agent
UDP	User Datagram Protocol
UDPSlp	SLPManet-compatible UDP agents
UPnP	Universal Plug and Play
UR	User Agent ratio
URL	Universal Resource Locator
US	Unique Services
UTF	Universal Transformation Format
VINT	Virtual InterNetwork Testbed
vp	Vehicle-Passenger scenario
vr	Vehicle-Roadside scenario
X.500	An ISO and ITU standard that defines how global directories should be structured
XID	SLP message unique Identification
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Motivation

Due to the fundamental role that Mobile Ad Hoc Networks (MANETs) play in future pervasive computing environments, providing service discovery in such networks is essential. Compared to wired infrastructure-based networks, wireless ad hoc networks levy a number of unique challenges to the problem of service discovery. In MANETs, heterogeneous nodes are constantly moving, entering and leaving the network, hence continually changing the network's topology. Ad hoc networks must be fully decentralized with no single controlling entities such as directories and lookup tables. Often nodes can not communicate directly with each other, forcing routes between nodes to transverse several hops. Finally, mobile devices have limited battery and processing power, and therefore can not afford to perform complex tasks.

It is fundamental that we not only provide a Service Discovery Protocol (SDP) in MANETs, but also highlight its assets and deficiencies. Essentially, the most accurate way to evaluate a protocol is to test it under environments that are as realistic as possible. Each protocol can ultimately out-perform another in a particular networking scenario. This necessitates the availability of reference configurations for a fair comparison of MANET SDPs. Evaluating SDPs using

this common benchmark facilitates recognizing the strengths and weaknesses of the protocols in the various realistic networking contexts. Such benchmark also serves as a common ground for a fair comparison of SDPs.

1.2 Research Contributions

This research concentrates on the problem of service discovery in MANETs. Service Location Protocol (SLP) Version 2 was the protocol of choice for this work. We first adapted SLP to MANETs. We then implemented the adapted protocol, Service Location Protocol for MANET (SLPManet), and tested it using Network Simulator Version 2 (NS-2). We devised MANET reference configurations in the form of a benchmark, Benchmark for MANET (BENCHManet), consisting of scenarios mimicking each of the different classes of existing MANET applications. We evaluate the performance of SLPManet using the developed benchmark, and hence draw qualitative as well as quantitative conclusions. We finally suggest a simple improvement to SLPManet, and show that the proposed caching modification indeed improves the protocol's performance.

1.3 Organization of Thesis

This thesis begins with a brief overview of this work. Chapter 2 reviews the background associated with Mobile Ad Hoc Networks. The Chapter also introduces the problem of service discovery. Chapter 3 is a review of the state

of the art relevant to SDPs and performance evaluation frameworks. The Chapter also discusses reasons for choosing SLP for service discovery in MANETs. Chapter 4 discusses the adaptations we made to SLP, and gives a detailed specification of the implemented SLPManet protocol. The chapter also presents a proposed extended caching modification to the SLPManet protocol. Chapter 5 presents the evaluation framework based on reference configurations established to realistically test SDPs, thus, demonstrates the evolution of BENCHManet. Chapter 6 discloses an evaluation of SLPManet, and presents the improvement in the protocol's performance with the proposed extended caching modification. Finally, Chapter 7 ends this thesis with conclusions and suggestions for future research.

Chapter 2

Background

2.1 Introduction to Mobile Ad Hoc Networks

The decreasing cost of networking technologies has led to the rapid growth of distributed systems and the deployment of millions of both wired and wireless network enabled devices. Mobile Ad Hoc Networks (MANETs) is currently one of the most eminent areas in mobile computing and networking. MANETs consist of a collection of mobile devices (nodes) that wish to communicate with each other without a fixed infrastructure or a pre-determined organization of available links. MANETs can be built around any wireless technology. The most attractive features of MANETs include ease of deployment, speed of deployment, and independence of infrastructure.

There does not exist a standard classification for MANETs. However, such networks usually fall into one of the following three categories:

1. Isolated ad hoc networks

   Very large ad hoc networks with thousands of nodes have been proven unpractical for communication networks that need to transport a large amount of information. This is due to their poor security, high organization cost and very low traffic performance. On the contrary, small sized ad hoc networks (usually tens or hundreds of nodes) are more ubiquitous

and therefore have a high usage potential (see Section 2.2). Such networks are the ones that our work targets.

2. Integrated ad hoc networks / mesh networks

   Integration of small sized ad hoc networks with Internet Protocol (IP) based broadband access networks and the global Internet can be realized by ad hoc gateways.

3. Beyond 3rd Generation (3G) cellular ad hoc networks

   Cellular ad hoc networks are self-organizing multi-hop networks with multiple access points connected with broadband cores. Few researchers are currently looking at such beyond 3G possibilities.

Compared to wired infrastructure-based networks, MANETs possess a number of unique challenges. In MANETs, heterogeneous nodes constantly move in the environment, hence constantly changing the network's topology. Ad hoc networks are decentralized with no single controlling entity. Often nodes can not communicate directly with each other; therefore, routes between nodes may need to traverse several hops in order to reach a destination. Such properties of ad hoc networks thus invalidate many of the assumptions that protocols are based upon. Adaptations need to be introduced starting from the physical layer up to the applications.

Routing in ad hoc networks was among the main problems that caught the

attention of the research community. The MANET working group of the Internet Engineering Task Force (IETF) [1] was formed to investigate and develop candidate standard Internet routing support for mobile, wireless IP autonomous segments. Today, hundreds of routing protocols already exist. Another challenging task that researchers are currently looking into is the problem of service discovery introduced in Section 2.3.

2.2 Applications of Mobile Ad Hoc Networks

The only limit to the list of possible applications of wireless mobile ad hoc networks is one's imagination. Nonetheless, existing applications were examined and classified. Hence, the following taxonomy of MANET applications was devised.

1. Business and Commercial Applications

   (a) Low mobility situations

   - Meeting to brief clients on a given assignment; a presenter may use the equipment available at the client's office or meeting room.

   - Collaborative conference where attendees share each others' resources. In such scenarios, mobile nodes are static most of the time. Movement is infrequent, and when it occurs, nodes travel with low speeds in small areas. The number of users and services is almost equal.

(b) High mobility situations

Event coverage scenario, where the media and audience follow one or multiple spokesmen or events. While mobile nodes may travel in bigger area in such scenarios, they usually travel together. There are typically more users than available services. Moreover, many users may need to access the same set of services simultaneously.

2. Crisis Management Applications

(a) Almost-static Target

- Physician in a mall answering a rescue call from another shopper.

- Fire/Safety/Rescue operations. In such scenarios, few nodes span large areas. Hence, the nodes are usually far from each other.

(b) Moving Target

Police pursuits; where police cars are trying to catch an escaped criminal for instance. In such scenarios, nodes are constantly following each other with high speeds.

3. Personal Area Networking Application

To eliminate wires between various devices at home or office. For instance, communicate via a Personal Digital Assistant (PDA) device with heating, phone, TV, and alarm systems. Hence, there are typically more services than users.

4. Residential Mesh Networks

   Such networks allow neighbors to connect their home networks together through wireless technologies. Neighborhood community networks allow faster and easier dissemination of cached information that is relevant to the local community. A mesh network is reliable and offers redundancy. Such networks are usually huge in terms of spatial size, and are composed of numerous nodes. Almost all services offered are identical. The network is typically static, with very little mobility taking place.

5. Vehicle Applications

   (a) Vehicle-roadside networking

   - Vehicles communicating with each other to provide an active safety system.

   - Real-time traffic congestion and routing information conveyed to vehicle by road-side mounted stations. Such networks hence are usually fairly large, with many duplicated services.

   (b) Vehicle-Passenger Networking

   Passenger networking with devices in an automobile. An “office car” may contain a fax, phone, and a printer that a passenger may want to utilize via his PDA or laptop. Such networks are hence usually fairly small, in terms of spatial area and number of nodes. Movement is also very limited.

6. Military Applications

   (a) Organized motion

   Communication between soldiers and base stations during a military march. Although such networks are huge, nodes are fairly close to each other, usually organized in a column fashion, and travel in similar speeds and fashion. The number of users and services is fairly balanced.

   (b) Unexpected continuous motion

   Communication between soldiers, vehicles, helicopters, and base stations during a combat situation. Nodes in such scenarios are moving constantly in a random unpredictable fashion with fairly high speeds. They travel in huge spatial areas. There are many nodes constituting the network. Furthermore, there are many more users than available services; most users typically request services simultaneously.

2.3 Introduction to Service Discovery

The pervasive ubiquitous computing infrastructure should, at least in theory, facilitate the availability of a large number of various services to end-users. Such services vary from standardized ones such as email Internet Message Access Protocol (IMAP) and X.500 directory services to non-standardized services such as locating and using the nearest color laser printer capable of printing A3 sized documents.

Service discovery is thus needed to simplify the task of building, maintaining and introducing new devices and services to a network. It is essential in encouraging the development of mobile computing environments.

Imagine waiting in the transit lounge of an airport for your flight. You reach for your Pocket PC, switch it on, connect to an email server, download your email messages, and finally print the new ones to the nearest printer. This very typical scenario manifests the impact of service discovery on mobile computing.

The main challenge is for users and applications to discover the existence of services, locate the desired service among thousands of such services, and to finally securely interact and use these services. This interaction is required to take place without prior knowledge of the networking environment details. Thus, the discovery and usage of devices is required to take place with little or no prior configuration tasks.

2.4 Design Principles for Service Discovery Frameworks

In the past few years, developers and researchers have been seeking solutions to the problem of automatic service discovery. Today, tens of SDPs exist. Some of them are commercialized such as SLP [2], Jini [3], Salutation [4], Universal Plug and Play (UPnP) [5], and the Bluetooth SDP [6]. Others are being proposed such as DEAPspace [7], MARE [8], and Splendor [9].

It is important to realize however that a One-Size-Fits-All solution neither exists nor is possible. Each SDP makes choices for a set of design parameters.

Different protocols fit different contexts better. Zhu et al. provides a good classification of service discovery [10]. In the subsequent subsections, we summarize the basic design principles common to most SDP frameworks.

2.4.1 Architectures

Two different architectures of SDPs exist. Figure 1 illustrates the Non Directory-Based model, where clients and services interact directly with each other. The second model, depicted in Figure 2, shows the Directory-Based model, where users and services interact together through brokers, called directories.

1. **Non Directory-Based Model**

   Clients

   - Run on behalf of users or applications.
   - They either query a particular service or query all services available.
   - Alternatively, clients may wait for periodic service announcements from services.
   - Clients then select a service from a list of services that responded to their queries.
   - Finally, clients interact and use that service through its service interface.

Services

- Possess a name and a set of attributes.
- They either announce their types and attributes periodically to clients.
- Or alternatively wait to answer client query requests.
- They may then authenticate and/or authorize a user when its client selects a service.
- Finally services interact with clients through their service interface.

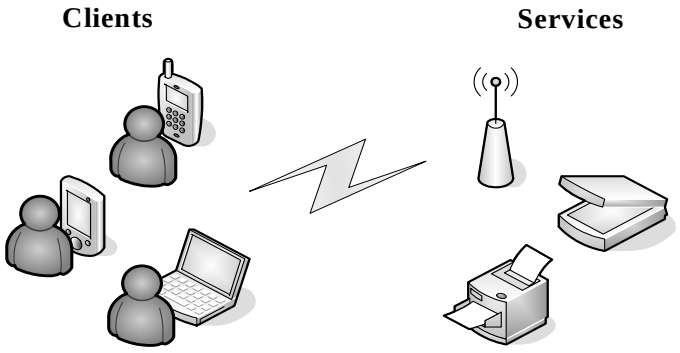


Figure 1: Non Directory-Based Model

2. Directory Based Model

Clients

- Inquire directories for services.
- Receive service information from directories.

Services

- They either announce their types and attributes periodically to directories.
- Or alternatively wait to answer service requests from directories.
- Finally, services interact with clients through their service interface.

Directories

- Act as intermediate caches of service information.
- Query available services.
- Send service information to clients (periodically, or upon request).
- Authenticate and authorize users to use services.

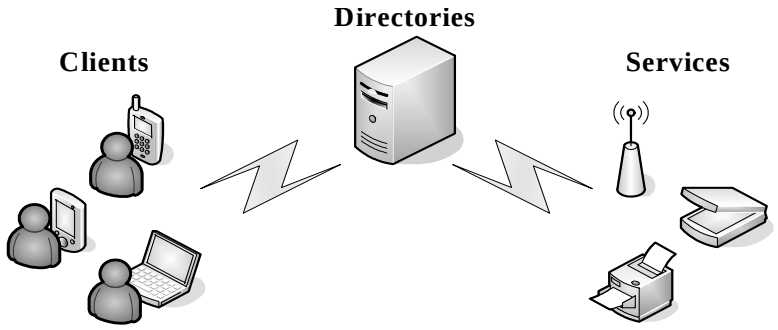


Figure 2: Directory-Based Model

2.4.2 Service Design

Each service must have a name (e.g. 'print', 'imap', 'fax'). A service name is needed to identify the type of the service. Protocols usually define naming

standards to avoid naming conflicts (e.g. instead of naming printers 'printing', 'printer', or 'prn'; they should all be named 'print'). While a printer or a fax machine are resources rather than services, the goal of service discovery protocols is to locate an address or handler of the service interface or embedded software daemon running on these resources. These service interfaces then allows direct communication and data exchange with users and applications.

A service must also have a set of attributes. These describe the properties of the service. A client requests a service by name (e.g. 'print') and possibly some attributes (e.g. 'postscript+color+duplex'). The request is then matched against available service names and attributes.

After the service is discovered, it needs to be invoked via its service interface. Some SDPs define these interfaces while others leave it for applications or other protocols to define.

Finally, a client might need to learn the status of a service (e.g. 'busy', '70% completed', 'idle'). There are two methods for conveying the status to the client. In the first, status information is 'polled' from the services when needed (pull model). In the alternative method, a service notifies a client of its status either periodically or when an event occurs (push model).

2.4.3 Service Lookup Design

There are generally two methods to look for a service. The first is the 'query' method (a.k.a. active or pull). In 'query', clients request one or more services.

Services must in turn respond to the client requests. The drawback to this method is that several clients may ask separately for the same service. The second method of service lookup is the 'announcement' method (a.k.a. passive or push), where services send periodic announcements to all clients. However, in this method, clients must handle all service announcements whether they are interested or not. Moreover, a client may have to wait for the interval between service announcements before it can learn about the available services.

In directory-based models, clients query directories for services; similarly, services send their service information to directories. Hence, directories add a second level of abstraction to the lookup process.

Services may also be 'leased'. This means that the service information distributed to clients and directories is only valid for a limited time. This is a simple way to provide fault tolerance. In case a device is powered off, breaks, or gets disconnected, all its information will eventually be invalidated.

2.4.4 Service Selection Design

After locating all services that matched the client's request, a service has to be selected for the client to use. Most protocols usually return a list of the matched services to the user and let him or her manually select a service. However, if the number of matched services is large, this process could be very time consuming and inefficient. On the contrary, other protocols may select one of the services on the user's behalf. This method, although simple, could result

in choosing a service that is against the user's will. Furthermore, protocols may implement more sophisticated selection schemes. For instance, a protocol that supports 'scope-awareness' selects a service that is within the user's current scope (e.g. within the same department or floor). 'Context-aware' protocols would similarly select the service that best suits the user's current context (e.g. services that are suitable for a cell-phone are different from those suitable for a powerful desktop). Other proposed protocols attempt to support Quality of Service (QoS). Such 'QoS-aware' protocols select the service with the least load, or one that provides a higher quality (e.g. choose printer with shortest job queue, or highest color resolution).

2.4.5 Directory Design

When directory-based SDPs are employed, one of the most important concerns to clients and servers is reaching the directory. One of four methods is normally employed. In the first method, clients and services listen on known multicast channels for directory agent advertisements. Secondly, clients may choose to query for a directory server by multicasting or broadcasting a request. In the third method, Dynamic Host Configuration Protocol (DHCP) servers are used to configure the clients and services with the address of their directory. The last and least favorable solution is the manual configuration of the directory's address at every client and server.

Another attribute associated with the design of directories is whether they

support and allow for Directory-to-Directory communication. Directory-to-Directory communication can enhance the performance and scalability of a SDP. However, consistency mechanisms have to be employed in order to assure that service information is consistent across directories.

Chapter 3

Related Work

3.1 Review of Existing Service Discovery Protocols

3.1.1 Commercialized SDPs Adaptable to Ad Hoc Environments

Bluetooth SDP [6]: The Bluetooth SDP was developed by the Bluetooth Special Interest Group (SIG). The protocol targets Bluetooth capable devices. A client queries services directly by type and attributes. All matching services respond, thus allowing the client to locate services. There are no directories involved due to the ad hoc nature of the environment. Since the protocol does not offer functionality to access these services, other complementing protocols may be used.

Service Location Protocol (SLP) [2]: SLP is the IETF solution to service discovery. Services are represented by Service Agents (SAs), while User Agents (UAs) mediate for users and clients. Although directories, represented by Directory Agents (DAs) are optional, they are required for scalability and efficiency. String based querying is used to search for services. Alternatively, services could be browsed. Universal Resource Locators (URLs) are used to access services. Scope awareness is supported, and can be utilized to search for services within a particular scope.

Universal Plug and Play (UPnP) [5]: UPnP, initiated by Microsoft, relies on a suite of widely available web protocols and open standards in order to provide a powerful solution to the problem of service discovery. UPnP is based upon the TCP/IP protocol stack. New clients and services joining a network acquire an IP address either via a DHCP server or Auto-IP. In UPnP, directories, called control points, are optional. Services announce their presence using the Simple Service Discovery Protocol (SSDP), which in turn uses HyperText Transfer Protocol (HTTP) via User Datagram Protocol (UDP). Service names and attributes are described using the eXtensible Markup Language (XML). A URL pointing to the XML file is included in the advertisement messages. To invoke and control a service, the Simple Object Access Protocol (SOAP) is used. The General Event Notification Architecture (GENA) is used to format status messages, which are sent when the service status changes. Clients and directories interested need to register with services to receive these status messages. Moreover, UPnP is capable of providing URLs for user interfaces, allowing users to easily control as well as view the status of a service.

3.1.2 SDPs Proposed for Ad Hoc Environments

DEAPspace [7]: DEAPspace is proposed by IBM Research to address the problem of service discovery in single-hop ad hoc networks. The goal is to

minimize the time needed to discover new devices in a network without incurring a significant increase in bandwidth and power requirements. The protocol uses the proactive push method for flooding service announcements. However, each device not only announces the service it provides, but also announce all those it is aware of. This increases responsiveness, since a lost advertisement will not significantly affect the "world-view" that the devices possess. Moreover, a new device joining the network does not need to wait for all other devices offering services to advertise in order for it to acquire a complete overview of the available services.

Konark [11]: Konark, proposed by Helal et. al., is a service discovery and delivery protocol designed for large wireless multi-hop ad hoc networks. The discovery process is completely distributed. Each device includes a Konark Application that facilitates human control of service discovery functions, such as advertising and service invocation. The Konark application sits on top of managers and service registries, where the service discovery process and data structures are managed. The protocol uses XML-based templates to describe services. Service invocation is achieved via a micro-HTTP server present on each device, and which is based on SOAP. All service discovery functionalities can be accessed via a set of Application Program Interfaces (APIs), which actual services can easily be equipped with.

Koodli and Perkins [12]: Koodli and Perkins proposed the integration of the service discovery problem with the mobile ad hoc network routing protocols. Extensions to both table-driven and on-demand routing protocols are added. In the first case, service information can be made available along with the route information stored in the routing tables. In the second case, service discovery uses the same operations used in route discovery by adding a service request extension to the route request packet.

Li and Lamont [13]: Li and Lamont proposed a lightweight service discovery mechanism for mobile ad hoc pervasive environments. They integrate service discovery with the routing layer. This integration allows the protocol to automatically identify the appropriate service discovery model for the current network configuration; that is detect whether a directory exists or not, as well as decide whether to pursue active or passive discovery. Service discovery is achieved by extending the messaging mechanism of the Optimized Link State Routing (OLSR) layer. The topology information obtained from the routing layer is used to enhance the protocol's capability in accommodating topology change and improve its scalability for hierarchical networks.

MARE [8]: MARE uses a combination of mobile agents and tuple spaces to address the problem of service discovery in highly dynamic ad hoc networks. The goals of the protocol are to achieve a near consistent view

of a constantly changing ad hoc network without reliance on a single central entity, and to reduce the amount of communication between the parties. The authors report results showing that the number of messages exchanged while achieving a consistent view using MARE is much less than that of SSDP and SLP. This is achieved by using agents and moving them to the point of operation rather than transmitting information back and forth across the network. Agents may also aggregate information about few services all at once (e.g. aggregate all information pertaining to all services of the same service provider at once). The authors do acknowledge that the proposed MARE lacks security mechanisms.

Zhu et al. [14]: Zhu et al. propose another service discovery protocol that is also based on proxies, but customized for ad hoc environments. The protocol solves some of the security and trust concerns that evolve in ad hoc networks when accessing unfamiliar public services. The protocol presumes a non-pure ad-hoc environment, one that has access to other network connections. Services are assumed to be connected to their service providers via the Internet. Clients must communicate with their trusted proxies using the Internet connection that the services possess with their service providers. The Internet is used to setup trust relationships and exchange security keys between the communicating partners while leaving the ad-hoc environment intact.

3.1.3 Other SDPs

Jini [3]: Jini, developed by Sun Microsystems, leverages on Java technology to provide a platform-dependent service discovery solution. It runs on Java-enabled devices, and requires the use of directories, called lookup tables. Clients and services need to first discover these lookup tables. A service provider registers its service object and attributes with the lookup table. A client requests a service by Java type and maybe some attributes. The service object code is moved during lookup to the client where it is used to interact with the service.

Salutation [4]: Salutation is developed by an open industry Salutation Consortium. Salutation runs independent of the underlying physical network and transport protocol. A Transport Manager (TM) is needed per underlying infrastructure to provide a reliable communication channel. Directories, called Salutation Managers (SMs), must sit on top of one or more TM(s). The SM acts mainly as a service broker, which provides discovery, lookup and access capabilities via a protocol-independent communication interface. SMs distributed throughout a network may exchange information with each other to increase the breadth of the search for services. Salutation also offers a version of its protocol, Salutation Lite, that is more suitable for mobile devices with limited capabilities.

INS/Twine [15]: INS/Twine is developed by the Massachusetts Institute of

Technology. The protocol goal is to handle sophisticated service attributes and queries in highly dynamic environments. More importantly, INS/Twine provides a highly scalable solution to service discovery that extends across a wide network of different administrative domains. Directories, called resolvers, co-operate with each other in a peer-to-peer fashion in order to distribute resource information and resolve queries. This is achieved by mapping attributes to numeric keys which are hashed into these directories. The keys are created in a manner that preserve expressiveness and facilitates an even distribution of data across resolvers.

Splendor [9]: Splendor, proposed by Zhu et al., is a service discovery protocol geared towards security and location-awareness for supporting mobile services. Splendor targets environments where clients and services may not have accounts in the infrastructure-based system, and therefore can not establish trust relationships. Splendor enables parties to mutually authenticate each other, as well as provide service authorization, confidentiality, integrity and non-repudiation. This is achieved using proxies whose job is to manage services. Clients querying services through directories may receive service information represented by these proxies. Proxies also use aggregation and filtering to support the mobile and dynamic nature of devices. Finally, Splendor integrates location information, which is used by location dependent services.

3.2 Review of Evaluation Frameworks

3.2.1 Review of SDP Evaluations

Zhu et al. qualitatively classify eight of the most prominent SDPs in [10]. They compare protocols according to their inherent features such as announcement and lookup method, service selection method, and directory design strategy.

Bettstetter and Renner also qualitatively compare SLP, Jini, Salutation, and UPnP based on their features in [16]. Additionally, they implement SLP using UDP sockets with the aim of demonstrating service discovery in wireless Local Area Networks (LANs). They provide tools to generate and send, as well as to listen and output SLP messages.

Barbeau realizes the importance of bandwidth usage in mobile wireless devices. He therefore characterizes the usage of bandwidth that SLP consumes and compares it to that of Jini in [17]. This task was accomplished by deriving mathematical equations for the upper bounds on bandwidth usage for both protocols.

Govea and Barbeau also compare the latency of SLP and Jini in [18]. They implement two Service Agents (SAs), one for a print service, and another for a "hello world" service along with a directory agent. The equivalent entities were implemented in Java. They find the latency of the discovery process, registration process, and lookup process.

Engelstad and Zheng evaluate several discovery architectures in [19]. They

investigate whether the benefits induced by architectures that offer additional levels of service availability (through implementing service coordinators or directories) outweigh the cost of the additional overhead.

3.2.2 Review of general MANET Protocol Evaluation Frameworks

Broch et al. provide an extensive performance comparison of multi-hop wireless ad hoc network routing protocols in [20]. Their simulations consisted of 50 wireless nodes moving in a 1500m x 300m flat space for 900 seconds. They used the "random waypoint" model to model movement; they tested with pause times of 0, 30, 60, 120, 300, 600, and 900 seconds. Two different maximum speeds of 1 m/s and 20 m/s were tested. They employed 10, 20, and 30 constant bit rate traffic sources with sending rates of 1, 4, and 8 packets per second. Their performance evaluation methodology became the "de-facto" standard in evaluating MANET routing protocols. So far, most researchers still use simulation parameters very similar to that set by Broch et al. to evaluate the performance of MANET protocols, including SDPs.

Johansson et al. compares three routing protocols in [21]. They test the protocols in three realistic contexts. These contexts consist of a conference scenario, an event coverage scenario, and a disaster area scenario. They vary the transmission range, environment size, and the speed of the nodes. More interestingly, they induce different spacial obstacles that cause various network partitions. They also present another set of results as a function of a mobility

metric designed to reflect the relative speeds of the nodes in a scenario.

3.3 Analysis and Protocol Choice

We have examined in Section 3.1 twelve of the most popular and recent Service Discovery Protocols available today. Some of them are standardized and commercialized, while others are still being proposed and developed. It is clear that no single protocol does the job better than the other. Each protocol tackles a subset of the challenges faced in the problem of service discovery. Therefore, each SDP tends to be better suited for a particular environment than another. A comprehensive SDP that is efficient, secure, reliable and works with any underlying network is unrealistic.

Two of the commercialized protocols, namely UPnP and SLP, might be adaptable to multi-hop MANETs. The Bluetooth SDP, although designed for such ad hoc networks, only supports Bluetooth-capable devices. DEAPspace, Konark, Koodli et al.'s proposal, Li et al.'s proposal, Mare, and Zhu et al.'s proposal are all service discovery proposals under construction, specifically proposed for MANETs.

Although the majority of service discovery protocols address service discovery in the application layer, some proposals as in [12] and [13], integrate the problem of service discovery with routing. While this idea may bring considerable optimizations, the regular layering approach to networking is sacrificed. In this work, we assume that the architecture employed in the ad hoc network

preserves the modular Open System Interconnection (OSI) layered approach, we therefore do not consider cross-layer approaches to service discovery.

Service Location Protocol (SLP) was the protocol of choice for our work. Firstly, it is the most commonly used service discovery protocol with multi-platform implementations available for both commercial and non-commercial applications. Some SLP Version 2 implementations are found in Sun Microsystems Solaris [22], OpenSLP [23], Novell NetWare servers [24], as well as in Axis communication server products [25]. It is an application-layer protocol that integrates smoothly with the common OSI layered approach. Furthermore, the fact that SLP is being standardized by the IETF, attracts the attention of the majority of researchers and users to adopt and develop. Moreover, unlike its newly proposed counterparts of MANET service discovery protocols, SLP has a fairly mature and complete specification publicly available through Request For Comments (RFC) 2608 [2]. Additionally, SLP is a string-based stand-alone very light-weight protocol that does not depend on a suit of other protocols to function (as in the case with UPnP). Finally, its relative simplicity greatly facilitates the development and implementation the protocol. These outlined reasons make SLP the most suitable candidate for MANETs.

Section 3.2 gave a representative review of the currently employed evaluation methodologies. Work in [10] and [16] classifies SDPs based on inherit features. On the other hand, [17], [18], and [19] tested the effect of one or more attribute on the performance of SDPs. Broch et al. [20] set a landmark by providing an

evaluation framework that researches widely adopted in testing MANET routing protocols. Although this framework is a valuable contribution, it is neither sufficiently representative of real MANET applications, nor does it suffice for purposes of service discovery evaluation. Lastly, Johansson et al. examined routing protocols through a scenario-based performance analysis where they tried to re-produce three MANET contexts in [21].

3.4 Problem Statement

This work aims to adapt, implement, evaluate, and improve the Service Location Protocol (SLP) Version 2 in the context of Mobile Ad Hoc Networks (MANETs). Additionally, this work aims to develop a framework based on reference configurations by which the performance of Service Discovery Protocols (SDPs) could be realistically evaluated in MANET environments.

Chapter 4

Service Location Protocol for MANET (SLPManet)

4.1 Introduction to SLP

Service Location Protocol (SLP) is the Internet Engineering Task Force (IETF) standard for enabling network-based applications and users to automatically locate services in Local Area Networks (LANs) with cooperative administrative control. The current Version 2 of the protocol, documented in RFC 2608 [2], was promoted to the Internet Standards Track in June 1999.

In SLP, service discovery is achieved using three entities: User Agents (UAs), Service Agents (SAs), and Directory Agents (DAs). Services are represented by SAs, while UAs mediate for users and applications. SAs broadcast service information to DAs (passive discovery), or reply to service requests (active discovery). UAs request service information (active discovery), or listen to service broadcasts from DAs (passive discovery). Directories, represented by DAs, are intermediate centralized service information repositories, which cache service information from SAs, and satisfy UAs requests accordingly. Similarly, DAs also operate in either an active or a passive fashion. Although DAs are optional, they are required when scalability or efficiency is desired. String based

querying is used to search for services. Universal Resource Locators (URLs) are used to access services; they provide users and applications with necessary service information to enable them to directly contact the discovered service. Scope awareness is supported, and can be utilized to search for services within a particular administrative scope.

4.2 Adaptations to SLP

The original Service Location Protocol (SLP) description is documented in RFC 2608 [2]. The original SLP specification is mainly intended to function within LANs under cooperative administrative control. In our work, we have adapted SLP Version 2 to Mobile Ad Hoc Network (MANET) environments, we called the adapted protocol Service Location Protocol for MANET (SLPManet). SLPManet is hence not a new protocol, instead it provides a subset of the original SLP specification that is adaptable to MANET environments. Section 4.3 discusses the adapted components of the original specification.

SLPManet implements all the SLP Version 2 required features described in [2]. All features not implemented in SLPManet were described as optional in the RFC specification of SLP, and were left out because they do not suit MANET environments. The most notable of these excluded features are:

- Directory Agents (DAs): Directory Agents are centralized stores for service information. Directory agents could exist in LANs to enhance performance and scalability of SLP. DAs do not lend themselves to MANETs

since such networks must operate without requiring the existence of certain preexisting and continuously existing nodes. As a consequence of the absence of DAs, only active discovery can take place.

- Authentication Blocks: Agents are neither configured to generate authentication blocks nor do they verify them. This optional feature is needed to prevent control of services by an adversary. Since security is not the goal of our research, we did not implement this functionality in SLPManet.

- Optional SLP messages: Messages such as Attribute Request and Service Type Request were not implemented. In a MANET context, such optional features add complexity and consumes more resources. The focus of our work is to implement the required core of the SLP protocol, which provides enough features to implement a service discovery mechanism for MANET environments.

4.3 SLPManet Specification

All features of SLP Version 2 required for proper functioning in MANETs described in this section were implemented in our work. This section also alludes to specific design decisions and assumptions.

4.3.1 Introduction

The Service Location Protocol for MANET (SLPManet) provides a flexible framework for providing hosts with access to information about the existence,

location, and configuration of networked services. Traditionally, users have had to find services by knowing the Domain Name System (DNS) name of a network host (a human readable text string), which is an alias for a network address. SLP eliminates the need for a user to know the name of a network host supporting a service. Rather, the user supplies the desired type of service. Based on that description, SLPManet resolves the network address of a matching service for the user.

SLPManet provides a dynamic configuration mechanism for applications in MANETs. Applications are modeled as clients that need to find services associated with any of the nodes within a MANET.

Applicability Statement

SLPManet is specifically intended to function within mobile ad hoc networks under cooperative administrative control. Such networks permit a policy to be implemented regarding routing and the organization of services and clients into groups which are not feasible on a larger scale, such as the Internet.

SLPManet has been designed to serve small mobile multi-hop wireless ad hoc networks with shared services, and it may not necessarily scale for wide-area service discovery, or in networks where there are hundreds of thousands of clients or tens of thousands of services.

Terminology and Notation Conventions

User Agent (UA) : A process working on the user's behalf to establish contact with some service. The UA retrieves service information from the Service Agents.

Service Agent (SA) : A process working on behalf of one or more services to advertise the services.

Service Type : Each type of service has a unique Service Type string.

Scope : A set of services, typically making up a logical administrative group.

URL : A Universal Resource Locator [26] conveys at least the name of the access protocol as well as the address locating a service. This information will be used by the UA after the discovery process in order to directly contact the discovered service.

Strings : All strings are encoded using the Universal Transformation Format (UTF)-8 [23] transformation of the Unicode [6] character set and are not null terminated when transmitted. Strings are preceded by a two byte length field.

<**string-list**> : A comma delimited list of strings with the following syntax:

string-list = string / string ',' string-list

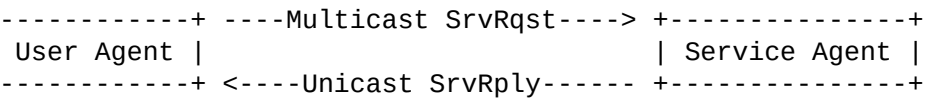
In format diagrams, any field ending with a \ indicates a variable length field, given by a prior length field in the header.

4.3.2 Protocol Overview

The Service Location Protocol for MANET supports a framework by which client applications make use of 'User Agents (UAs)' and services are advertised by 'Service Agents (SAs)'.

The User Agent issues a 'Service Request (SrvRqst)' on behalf of the client application, specifying the characteristics of the service which the client requires. The User Agent will receive a 'Service Reply (SrvRply)' specifying the location of each of the services in the network which satisfy the request.

The SLPManet framework allows the User Agent to directly issue requests to Service Agents. User Agents multicast service requests to the network. Service Agents receiving a request for a service which they advertise unicast a reply containing the service's location.



Services may be grouped together using 'scopes'. These are strings which identify services which are administratively identified. A scope could indicate a location, administrative grouping, proximity in a network topology or some other category. Service Agents are always assigned a scope string.

A User Agent is normally assigned a scope string, in which case the User Agent will only be able to discover that particular grouping of services. This allows a network administrator to 'provision' services to users.

Alternatively, the User Agent may be configured with no scope at all. In such case, it will discover all available scopes and allow the client application to issue requests for any service available on the network. A user agent hence may request Service Agents in order to discover their scope configuration. In this case, Service Agents would reply with 'Service Advertisements (SAAdverts)'.

4.3.3 URLs used with Service Location

A Service URL indicates the location of a service. In SLPManet a Service URL is of the form:

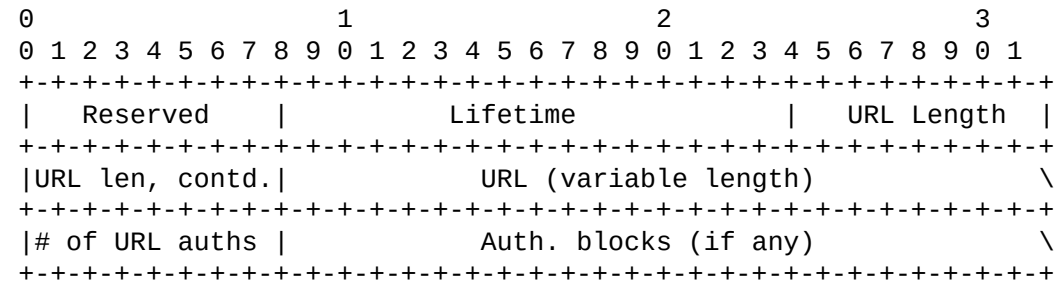
"service:"<abstract srvtype>:<concrete srvtype>:"://"<addrspec>

For example, a tftp service URL would look like:

"service:tftp://bad.glad.org:8080"

The service type string "service:<abstract-type>" matches all services of that abstract type. If the concrete type is included also, only these services match the request. For example: a SrvRqst which specifies "service:printer" as the Service Type will match the URL "service:printer:lpr://hostname" and "service:printer:http://hostname". If the requests specified "service:printer:http" only the latter URL is matched.

4.3.4 URL Entries



SLP stores URLs in protocol elements called URL Entries, which associate a length and a lifetime along with the URL. The authentication information is optional, and is not used in our implementation of SLPManet. Hence the "# of URL auths" is set to zero. URL Entries, defined as shown above, are used in Service Replies.

4.3.5 Use of Ports, UDP, and Multicast

The default reserved listening port for SLPManet in our implementation is 18. This is the destination port for all SLPManet messages. Replies and acknowledgments are sent to the port from which the request was issued. The default maximum transmission unit for UDP messages is 1400 bytes excluding UDP and other headers.

In SLPManet, transmission is carried over UDP only. If an SLP message does not fit into a UDP datagram, it will not be sent by the UDP agent.

SLP Requests messages are multicast to the Network Simulator Version 2 (NS-2) flat address:0xE000000. The default Time To Live (TTL) to use for

multicast is 255. Setting multicast TTL to less than 255 (the default) limits the range of SLPManet discovery in a network, and localizes service information in the network.

4.3.6 Retransmission of SLPManet Messages

Requests which fail to elicit a response are retransmitted. The initial retransmission occurs after a CONFIG_RETRY wait period (default=2 sec). Retransmissions are made with exponentially increasing wait intervals (doubling the wait each time).

Multicast requests are reissued over CONFIG_MC_MAX seconds (default=15 sec) until a result has been obtained. UAs may only wait till they obtain the first reply which matches their request. That is, retransmission is not required if the requesting agent is prepared to use the 'first reply'. Alternatively they may choose to wait for 'as many replies as possible within the bounded time interval'. This is controlled in SLPManet by a configuration parameter, maxSearch, which is set by default to "false".

When SLPManet SrvRqst messages are multicast, they contain a <PRList> of previous responders. Initially the <PRList> is empty. Any SA which sees its address in the <PRList> will not respond to the request.

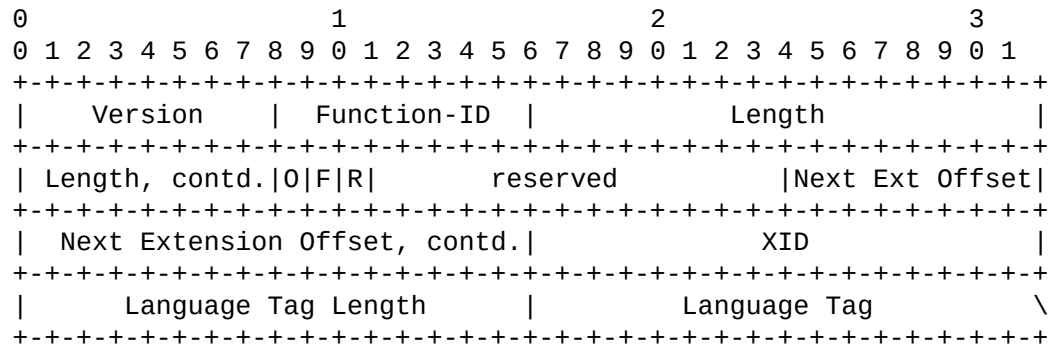
The message is retransmitted until the <PRList> causes no further responses to be elicited, or the previous responder list and the request will not fit into a single datagram, or until CONFIG_MC_MAX seconds elapse.

UAs which retransmit a request use the same XID. XIDs should be randomly chosen to avoid duplicate XIDs in requests if UAs restart frequently. To avoid duplicate XIDs in SLPManet, we kept a global counter common to all UAs, the counter is incremented every time a new unique request is issued. This unrealistic solution is possible due to the simulation environment, however, the original method of randomly choosing XIDs should be used instead.

4.3.7 SLPManet Messages

All length fields in SLP messages are in network byte order. Where 'tuples' are defined, these are sequences of bytes, in the precise order listed, in network byte order.

SLP messages all begin with the following header:

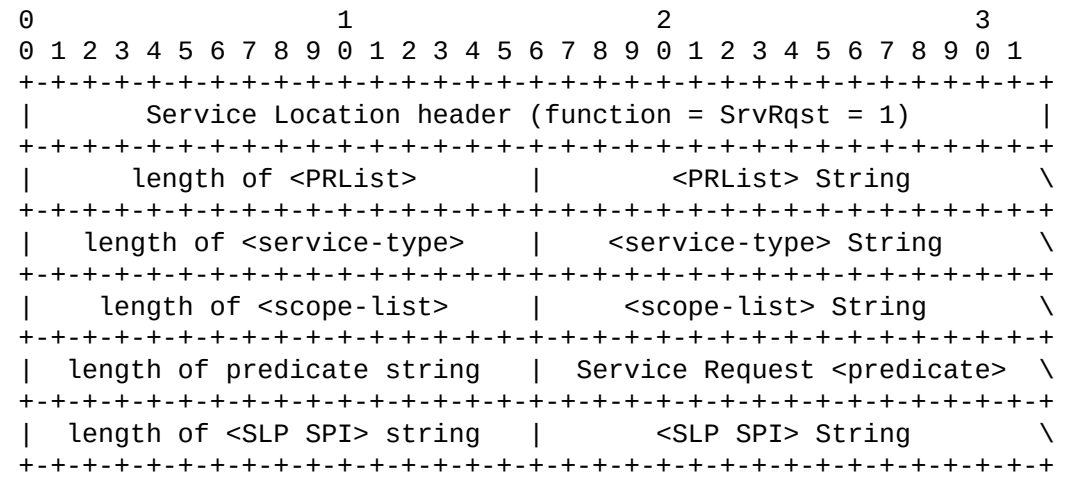


Three SLP messages are used in SLPManet. They are:

Message Type	Abbreviation	Function-ID
Service Request	SrvRqst	1
Service Reply	SrvRply	2
SA Advertisement	SAAdvert	11

- Length is the length in bytes of the entire SLP message, header included.
- The flags used: REQUEST MCAST (0x20) is set when multicasting or broadcasting requests. Reserved bits are 0.
- Next Extension Offset is set to 0 since no extensions are used.
- XID is set to a unique value for each unique request. If the request is retransmitted, the same XID is used. Replies set the XID to the same value as the XID in the request.
- Language Tag Length is the length in bytes of the Language Tag field.
- Language Tag is set to "en" in SLPManet.

Service Request



In order for a Service to match a SrvRqst, it must belong to at least one requested scope, support the requested service type, and match the predicate (if present).

<PRList> is the Previous Responder List. This <string-list> contains NS-2 flat addresses, and is iteratively multicast to obtain all possible results.

A SA silently drops all requests which include the SA's address in the <PRList>. In our implementation, an SA can only have one network interface, therefore, only a single NS-2 address could belong to a node.

Once a <PRList> plus the request exceeds the path Maximum Transmission Unit (MTU), multicast convergence stops. This algorithm is not intended to find all instances; it finds 'enough' to provide useful results.

The <scope-list> is a <string-list> of configured scope names. SAs which have been configured with any of the scopes in this list will respond.

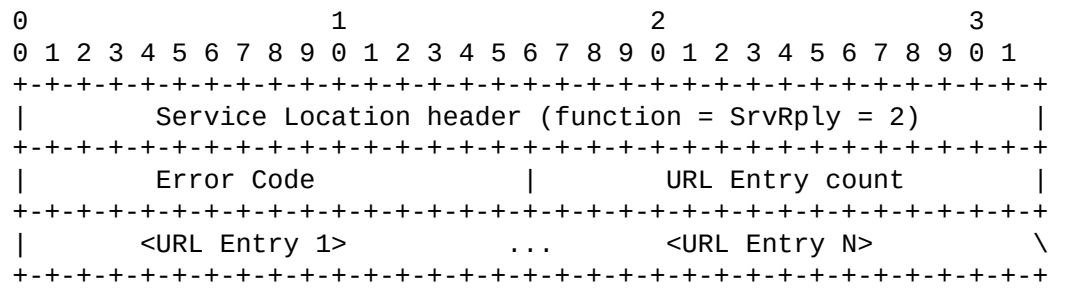
The <service-type> string is discussed in Section 4.3.3. Normally, a SrvRqst elicits a SrvRply. The only exception is when the <service-type> is set to "service:service-agent", in this case, SAs respond with SAAdvert messages. In SLPManet, an SA will not reply unless it has a matching <service-type>, therefore this field will never be omitted.

The <predicate> is a Lightweight Directory Access Protocol Version 3 (LDAPv3) search filter. This field is optional and not implemented in SLPManet. Hence, the <predicate> length field is always set to zero and the <predicate> string is always empty. Services are discovered simply by type and scope.

The <SLP SPI> string indicates a SLP Security Parameter Index (SPI) that the requester has been configured with. This feature is intended for UAs to

verify service information through digital signatures. This field is optional and is omitted in SLPManet, hence, the responder does not include any Authentication Blocks in its reply. This signifies that the <SLP SPI> length field is always set to zero, and the <SLP SPI> string is always empty.

Service Reply



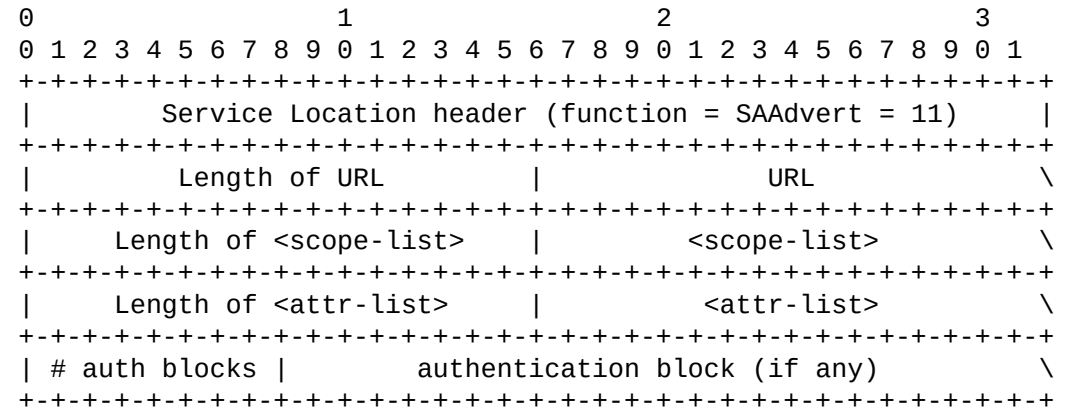
The service reply contains one or more URL entries (see Section 4.3.3). A service reply with zero URL entries is not sent in response to a multicast or broadcast service request (instead, if there was no match found, the service reply will not be generated at all).

Since, UDP is used to send the SLPManet packets, a URL Entry will not be included unless it fits entirely without truncation.

When a UA receives a service reply, it caches the service information provided in each of the URL entries for the duration indicated in each URL Entry. Every time a service is required, the UA first searches its internal cache for an entry whose service type matches the requested service. Only if none of the cached services match the required service, a new service request is issued.

Service Agent Advertisement

User Agents can not solicit SA Advertisements if they have been configured with a <scope-list>. UAs solicit SA Advertisements only when they are explicitly configured to use User Selectable scopes in order to discover the scopes that SAs support (see Section 4.3.8).



The SA responds only to multicast SA discovery requests which either include no <scope-list> or a scope which they are configured to use.

The URL is "service:service-agent://"<addr> of the SA, where <addr> is the NS-2 address of the SA.

The SAAdvert may include a list of attributes the SA supports. This attribute list should be kept short so that the SAAdvert will not exceed the path MTU in size. However, since attributes are not used in SLPManet, the length and string field of the <attr-list> are set to zero and null respectively.

The SAAdvert contains no SAAdvert authentication blocks in our implementation since digital signature verification was not implemented in SLPManet.

4.3.8 Scopes

Scopes are sets of services. The primary use of Scopes is to provide the ability to create administrative groupings of services. A set of services may be assigned a scope by network administrators. A client seeking services is configured to use one or more scopes. The user will only discover those services which have been configured for him or her to use. By configuring UAs and SAs with scopes, administrators may provision services. Scope strings are case insensitive. The default scope string is "DEFAULT". Scopes are the primary means an administrator has to scale SLPManet deployments to larger networks.

SLPManet messages which fail to contain a scope that the receiving Agent is configured to use are dropped. Every SrvRqst (except for SA discovery requests) and SAAdvert message must include a <scope-list>.

Administrative and User Selectable Scopes

All requests and services are scoped. The only exception is SrvRqsts for "service:service-agent". These may have a zero-length <scope-list> when used to enable the user to make scope selections. In this case UAs obtain their scope from SAAdverts.

Otherwise, if SAs and UAs are to use any scope other than the default (i.e. "DEFAULT" scope), the UAs and SAs are configured with lists of scopes to use. In practical deployments, this could be achieved perhaps automatically by way of DHCP option 78 or 79. Such administrative scoping allows services to

be provisioned, so that users will only see services they are intended to see.

User configurable scopes allow a user to discover any service, but require them to do their own selection of scope. This is similar to the way AppleTalk and Server Message Block (SMB) networking allow user selection of AppleTalk Zone or workgroups.

Note that the two configuration choices are not compatible. One model allows administrators control over service provision. The other delegates this to users (who may not be prepared to do any configuration of their system).

In our implementation of SLPManet, if Service Agents are not configured with particular scopes, they default to "DEFAULT". Scopes of User Agents are either manually configured, or learned from the service advertisements received in response to a service request for "service:service-agent" if the UAs were configured to use user selectable scopes. In SLPManet, UAs are configured to use user selectable scopes if no scopes were manually assigned to them at the time the UAs are enabled (refer to Appendix A.3).

4.4 Modified SLPManet

4.4.1 Review of Extended Caching Approaches

Raman et al. argues for extensive cross-layer optimizations in the context of Bluetooth scatternets in [27]. They propose an integrated approach that combines both routing and service discovery in a single protocol layer. In this integrated approach, they propose to minimize scatternet-wide floods through

caching of service discovery information at all intermediate nodes. Consider the network shown in Figure 3. When client C1 looks for service **X**, a service discovery query is flooded, nodes S1 and S2 have matching services, and thus unicast their replies to C1. The service information offered by node S1 is cached in the intermediate nodes N1 and N2. When C2 initiates a search for **X**, N1 would send a service reply back to C2, without re-broadcasting the service request since S1's service information was cached at N1 from the previous discovery process (i.e. cache hit).

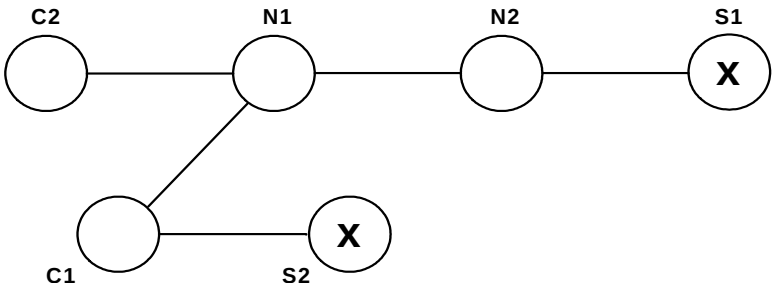


Figure 3: Extended caching approaches

Motegi et al. present in [28] another approach to service discovery closely related to Raman et al.'s caching approach described above. The authors foresee that the approach described in [27] has the potential drawback of lowering the number of discoverable services, since clients can not discover other services whose information is not cached at the responding (intermediate) nodes. For instance, in the service discovery scenario described above, C2 will not learn about service **X** offered by S2.

Motegi et al. propose to induce a new service-information distribution process to make up for the lack of cached configuration information of the existing method. They extend SLP message formats to include next-hop information for routing to clients and responded nodes.

When C1 sends a service request for service **X**, it receives two service replies, one from S1 and another from S2. Different service replies signify that there are some services that the responding and intermediate nodes does not know about. Therefore, the client, C1, generates a service reply including service information that the responding and intermediate nodes do not have. It sends this reply to the nodes that responded. When an intermediate node receives a service reply from the client, it caches the service information and then sends the service reply to the next-hop node for routing to the responded node.

4.4.2 Analysis of Existing Caching Approaches

We identify three major disadvantages with the caching approaches identified above:

1. Both approaches described involve cross-layering through integrating routing with service discovery, thus the OSI layered approach is sacrificed. Consequently, in [28], the SLP headers must be extended to include next-hop information for routing from client to responded nodes.

2. The first approach has the potential drawback of lowering the number of discoverable services as the authors of [28] have identified.

3. The latter approach can also induce increased flooding of messages due to the distribution of service replies. In MANET scenarios where a service is not like to be used repeatedly, or the service advertisement lifetime is short, this method will result in decreased efficiency. The distribution of service replies in this case will only result in useless flooding.

4.4.3 Proposed Extended Caching Approach

We propose a simple caching optimization. All intermediate nodes that are also participating as clients or User Agents (UAs) in the service discovery process shall cache service information found in the Service Replies (SrvRplys) that they relay. This way, if one of these intermediate UAs wish to use a service that was previously cached, it need not issue a new Service Request (SrvRqst). UAs must not generate SrvRplys for cached service information.

Therefore, in the discovery scenario previously described, and depicted in Figure 3 on page 46, when C2 requests service **X**, C1, N1 and N2 will all re-broadcast the SrvRqst as usual. However, when N1 or N2 need service **X**, they will not send out a SrvRqst since service **X** is locally cached (assuming that the cached service URL has not yet timed out).

The proposed modification to SLPManet:

1. Increases the performance and efficiency of SLPManet. Increased caching at UAs results in reduced flooding of messages since less SrvRqsts need to

be issued. Reduced message flooding translates into less network bandwidth consumption. Similarly, increased cache hits means that the average discovery latency is reduced, and the overall discovery success rate should improve.

2. Does not sacrifice the layered OSI approach to networking. No routing information is needed by the service discovery application, hence, SLPManet headers need not be amended.

3. No extra distribution of information is needed. Service information is learned by intermediate UAs nodes for free. The only cost is that of caching. Therefore, scenarios with short service advertisement lifetimes, or where services are not likely to be re-used will not result in decreased protocol efficiency.

4. The number of discoverable services is not lowered as much as in [27]. All intermediate nodes still re-broadcast service requests even if they have cached hits.

5. The modification conforms to the original specification of SLPManet; UAs issue SrvRqsts while only SAs respond with SrvRplys.

However, a possible drawback to caching is that in the event that a network topology change occurs, a cached service may no longer be the optimal (or closest) service. For instance, if N1 in Figure 3 on page 46 moved beside S2,

then accessing service **X** provided by S2 is probably more optimal than the already cached service (i.e. service **X** provided by S1). This is a general problem associated with designs that do not have access to routing information (i.e. non-cross layer designs). Therefore, it is essential that the service URL lifetimes are representative of the rate of network topology change.

Chapter 5

MANET Reference Configurations

In order to fairly evaluate service discovery protocols, we need to establish realistic reference configurations that reflect the various MANET applications. We therefore develop a benchmark, Benchmark for MANET (BENCHManet), consisting of a test for each of the applications described in Section 2.2. It is important that each scenario mimic as closely as possible the environment and unique features found in the application the scenario represents. Section 5.1 examines the mobility behavior found in each of the applications, and associates mobility parameters to each benchmark test accordingly. Section 5.2 looks at other networking characteristics in order to appropriately choose values for the other simulation parameters.

5.1 Modeling Mobility

Throughout recent years, researchers have tended to base most of their results on the random way-point mobility model. However, many of the applications described in real life possess different mobility models.

Mobility models fall into two major groups: entity mobility models, and group mobility models. In the first, the movement of Mobile Nodes (MNs) are

independent of each other, while in the latter, decisions on movement depend upon other mobile nodes in the group.

Camp et al. survey and analyze the various mobility models in [29]. They show that:

- The performance of an ad hoc network protocol can vary significantly with different mobility models.
- The performance of an ad hoc network protocol can vary significantly when the same mobility model is used with different parameters.
- The performance of an ad hoc network protocol should be evaluated with the mobility model that most closely matches the expected real-world scenario.

5.1.1 Entity Mobility Models

We describe three entity models that could be easily employed in our simulation environment. Network Simulator Version 2 (NS-2) code is provided by the authors of [29], [30], and [31] in order to generate movements patterns. Other entity mobility models not described here may possibly model some of the applications better. However, readily applying them is hindered by lack of simulation support (e.g. a Boundless Simulation Area Mobility Model needs a radio propagation models that wraps transmissions across the edges of the simulation area).

1. Random Walk Mobility: A simple mobility model based on random directions and speeds. A MN moves from its current location to a new location by randomly choosing a direction and speed in which to travel. It is a memory-less model where no knowledge of past locations and speed values is retained. If a MN reaches the simulation boundary, it bounces off with an angle equal to the incident angle, movement then continues in the new direction. When the parameter (i.e. distance or time) to change a MN's direction and speed is small, the movement pattern results in a random roaming pattern restricted to a small portion of the simulation area. Therefore, when a semi-static network is being evaluated, this parameter should be small, otherwise, a large value should be given.

2. Random Waypoint Mobility Model: A model that includes pause times between changes in destination and speed. Once this time expires, the MN chooses a random destination in the simulation area as well as a speed by which it travels. MNs in this model tend to choose a destination near the center of the simulation area, or a destination that requires travel through the center of the simulation area. The authors of [30] and [31] show that the traditional random waypoint model could lead to unreliable results due to its instability. They propose a modified version that prevents this problem by setting a non-zero minimum speed, hence converging the simulation results to a constant and stable level. We adopt this improved

model in our simulations.

3. Random Direction Mobility Model: A model that forces MNs to travel to the edge of the simulation area before changing direction and speed. Hence, this model prevents clustering of nodes in one part of the simulation area (unlike the random waypoint model). This model is useful when MNs are likely to spread themselves evenly in the simulation area.

5.1.2 Group Mobility Models

1. Column Mobility Model: A group mobility model where a set of MNs form a line and are uniformly moving forward in a particular direction. Initially, each MN has a reference point, where all points form a line. Individual MNs are allowed to move randomly around their respective reference points using an entity mobility model. The new reference points are found by advancing the current reference points forward.

2. Nomadic Community Mobility Model: A group mobility model where a set of MNs move together from one location to another. Each MN roams randomly around a single reference point using an entity mobility model. When the reference point changes, all MNs in the group travel to the new area defined by the reference point, then start roaming again around the new reference point.

3. Pursue Mobility Model: A group mobility model where a set of MNs follow

a given target. The amount of randomness of the individual pursuing MNs is limited in order to maintain effective tracking of the MN being pursued.

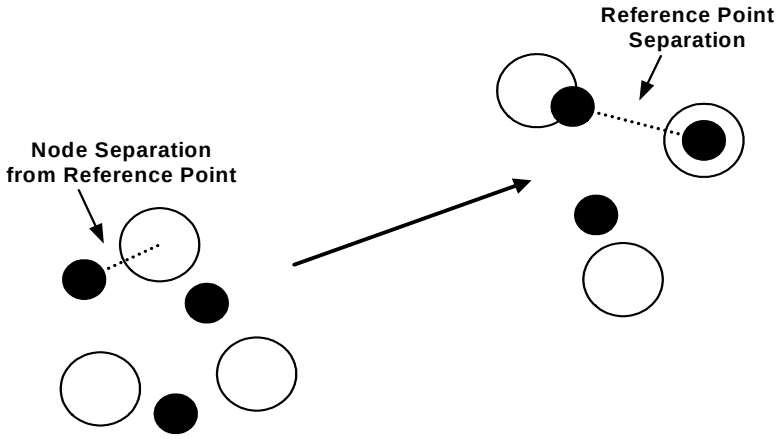


Figure 4: Reference Point Group Mobility (RPGM) model

The Reference Point Group Mobility (RPGM) model is a generic method for handling group mobility. An entity mobility model needs to be specified to handle both the movement of a group of MNs and the movement of the individual MNs within the group. This mobility model is also implemented by [29]. In Figure 4, the movement of a group of three MNs (white circles) is based on the path traveled by a logical center for the group. Individual mobile nodes move randomly about their own predefined reference points (black circles), whose movements depend on the group's movement. Varying the reference point separation and node separation from reference point parameters can yield the three group mobility models previously described (i.e. Column, Nomadic, and Pursue). To derive the column mobility model from RPGM, a

restriction that reference points must be arranged in a column which is either parallel or perpendicular to the direction of travel must be enforced.

5.1.3 Mobility models for MANET applications

We used the mobility models' anatomy summarized in Sections 5.1.1 and 5.1.2 to aid us in affiliating a suitable mobility model with each of the scenarios constituting BENCHManet (refer to Section 2.2 on page 6 for application types).

Table 1 on page 59 describes the mobility characteristics of each of the applications. Table 2 on page 59 shows the mobility model and parameters used for each of the MANET applications composing BENCHManet. $\overline{Speed}$ is the mean speed in m/s. $\Delta Speed$ describes the variation in speed. For instance, if $\overline{Speed} = 2$ and $\Delta Speed = 1$, the speed varies between 1 m/s and 3 m/s. $\overline{Pause}$ and $\Delta Pause$ are similarly the mean pause time and pause time variation in seconds. The pause times in Table 2 assume 2000 seconds of simulation.

The movement pattern observed in application type 6b (refer to Section 2.2 on page 6) such as that found in a Military Combat scenario (**combat**), could be best represented by the random walk mobility model. Jet fighters, tanks, helicopters and soldiers move unexpectedly with speeds and directions independent of past speeds and directions, hence generating sudden stops and sharp turns. Such patterns are found in the random walk model. Figure 14 on page 64 shows the movement trace of one node in **combat**.

Motion prevailed in application types 1a, 3, 4, 5a, and 5b such as that

found in Collaborative Conference scenario (**conf**), Personal Area Networks scenario (**pan**), Residential Mesh Networks scenario (**mesh**), Vehicle-Roadside scenario (**vr**), and Vehicle-Passenger scenario (**vp**) respectively, is best described by the random-waypoint model. In such applications, nodes move to random destinations with varying speeds, they then pause for varying durations and then start traveling to the next destination. Speeds and pause times are chosen to mirror as closely as possible speeds and pauses found in the different applications. Figures 5, 9, 10, 11, and 12 on pages 60 to 63 are movement traces of a single node in each of the **conf**, **pan**, **mesh**, **vr** and **vp** scenarios respectively.

MANET applications of type 2a, such as that found in a Rescue Operation scenario (**rescue**) could be best represented by the random direction model. The rescue team usually spread equally apart in order to find their target. This behavior is induced by the random direction model. Figure 7 on page 61 shows the movement trace of a node in the **rescue** scenario.

In applications of type 6a, such as a Military March scenario (**march**), entities move together forward towards a target. Therefore, the column mobility model best suits this group mobility behavior. Moreover, entities move within a small vicinity around their reference points (while still marching forward with the group), therefore, a random waypoint model is used in conjunction with the column mobility model to represent motion of the individual entities. The RPGM model was used to derive the column mobility model by enforcing reference points to be parallel to the direction of travel. Figure 13 on page 64 shows

the movement traces of the nodes in the **march** scenario.

Applications of type 1b, such as in an Event Coverage scenario (**event**), groups of reporters, cameramen, and journalists maintain random distances and motion with respect to each other, while they follow an event or spokesman. This behavior is thus best modeled with the nomadic community mobility model. The RPGM mobility model was also used to derive the nomadic mobility model. Figure 6 on page 60 shows the movement traces of the nodes in the **event** scenario.

Finally, the pursuit mobility model best suits applications of type 2b, such as a Police Pursuit scenario (**pursuit**). Usually, police cars are involved in a high speed chase of a suspect or an escaped criminal. Police cars do not move quiet randomly within the area of their individual reference points because their main aim is to catch the target. The RPGM model was similarly used to derive the pursuit mobility model. Figure 8 on page 61 shows the movement traces of the nodes in **pursuit**.

	Application	**Characteristics**
Business & Commercial	(1a) conf	Human speed, long pauses
	(1b) event	Human speed, long pauses
Crisis Management	(2a) rescue	Human and vehicle low speeds, moderate pauses
	(2b) pursuit	Vehicle high speeds, no pauses
Home Networks	(3) pan	Very low speeds, long pauses
Mesh Networks	(4) mesh	Almost static; very low speeds with very long pauses
Vehicle Applications	(5a) vr	Vehicle moderate speeds, short pauses
	(5b) vp	Almost static; very low speeds with moderate pauses
Military Applications	(6a) march	Human speeds, no pauses
	(6b) combat	High speeds, no pauses

Table 1: Mobility Characteristics of MANET applications

Application	**Mobility Model**	$\overline{\textbf{Speed}}$	**Δ Speed**	$\overline{\textbf{Pause}}$	**Δ Pause**
(1a) conf	Random Waypoint	0.60	0.50	60	60
(1b) event	Nomadic	0.50	0.50	60	60
(2a) rescue	Random Direction	5.00	4.00	30	10
(2b) pursuit	Pursuit	20.00	10.00	0	0
(3) pan	Random Waypoint	0.30	0.25	60	60
(4) mesh	Random Waypoint	0.30	0.25	250	0
(5a) vr	Random Waypoint	14.00	13.00	10	10
(5b) vp	Random Waypoint	0.60	0.50	30	10
(6a) march	Column	1.00	1.00	0	0
(6b) combat	Random Walk	20.00	15.00	0	0

Table 2: Mobility Model and Parameters of MANET applications

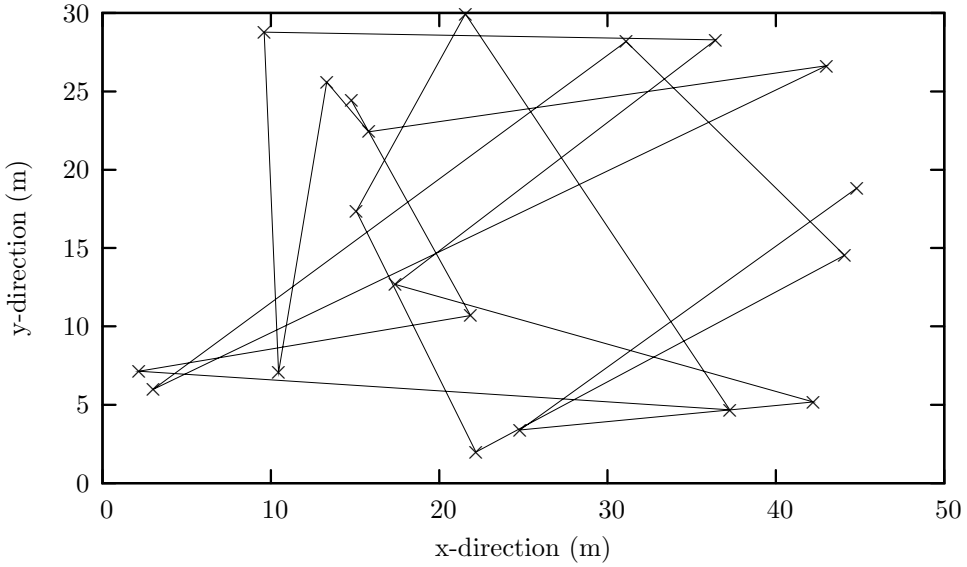


Figure 5: Movement trace of 1 out of 40 nodes in **conf** during 2000 s. Random Waypoint mobility model, $Speed = 0.60 \pm 0.50\ m/s$, $Pause = 60 \pm 60\ s$.

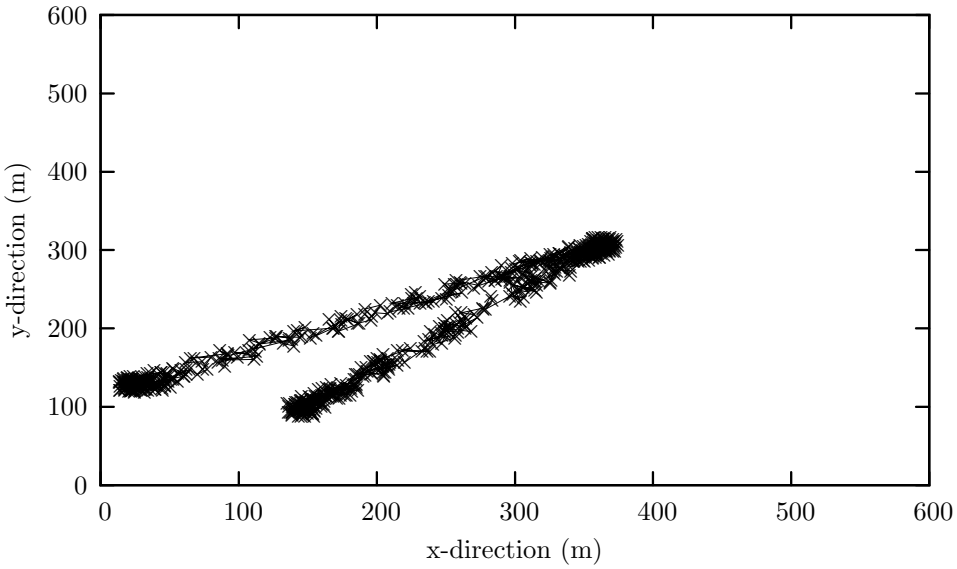


Figure 6: Movement trace of a group of 40 nodes in **event** during 2000 s. Nomadic mobility model (using RPGM), Reference points separation = 0 m, Node separation from reference point = 10 m, $Speed = 0.50 \pm 0.50\ m/s$, $Pause = 60 \pm 60\ s$.

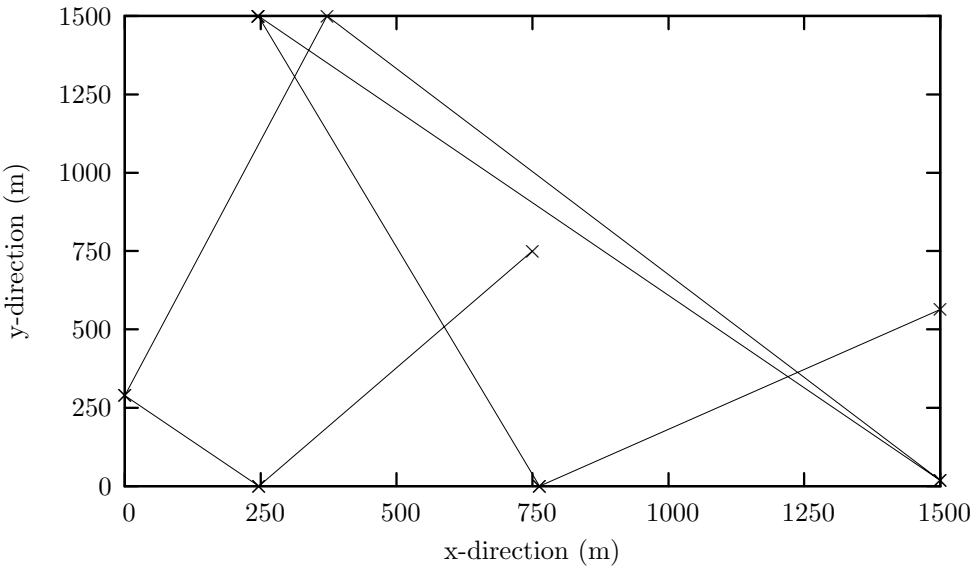


Figure 7: Movement trace of 1 out of 100 nodes in **rescue** during 2000 s. Random Direction mobility model, $Speed = 5.00 \pm 4.00\ m/s$, $Pause = 30 \pm 10\ s$.

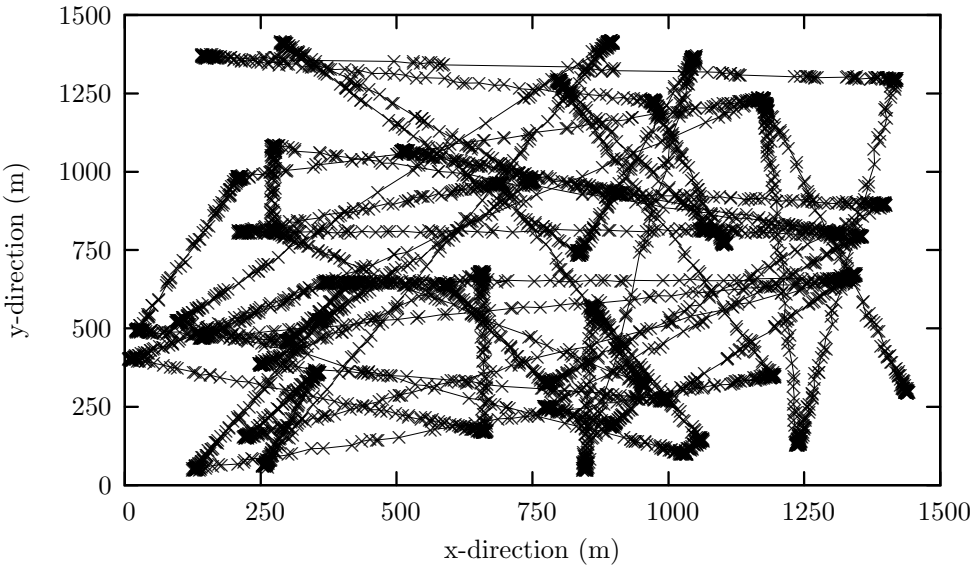


Figure 8: Movement trace of a group of 10 nodes in **pursuit** during 2000 s. Pursuit mobility model (using RPGM), Reference points separation = 0 m, Node separation from reference point = 5 m, $Speed = 20.00 \pm 10.00\ m/s$, $Pause = 0 \pm 0\ s$.

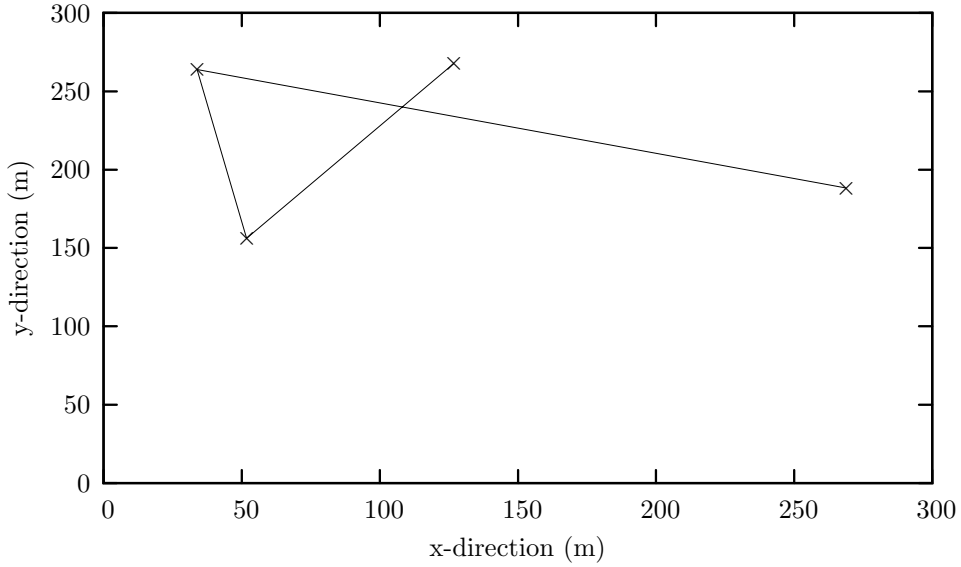


Figure 9: Movement trace of 1 out of 20 nodes in **pan** during 2000 s. Random Waypoint mobility model, $Speed = 0.30 \pm 0.25\ m/s$, $Pause = 60 \pm 60\ s$.

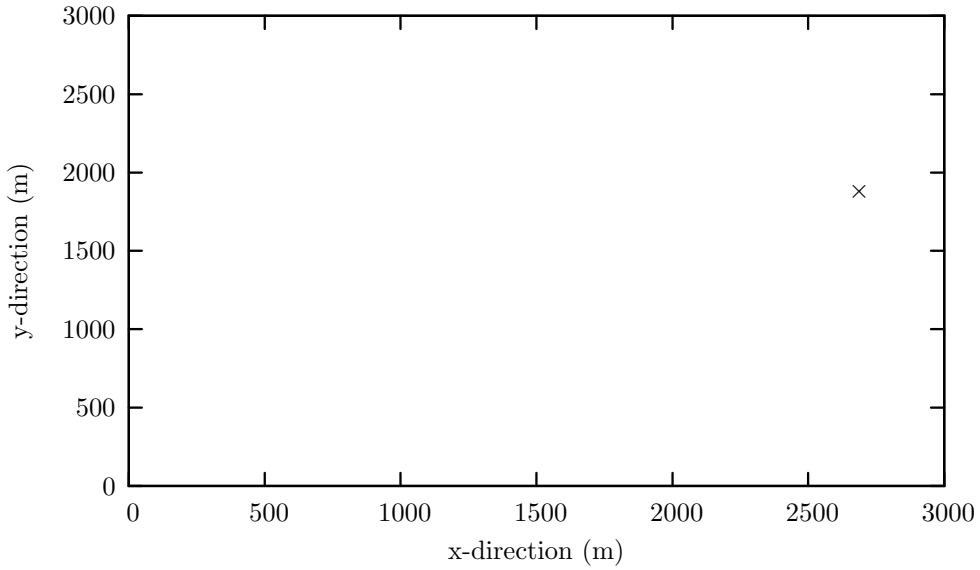


Figure 10: Movement trace of 1 out of 300 nodes in **mesh** during 2000 s. Random Waypoint mobility model, $Speed = 0.30 \pm 0.25\ m/s$, $Pause = 250 \pm 0\ s$.

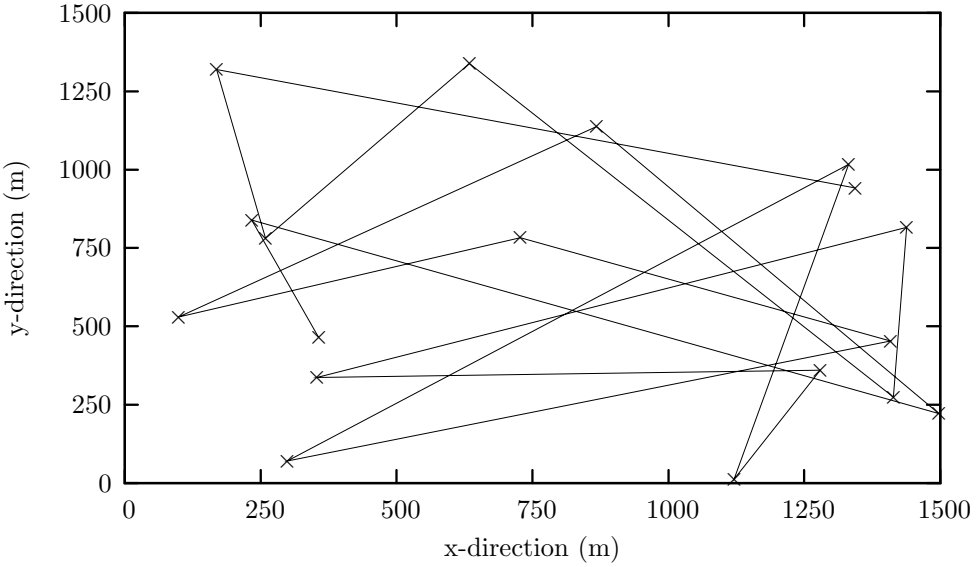


Figure 11: Movement trace of 1 out of 80 nodes in **vr** during 2000 s. Random Waypoint mobility model, $Speed = 14.00 \pm 13.00\ m/s$, $Pause = 10 \pm 10\ s$.

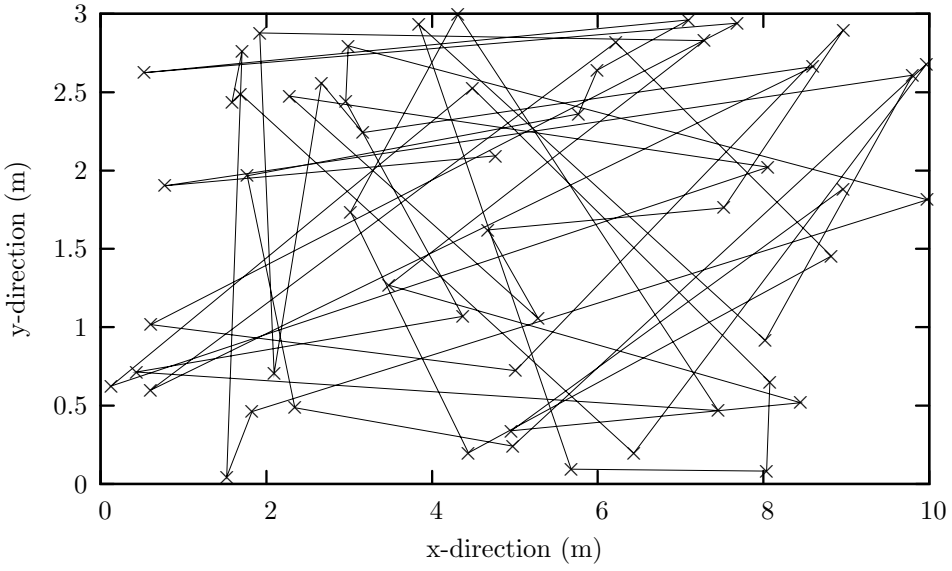


Figure 12: Movement trace of 1 out of 6 nodes in **vp** during 2000 s. Random Waypoint mobility model, $Speed = 0.60 \pm 0.50\ m/s$, $Pause = 30 \pm 10\ s$.

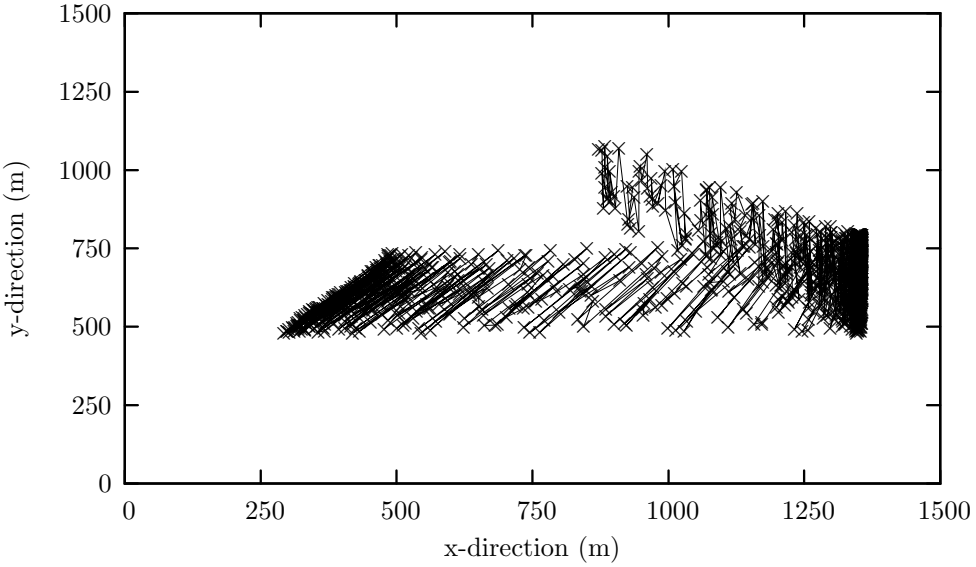


Figure 13: Movement trace of a group of 50 nodes in **march** during 2000 s. Column mobility model (using RPGM), Reference points separation = 10 m, Node separation from reference point = 5 m, $Speed = 1.00 \pm 1.00\ m/s$, $Pause = 0 \pm 0\ s$.

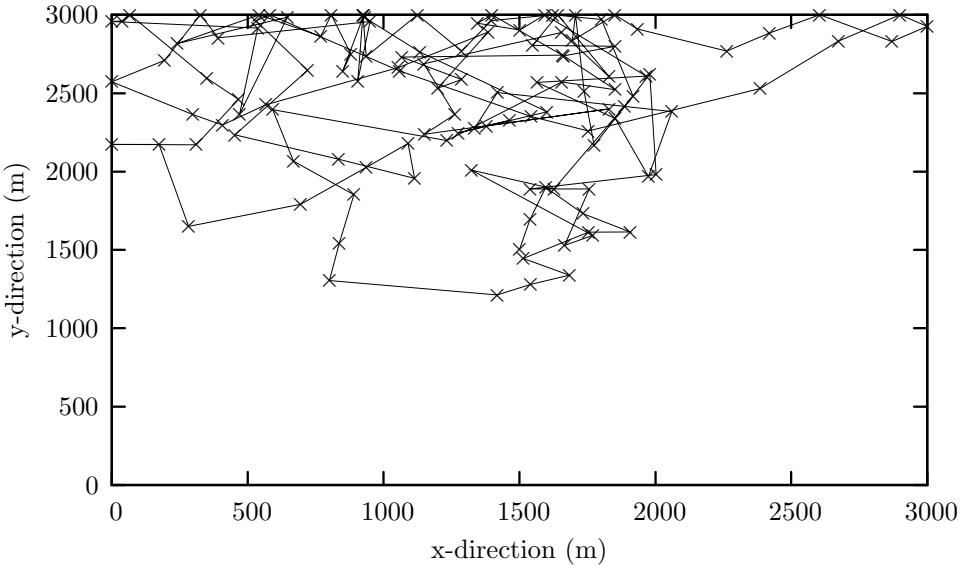


Figure 14: Movement trace of 1 out of 250 nodes in **combat** during 2000 s. Random Walk mobility model, Travel time before changing direction = 20 s, $Speed = 20.0 \pm 15.00\ m/s$, $Pause = 0 \pm 0\ s$.

5.2 Other Configuration Parameters

Apart from the unique mobility behavior of each of the MANET applications described in Section 5.1.3, each application also possess a set of other scenario-specific features. Below are other parameters that need to be carefully chosen for each scenario.

- Area: the size of the spatial area in which nodes are allowed to move (in meters)
- Network Size (NS): the total number of nodes constituting the MANET.
- Service Agent ratio (SR): the fraction of the total nodes that are Service Agents (SAs).
- User Agent ratio (UR): the fraction of the total nodes that are User Agents (UAs).
- Services per Server Node (SP): the number of distinct services that each Service Agent (SA) offers.
- Service Advertisement Lifetime (SL): Service URL advertisement lifetime (in seconds). That is the duration that a service URL advertised in a SrvRply is valid (and cached) at the UA.
- Simultaneous Requests (SQ): fraction of UAs that query services simultaneously.

- Service Duplication ratio (SD): the fraction of SAs that offer identical services.

Table 3 on the following page shows the parameter values that were chosen for each of the applications in BENCHManet. L, M, and H used in the Table signifies low, medium, and high quantities respectively. The numbers below each of the qualitative quantities or ratios shows the actual values used in our simulations. Although the total simulation time is 2000 seconds, the service discovery process only starts after 1000 seconds has elapsed (see Section 6.2 on page 71). Hence, a simulation time of 1000 seconds was assumed when assigning values to service URL advertisement lifetimes.

In the **conf** scenario for instance, the spacial area was 50m×30m, there were a total of 40 nodes in the network (NS=40), 10 of which were Service Agents (SR=10), while another 10 were User Agents (UR=10). Each Service Agent offered one service (SP=1). The service URL lifetimes were 60 seconds (SL=60). Three User Agents always requested services simultaneously (SQ=3). Finally, three of the Service Agents offered identical services (SD=3).

Application	**Area**	**NS**	**SR**	**UR**	**SP**	**SL**	**SQ**	**SD**
(1a) conf	small 50×30	M 40	M 10	M 10	L 1	H 60	L 3	L 3
(1b) event	medium 600×600	M 40	L 10	H 20	L 1	L 10	H 15	L 3
(2a) rescue	large 1500×1500	M 100	H 15	L 5	L 1	M 30	L 0	L 5
(2b) pursuit	large 1500×1500	L 10	M 5	M 5	L 1	L 10	M 3	L 0
(3) pan	medium 300×300	L 20	H 15	L 5	L 1	H 60	L 0	L 2
(4) mesh	large 3000×3000	H 300	H 100	H 100	L 1	L 100	L 10	H 100
(5a) vr	large 1500×1500	M 80	H 38	L 12	L 1	L 10	L 3	H 30
(5b) vp	small 10×3	L 6	H 4	L 2	L 1	M 30	L 0	L 0
(6a) march	large 1500x1500	M 50	M 25	M 25	L 1	M 30	L 5	L 0
(6b) combat	large 3000×3000	H 250	L 25	H 75	L 1	L 10	M 38	L 5

Table 3: Other Configuration Parameters

Chapter 6

Evaluation of SLPManet

6.1 Simulation using NS-2

We used Network Simulator Version 2 (NS-2) developed by the Virtual InterNetwork Testbed (VINT) project at the University of California at Berkley for our simulations. NS-2 is an open-source discrete event simulator targeted at networking research. NS-2 provides substantial support for simulation of Transmission Control Protocol (TCP), routing, and multicast protocols over wired and wireless (local and satellite) networks [32]. Tools for supporting multihop wireless networks with models for physical, data link, and Media Access Control (MAC) layers is provided by the Monarch research group at Carnegie-Mellon University.

NS-2 is an object oriented simulator, written in C++, with an Object Tool Command Language (OTcl) interpreter as a front-end. The simulator supports a class hierarchy in C++ and a similar class hierarchy within the OTcl interpreter. The two hierarchies work in harmony with each other. From the user's perspective, there is a direct correspondence between a class in the OTcl hierarchy and one in the C++ hierarchy. Users create new simulator objects using OTcl scripts. These objects are instantiated within the interpreter, and are

closely mirrored by a corresponding object in the C++ hierarchy. User instantiated objects are mirrored through methods defined in the corresponding C++ class. This split-programming approach and one-to-one mapping between hierarchies allow users to easily configure, setup, and control simulation parameters through OTcl scripts without the need of manipulating and recompiling C++ objects.

In NS-2, each mobile node on a flat ground uses an omni-directional antenna with transmitter and receiver gain equal to 1. The transmitters' and receivers' antennas are 1.5 m high. The wireless interface mimics the commercial 914 MHz Lucent WaveLAN Direct-Sequence Spread Spectrum (DSSS) radio [33]. It is implemented as a shared-media radio with nominal bit rate of 2 Mb/s. The system is assumed to be an ideal one, with system loss equal to 1. The signal propagation model combines both a free space propagation model and a two-ray ground reflection model. When a transmitter is within the reference distance of the receiver, the free space model with a signal attenuation of $1/r^2$ is used. Otherwise, the ground reflection model, where the signal falls off as $1/r^4$, is used. The crossover point is around 86.14 m. The transmitted signal power is around 0.2818 W, while the correct-received signal threshold is around 3.652e-10 W. Therefore, the transmission range is in the range of 250 m.

The MAC layer protocol used is the IEEE 802.11 Distributed Coordination Function (DCF). Each node maintains a network interface priority queue whose job is to queue packets until the MAC layer can transmit them. The queue's

length is 50 packets. Routing packets possess higher priority, and therefore are always inserted at the head of the queue.

In order to implement Service Location Protocol for MANET (SLPManet), it was required to use a MANET routing protocol that supports multicasting or broadcasting as well as unicasting. We used an efficient scalable BroadCAST routing protocol for MANET (BCAST), developed by Kunz [34]. An NS-2 implementation of BCAST is publicly available at [35].

However, BCAST does not support unicasting. SLPManet requires that a Service Agent (SA) unicasts Service Replies (SrvRplys) back to the requesting User Agent (UA). In order to work around this obstacle, we made each UA join a BCAST group of its own. SAs would then send Service Replies (SrvRplys) to the one-node BCAST group of the requesting UA. For instance, a UA residing at node 4 would join a BCAST group of its own, the address of the BCAST group is 0xE000004. When the SA receives Service Requests (SrvRqsts) from node 4, they would broadcast their SrvRplys to the single-node group 0xE000004. The penalty of this setup is the exchange of extra routing messages by the BCAST routing agents. Since routing performance and evaluation is not the goal of this work, this will not affect the performance or validity of our results. It is noteworthy to mention that our SLPManet implementation will work with any MANET routing protocol that supports both multicasting and unicasting. The above work-around will only take effect if BCAST was defined to be the underlying routing protocol (by means of a "BCAST" flag, refer to Appendix A.2).

In order to implement our extended caching modification of SLPManet, we made each UA join every other UA BCAST group (in addition to it's own BCAST group). Hence, any SrvRply that is being unicast back to the querying UA will also be picked up and cached by the intermediate UAs. We did not impose a limit on the size of the node's cache. However, a URL entry is removed from the cache as soon as it times out. Hence, the number of entries cached at a UA or a SA at any one time was relatively small.

SLPManet requires packets to be transported via UDP. Since a UDP agent allocates and sends network packets, all the information needed for application level communication (i.e. needed by the UA and SA of the SLPManet application) should be handed to the UDP agent as a data stream. Unfortunately, the UDP implementation in NS-2 allocates packets that only contains a header stack. Therefore, we had to modify the UDP implementation so that they correctly handle application level data sent to and from the SLPManet application agents. We called the SLPManet-compatible UDP agents UDPSlp.

6.2 Experimental Setup

We determined in Chapter 5 reference configurations for each of the scenarios composing our benchmark. A test for each of the 10 applications was crafted according to the configurations in Table 3 on page 67. NS-2 implementations of the mobility models provided by [29], [30], and [31] were used to generate 10 movement scenarios according to the values in Table 2 on page 59.

SR nodes (refer to Table 3 on page 67) serving as Service Agents (SAs) are picked randomly from a uniform distribution on [0,NS). Similarly, UR nodes serving as User Agents (UAs) are picked randomly from a uniform distribution on [0,NS). However, a node that has already been chosen can not be picked again. This signifies that there can be at most 1 UA or 1 SA per node. The rest of the nodes, $NS - SR - UR$, do not play a role in the service discovery process except relaying packets.

The number of Unique Services (US) available in a given scenario is the number of Service Agent nodes less the number of duplicated services, $US = SR - SD + 1$. Therefore, $US - 1$ Service Agents offer different services, while SD server agents offer identical services.

Research in [29] shows that the initial random distribution of mobile nodes is not representative of the manner in which nodes distribute themselves when moving. There is a high variability in the number of nodes within a mobile node transmission range during the first 600 seconds of simulation. This problem may have severe consequences in some of the mobility models, such as the random waypoint model. In order to solve this problem, we run simulations for 2000 seconds, and discard the initial 1000 seconds of simulation. In fact, the service discovery process does not start until after 1000 seconds of simulation time has elapsed.

There are a total of 1000 requests per simulation. The inter-arrival time for requests is exponentially distributed in the range [1000, 2000) seconds with

a mean at 1360 seconds. Every time a User Agent queries a service, it randomly requests a service from the US unique services available. Each User Agent, in the range [0,UR), issues an approximately equal number of requests (i.e. $1000/UR$ requests each). In every UR requests, SQ requests are issued simultaneously.

SLPManet parameters were set to their default values as described in Section 4.3.6 on page 38. Since it is more reasonable in MANET environments to find one matching service, versus finding as many matching services as possible, the maxSearch parameter is left to its default "false" value. Furthermore, SLPManet timing parameters, specifically, CONFIG_RETRY and CONFIG_MC_MAX, were set to their default values of 2 seconds and 15 seconds respectively. Similarly, the Maximum Transmission Unit (MTU) for SLPManet messages, excluding UDP headers, is kept at its default value of 1400 bytes.

6.3 Performance Metrics

Note that a new "discovery transaction" is said to be initiated whenever a UA requires a service. A "discovery transaction" is said to be "successful" if either the service that the UA requires was already locally cached (i.e. cache-hit), or at least one SrvRply was successfully received by the querying UA in reply to one or more (if re-transmitted) SrvRqst(s).

We define below four metrics we used in our evaluation of SLPManet.

1. Overall discovery success: the percentage of discovery transactions that

were successful by the end of the simulation period. This metric measures the overall performance and throughput of the protocol. The overall discovery success is composed of:

(a) Discovered services: The fraction of the successful discovery transactions that were a result of newly discovered services (i.e. was a result of a successfully received SrvRply to a SrvRqst).

(b) Cached services: The fraction of the successful discovery transactions that were a result of matching services already cached at the querying UA (thus the discovery transaction did not lead to broadcasting SrvRqsts).

2. Service lookup bandwidth of successful discovery transactions: bandwidth consumed by SLPManet messages from the time a service is first requested by a UA until the first matching service information is available at the UA. This metric measures the efficiency of the protocol. We measure:

(a) Peak bandwidth: The maximum of all successful discovery transactions' bandwidth consumption.

(b) Average bandwidth: The mean of all successful discovery transactions' bandwidth consumption.

(c) Standard deviation: Measures the dispersion in service lookup bandwidth of all successful discovery transactions.

3. Aggregate service discovery bandwidth: bandwidth consumed by the whole lookup process. This metric measures the efficiency of the protocol, and demonstrates the degree by which the protocol is conservative of network resources. Aggregate bandwidth is composed of:

   (a) Useful bandwidth: Total bandwidth induced by all successful discovery transactions.

   (b) Wasted bandwidth: SrvRqsts that do not yield SrvRplys, or where the SrvRplys are not received by the querying UA result in wasted bandwidth.

4. Service lookup latency of successful discovery transactions: time elapsed from the time a service is first needed by a UA and the first matching service information is available at the UA. This metric measures the responsiveness of the protocol. We measure:

   (a) Peak latency: The maximum of all successful discovery transactions' lookup latencies.

   (b) Average bandwidth: The mean of all successful discovery transactions' lookup latencies.

   (c) Standard deviation: Measures the dispersion in service lookup latencies of all successful discovery transactions.

6.4 Evaluation of SLPManet

Since the performance of SLPManet is likely to be sensitive to movement patterns, we ran BENCHManet with different sets of mobility files (i.e. ten movement files per benchmark run) until the standard deviation of the samples became fairly small for the majority of the benchmark tests. The number of necessary samples was thus determined to be 10.

In the subsequent sections, all metrics and quantities resemble the mean of the 10 samples. We also calculate the 95% confidence interval for each of these metrics, and illustrate the confidence intervals by means of vertical y-bars on the bar-charts.

6.4.1 Overall Service Discovery Success

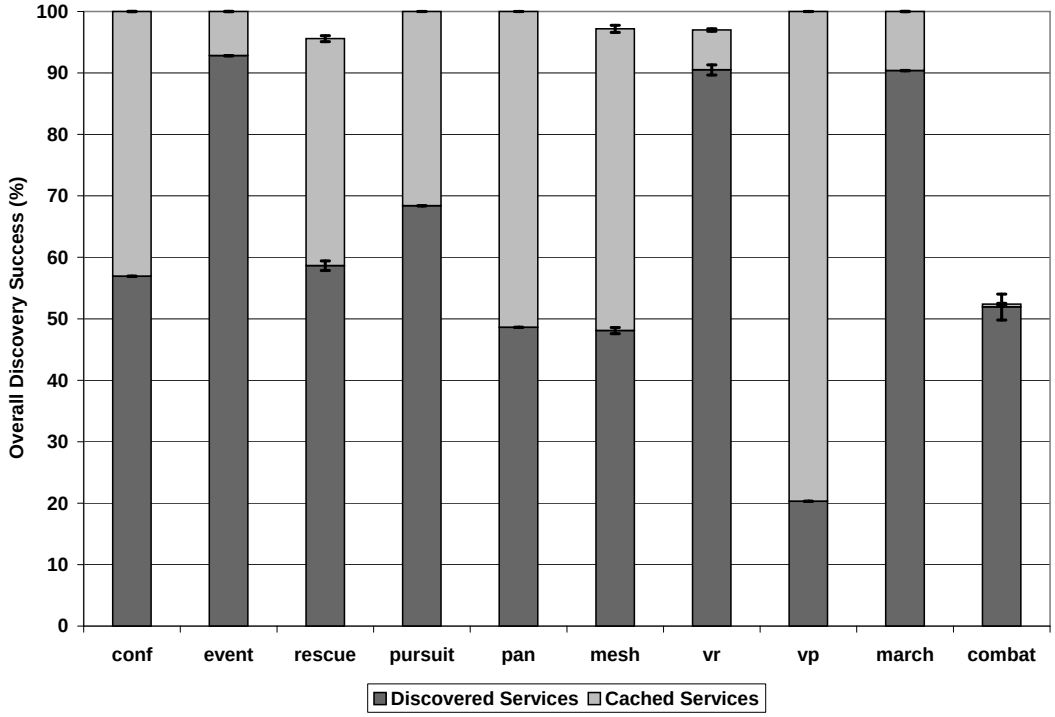


Figure 15: SLPManet overall service discovery success

Figure 15 shows the overall discovery success percentage of SLPManet across the 10 different scenarios in BENCHManet. The bar-chart also shows the proportions that cached services and newly discovered services contributed toward the overall success.

It is notable that 9 out of the 10 benchmark scenarios achieve an overall discovery success greater than 95%. 6 of these 9 scenarios achieve a discovery success of 100%, meaning that every time a service was needed by a UA, either a SrvRply was successfully received, or matching service information was already cached at the UA.

The **combat** scenario performs very poorly, with overall discovery success of barely above 50%. Nodes traveling with high speeds without pausing makes **combat** the scenario with the highest degree of mobility (refer to Tables 1 and 2 on page 59). Furthermore, the simulation area in **combat** is the largest; nodes travel using the random walk mobility model (refer to Figure 14 on page 64). This causes the network topology to be continuously changing, resulting in volatile connections. Although **pursuit** possess comparable speeds and pauses, nodes in **pursuit** are modeled using the pursuit mobility model (refer to Figure 8 on page 61). Hence, the pursuit network is much more strongly connected than in **combat**.

Note that in the **vr** scenario, most of the services were newly discovered. Cache hits did not contribute much to the overall discovery success. The **vp** scenario on the contrary witnessed the opposite trend; most of the discovery success was due to cached service information. This phenomenon is due to several factors, two of which are focal. Firstly, the simulation area and amount of node mobility in **vr** is much greater than that of **vp**. Therefore, unlike **vp**, most nodes in **vr** are not within one-hop reach of each other, resulting in less SrvRplys received and cached at UAs. Moreover, the service URL advertisement lifetime in **vr** is shorter than that of **vp**, therefore, service information cached at UAs in **vr** is more likely to expire before they are needed again.

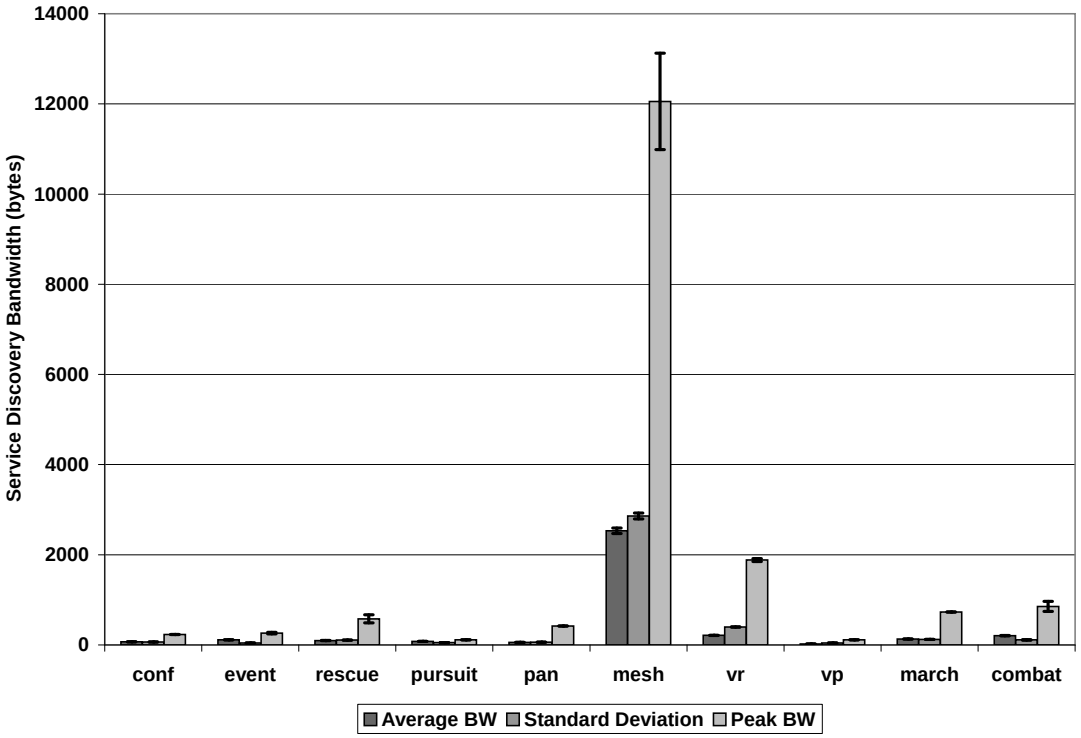


Figure 16: SLPManet bandwidth of successful discovery transactions

6.4.2 Discovery Bandwidth of Successful Discovery Transactions

Figure 16 shows the bandwidth consumed by SLPManet messages of successful discovery transactions for each of the 10 scenarios. If a UA requests a service that was already cached, the bandwidth consumed for this discovery transaction is assumed to be 0. A user request that was never satisfied by the end of the simulation period is not taken into account in this metric. This metric captures the efficiency of SLPManet given that there exists a reachable service that satisfies the UA's request.

It is observed that, except in the **mesh** scenario, the average bandwidth is inversely proportional to the fraction of cached services (refer to Figure 15 on

page 77). This is due to the fact that a cached hit induces no network bandwidth, thus lowering the average bandwidth consumed per successful discovery transaction.

The **mesh** scenario does not follow the trend described above. The peak bandwidth in **mesh** is much greater than in other scenarios. Note from Table 3 on page 67 that the number of Service Agents (SAs) in mesh is evidently higher than the other scenarios (SR=100). Moreover, all 100 SAs are duplicated, thus offering identical services. Hence, when a User Agent requests a service, all 100 SA nodes have a matching service (i.e. a maximum of 100 SrvRplys per SrvRqst). A detailed analysis of the trace files generated reveals that some SrvRqsts were retransmitted three times, each time causing a high fraction of the 100 SAs to send SrvRplys. It is only after the third SrvRqst is retransmitted that the first SrvRply caused by the first SrvRqst is received by the UA. Thus, the total number of SrvRplys that were generated reached close to $3 \times SR$ messages. This phenomenon was particularly eminent in **mesh** since the huge spatial area and large number of nodes amplified the transmission and queuing delays. Consequently, the initial CONFIG_RETRY of 2 seconds (refer to Section 4.3.6 on page 38) was insufficient for sent SrvRplys to be received by the UA before a retransmission is triggered.

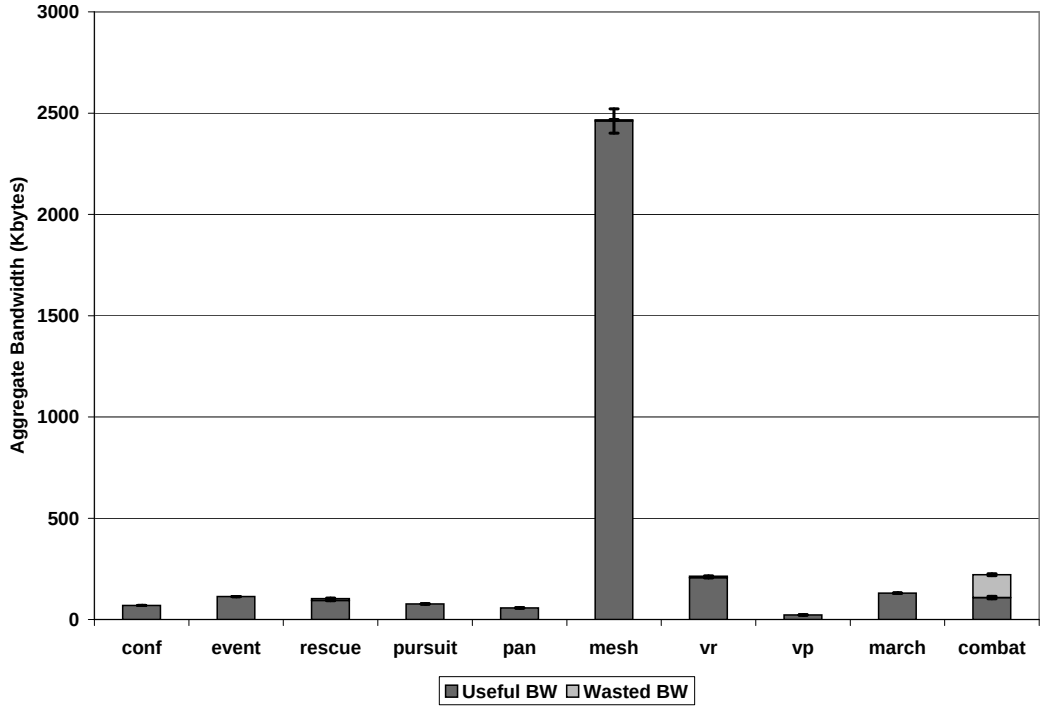


Figure 17: SLPManet aggregate bandwidth consumption

6.4.3 Aggregate Bandwidth Consumption

Figure 17 shows the aggregate bandwidth consumed by service discovery messages during the entire simulation for each of the ten scenarios. The bar-chart also shows the fraction of the aggregate bandwidth that was wasted. Wasted bandwidth is the result of service discovery messages sent, without at least one SrvRply successfully reaching the querying UA. Wasted bandwidth is thus inversely proportional to the overall service discovery success depicted in Figure 15 on page 77. Evidently, the failure percentage (100%−Overall Discovery Success%) of **combat** was around 50%, hence, 50% of the aggregate bandwidth consumed by **combat** was wasted.

The aggregate bandwidth consumed in **mesh** is notably higher for the same

reason described previously in the analysis of the **mesh** case on page 80.

6.4.4 Service Lookup Latency of Successful Discovery Transactions

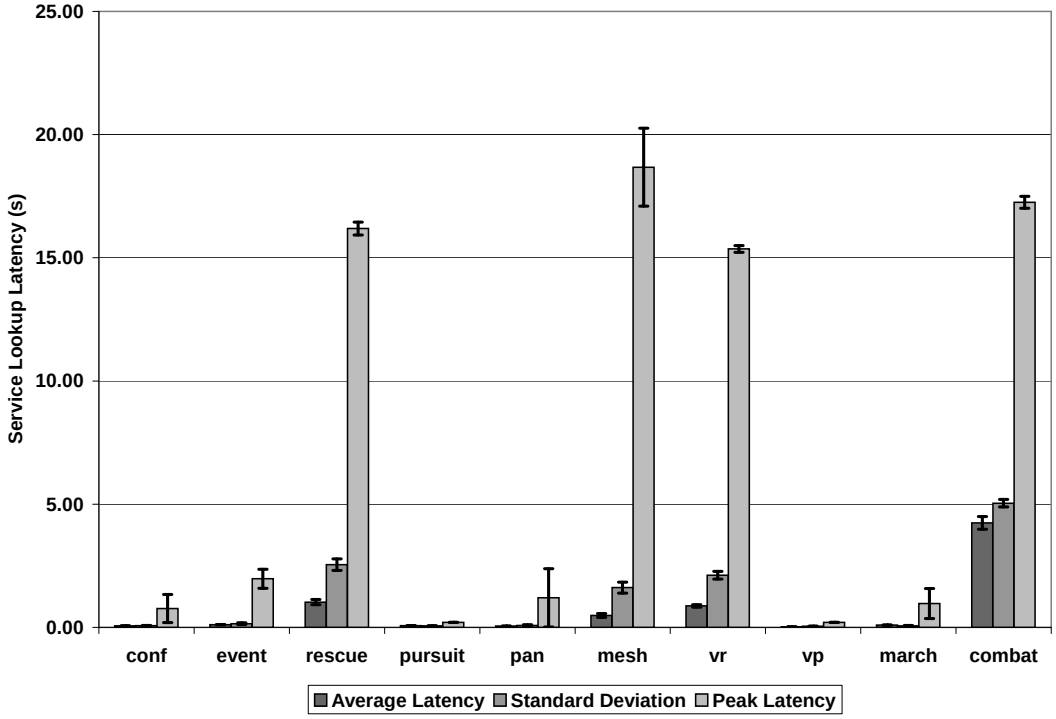


Figure 18: SLPManet lookup latency of successful discovery transactions

Figure 18 shows the SLPManet service lookup latency of successful discovery transactions for each of the 10 BENCHManet scenarios. If a UA requests a service that was already cached, the latency for this discovery transaction is assumed to be 0. Again, a user request that was never satisfied by the end of the simulation period is not taken into account in this metric. This metric captures the efficiency of the lookup process in SLPManet, given that there exists a reachable service that satisfies the UA's request.

The peak latency captures the longest successful discovery process. Recall from the specification of SLPManet in Section 4.3.6 on page 38 that SrvRqsts will keep being retransmitted within a bounded time interval (CONFIG_MC_MAX) until a reply is received by the issuing UA. Also note that CONFIG_MC_MAX is set to its default of 15 seconds, explaining why the longest service discovery latency is not much greater than 15 seconds.

The general trend observed is that the further apart nodes are during the simulation, the longer it takes to lookup for a service. Distant nodes induce more retransmissions and cause longer transmission delays, thus prolonging the peak latency. Several other factors affect the average latency, most prominent of which is the amount of cache hits induced by a scenario. This explains why **rescue**, **mesh**, **vr**, and **combat** scenarios suffer from longer discovery latencies.

6.4.5 Summary

In general, SLPManet performance was acceptable for most scenarios in BENCHManet. Nonetheless, some applications performed poorly in terms of the various metrics. Table 4 on the following page summarizes the weaknesses found in our evaluation of SLPManet.

Metric	**Scenarios**	**Reason**
Discovery Success	**combat**	Highly mobile, weakly connected topology
Average Bandwidth	**mesh**	Huge spatial area, many SrvRplys, longer delays, retransmissions triggered before initial SrvRplys received
Peak Bandwidth	**mesh**	
Aggregate Bandwidth	**mesh**	
Peak Latency	**mesh** **rescue** **vr** **combat**	Distant nodes, longer transmission delays, lost messages trigger retransmissions

Table 4: SLPManet weaknesses using BENCHManet

6.5 Evaluation of Modified SLPManet

We ran the same 10 experiments (same sets of movement files and identical initial seeds) that were previously used in evaluating SLPManet in order to evaluate SLPManet with our extended caching modification.

In subsequent sections, all metrics and quantities resemble the mean of the 10 samples. We also calculate the 95% confidence interval for each of these metrics, and illustrate the confidence intervals by means of vertical y-bars on the bar-charts.

Furthermore, we calculate the percentage improvement in each of the metrics. We illustrate the improvement percentages also using bar-charts. We used the mean values of the 10 sample runs in calculating the improvement percentages. We assume that the control group (baseline) is the original SLPManet,

while the experimental group is the modified SLPManet.

In order to verify the validity and statistical significance of the improvement, we carried out two statistical tests on each pair of metrics. The first was a two-sample one-tailed t-test for two means (original and modified SLPManet metric means). The second test we the 95% confidence interval of the difference between the two metrics' means.

Two-sample one-tailed t-test:

For the two-sample one-tailed t-test, the T-statistic was calculated using:

$$T = \frac{\bar{M_O} - \bar{M_M}}{\sqrt{s_O^2/N_O + s_M^2/N_M}} \tag{1}$$

where $\bar{M_O}$ and $\bar{M_M}$ are the sample means of the metrics in the original and modified SLPManet respectively. Similarly, s_O^2 and s_M^2 are the sample variances and N_O and N_M are the samples sizes. The hypotheses for the two-sample one-tailed t-test are:

$$H_0 = \mu_O \geq \mu_M \qquad H_1 = \mu_O < \mu_M \tag{2}$$

in which case we reject H_0 if $T < -t_{(\alpha,v)}$. $t_{(\alpha,v)}$ is the (one-tailed) critical value of the t-distribution with significance level α, and v degrees of freedom, where $v = N_O + N_M - 2$. We used the one-tailed t-test in Equation 2 when we expect that the population mean of the original SLPManet (μ_O) to be greater than that of the modified SLPManet (μ_M). This is the case for the bandwidth, aggregate bandwidth and latency metrics (i.e. Metrics 2, 3, and 4). On the contrary, in the overall service discovery success (Metric 1), we expect the population mean

of the overall success to be greater in the modified SLPManet. Therefore, we used the two-sample one-tailed t-test whose hypotheses are in Equation 3:

$$H_0 = \mu_O \leq \mu_M \qquad H_1 = \mu_O > \mu_M \tag{3}$$

in which case we reject H_0 if $T > t_{(\alpha,v)}$.

For our two-sample one-tailed t-tests, we used a significance level of 0.05 and 18 degrees of freedom ($N_O = N_M = 10$), thus $t_{(\alpha,v)}$ was 1.734. Acceptance of the hypothesis, H_0 signifies that the population mean of the modified SLPManet is not any "worse" than that of the original SLPManet, otherwise the hypothesis H_0 is rejected.

Confidence interval for the difference of two means:

For the second statistical test, we find the 95% confidence interval of the difference between two means, M_d, where $M_d = \bar{M}_M - \bar{M}_O$ for the overall service discovery success metric (Metric 1), and $M_d = \bar{M}_O - \bar{M}_M$ for the rest of the metrics (Metrics 2, 3, and 4). The confidence intervals upper (UL) and lower (LL) limits are:

$$UL = M_d + t_{(\alpha,v)}\, s_{M_d} \qquad LL = M_d - t_{(\alpha,v)}\, s_{M_d} \tag{4}$$

where s_{M_d} is the standard error of the difference between means. Since the variances in the two populations are assumed to be the same, and so is the number of observation ($N = N_O = N_M$), we can calculate s_{M_d} using:

$$s_{M_d} = \sqrt{2MSE/N} \qquad MSE = \frac{s_O^2 + s_M^2}{2} \tag{5}$$

$t_{(\alpha,v)}$ is the (two-tailed) critical value of the t-distribution with significance level α, and v degrees of freedom. Again, s_O^2 and s_M^2 are the sample variances in the original and modified SLPManet. In the event that the confidence interval computed includes 0, we can say that there is no statistically significant difference between the means of the two populations at the given level of confidence. The level of confidence we used in our tests was 95%.

6.5.1 Overall Service Discovery Success

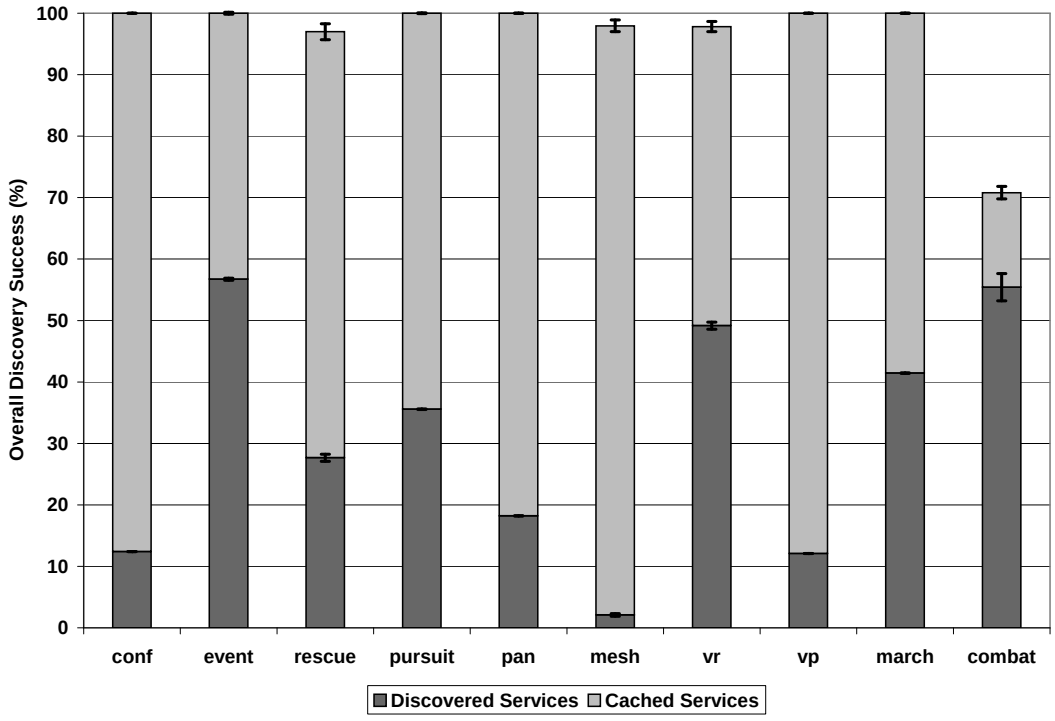


Figure 19: Modified SLPManet overall service discovery success

Figure 19 shows the overall discovery success of the modified SLPManet. Figure 20 on the following page shows the improvement in the overall discovery success.

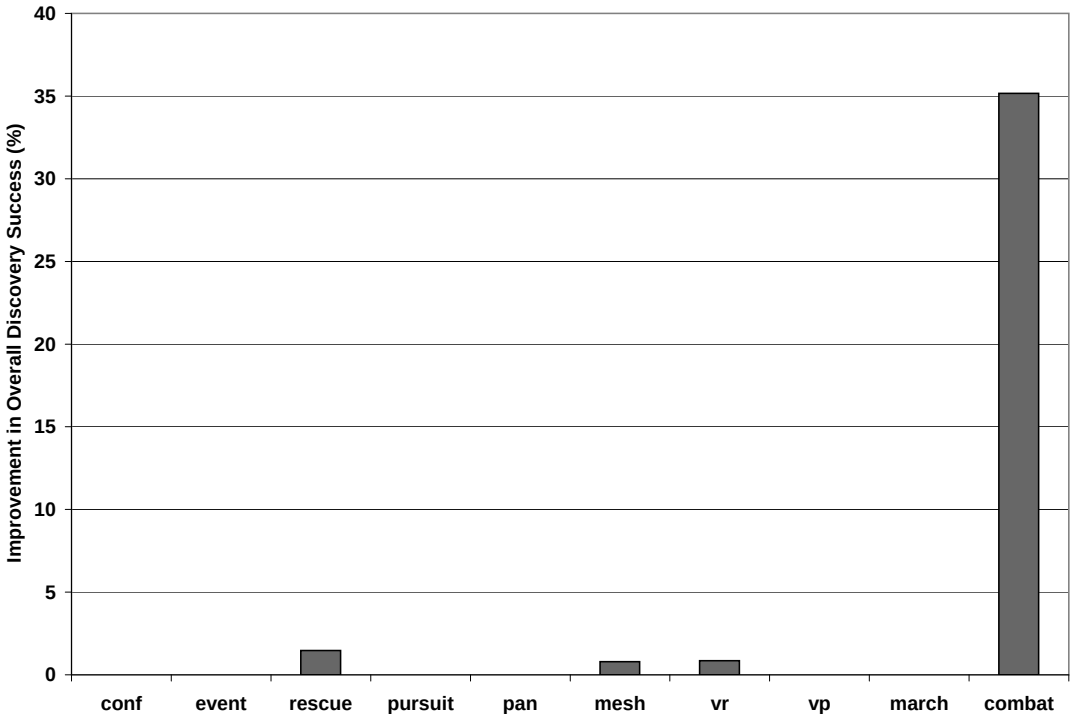


Figure 20: Improvement in SLPManet overall service discovery success

Application	$H_0 = \mu_O \leq \mu_M$	M_d
conf	accept	not significant
event	accept	not significant
rescue	accept	not significant
pursuit	accept	not significant
pan	accept	not significant
mesh	accept	not significant
vr	accept	not significant
vp	accept	not significant
march	accept	not significant
combat	accept	significant

Table 5: t-test and significance for overall discovery success

We observe that discovery success was maintained at 100% for 6 of the scenarios. Discovery success in **rescue**, **mesh**, and **vp** increased by around 1%, making them approach 100%. More interestingly, **combat**, the application with the lowest overall discovery success in the original SLPManet improved by more than 35%. We also note that compared to the original SLPManet, the fraction of services that were cached have increased significantly in all 10 scenarios.

Table 5 on the previous page shows that the hypothesis, $H_0 = \mu_O \leq \mu_M$, was never rejected for any of the scenarios. This signifies that the population mean for the overall discovery success in the modified SLPManet is at least equal, if not greater than that of the original SLPManet. The table also verifies that the improvement in the overall service discovery metric is statistically significant for the **combat** scenario.

6.5.2 Discovery Bandwidth of Successful Discovery Transactions

Figure 21 on the following page shows the bandwidth of successful discovery transactions in the modified SLPManet for each of the 10 scenarios. Recall that if a UA requests a service that was already cached, then the bandwidth consumed for this discovery transaction is assumed to be 0 bytes. A discovery transaction that was not satisfied by the end of the simulation period is not taken into account in this metric. This metric captures the efficiency of the modified SLPManet given that there exists a reachable service that satisfies the UA's request.

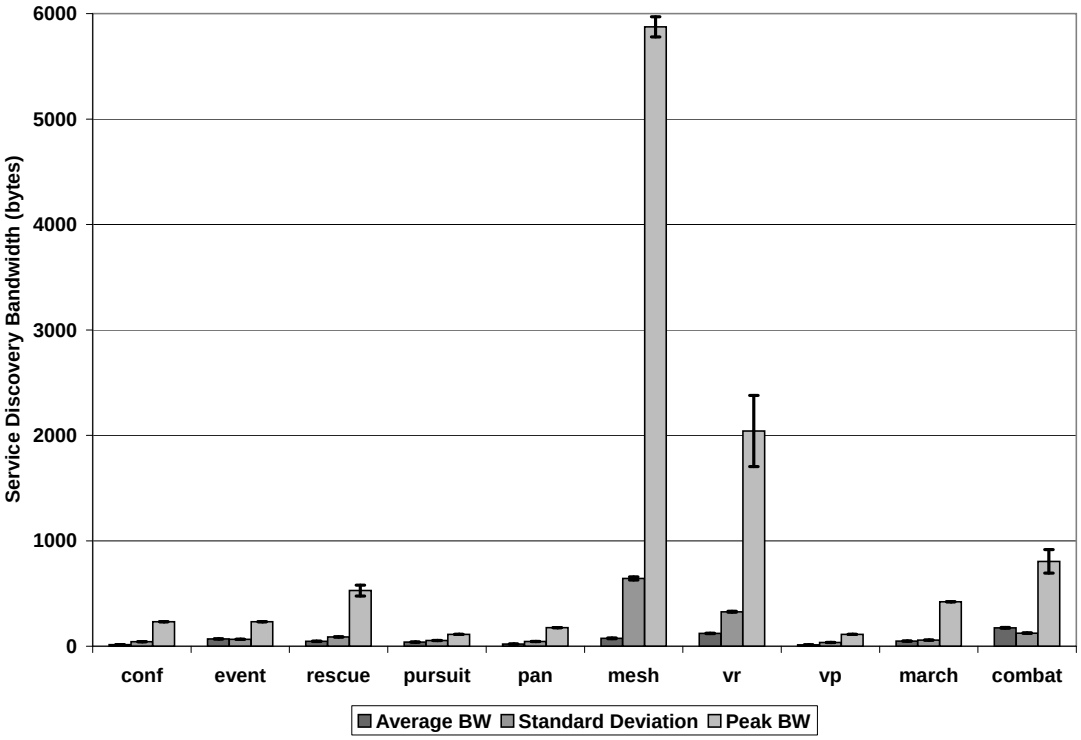


Figure 21: Modified SLPManet bandwidth of successful discovery transactions

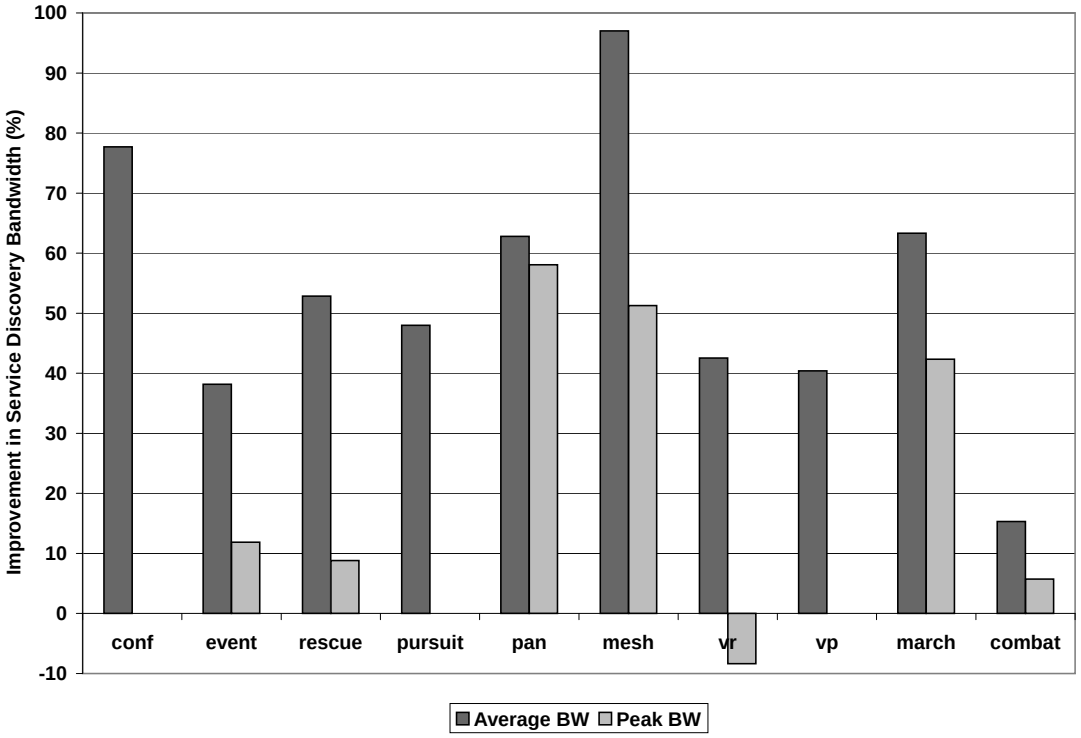


Figure 22: Improvement in SLPManet bandwidth of successful discovery transactions

Figure 22 on the previous page shows the improvement in the average and peak bandwidth of successful discovery transactions. We observe that the average bandwidth has improved significantly for all 10 scenarios due to increased cache hits. The improvement varied between 15% for **combat** and 97% for **mesh** (the scenario with the highest bandwidth consumption in the original SLPManet).

We also note that the peak bandwidth of successful discovery transactions improved (except for **vr**) by up to 58% (in **mesh**). However, the peak bandwidth of successful discovery transaction has deteriorated by almost 8% on average in **vr**. An inspection of the peak bandwidth in the individual **vr** sample runs shows that only 1 of the 10 samples led to an average peak bandwidth higher in the modified SLPManet than in the original SLPManet. If we assumed that this sample was an outlier, then the 9-sample mean of the peak bandwidth in **vr** is 1869 bytes (i.e. 14 bytes less than in the original SLPManet).

A detailed investigation of the troublesome **vr** sample reveals that the peak bandwidth was a result of a UA request for the specific service type that was duplicated in 30 SAs. Shortly after the SrvRqst was sent, the querying UA traveled to a new location. However, at the time the SrvRqst was transmitted, all 30 matching SAs were within reach, hence, 30 SrvRplys were transmitted but not successfully received by the querying UA (since it changed location). Consequently, the UA retransmits the SrvRqst, resulting in 30 new SrvRplys (which were successfully received by the querying UA the second time). Note

that compared to the original SLPManet, the timing of the generated SrvRqsts is different (even though the same movement patterns and seeds were used), due to extended caching. This explains why this particular coincidence was not observed in any of the 10 sample runs in the original SLPManet evaluation. Therefore, the deterioration in one of the samples was a coincidence of several factors, none of which is a shortcoming of the proposed caching modification.

Application	$H_0 = \mu_O \geq \mu_M$	M_d
conf	accept	significant
event	accept	significant
rescue	accept	significant
pursuit	accept	significant
pan	accept	significant
mesh	accept	significant
vr	accept	significant
vp	accept	significant
march	accept	significant
combat	accept	significant

Table 6: t-test and significance for average bandwidth

Table 6 shows the validity of the improvement in the average bandwidth, as the hypothesis, $H_0 = \mu_O \geq \mu_M$, was accepted, and the improvement was statistically significant for all 10 scenarios. Table 7 on the next page also shows that the improvement in peak bandwidth was statistically significant in four of

Application	$H_0 = \mu_O \geq \mu_M$	M_d
conf	accept	not significant
event	accept	significant
rescue	accept	not significant
pursuit	accept	not significant
pan	accept	significant
mesh	accept	significant
vr	accept	not significant
vp	accept	not significant
march	accept	significant
combat	accept	not significant

Table 7: t-test and significance for peak bandwidth

the ten scenarios, including **mesh** (the scenario with the worst peak bandwidth in the original SLPManet). Note that Table 7 also shows that H_0 was not rejected for the **vr** scenario, signifying that although the sample mean of the peak bandwidth in **vr** has slightly deteriorated on average after caching, this will probably not be the case in the population mean (μ_M).

6.5.3 Aggregate Bandwidth Consumption

Figure 23 on the next page shows the aggregate bandwidth consumed by service discovery messages in the modified SLPManet during the entire simulation for each of the ten scenarios of the benchmark. The bar-chart also shows the fraction of the aggregate bandwidth that was wasted. Wasted bandwidth is the result of SrvRqsts messages sent, without at least one SrvRply successfully

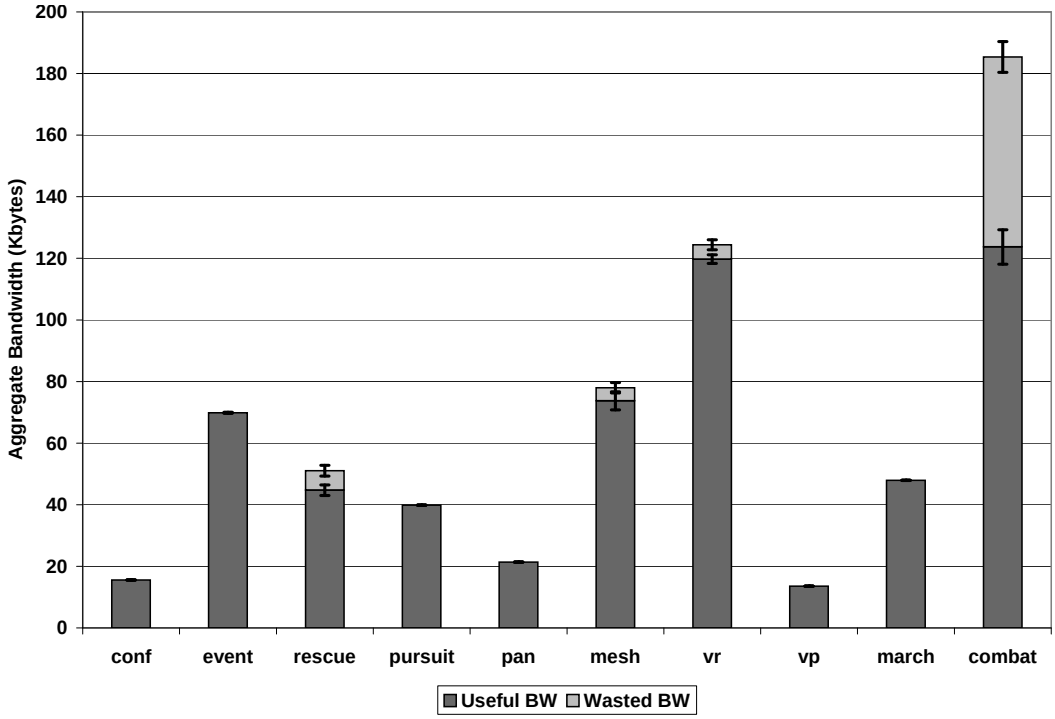


Figure 23: Modified SLPManet aggregate bandwidth consumption

reaching the querying UA. This fraction of wasted bandwidth is inversely proportional to the overall service discovery success shown in Figure 19 on page 87.

Figure 24 on the next page shows the improvement in the aggregate bandwidth consumed, that is the percentage savings in aggregate bandwidth consumption. We note that the aggregate bandwidth consumption has significantly improved for all 10 scenarios. The improvement varied between 16% in **combat**, and 96% in **mesh** (the scenario with the highest aggregate bandwidth consumption in the original SLPManet).

Table 8 on the following page shows the validity of the improvement in aggregate bandwidth savings, as the hypothesis, $H_0 = \mu_O \geq \mu_M$, was accepted, and the improvement was statistically significant for the entire benchmark.

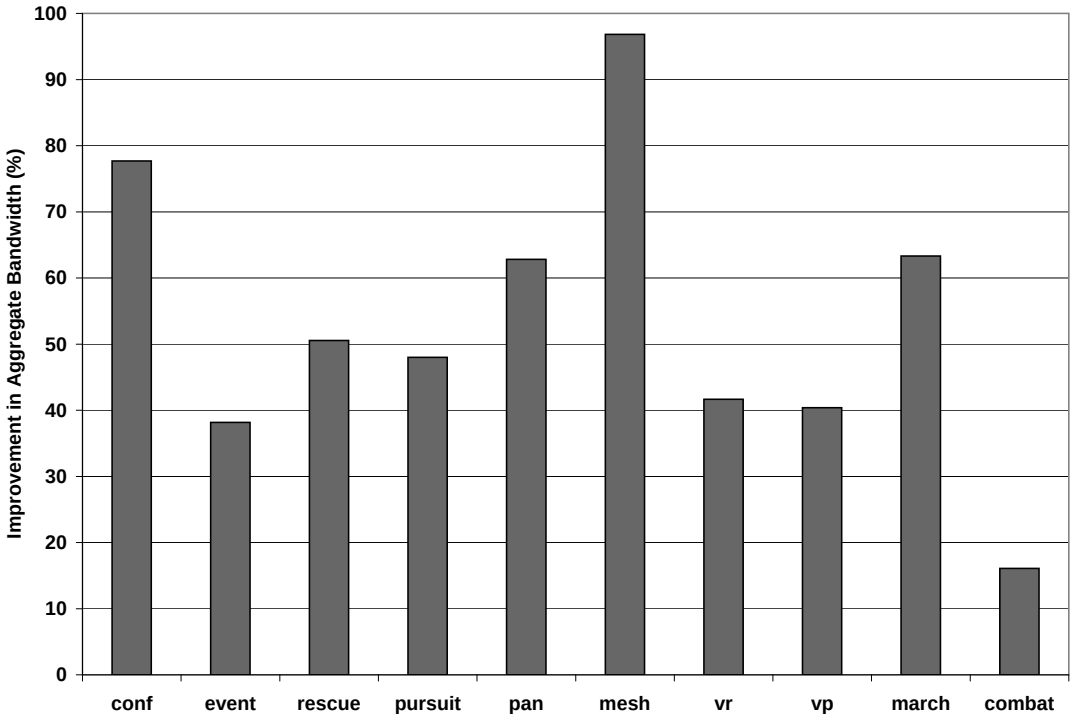


Figure 24: Improvement in SLPManet aggregate bandwidth consumption

Application	$H_0 = \mu_O \geq \mu_M$	M_d
conf	accept	significant
event	accept	significant
rescue	accept	significant
pursuit	accept	significant
pan	accept	significant
mesh	accept	significant
vr	accept	significant
vp	accept	significant
march	accept	significant
combat	accept	significant

Table 8: t-test and significance for aggregate bandwidth

6.5.4 Service Lookup Latency of Successful Discovery Transactions

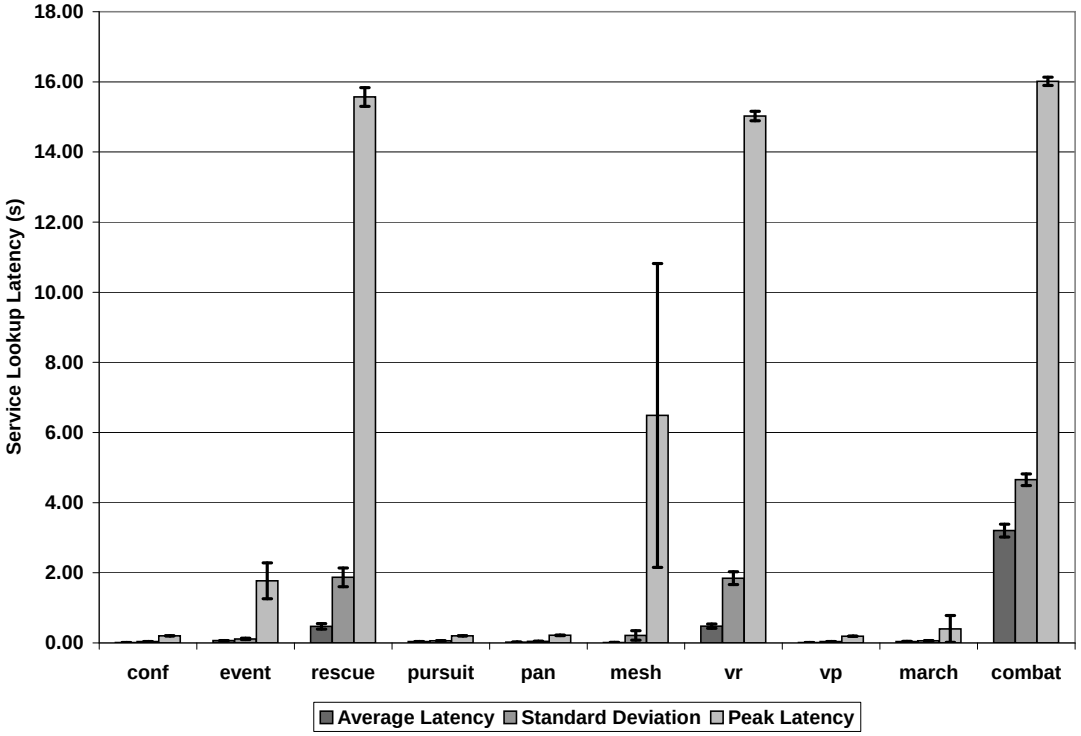


Figure 25: Modified SLPManet lookup latency of successful discovery transactions

Figure 25 shows the service lookup latency of successful discovery transactions in the modified SLPManet for each of the 10 scenarios. If a UA requests a service that was already cached, then the latency for this discovery transaction is assumed to be 0. Again, a discovery transaction that was never satisfied by the end of the simulation period is not taken into account in this metric. This metric captures the efficiency of the lookup process in the modified SLPManet given that there exists a reachable service that satisfies the UA's request.

Figure 26 on the following page shows the improvement in the service lookup

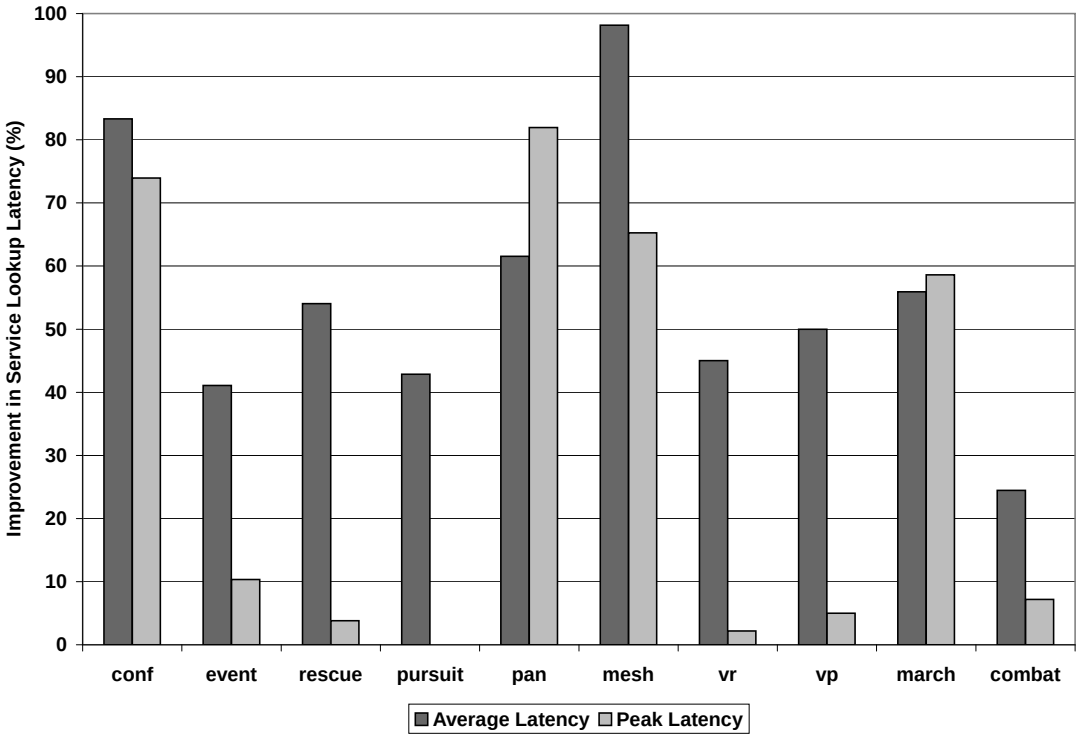


Figure 26: Improvement in SLPManet lookup latency of successful discovery transactions

Application	$H_0 = \mu_O \geq \mu_M$	M_d
conf	accept	significant
event	accept	significant
rescue	accept	significant
pursuit	accept	significant
pan	accept	significant
mesh	accept	significant
vr	accept	significant
vp	accept	significant
march	accept	significant
combat	accept	significant

Table 9: t-test and significance for average latency

Application	$H_0 = \mu_O \geq \mu_M$	M_d
conf	accept	not significant
event	accept	not significant
rescue	accept	significant
pursuit	accept	not significant
pan	accept	not significant
mesh	accept	significant
vr	accept	significant
vp	accept	significant
march	accept	not significant
combat	accept	significant

Table 10: t-test and significance for peak latency

latency. We note that the average latency of successful discovery transactions has greatly improved for all 10 scenarios due to increased cache-hits. The improvement varied between 24% for **combat** and 98% for **mesh**. The peak latency has similarly improved for most scenarios. The improvement reached up to 82%.

Table 9 on the previous page shows the validity of the improvement in the average latency, as H_0 was accepted, and the improvement was statistically significant for all 10 scenarios. Table 10 also shows that the improvement in peak latency was statistically significant in half of the scenarios, including scenarios whose peak latency was notably high in the original SLPManet (i.e. **rescue**, **mesh**, **vr**, and **combat**).

6.5.5 Summary

In summary, the caching modification has improved the performance of the SLPManet protocol under BENCHManet tests.

Although the overall service discovery success of **combat** has improved by 35%, it is only achieving 70%. Whether this is satisfactory or not, is indeed dependent on the type of application. In a **combat** scenario, with a high degree of mobility and topology complexity, it might be reasonable not to expect a much better performance.

The caching modification has dramatically improved the performance in terms of the bandwidth and latency metrics for almost every scenario. The only exception was an 8% deterioration in the sample mean of the peak bandwidth metric in **vr**. However, a detailed analysis revealed that this was due to a rare timing coincidence that occurred in only 1 of the 10 sample runs, and was not a shortcoming of the caching modification. Hence, the statistical tests showed that the deterioration was not of any statistical significance.

6.6 Comparing SLPManet and modified SLPManet

Original SLPManet:

- Original SLPManet achieved an overall service discovery success of at least 95% for all scenarios except **combat** which only achieved 50%. This is due to **combat**'s high mobility and troublesome network topology.

- The average, peak, and aggregate bandwidth consumption in the original

SLPManet was particularly high for **mesh**. This was due to its large number of duplicated services, and large number of nodes which increased the delays, hence triggering several retransmissions of a large number of SrvRplys.

- The peak latency in the original SLPManet was particularly higher for **rescue**, **mesh**, **vr**, and **combat** since nodes in these topologies are generally further apart, resulting in several retransmissions before a SrvRply is successfully received by the querying UA.

Modified SLPManet:

- The overall service discovery success for **combat** increased to 70%, hence achieving a 35% improvement. The overall success in the rest of the scenarios was either maintained at 100% or improved by around 1%. The fraction that cache hits contributed to the overall service discovery success has increased noticeably for all scenarios in BENCHManet compared to the original SLPManet.

- The average bandwidth has decreased for all scenarios in the benchmark. The improvement varied between 15% and 97%, with the highest improvement in the **mesh** scenario. The peak bandwidth has similarly improved (except in **vr**) by up to 58%, with the highest improvement also observed in **mesh**.

- The aggregate bandwidth improved for all 10 scenarios. The improvement varied between 16% and 96%, with the highest improvement observed in **mesh**.

- The average latency has similarly improved for all 10 scenarios. The improvement varied between 24% and 98%. The peak latency has also improved by up to 82%.

Improvement Accuracy:

The two-sample t-test ($\alpha = 0.05$) did not reject any of the hypothesis, H_0, signifying that the population means of the metrics in the modified SLPManet is not any "worse" than in the original SLPManet. This includes the hypothesis, $H_0 = \mu_O \geq \mu_M$, of the peak bandwidth in **vr**.

The 95% confidence level for the difference of two means test revealed that:

- The improvement in average bandwidth, aggregate bandwidth, and average latency for all scenarios in the BENCHManet to be statistically significant.

- The improvement in overall discovery success for the **mesh** scenario to be statistically significant.

- The improvement in the peak bandwidth of 4 scenarios (including **mesh**) to be statistically significant.

- The improvement in the peak latency of 5 scenarios (including **rescue**, **mesh**, **vr**, and **combat**) to be statistically significant.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, a service discovery protocol, Service Location Protocol (SLP) Version 2 [2], has been adapted to suit Mobile Ad Hoc Networks (MANETs). The adapted protocol, Service Location Protocol for MANET (SLPManet), was implemented using Network Simulator Version 2 (NS-2). Various existing and potential applications of MANETs were carefully examined in order to determine unique mobility and networking characteristics for each different class of applications. A benchmark, Benchmark for MANET (BENCHManet), consisting of a suit of tests, each of which is based on the determined reference configurations of the various representative MANET applications, was developed. Hence, BENCHManet facilitated a thorough performance evaluation of SLPManet.

The performance evaluation of SLPManet shows that the overall success percentage for most of the tests in BENCHManet exceeded 95%. However, scenarios with a high degree of mobility, and fast changing network topologies, such as **combat**, performed poorly in terms of overall discovery success. Furthermore, the performance evaluation reveals that SLPManet can be very inefficient in terms of bandwidth consumption for service discovery scenarios

that are characterized by a high ratio of duplicated services. Finally, an evaluation of the service discovery lookup latency shows that the time SLPManet needs to successfully locate the desired service tends to increase unsatisfactorily with scenarios whose nodes are usually distanced further apart.

We suggested a simple caching modification to SLPManet where intermediate User Agents (UAs) cache service information found in Service Replies (SrvRplys) which they relay. The modification increases the performance and efficiency of SLPManet by reducing the flooding and exchange of service discovery messages. The modification conforms to the general specification of the SLP protocol; it neither requires additional generation or exchange of messages, nor an integration of the network layers.

The performance of the modified SLPManet was also evaluated using the same BENCHManet parameters. The overall service discovery improved by 35% for the scenario that performed poorly in the original SLPManet evaluation (i.e. **combat**). The average bandwidth has decreased significantly for all scenarios; the improvement varied between 15% and 97%. The peak bandwidth consumption improved by up to 58% (except in **vr**). The peak bandwidth in **vr** deteriorated by 8% due to a rare timing coincidence that occurred in 1 of the 10 sample runs. The aggregate bandwidth improved significantly for all scenarios; the improvement varied between 16% and 98%. Moreover, the average latency improved significantly for all scenarios; the improvement varied between 24% and 98%. Similarly, the peak latency improved by up to 82%. Expectedly,

scenarios that performed the worst in terms of bandwidth and latency in the original SLPManet improved the most with the caching modification.

Finally, we carried out two statistical tests, the two-sample one-tailed t-test and the 95% confidence level for the difference between two means test. Both tests verified the legitimacy and statistical significance of the improvements witnessed using the modified SLPManet. The deterioration in the sample mean of the peak bandwidth in the **vr** scenario was deemed statistically insignificant.

7.2 Summary of Contributions

This thesis provided the following contributions:

1. Adapted SLP Version 2 to MANET environments.

2. Implemented the adapted protocol, SLPManet, in NS-2.

3. Classified existing MANET applications and determined reference configurations associated with each unique application class. The determined reference configurations could be easily adopted in evaluating any MANET application-level protocol.

4. Provided a benchmark, BENCHManet, consisting of a test for each of the reference applications. The benchmark allows a direct practical and comprehensive evaluation of any MANET Service Discovery Protocol (SDP).

5. Evaluated the performance of SLPManet using BENCHManet.

6. Proposed a simple extended caching modification to SLPManet and implemented the modification in NS-2.

7. Evaluated and compared the performance of the improved SLPManet using BENCHManet.

7.3 Future Research

Several avenues of future work exist, most prevailing of which may include:

1. Developing a purely passive service discovery version of SLPManet and comparing its performance to the current purely active SLPManet (note that the original SLP specification [2] does not permit passive discovery without DAs). Moreover, the current active SLPManet can be modified so that every Service Agent (SA) floods a service announcement when joining the network, or in a periodic fashion, thus, leading to a protocol that combines the benefits of both discovery models. A detailed comparison of the three discovery models (active, passive, and combined) using the developed benchmark, BENCHManet, will lead to constructive conclusions on which model best suits which MANET application. Our caching modification to SLPManet will be particularly useful with models of SLPManet that provides some sort of passive discovery.

2. The developed benchmark, BENCHManet, calls for a quantitative performance comparison of SLPManet against other Service Discovery Protocols

such as UPnP [5], Bluetooth SDP [6], as well as other proposed protocols such as DEAPspace [7].

3. A comparison between application-level service discovery approaches such as SLPManet and cross-layer service discovery proposals such as in [12], [13], and [36] is important. Typically, integration of service discovery with routing will result in better performance due to reduced infrastructure and protocol overhead. However, the close binding between the layers results in giving up the many benefits associated with modularity. If the performance penalty of application-level service discovery is not too severe, adhering to the principles of a layered protocol stack and its inherent flexibility may prove to be the superior choice.

4. BENCHManet could be enhanced to utilize more complicated mobility models. For instance, [37] proposes the design of a mobility model and a signal propagation model that allows the placement of obstacles that restrict movement and signal propagation, as well as emulate properties of fading around imposed obstacles. The obstacle mobility model proposed could be combined with the entity and group mobility models used in BENCHManet to further improve the practical relevance of the reference configurations.

5. Two additional metrics should be considered and evaluated. The first measures the percentage of discovered services that are actually reachable

at the time of service invocation. Typically, services in highly dynamic topologies should be assigned shorter service URL lifetimes in order for cached service information to be invalidated before a network topology change occurs (and the cached service URL becomes no longer reachable). Hence, this metric will test how representative is the assigned URL lifetimes to the rate of topology change. This metric is particularly relevant if the caching modification to SLPManet is employed. The second potentially useful metric measures the number of times a message is exchanged. This metric reflects the frequency by which the channel needs to be contended for, thus revealing the combined efficiency of both routing and service discovery protocols.

6. According to the specification of SLP in [2] and thus, the specification of SLPManet in Section 4.3, in the event that a discovered service is unreachable, the user will not be able to discover alternative services until the cached URL for the desired service type times-out. Therefore, a mechanism that allows re-discovery of new services for cached service types that are no longer reachable, must be supplemented.

7. Finally, in the current NS-2 implementation of SLPManet, a mobile node can either act as a User Agent (UA), or as a Service Agent (SA). Future enhancements of SLPManet should allow the co-existence of both a UA and a SA on the same mobile node.

List of References

[1] J. Macker and I. Chakeres, "Mobile Ad-hoc Networks (manet) IETF Charter." http://www.ietf.org/html.charters/manet-charter.html.

[2] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," Request for Comments Standards Track 2608, Internet Engineering Task Force, June 1999.

[3] Sun Microsystems, *Jini Device Architecture Specification, Version 2.0*. http://www.sun.com/software/jini/specs/jini2_0.pdf.

[4] "Salution Architecture Overview." White Paper, 1998. http://www.salutation.org/whitepaper/originalwp.pdf.

[5] Universal Plug and Play Forum, *Universal Plug and Play Device Architecture, Version 1.0*, June 2000. http://www.upnp.org/download/UPnPDA10_20000613.htm.

[6] Bluetooth SIG, *Specification of the Bluetooth System, Volume 1*, 2001. http://www.bluetooth.com.

[7] M. Nidd, "Service Discovery in DEAPspace," *IEEE Personal Comm.*, pp. 39–45, August 2001.

[8] M. Storey, G. Blair, and A. Friday, "MARE: resource discovery and configuration in ad hoc networks," *Mobile Networks and Applications*, vol. 7, pp. 377 – 387, 2002.

[9] F. Zhu, M. Mutka, and L. Ni, "Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services," in *Proceeding of the 1st IEEE Annual Conference on Pervasive Computing and Communications*, pp. 235–242, IEEE Computer Society Press, March 2003.

[10] F. Zhu, M. Mutka, and L. Ni, "Classification of Service Discovery in Pervasive Computing Environments," Technical Report MSU-CSE-02-24, Michigan State University, East Lansing, 2002.

[11] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks," in *Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC)*, vol. 3, pp. 2107–2113, March 2003.

[12] R. Koodli and C. Perkins, "Service Discovery in On-Demand Ad Hoc Networks," Internet-Draft, Internet Engineering Task Force, October 2002.

[13] L. Li and L. Lamont, "A Lightweight Service Discovery Mechanism for Mobile Ad Hoc Pervasive Environment Using Cross-Layer Design," in *Proceeding of the Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'05)*, pp. 55–59, March 2005.

[14] F. Zhu, M. Mutka, and L. Ni, "Facilitating secure ad hoc service discovery in public environments," in *Proceedings of the 27th Annual International Conference on Computer Software and Applications (COMPSAC)*, pp. 433–438, November 2003.

[15] M. Balazinska, H. Balakrishnan, and D. Karger, "INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery," in *Proceeding of the International Conference on Pervasive Computing (Pervasive)*, (Springer-Verlag, Zurich, Switzerland), 2002.

[16] C. Bettstetter and C. Renner, "A Comparison Of Service Discovery Protocols and Implementation of The Service Location Protocol," in *Proceedings of 6th EUNICE Open European Summer School: Innovative Internet Applications (EUNICE'00)*, (Twente, Netherlands), Septemeber 2000.

[17] M. Barbeau, "Bandwidth Usage Analysis of Service Location Protocol," in

International Workshop on Parallel Processing, (Toronto, Canada), pp. 51–56, August 2000.

[18] J. Govea and M. Barbeau, "Results of Comparing Bandwidth Usage and Latency: Service Location Protocol and Jini," in *Workshop on Ad hoc Communications, held in conjunction with the Seventh European Conference on Computer Supported Cooperative Work (ECSCW 2001)*, (Bonn, Germany), September 2001.

[19] P. Engelstad and Y. Zheng, "Evaluation of Service Discovery Architectures for Mobile Ad Hoc Networks," in *Proceeding of the Second Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 2–15, January 2005.

[20] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," in *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobi-Com)*, (Dallas, TX), pp. 85–97, October 1998.

[21] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks," in *Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, pp. 195–206, August 1999.

[22] Sun Microsystems, *System Administration Guide: Resource Management and Network Services, SLP Implementation*, May 2002. http://docs.sun.com/app/docs/doc/806-4076.

[23] M. Peterson, J. Carey, D. McCormack, G. Rajagopal, M. Desmons, E. Hughes, J. Calcote, and M. Day, "OpenSLP project," February 2004. http://www.openslp.org.

[24] "Novell NetWare servers products." http://www.novell.com/.

[25] "Axis Communications server products." http://www.axis.com/.

[26] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifiers(URI): Generic Syntax," Request for Comments Standards Track 2396, Internet Engineering Task Force, August 1998.

[27] B. Raman, P. Bhagwat, and S. Seshan, "Arguments for Cross-Layer Optimizations in Blutooth Scatternets," in *Proceeding of the Symposium on Applications and the Internet (SAINT)*, pp. 176–184, January 2001.

[28] S. Motegi, K. Yoshihara, and H. Horiuchi, "Service Discovery for Wireless Ad Hoc Networks," in *Proceeding of the 5th International Symposium on Wireless Personal Multimedia Communications*, vol. 1, pp. 232–236, October 2002.

[29] T. Camp, J. Boleng, and V. Davies, "A survey of Mobility Models for Ad Hoc Network Research," *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2(5), pp. 483–502, 2002.

[30] W. Navidi and T. Camp, "Stationary Distributions for the Random Waypoint Mobility Model," *IEEE Transactions on Mobile Computing*, vol. 3(1), pp. 99–108, January/March 2004.

[31] W. Navidi, T. Camp, and N. Bauer, "Improving the Accuracy of Random Waypoint Simulations Through Steady-State Initialization," in *Proceedings of the 15th International Conference on Modeling and Simulation (MS '04)*, pp. 319–326, March 2004.

[32] K. Fall and K. Varadhan, *The ns Manual*. VINT Project - UC Berkley, December 2003. http://www.isi.edu/nsnam/ns/ns-documentation.html.

[33] B. Tuch, "Development of WaveLAN, and ISM Band Wireless LAN," *AT&T Technical Journal*, vol. 72(4), pp. 27–33, July/August 1993.

[34] T. Kunz, "Multicasting in mobile ad-hoc networks: Achieving high packet delivery ratios," in *Proceedings of the 2003 Center for Advanced Studies Conference (CAS)*, (Toronto, Canada), pp. 156–170, IBM Canada Ltd. Laboratory, Center for Advanced Studies, October 2003.

[35] T. Kunz, "Public Domain Code," March 2004. http://www.sce.carleton.ca/wmc/code.html.

[36] A. Varshavsky, B. Reid, and E. de Lara, "Cross-Layer Support for Timely and Efficient Service Selection in MANETs." Department of Computer Science, University of Toronto.

[37] A. Jardish, E. Belding-Royer, K. Almeroth, and S. Suri, "Real-World Environment Models for Mobile Network Evaluation," *IEEE Journal on Selected Areas in Communications (ISACEM)*, vol. 23(3), pp. 622–632, March 2005.

Appendix A

SLPManet User Manual

A.1 Required Files

First, the NS-2 simulator must be installed. Download the NS-2 package from http://www.isi.edu/nsnam/ns/, and type **`./install`** in the extraction folder: **`ns-allinone-2.27/`**.

In the SLPManet Package, **`SLPManet.zip`**, there are two directories. The first is the **`slp/`** directory which contains all source code. It consists of the following seven files:

```
1) udp-slp.h
2) udp-slp.cc
3) slp-sa.h
4) slp-sa.cc
5) slp-ua.h
6) slp-ua.cc
7) slp-common.h
```

The second directory is the **`slp_misc/`** directory, and which contains the NS-2 files that need to be altered in order to support SLPManet. The directory consists of the following seven files:

```
FILE                    Destination
-----------------------------------
1)  packet.h           -> common/.
2)  ns-packet.tcl      -> tcl/lib/.
3)  agent.h            -> common/.
4)  app.h              -> apps/.
5)  ns-default.tcl     -> tcl/lib/.
```

```
6)  cmu-trace.h       -> trace/.
7)  cmu-trace.cc      -> trace/.
8)  replaceFiles.bat
```

A.2 Installation Steps

In order to install SLPManet, perform the following steps:

1. Copy **SLPManet.zip** to **ns-2.27/.**
2. Extract the source files: **unzip SLPManet.zip**.
3. Add SLPManet support to current NS-2 files by pursuing Option 1 or 2 described below.
4. Modify the top-level **Makefile.in** by adding: **slp/udp-slp.o**, **slp/slp-sa.o**, and **slp/slp-ua.o** to the compilation list.
5. Under **ns-2.27/** type **./configure**.
6. Under **ns-2.27/** type **make clean** to remove all *.obj files.
7. Under **ns-2.27/** type **make** to finally compile the new simulator.

Option 1

Our SLPManet implementation works with any Ad Hoc Routing protocol that supports multicasting. Option 1 will pursue installation of SLPManet assuming an underlying BCAST [34] routing protocol. If you do have extra NS-2 components installed (other than BCAST), or planning to use a different routing protocol, you must pursue Option 2.

The files in the **slp_misc** directory are based on a plain NS-2 installation with BCAST support only. The script **replaceFiles.bat** copies these files to the intended destinations.

3a. Install BCAST by following the instructions in:
http://kunz-pc.sce.carleton.ca/Thesis/README.ns2code.txt

3b. Under **slp_misc/**, execute **./replaceFiles.bat** to update the NS-2 files.

Option 2

You will need to look in files under the directory **slp_misc** for SLPManet-specific code and merge it to your existing NS-2 files. SLPManet-specific code is annotated with the word **''SLP''** in these files (to facilitate a 'grep').

3a. Register SLPManet application header by modifying **common/packet.h** and **tcl/lib/ns-packet.tcl**

3b. Add **supportSLP()** and **enableSLP()** methods to the **Agent** class in **common/agent.h**.

3c. Add **recv_slpmsg** method to the **Application** class in **apps/app.h**.

3d. Set default values for the newly introduced configurable parameters in **tcl/lib/ns-default.tcl**.

3e. To support more meaningful trace file output, add necessary routines to **`cmu-trace.cc`** and **`cmu-trace.h`** in **`trace/`**.

3f. Comment out the **`#include BCAST`** line in **`slp/slp-common.h`** if you are not intending to use BCAST.

A.3 SLPManet Commands

The following commands are used to set up SLPManet in simulation scripts:

`set $udpagent [new Agent/UDP/UDPSlp]`

Creates a UDP agent that is capable of sending and receiving SLPManet packets to and from the SLPManet application layer. Note that original UDP agents in NS-2 are not capable of handling application data.

`$udpagent packetSize_` <**`pkt size in bytes`**>

Sets the maximum size of the datagram sent by the UDP agent. Command is optional; default is 1400 bytes.

`$udpagent ttl_` <**`ttl value`**>

Sets the time-to-live of UDP packets. Command is optional; default is 255 hops.

`set $slpua [new Application/SLPua]`

Creates an SLPManet User Agent (UA).

`set $slpsa [new Application/SLPsa]`

Creates an SLPManet SA.

$slpua set dst_port_ <slp listening port>

Sets slpua's listening port. The command is applicable to UAs only. Command is optional; default port is 18.

$slpagent set pktsize_ <pkt size in bytes>

Sets the maximum size of the packet sent by slpagent. The command is applicable to both UAs and SAs. Command is optional; default size is 1400 bytes.

$slpagent start

Enables the slpagent to take part in service discovery. The command is applicable to both UAs and SAs.

$slpagent add-scope <scope-name>

Configures slpagent with <scope-name>. If no scope is added by the time slpagent is enabled, SAs configure themselves with the "DEFAULT" scope. However, if UAs are not configured with a scope by the time they are enabled, they will have no configured scopes, and must solicit Service Advertisements (SAAdverts) to learn scopes in the vicinity (i.e. User-Selectable Scopes mode). The command is applicable to both UAs and SAs.

$slpagent remove-scope <scope-name>

Removes <scope-name> from the scopes configured for slpagent. The command is applicable to both UAs and SAs.

$slpsa add-service <**url**> <**srv lifetime**> <**url lifetime**>

Adds a service with service URL, <url>, to slpsa. This service is valid at the SA node for <srv lifetime> seconds from the time the command is issued. The service URL advertised in SrvRplys is only valid for <url lifetime> seconds from the time the SrvRply is received by the UA. The command is only applicable to SAs.

$slpua rqst-service <**service type**> <**multicast address**>

Multicasts to <multicast address> a Service Request (SrvRqst) for <service type>. The command is only applicable to UAs.

$slpua rqst-service service:service-agent <**multicast address**>

Multicasts to <multicast address> a SrvRqst that solicits SAAdverts. This command is used if the UA was not configured with any scopes, and User-Selectable scopes are desired. UAs will configure themselves with the scopes learned from SAAdverts. The command is only applicable to UAs.

$slpua set maxSearch_ <**true/false**>

Allow the UA to either use the first SrvRply received, or collect as many as possible (i.e. until maximum datagram size is reached, or no further replies, or MC_CONFIG_MAX has elapsed). The command is applicable to UAs only. Command is optional; default is false.

Appendix B

BENCHManet User Manual

B.1 Required Files

The package, **`BENCHManet.zip`**, contains the benchmark we have developed along with some useful scripts. The files and directories included in the package are:

```
(1)   slpRunScenario.bat
(2)   slpRunAll.bat
(3)   slp_tools/slpMotion/
(4)   slp_tools/slpTraces/
(5)   slp_tools/slpAnalysis/
(6)   slp_tools/install
(7)   slp_tools/MobileModels/
(8)   slp_tools/slpSimulationSetup.bat
(9)   slp_tools/tr2Excel.bat
(10)  slp_tools/combineAnalysis.bat
(11)  slp_tools/eval.awk
(12)  slp_tools/original_common.tcl
(13)  slp_tools/cache_common.tcl
(14)  slp_tools/conf.tcl
(15)  slp_tools/event.tcl
(16)  slp_tools/rescue.tcl
(17)  slp_tools/pursuit.tcl
(18)  slp_tools/pan.tcl
(19)  slp_tools/mesh.tcl
(20)  slp_tools/vr.tcl
(21)  slp_tools/vp.tcl
(22)  slp_tools/march.tcl
(23)  slp_tools/combat.tcl
```

The files (14)-(23) contain network configuration parameters particular to each of the 10 scenarios constituting BENCHManet. In order to generate appropriate movement patterns and setup the directory structures, the script in (8) is run. This script in turn uses the implementations of the mobility models in (7) provided by [29], [30], and [31]) to generate movement files for each of the scenarios in BENCHManet; these models must be first compiled by invoking (6). The movement files generated are stored in (3). The core simulation drivers are found in (12) and (13); the former is used when the original SLPManet is desired, while the later is used for the modified SLPManet which supports extended caching. NS-2 output traces are stored in (4). Script (11) analyzes trace files. To combine the trace analysis of all tests in a benchmark run, the script in (9) is invoked. To combine the results of several benchmark runs, the script in (10) is invoked. Analysis results are stored in (5).

The scripts in (1) and (2) automate the whole simulation and analysis process; the first automates running the procedure of simulating a particular scenario test, while the latter automates running the entire benchmark.

B.2 Installation Steps

1. Copy the package **`BENCHManet.zip`**, to **`ns-2.27/`**.

2. Unpack the package: **`unzip BENCHManet.zip`**.

3. Compile the mobility models by typing **`./install`** under **`slp_tools/`**.

B.3 Running Simulations

B.3.1 To Run a Particular Scenario

Run the script:

`./slpRunScenario.bat scenario runDirectory seed <-cache>`

The script automates the process of simulating `scenario`. `runDirectory` is the name of the sample. `seed` is used to set the seed of the random number generators. If the `-cache` option is specified, the modified version of SLPManet that allows extended caching is used instead of the original SLPManet. The script in turn invokes other scripts to set up directory structures, generate new movement files (if not already existing) and run the simulation for the given scenario.

For instance, in order to run the conference scenario with the original SLPManet, setting a seed of 2, one would type:

```
./slpRunScenario.bat conf run0 2 -cache
```

The movement file will be stored in:

```
slp_tools/slpMotion/run0/conf.motion
```

The trace file will be stored in:

```
slp_tools/slpTraces/run0/original/conf.tr
```

To analyze the generated output trace file, run:

`awk -f slp_tools/eval.awk slp_tools/slpTraces/run0/original/conf.tr`

B.3.2 To Run the Entire Benchmark

Run the script:

```
./slpRunAll.bat runDirectory seed <-cache>
```

The script is used to run all scenarios in the benchmark. `runDirectory` is the name of the sample run. `seed` is used to set the seed of the random number generators. If the `-cache` option is specified, the modified version of SLPManet that allows extended caching is used instead of the original SLPManet. The script in turn invokes other scripts to setup directory structures, generate new movement files (if not already existing), run the simulations, and analyze trace files.

For instance, in order to run the benchmark, with a seed of 2, and enabling caching, one would type:

```
./slpRunAll.bat runAll0 2 -cache
```

The movement files will be stored in:

```
slp_tools/slpMotion/runAll0/<scenario_name>.motion
```

The trace files will be stored in:

```
slp_tools/slpTraces/runAll0/cache/<scenario_name>.tr
```

The summary of analysis of all scenarios will be stored in:

```
slp_tools/slpAnalysis/runAll0_cache.txt
```

Note that the script in (8), which is invoked by `slpRunScenario.bat` and `slpRunAll.bat`, randomly generates a new set of movement files for every new `runDirectory`. If the simulation was run for an already existing `runDirectory`, then a set of movement files must have already been previously generated for this sample run, and are hence used in the simulation.

The `seed` only controls the random generators used in the NS-2 simulation. Random generators were used to randomly assign agents to nodes, and to generate random exponential inter-arrival times for user requests.

If you wish to run several samples of the benchmark, and then merge the analysis files into a single file, run:

`./combineAnalysis.bat` <**`-cache`**>

The script combines all the benchmark summary files:

`slp_tools/slpAnalysis/*_original.txt`

and produces

`slp_tools/slpAnalysis/original_results.txt`.

If the `-cache` option was specified, it will similarly merge all the cache analysis summary files and produces `cache_results.txt`.

B.4 Understanding the Analysis Output

The output of the trace file analysis is organized in rows, each of which represents the analysis of a simulation sample run of a given scenario. Each row contains 16 columns, whose headers are described below:

1. **`scenario`**: Name of the scenario test (e.g. **conf**).
2. **`Rqsts`**: Total number of discovery transactions.
3. **`tRplys`**: Total number of successful discovery transactions.
4. **`uRplys`**: Portion of discovery transactions satisfied by new SrvRplys.
5. **`Cached`**: Portion of discovery transactions satisfied through cache hits.
6. **`Succ`**: Overall Discovery Success Percentage.
7. **`minLat`**: The minimum latency consumption of all successful discovery transactions (in seconds).
8. **`maxLat`**: The maximum latency consumption of all successful discovery transactions (in seconds).
9. **`avgLat`**: The average latency consumption of all successful discovery transactions (in seconds).
10. **`stdLat`**: The standard deviation in latency consumption of all successful service discovery transactions (in seconds).

11. **minBW**: The minimum bandwidth consumption of all successful discovery transactions (in bytes).

12. **maxBW**: The maximum bandwidth consumption of all successful discovery transactions (in bytes).

13. **avgBW**: The average bandwidth consumption of all successful discovery transactions (in bytes).

14. **stdBW**: The standard deviation in bandwidth consumption of all successful service discovery transactions (in bytes).

15. **aggBW**: Aggregate bandwidth consumption by the entire discovery process (in bytes).

16. **wasBW**: Total wasted bandwidth due to unsuccessful discovery transactions (in bytes).