Application Layer Architectures for Disaster Response Systems

by

Mohammadmajid Hormati

A Thesis

in

The Department of Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montreal, Quebec, Canada

June 2013

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Mohammadmajid Hormati

Entitled: Application Layer Architectures for Disaster Response Systems

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (Electrical & Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. J. Grégoire External Examiner

Dr. J. Rilling External to Program

Dr. R. Dssouli Examiner

Dr. A. Agarwal Examiner

Dr. F. Khendek Thesis Co-Supervisor

Dr. R. Glitho Thesis Co-Supervisor

Approved by

_____
Chair of Department or Graduate Program Director

August, 2013

_____
Dean of Faculty

ABSTRACT

Application Layer Architectures for Disaster Response Systems

Mohammadmajid Hormati, Ph.D.
Concordia University, 2013

Traditional disaster response methods face several issues such as limited situational awareness, lack of interoperability and reliance on voice-oriented communications. Disaster response systems (DRSs) aim to address these issues and assist responders by providing a wide range of services. Since the network infrastructure in disaster area may become non-operational, mobile ad-hoc networks (MANETs) are the only alternative to provide connectivity and other network services. Because of the dynamic nature of MANETs the applications/services provided by DRSs should be based on distributed architectures. These distributed application/services form overlays on top of MANETs.

This thesis aims to improve three main aspect of DRSs: interoperability, automation, and prioritization. Interoperability enables the communication and collaboration between different rescue teams which improve the efficiency of rescue operations and avoid potential interferences between teams. Automation allows responders to focus more on their tasks by minimizing the required human interventions in DRSs. Automation also allows machines to operate in areas where human cannot because of safety issues. Prioritization ensures that emergency services (e.g. firefighter communications) in DRSs have higher priority to receive resources (e.g. network services) than non-emergency services (e.g. new reporters' communications). Prioritizing vital services in disaster area can save lives.

This thesis proposes application layer architectures that enable three important services in DRSs and contribute to the improvement of the three aforementioned aspects of DRSs: overlay interconnection, service discovery and differentiated quality of service (QoS). The overlay interconnection architecture provides a distributed and scalable mechanism to interconnect end-user application overlays and gateway overlays in MANETs. The service discovery architecture is a distributed directory-based service discovery mechanism based on the standard Domain Name System (DNS) protocol. Lastly, a differentiated QoS architecture is presented that provides admission control and policy enforcement functions based on a given prioritization scheme.

For each of the provided services, a motivation scenario is presented, requirements are derived and related work is evaluated with respect to these requirements. Furthermore, performance evaluations are provided for each of the proposed architectures. For the overlay interconnection architecture, a prototype is presented along with performance measurements. The results show that our architecture achieves acceptable request-response delays and network load overhead. For the service discovery architecture, extensive simulations have been run to evaluate the performance of our architecture and to compare it with the Internet Engineering Task Force (IETF) directory-less service discovery proposal based on Multicast DNS. The results show that our architecture generates less overall network load and ensures successful discovery with higher probability. Finally, for the differentiated QoS architecture, simulations results show that our architecture not only enables differentiated QoS, it also improves overall QoS in terms of the number of successful overlay flows.

Acknowledgement

This thesis would have not been possible without the support and guidance I received from my supervisors, Dr. Ferhat Khendek and Dr. Roch Glitho. I thank Dr. Khendek for his generous support, research insights and constructive comments for this work. His advises have helped me to improve my research and professional skills beyond the reach of this work. I thank Dr. Glitho for his concrete research ideas, comments and suggestions during the preparation of this work. I was privileged to have continuous valuable discussions with him through my entire Ph.D. program.

I would like to also thank Dr. Fatna Belqasmi for her kind and crucial contributions in this work. Her knowledge about the research domain and her detailed comments was very helpful.

Special thanks to Concordia University, Ericsson and the Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support.

I would like to thank my supervisory committee members, Dr. Dssouli, Dr. Agarwal, and Dr. Rilling, for their advices and comments at different stages of this research and for their effort in the final evaluative process.

I express my profound appreciation to my wife Farnaz for her love and inspiration during the past few years. Her support and encouragement was in the end what made this dissertation possible.

My parents, Mohammad Reza and Minoo, receive my deepest gratitude and love for their unconditional support through my entire life. This thesis is dedicated to them. I would

like to thank my sister Maryam and my brother Mojtaba for always supporting me during the various stages of my life.

Last but not least, I would like to thank all my colleges and friends in our lab, specially Saba Hamedi, Pejman Salehi, Ali Kanso, and Razieh Safaripour, for their friendship and support that helped me in this work.

Table of Contents

List of Figures

List of Abbreviations

3G	Third Generation Wireless System
API	Application Programming Interface
CDNS	Cluster-based DNS
DHT	Distributed Hash Table
DiffServ	Differentiated Service
DNS	Domain Name System
DoS	Denial of Service
DRS	Disaster Response System
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
iMANET	Internet based Mobile Ad hoc Network
IMS	Internet Multimedia Subsystem
IntServ	Integrated Service
IP	Internet Protocol
M2M	Machine-to-Machine
MANET	Mobile Ad hoc Network
mDNS	Multicast Domain Name System
P2P	Peer-to-Peer
QoS	Quality of Service
SIP	Session Initiation Protocol

SLP	Service Location Protocol
VANET	Vehicular Area Network
WBAN	Wireless Body Area Network
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Chapter 1: Introduction

This chapter starts by presenting the motivation of this thesis. It then states the problem, presents the thesis objectives, general requirements, and summarizes the contributions along with the related publications. Finally it describes the thesis organization.

1.1 Motivations and Problem Statement

When a large-scale natural or human made disaster such as an earthquake, hurricanes or a terrorist attack happens there is a need to respond quickly and in organized fashion. Disaster response includes the mobilization of emergency services and first responders to the disaster area. In the disaster area, the rescue teams are responsible to evacuate the victims, triage the patients and transfer them to hospitals. Several teams may be involved in this process such as firefighters, paramedics, police department and specialists.

An effective disaster response operation should provide help and rescue to the victims at the right time. Traditional disaster response methods face several issues such as limited situational awareness, lack of interoperability and reliance on voice-oriented communications [1]. Low situational awareness (e.g. inability to track units and equipment or to receive notifications about the events in the disaster area) has negative impacts on rescue teams' decisions and operations. Lack of interoperability between different rescue teams in the disaster area, e.g. firefighters and first responders, may result in coordination issues. Finally, sole reliance on voice communication limits the richness of information that can be communicated and unnecessary communication overhead.

Disaster response systems (DRSs) enhance the situational awareness of responders and assist them by providing a wide range of applications and services for communication and collaboration. For instance monitoring applications can be used to collect information about environmental situation in the disaster area using sensor networks [2]. Health monitoring and triage applications can be used to collect ambient information about victims' vital signs [3]. Location tracking applications can be used to locate and track patients and rescue teams and resources in disaster area [4]. Human limitations such as rescue safety and effectiveness, long preparation time and limited number of trained individuals for various tasks also affect disaster response process. To overcome these limitations DRSs can use robots to operate on areas that humans cannot because of size, extreme heat, toxicity of environment, etc. [5]. Furthermore, DRSs enable and benefit from various services such as machine-to-machine (M2M) communication, automated discovery, interoperability, security and privacy to provide the aforementioned applications.

Existing network infrastructure may become non-operational in the disaster area. In such cases, mobile ad-hoc networks (MANETs) may be the only alternative to provide connectivity. MANETs are transient networks, formed dynamically by a collection of arbitrarily-located wireless mobile nodes, with no reliance on any existing infrastructure or centralized administration [6]. Because of the ad hoc nature of MANETs, the applications provided by DRSs should follow distributed architectures. These distributed application/services form overlays on top of MANETs. Overlay networks are logical application layer networks that enables various services such as naming and addressing,

scalability and reliability [7]. A detailed background on MANETs and overlay networks is provided in Chapter 2.

DRSs should provide various services to enable the correct operation of different applications. This thesis focuses on three major areas in DRS: interoperability, automation and prioritization. A wide range of other services may be required such as security, privacy, reliability, availability, etc. However for simplicity we assume that the underlying infrastructure already provides these services if needed. The following presents the motivations behind the focused areas of this thesis.

- Interoperability: Interoperability between applications enables communication and collaboration between different rescue teams, e.g. firefighters and first responders. As we discussed earlier, this is critical to improve the efficiency of rescue operations and avoid potential interferences between teams.
- Automation: Automation aims to minimize the required human intervention in applications. First, it allows responders to focus on more complex tasks, which is an advantage in disaster situation. Second, it allows for operation in special conditions where humans cannot intervene. For instance robots will be able to provide operations in areas with hazardous material or search places that human cannot reach.
- Prioritization: It is vital to ensure that the communication in emergency applications has more reliability and quality than non-emergency applications. For instance the communication between firefighters is certainly more critical than the communication between news reporters in the disaster area.

There are three objectives in this thesis that are associated to the aforementioned focus areas, i.e. interoperability, automation, and prioritization.

- Provide an overlay interconnection architecture to enable communication between various types of overlays. Interconnecting various types of overlays is an important step for enabling the interoperability. For instance, it enables the medical staff and firefighters to coordinate their organization according to the situation in the disaster area.
- Provide a service discovery architecture that enables overlay nodes to discover existing services in the network. Service discovery ensures that no human intervention is required for configuring overlays nodes to know how to find existing application or services. Therefore it satisfies the goal of automation. Service discovery enables M2M communications which is critical for seamless interactions between devices such as sensors and robots in the disaster area.
- Provide a differentiated quality of service (QoS) architecture that enables the enforcement of a prioritization scheme between overlays and users in the network.

1.2 General Assumptions and Requirements

Providing services in DRSs faces several challenges that raise from the dynamic nature of MANETs, the characteristics of disaster response operations and devices used during these operations. To define clearly the scope of the problem and the contributions reported in this thesis, we set the following assumptions regarding the network and devices in the DRSs. First, we assume that all the nodes in the MANETs are IP enabled. Second, we assume that MANETs support multicasting and a specific multicast address

is pre-known to all nodes. Third assumption is that each node stores its capability value, which is dynamically updated according to its available host device's battery status and processing power.

On the other hand, when devising architectures to enable services in DRSs, the following general requirements (also seen as constraints) should be taken to account:

1. **No permanent centralized entity**: Since nodes in the MANETs may leave and join any time, the architectures should not include any permanent centralized entity.
2. **Lower layer independence**: Several types of networks (e.g. wireless local area networks, body area networks) using different protocols may exist in disaster response scenarios. The architectures should be provided in the application layer to ensure that they are not limited to specific network types or devices. Therefore, the architectures should not use any parameter that is computed based on a specific lower layer protocol (e.g. connectivity degree, physical layer information, etc.).
3. **Scalability in terms of number of nodes**: In a large-scale disaster the number of first responders and vehicles equipped with communication devices is on the order of hundreds of devices [8]. Therefore, MANETs of such scale should be supported.
4. **Support low to medium mobility**: Node mobility in the disaster area, ranging from pedestrians with low mobility (1-2 m/s) to vehicles with medium mobility (10-12 m/s) should be supported [9].

5. **Support resource-constrained devices**: Considering that resource-poor and battery powered devices such as sensors and actuators are commonly used in disaster response scenarios, the provided services should be able to function on constrained devices.

Specific requirements are also derived for each of the services, which are discussed later in the corresponding chapters.

1.3 Contributions of the Thesis

This section summarizes the contributions of the thesis. Each of our contribution corresponds to an objective of the thesis.

- **Overlay Interconnection Architecture** ([10]): We propose a scalable and lightweight architecture to enable the interconnection of wireless sensor network (WSN) gateways and applications. Based on our critical literature review with respect to our requirements, we show that existing approaches have mainly focused on interconnecting specific type of overlays or were not independent of lower layers. We also provide software architectures for the main components of our architecture, along with a prototype implementation as proof of concept. Our evaluation based on the implemented prototype shows that our architecture achieves an acceptable interconnection delay and network overhead.
- **Service Discovery Architecture** ([11]): Our critical review of the MANETS' service discovery architectures shows that the existing methods mainly lack the support of required level of scalability and mobility or ability to function on resource-constrained devices. To address these issues we propose a novel distributed directory-based service discovery architecture based on the DNS

protocol. In particular, we extend the semantics of DNS resource records to enable dynamic formation of a distributed service directory in MANETs. We have run extensive simulations to evaluate the performance of our architecture and compared it with the IETF directory-less service discovery proposal based on Multicast DNS [12]. The results show that our architecture generates less overall network load and ensures successful discovery with a higher probability.

- **Differentiated QoS Architecture**: We propose a new differentiated QoS architecture for overlay-based DRSs to enforce a prioritization scheme between overlays as well as between users within overlays. We show that existing solutions mainly do not allow for the definition of arbitrary prioritization levels or they are not independent of lower layer protocols. Our architecture is based on a self-organizing overlay that provides admission control and policy enforcement functions. We have run extensive simulations to evaluate the performance of our architecture. The results show that our architecture not only enables differentiated QoS, it also improves overall QoS in terms of the number of successful overlay flows.

1.4 Thesis Organization

The rest of this thesis is organized as follows: Chapter 2 discusses the background information on MANETs, overlay networks, service discovery and quality of service. Chapter 3 presents our overlay interconnection architecture along with the provided prototype and evaluation results. Chapter 4 presents our service discovery architecture based on DNS protocol as well as the evaluation of the architecture. Chapter 5 presents our differentiated QoS architecture along with the simulation results that indicate the

efficiency of the proposed architecture. The work related to each of our contributions is discussed in the respective chapter. Chapter 6 concludes the dissertation and discusses potential future work.

Chapter 2: Background

2.1 Mobile Ad hoc Networks (MANETs)

A MANET is a self-organized wireless network of mobile devices which is not based on any infrastructure or centralized control [6]. Devices in MANETs can be heterogeneous and use wireless technologies such as the Institute of Electrical and Electronics Engineers (IEEE) 802.11, 802.15, or Bluetooth to connect to the network. Each node in the MANET plays both router and host roles and participates in multi-hop routing.

MANETs are used to provide communication when no infrastructure is available or to extend the network boundaries using multi-hop routing. MANETs are extensively used in military and emergency response situations to provide infrastructure-less communication between several agents involved. Vehicular Ad Hoc Networks (VANETs) [13] are also another type of MANETs which provide communication between vehicles and roadside equipment for safety and entertainment purposes. MANETs also have been applied on other application areas such as health, commercial, educational, entertainment and home scenarios. Several applications such as remote health monitoring, urban gaming, smart homes, robotics, traffic monitoring, etc. have applied MANETs.

MANETs can mainly be classified in two different views. First view classifies MANETs based on the relation between MANETs and other networks. Second view classifies MANETs based on their coverage area.

For relation of MANETs with other networks, two type of MANETs can exists, i.e. standalone and integrated networks. Standalone MANETs are ad hoc networks which are

isolated and each node can only communicate with other nodes in the same network. On the other hand integrated MANETs are capable to interconnect with other networks such as internet or third generation wireless networks (3G).

For instance, Internet based Mobile Ad hoc Networks (iMANET) [14] are ad hoc networks that link MANET nodes to the internet using fixed internet-gateway nodes. Several proposals are also introduced for interconnecting MANETs and 3G networks. A general assumption is that nodes are equipped with both MANET and 3G network interfaces so they are able to communicate with other nodes in both networks. It is also possible that some of nodes act as gateway to provide interconnection with 3G networks for other nodes in MANET. Main objectives of this interconnection are to extend the 3G network coverage [15] or to balance the load between cells [16].

We also can classify MANETs to three groups based on their coverage area, namely: Body, Personal and Local Area networks [17]. Wireless Body Area Networks (WBANs) [18] are designed to work on wearable computers. WBANs provide wireless communication between small wearable components such as pulse oximeter, and blood pressure cuff, head-mounted displays, etc. The coverage range of WBANs is around one meter and they use standards such as ZigBee or Bluetooth. Wireless Personal Area Networks (WPANs) [19] provide wireless connectivity between personal computer devices usually in a range of few meters such as telephones, PDAs, etc. WPANs use standards such as IrDA, Bluetooth, ZigBee, etc. Wireless Local Area Networks (WLANs) [20] provide wireless connectivity between devices using an access point or in ad hoc manner. WLANs use IEEE 802.11 standards and provide coverage to areas such

as homes or offices. Figure 2.1 illustrates different mobile ad hoc networks and their coverage range.

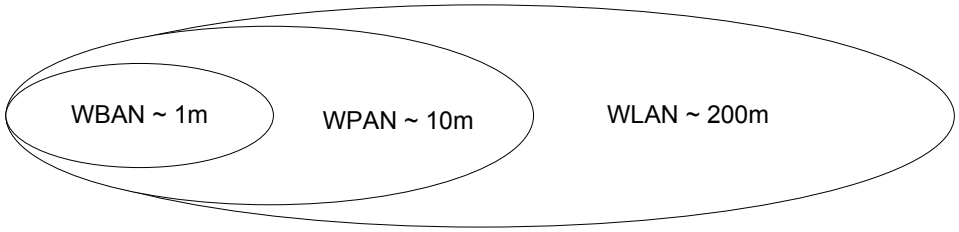

Figure 2.1: Ad Hoc Networks Classification

MANET environment characteristics such as mobility, infrastructure-less, network churn, unreliable channels, and heterogeneous and resource-constrained network nodes introduce new research issues. Research objectives are mainly about increasing network lifetime, reliability, scalability, performance. In layered network architecture, specific research issues are proposed for each layer. For instance, the physical layer should tackle the spectrum allocation and usage issue. The data link layer should address the media access control, error a correction issues. The network layer should provide robust and efficient routing protocols, addressing scheme, multicasting. The transport layer should address issues such as TCP adaptation and backoff window. Finally application, presentation and session layers need to address various issues including network auto-configuration, security (authentication, authorization), location services, QoS, etc.

Recently cross-layer designs have been also proposed to overcome the MANET challenges mostly on the issues such as energy conservation, QoS, reliability, scalability, network simulation, and performance optimization [6].

2.2 Overlay Networks

Peer to peer (P2P) overlay networks (in short, overlay networks or overlays) are logical application layer networks which try to overcome existing network challenges such as scalability, multicasting, naming and addressing, reliability, QoS and anonymity [7]. As Figure 2.2 illustrates, peers in the overlay networks are connected with virtual links. These are logical links and may not correspond to the underlying network physical links. Overlay networks generally provide various features such as self-organization, peer naming and discovery independent of underlying networks and reliable, robust, fault tolerant and scalable routing.

In contrast with client-server paradigm, peers in overlay networks act both as clients and servers to use and provide services or to share their computational and storage capacities with other peers in the network. These characteristics are used to create scalable and robust distributed applications such as file sharing, instant messaging, resilient networks, and content delivery networks. For instance several file sharing applications such as Napster [21], Gnutella [22], Bittorrent [23] and Kazaa [24] have used P2P overlays. Skype [25] as a well-known application in IP telephony domain uses a P2P overlay to deliver voice and video conferencing services.

Overlay networks have been shown very effective in MANETs by tackling existing network challenges such as robust routing, topology maintenance, multicasting and security [26].

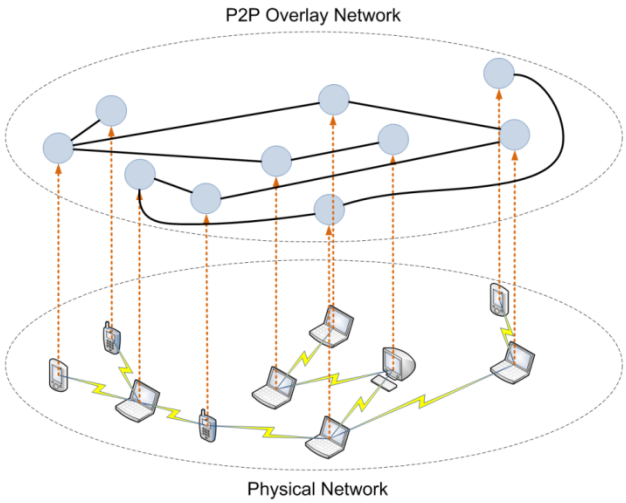

Figure 2.2: A P2P Overlay Network

Overlay networks are classified in two groups based on the organization mechanisms, namely structured and unstructured overlay networks.

2.2.1 Unstructured Overlays

Unstructured overlays such as Freenet [27], Gnutella [22] and Overnet [28] organize nodes in a random graph and use random search techniques to retrieve data from the overlay nodes. They do not impose any constraint on how to organize the network and how the data should be stored or retrieved. These characteristics lead to efficient self-organization and recovery mechanisms. However data retrieval will not be efficient especially in the case of rare data items, which raises scalability issue. Several techniques are proposed to address this issue such as forwarding, caching and overlay topology optimization [29]. In the forwarding technique, each peer selects a group of its neighbors based on network statistical information such as latency, and transmits the messages to them. The caching technique assign a group of peers to each peer based on an overlay distance metric. Each peer is then responsible to store list of resources that other peers in

the assigned group have stored. The third approach tries to optimize the overlay topology based on the real network topology to optimize the overlay performance metrics such as latency.

2.2.2 Structured Overlays

Structured overlays provide mechanisms to manage overlay network topology and deterministically and efficiently locate a node's real network address, e.g. IP address, from its logical overlay address. They also provide mechanisms to efficiently store and access data in/from the other nodes. So far structured overlays such as Chord [30], Pastry [31] and Tapestry [32] are implemented using distributed hash tables (DHTs). Like ordinary hash tables, DHTs provide store and retrieve functionalities but in the distributed manner which let them scale to extremely large number of nodes and handle nodes arrival, departure and failure. Each data item should have a unique key which is hashed by the DTH's hash function to find where this data should be stored to or retrieved from.

The following list includes some of main research challenges on overlay networks:

- Self-organization and self-recovery: overlays should organize and adopt a network topology based on the physical network characteristics. These problems are challenging specially in mobile ad hoc networks [33].
- Efficient and scalable routing and look up and caching: Huge amount of works have been done to provide efficiency and scalability in overlay networks in different physical networks [22] [34]. Hierarchical routing methods and super peer architectures are also proposed to address these issues [35].

- Location-aware overlays: Location-aware overlay networks are recently proposed to utilize physical network topology in overlay organization, recovery and routing [36] [37].
- Security: Security is a main concern in un-trusted P2P networks especially in military applications. Encryption techniques, distributed trusts and reputation methods are used to address security issues [38].
- Implementation: Several proprietary and open source projects are developing P2P overlay networks which are not yet stable. For instance, JXTA [39] by Sun Microsysytems, is an open source project which include protocol specification and implementation for overlay networks which strongly resemble Tapestry [32]. HyperCast [26], Open Chord [40], and RON [41] are a few other open source projects under development.

It worth noting that issues related to overlay networks depends on the applications and underlying network characteristics. For instance security may or may not be required by applications while hierarchical routing may depend on the mobility and resource limitation of underlying physical network.

2.3 Service Discovery

Service discovery is any mechanism that provides the capability to locate services in the network. A service is often described as a subset of resources in the network that could be used by other nodes, such as applications, network services, memory, computation capability, etc [42].

Two main services issues that should be addressed by a service discovery mechanism are naming and searching. Naming defines a mechanism to give and maintain names to services while searching is the mechanism to match a service discovery request with the service names that are available in the network.

Different categorization of service discovery approaches could be defined based on their characteristics. An interesting categorization could be made based on their architectures that partitions them into three groups: client-server, peer-to-peer or hybrid approaches. A brief description and examples of each group is provided as follows.

- **Client-server:** Service discovery mechanisms with client-server architecture clearly define a set of fixed servers or service directories that host the service information while other nodes in the network only act as clients. This architecture usually proposes mechanisms to collect the service information of the service providers as well as providing the address of service directories to the clients. Examples of such systems are Jini [43] and Service Location Protocol (SLP) [44]. Because of the structure of client-server service discovery approaches, they usually have best search performance but they could face single point of failure and suffer from Denial of Service (DoS) attacks.
- **Peer-to-peer (P2P):** P2P service discovery architectures can be based on structured or unstructured P2P overlay networks. With structured overlays, the service information is uniformly distributed between the peers based on a predefined mechanism. For instance DHTs could be used to determine which peer is responsible to store specific service information using a hash function [45]. Structured P2P approaches address the single point of failure issue and are

resilient to DoS attacks. However, they lack the ability to perform partial matching in their queries. In addition, maintaining the overlay structure, in case of nodes joining or leaving, imposes additional overhead [46].

In unstructured P2P approaches, service information could be hosted by service providers or it could be cached by other nodes in the network. Unstructured architectures, e.g. Gnutella [22], do not scale well in terms of number of nodes, because of the overhead imposed by their search mechanism. However they are able to provide complex and partial matching queries.

- **Hybrid:** Hybrid architectures mix structured and unstructured P2P architectures to take the advantages of both approaches. Often clustering is employed to define a loose structure in unstructured P2P architecture. Clustering techniques can be categorized into super node clustering and similarity clustering. In super node clustering (e.g. Gnutella2 [47] or FastTrack [48]), typically a cluster head (super node or super peer) is elected based on predefined set of parameters such as processing power or number of neighbors.

  In similarity clustering there is no notion of super node, instead, clusters are formed based on the similarity of nodes interests. Each node can belong to several clusters based on its interests. For instance, in interest-based locality [49], each cluster defines a set of nodes that are interested in a specific service. Shortcuts are defined between the nodes of each cluster and these shortcuts are ranked based on how many queries get answered via them.

2.4 Quality of Service

IETF has defined quality of service (QoS) as "A set of service requirements to be met by the network while transporting a flow" [50]. Network service requirements could be defined based on several QoS parameters such as:

- Bit rate: Number of bits per second that should be achieved while transferring data in a flow.
- Delay: Delay that packets in a flow experience while they are being transferred from one end-point to another.
- Jitter: Variation of delays that are experienced by packets of a flow.
- Packet loss rate: Number of packets that are not delivered over the total number of packets in a flow.

An important attribute of a QoS mechanism is whether it provides end-to-end QoS (hard QoS) or it operates in particulate domain or domains of the network (soft QoS). IETF has proposed Integrated Service (IntServ) [51] and Differentiated Service (DiffServ) [52] models as examples of hard and soft QoS, respectively. IntServ uses signaling to create an end-to-end path per flow and to reserve the required network resources along the path. Since network resources are pre-allocated in the path, IntServ is able to guarantee end-to-end QoS. However, IntServ model requires every router in the network to understand its signalling protocol and behave accordingly, which leads to several deployment issues. Furthermore, IntServ is not suitable for highly dynamic networks such as MANETs because of the unreliability of the network links and nodes and because of its signalling overhead.

DiffServ enables the classification of network flows and defines how individual routers should prioritize flows based on such classification. Therefore sensitive or important flows (e.g. voice, video, emergency communication) could be prioritized over non-sensitive flows (e.g. HTTP, FTP or email). Although DiffServ does not enforce any prioritization scheme, it recommends a set of Per Hop Behaviours (PHBs) to unify the behaviour of network routers. A PHB is associated with a set of traffic classes and defines how each hop should behave upon receiving a flow that is marked as those traffic classes. Three commonly used PHBs are as follows. Default PHB or best-effort forwarding do not require any specific behaviour in the hops. Expedited Forwarding (EF) PHB [53] ensures that the traffics in the group receive higher priority than other traffic classes. Assured Forwarding (AF) PHB [54] ensures that packets in the traffic in this group are delivered as long as they do not exceed a subscribed rate. Because DiffServ allows best-effort forwarding, it could provide a level of prioritization even if some of the routers in the network do not support DiffServ. However even if all the routers support DiffServ, since the detail behaviour of each individual router is implementation specific, it is difficult to predict the end-to-end behaviour.

Chapter 3: Overlays Interconnection Architecture

WSNs are sets of distributed nodes that collaborate to monitor physical, environmental and physiological conditions [55]. WSNs are commonly used in situations such as emergency response or modern warfare to provide health monitoring, location tracking and environmental monitoring. Ambient information collected by WSNs is made available to end-user applications through gateways. Enabling communication between gateways and end-user applications faces several challenges, especially when infrastructure-based networks are not viable such as in disaster response situations. This makes the use of WSNs in MANET settings a very pertinent issue.

P2P overlay-based systems have been proposed for WSN gateways for MANETs and in end-user applications in the same settings. For example, reference [56] presents an overlay-based gateway to enable integration of Internet Multimedia Subsystem (IMS) with mobile sink-based WSNs. Reference [57] proposes a resource monitoring scheme for group-based applications in MANETs, based on clusters of information that communicate with each other using a group-based overlay.

We assume that both WSN gateways and end-user applications are based on P2P overlays. This raises the critical issue of how to interconnect these two overlays to allow the applications to access ambient information via the gateways. Accessing the information brings a host of technical challenges, such as gateway discovery and

subscribe/notify messaging. We propose a decentralized, self-organized and scalable architecture for interconnecting end-user applications and gateway overlays.

The rest of this chapter is organized as follows: the next section presents a motivating scenario, derives specific requirements for overlay interconnection and examines related work. The second section is devoted to the proposed architecture. The implementation of our prototype is then described in the third section, and we conclude in the last section.

3.1 A Motivating Scenario, Requirements and Related Work

In this section, we first introduce a motivating scenario, and then use it to derive requirements. Related work is then discussed in light of these requirements.

3.1.1 A Motivating Scenario

Imagine that a large-scale disaster, such as an earthquake, has destroyed the existing network infrastructures in a city. Different emergency teams from different organizations such as first responders, paramedics and police departments are sent to respond to the situation.

The members of each team are equipped with devices that form a MANET, and the MANET hosts the applications used by the team members. For instance, for first responders the MANET will host applications that help in finding and identifying the victims while in the case of paramedics the applications will be help in treating patients. When it comes to the police department, the applications will aim at security issues.

Although the applications in different MANETs follow different goals, they mostly rely on the same ambient information collected from the disaster area. Therefore, it is

reasonable that an existing gateway will be used by applications in other MANETs to reduce the information collection effort and optimize the overall performance. For instance, the information collected by the first responders' WSN gateway thanks to the vital sign-monitoring sensors installed on the victims should be provided to other applications such as the paramedics' applications for triage and treatment when they arrive at the scene. These interactions can significantly increase the effectiveness of emergency response operations. This scenario is depicted in Figure 3.1.

The fact is that each team has its own organizational protocols and the MANET of each organization has its own middleware. It is important to note that in such settings, entities such as applications and gateways are usually distributed and built as overlays. This means, for instance, that the paramedics' health monitoring application overlay will need to interact with the overlay gateway where the first responders store the information collected by the vital sign monitoring sensors.

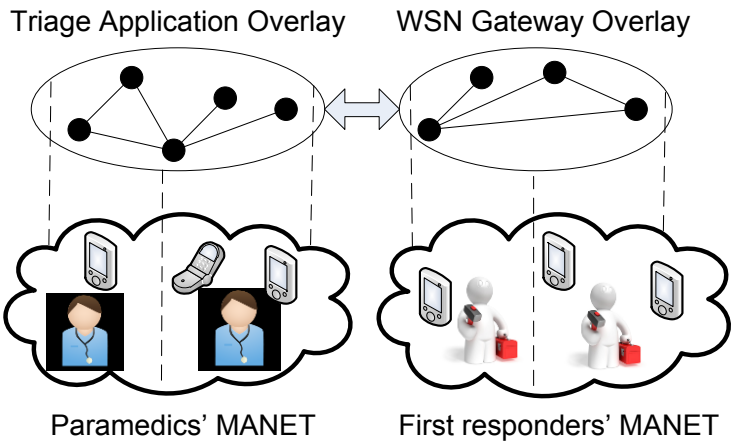

Figure 3.1. Motivating Scenario

3.1.2 Requirements

In addition to our general requirements discussed in Chapter 1, we introduce the specific requirements for the overlay interconnection architecture. We also present the requirements of the application layer protocol on which the architecture relies. This protocol is used for the purpose of message exchanges between the two overlays.

3.1.2.1 Interconnection architecture

First, the architecture should be independent of any P2P overlay architecture and middleware. This requirement ensures there will be various types of applications and gateways that utilize different P2P architectures (i.e. structured vs. unstructured) and middleware.

Second, the architecture should provide a publish/subscribe messaging service. The rationale behind this requirement is that it enables application overlays to receive notification about information changes in the WSN gateways.

Third, the architecture should provide gateway node discovery. The motivation for this requirement is that application nodes need to find a gateway node that provides specific services (e.g., a healthcare application may look for a gateway node that provides health status for a specific patient).

3.1.2.2 Interconnection protocol

In this section we specify the requirements for the interconnection protocol by refining our general requirements and the requirements on interconnection architecture.

First, the protocol should be simple and lightweight to make efficient use of resource-constrained mobile devices.

Second, the interconnection protocol should be suitable for P2P and distributed architectures.

Third, the protocol should be a standardized and commonly-used protocol so that it can be easily deployed.

Fourth, the protocol should support publish/subscribe message exchange patterns.

Fifth, the protocol should provide a mechanism for node discovery in a distributed manner, which is a required service enabling self-organization and self-discovery in MANET applications.

3.1.3 Related work

There are numerous middleware proposals that interconnect applications and gateways in WSNs and embedded systems. For instance, in [58] authors follow a disaster response scenario similar to our motivation scenario and present RUNES as a system middleware that allows interconnection of different application and gateway entities. RUNES and many other similar proposals provide great flexibility in terms of defining different types of applications, devices and communication patterns. But even with this flexibility it is not feasible to assume that in a practical scenario a single middleware can be adopted by all application and gateway entities. Our focus is on providing interconnection between applications and gateways that relay on different types of middleware and overlays.

To the best of our knowledge there has been no research specifically addressing the interconnection of application and WSN gateway overlays in MANETs. Therefore, we review proposals that address the interconnection of P2P overlays in general. These proposals mostly aim at extending the availability of overlays' common services, such as content sharing and multicasting. Other objectives addressed by these architectures are robust and efficient routing and network partitioning recovery.

In a broad view, we can classify these solutions into two categories. The first category introduces nodes that belong to two or to several overlays, which are used to translate and relay messages from one overlay to another. These nodes are often called co-located nodes or gateways. The second category is based on a hierarchy of overlays.

Synapse [59] is an interconnection architecture for overlays on the internet that lies in the first category. It introduces co-located nodes serving as inter-overlay bridges to enhance the overlay network metrics, such as scalability, failure recovery and resource discovery. The architecture works for both structured and unstructured overlay networks. The synapse protocol uses overlay message routing to route the messages inside each overlay which links the interconnection architecture to the underlying overlay technologies. Also, since the protocol presents a general overlay interconnection mechanism it does not satisfy our specific requirements on event notification and gateway node discovery.

Reference [60] presents a second example, which employs a gatewaying mechanism to bridge various distributed hash tables and to enable cross-DHT searching in wireless networks. This method provides the required routing and translation mechanisms, as well

as a performance evaluation, but since it was developed exclusively on DHT-based networks, it is not suitable for our general case.

Regarding the second category of solutions, these approaches try to select a group of nodes in overlays, called super peers, and then build a new ove rlay based on these nodes to interconnect the existing overlays. In [61] the authors present a vehicular overlay network that provides various services through a multi-layer overlay network. The super peer communication is not based on underlying overlays. However, communication between each peer and a super peer node is done using an overlay protocol and needs to be translated to the upper overlay's protocol in the hierarchy. This does not satisfy our requirement for the independence of interconnection architecture from the overlay architectures. Furthermore, it does not satisfy our requirements on gateway node discovery and event notification. Reference [62] proposes a hierarchical P2P system for DHT-based overlays. Super peers are selected in each overlay and then they form a top-level overlay which acts as an interconnection mechanism. As with [60], this work is also focused on DHT-based overlays and so does not satisfy our first specific requirement.

In summary, to the best of our knowledge, none of the existing proposed middleware/architectures address our discussed requirements such as self-organization, self-recovery and scalability in MANET environment.

3.2 The Overall Interconnection Architecture

In this section we first present our overall interconnection architecture and related assumptions. We then describe the operational procedures related to the interconnection architecture, and finally the interconnection protocol we have selected.

3.2.1 Assumptions and overall architecture

Figure 3.2 depicts our overall interconnection architecture. Several applications and gateways exist in separate MANETs. Each application is distributed through several Application Component (AppC) nodes forming an overlay. Similarly, each gateway is distributed through several Gateway Component (GWC) nodes forming an overlay. AppCs and GWCs are software nodes and can be hosted together on a single physical device. Some AppCs in an application need to interact with some GWCs in a gateway.

To enable this interconnection we introduce a new node called an interconnector. An interconnector is responsible for interconnecting a cluster of AppCs and GWCs that have registered to it. We present how these clusters are dynamically formed and how they will be used to enable the interconnection. We follow the work presented in [63] for splitting and merging the clusters as well as for dealing with unexpected node leaving. Furthermore, to address the scalability requirement, the architecture tries to balance the load on the interconnectors; an aspect we will discuss in the operational procedure section.

An interconnector node does not belong to any of the application or gateway overlays. It uses a third-party protocol (here is called interconnection protocol) to interact with the overlay nodes. The underlying MANETs provide the communication infrastructure

between the interconnectors and the overlay nodes. This will fulfill our requirement for the independence of the interconnection architecture from any overlay architectures and middleware.

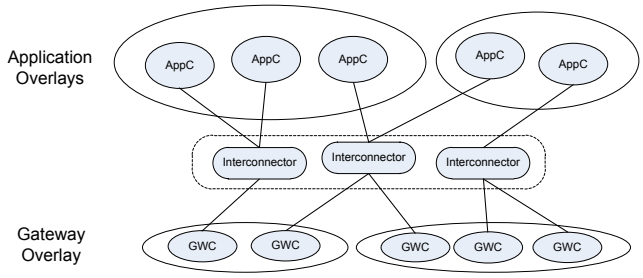

Figure 3.2. Overlay interconnection architecture

All of the interconnector nodes are joined to a predefined IP multicast group, which enables the overlay nodes to discover them. We suppose that the multicast address is known to all the nodes in the application and in the gateway overlays. For simplicity we assume that the GWC, the AppC and the interconnector nodes have static roles, i.e. nodes that will act these roles in the MANET have been preconfigured. It is clear that none of the entities defined in our architecture is permanently centralized, which satisfies our first general requirement.

In the following sections we describe the operational procedures as well as the interconnection protocol we use to realize the architecture.

3.2.2 Operational procedures

We divide the operational procedures into two groups, cluster management and intercommunication. Cluster management procedures describe how interconnectors are created and how AppCs and GWCs join interconnectors. Intercommunication procedures describe how AppCs and GWCs interact through interconnectors.

In the next section we use the term Overlay Component (OC) to refer to both AppC and GWC nodes. This shortens the description of procedures that are identical for AppCs and GWCs (e.g. joining and leaving).

3.2.2.1 Cluster management

a) Joining

- Overlay Component

  When an OC joins the network it first discovers and registers to an existing interconnector. The discovery is done by sending a multicast request to the pre-defined interconnectors' multicast address. If the OC receives responses from multiple interconnectors it will register to the interconnector with the least number of registered nodes. If the OC does not receive any response after a time threshold, it will initialize an interconnector on the same device and register to it.

- Interconnector

  When an interconnector node is initialized, it joins the interconnectors' multicast group. It listens to discovery messages sent by the OC nodes and returns responses. These responses should include the information that the OC nodes need to select the appropriate interconnector to register to.

b) Leaving

- Overlay Component

  When an OC node wants to leave, it informs the interconnector node it is registered to. The interconnector node then removes this node from its list of registered nodes.

- Interconnector

When an interconnector leaves the network, it should inform all of the OC nodes that have registered to it. These nodes then start the discovery procedure to find a new interconnector node and register to it.

3.2.2.2 Intercommunication

a) Information Exchange

We employ a publish/subscribe messaging pattern for information exchange between application and gateway nodes to fulfill our second specific requirement. The requests are triggered by AppC nodes. A request should be a subscription to an event managed by a gateway entity. Several notifications may be sent asynchronously to the application nodes, based on event occurrence. For example, if AppC nodes need to receive the temperature sensed by all sensors every t seconds, the gateway should provide an event called “give all temperatures” with the “period” parameter equal to t seconds.

b) GWC Node Discovery

We provide a gateway node discovery mechanism for AppC nodes to address our third specific requirement. An AppC node can discover a gateway node by sending a discovery message to the interconnector to which it is connected. The discovery message contains the required services or capabilities that the GWC node should provide. The interconnector node then forwards the discovery request to all registered GWC nodes and returns the responses accordingly. To reach all of the registered GWCs, the interconnector node forwards the message to the GWC nodes that are subscribed to it and also to the interconnectors’ multicasting address. Each receiving interconnector node then forwards the message to its connected GWCs.

3.2.3 Interconnection protocol

We have selected the Session Initiation Protocol (SIP) [64] as our interconnection protocol because it meets all our requirements. It is a light-weight, standard protocol that is easily extensible and widely deployed. It is interoperable with a variety of mobile devices. It is independent of P2P middleware and can carry the required information with an acceptable overhead.

We use a SIP User Agent in OC nodes to send and receive SIP messages, and use basic functions of SIP Registrar and SIP Proxy entities in interconnector nodes to register overlay nodes and to forward the SIP messages between overlay nodes.

For discovery purposes, we use the SIP multicasting method described in the SIP protocol. To discover an interconnector, the joining OC sends a SIP INFO message to the interconnectors' multicasting address and waits to receive the responses from all the available interconnector nodes. The discovery response is in the form of a SIP INFO message that contains the information to guide the OC node to choose the appropriate interconnector node.

An OC node then uses the SIP REGISTER method to register to an interconnector node and receive notifications related to node leaving. The SIP SUBSCRIBE/NOTIFY method is also used for subscribing to a gateway event and to receive response(s) as SIP NOTIFY messages.

3.2.4 Illustrative Scenario

Assume that in the emergency response scenario described in section 3.1.1, Alice, one of the medical staff, wants to receive periodic information about Bob's (one of the patients)

body temperature. Alice's end-user device runs an AppC node which should subscribe to an existing GWC node that provides notifications regarding Bob's body temperature.

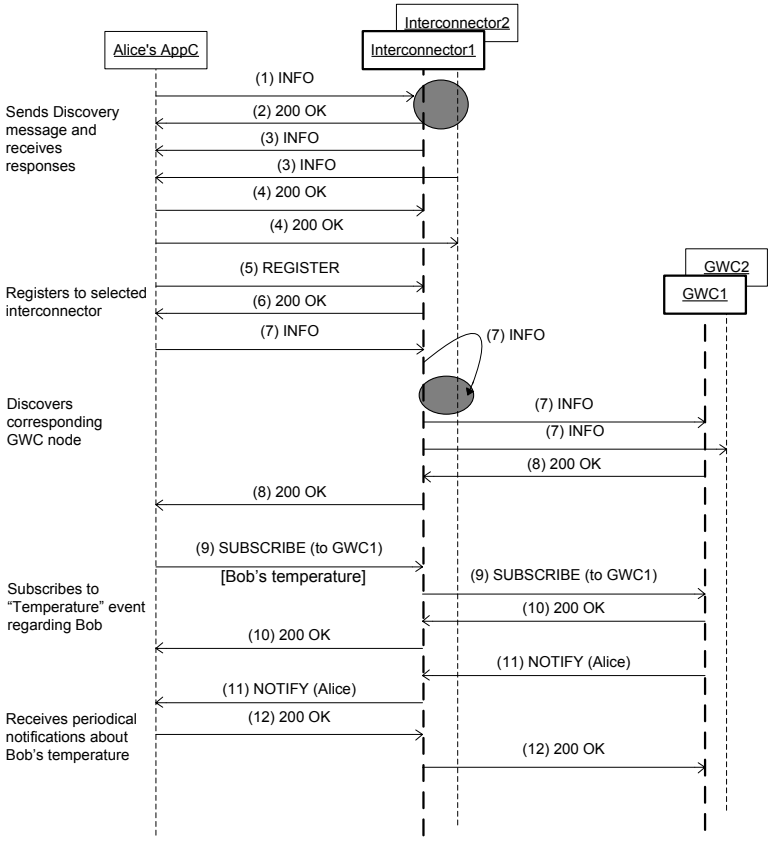

Figure 3.3. A subscription-notification scenario

Figure 3.3 depicts the message sequence required for this process. We assume that a gateway overlay already exists and that several interconnector nodes have already joined the network. When Alice switches her device on, the AppC joins the overlay and starts the discovery process (steps 1 to 4). It then registers to the selected interconnector, i.e. interconnector1 (steps 5 to 6).

Next, the AppC node needs to discover the GWC node that provides Bob's 'Temperature' event in the gateway overlay. For this purpose, the AppC node sends a SIP

INFO message to interconnector1, indicating the discovery parameters. Interconnector1 sends the discovery message to the GWC nodes registered to it (i.e. GWC1) and also to the interconnectors' multicasting address. Once the corresponding GWC node (i.e. GWC2) receives this request it will send back a SIP 200 OK that will be forwarded to the AppC node (steps 7 to 8). The AppC node then subscribes to the required event (steps 7 to 10). GWC2 will then be responsible for sending SIP notifications to the subscribed AppC node, which will be followed by a SIP 200OK message (steps 11 to 12).

3.3 Implementation

In this section, we present the software architecture of the interconnector node and a plug-in module which provides interconnection APIs that enable OC functionalities for the OC nodes. The plug-in module can be plugged into any node to make it run as an OC. We also present our prototype, including a performance evaluation.

3.3.1 Software architecture

3.3.1.1 Interconnector nodes

The software architecture of an interconnector node consists of three software modules and a SIP protocol stack, as illustrated in Figure 3.4. The self-organization module is responsible for managing interconnector joining and leaving. The SIP Registrar module acts as a simple SIP registrar server to receive SIP REGISTER messages and store the nodes' contact information. The SIP Proxy module, a simplified version of the SIP Proxy server, is used for forwarding request and response messages between overlays. Self-organization and SIP Proxy modules use the SIP Registrar to retrieve users' contact information.

Since the nodes in our architecture use SIP addresses, interconnector nodes need to map these addresses to actual network addresses. The SIP registrar module contains the mapping information for nodes that have already registered to it. When a unicast message is received by the SIP proxy, it asks the SIP registrar if it contains the mapping information for the receiver's SIP address. If that mapping exists, the SIP proxy will forward the message to the receiver. If the mapping does not exist, the proxy sends a SIP INFO message to the interconnector nodes' multicast address, asking for the mapping information. If any of the interconnector nodes has the mapping information, that node responds by sending a SIP INFO message containing the information.

Furthermore, to enable AppC nodes to discover GWC nodes, an AppC node sends a SIP INFO message containing the discovery criteria to the interconnector node. The interconnector will forward this message to all the GWC nodes registered to it and also to the multicast address assigned to the interconnector nodes. Other interconnectors forward the discovery message to their own registered GWC nodes. The corresponding GWC nodes will send back a SIP 200OK to inform the initial AppC node.

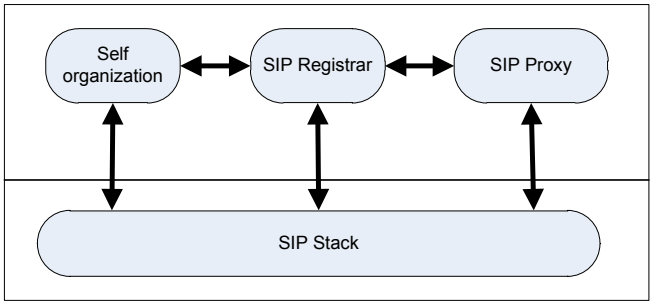

Figure 3.4. Software architecture of the interconnector node

3.3.1.2 Plug-in APIs and events

The plug-in module provides all the APIs and software events required for the operational procedures presented in our interconnection architecture.

It is worth to mention that certain software changes in the existing overlays are inevitable in order to enable the interconnection. However, the main issue is how to minimize these software changes. We believe that by providing the plug-in component, we achieve this goal for two reasons. First, there is no software changes required on the internal functionality of each overlay, i.e. the existing software works as it is for internal overlay operations. This is a huge advantage in contrast to porting the whole software to a new general overlay/middleware. Second, the plug-in hides the internal operations of our interconnection architecture (e.g. cluster management operations and subscriptions/notifications) from the existing software and provides simple APIs to be used by the software (join/leave, and send/receive). Therefore an existing overlay requires using these APIs only when it needs to interconnect with other overlays.

The list of APIs and events provided by the plug-in are as follows:

- **Join**:

  Called during node initialization, this API causes the plug-in module to start the interconnector node discovery procedure by sending a SIP INFO message to the interconnector nodes' multicast address.

- **Leave**:

This API is called upon to inform the plug-in about a voluntary node departure. The plug-in module sends a SIP REGISTER, with the "expire" value equal to 0, to the current interconnector node that this node is subscribed to.

- **Send_Message**:

This API sends a request or a response message to the interconnector node that this node is subscribed to. That interconnector node should then map the SIP address provided in the SIP message header to the destination node's IP address and forward the message to the destination node.

- **Receive_Message**:

This is an event which provides a mechanism for the AppC or the GWC node to get the request or response messages that are received by the plug-in module.

3.3.2 Prototype and experimental setup

As a proof of concept we implemented a prototype based on a simple evaluation scenario. As indicated in our requirements, we validated that the network load and request-response delay imposed by our architecture is acceptable in MANETs.

The implemented prototype includes an application overlay based on JXTA [39] (an open source P2P protocol specification and implementation developed by Sun Microsystems) and a gateway overlay based on Open Chord [40] (an open source implementation of Chord protocol [30]). JXTA and Open Chord have already been successfully used on MANETs in various projects. We used JAIN SIP as the SIP stack in the plug-in module and in the interconnector nodes. The plug-in APIs presented above were implemented and provided to AppC and GWC nodes. To avoid the overhead implied by a complete

SIP registrar module, we implemented a simple SIP registrar that receives SIP REGISTER messages and stores nodes' contact information.

3.3.2.1 Evaluation Scenarios

We implemented the scenario introduced in section 3.2.4. Each end-user device ran the same AppC node as Alice's device. Each AppC was subscribed to one of the GWC nodes to receive periodical notifications about the patient's body temperature every 5 seconds.

To include the effect of mobility in our measurements we simulated network churn. Network churn is the volume and pattern of the nodes moving in and out the network over time. Network churn depends on the physical network condition and the devices that are used. To simulate network churn, we developed a probability-based churn generator with two configurable values, P_c (connecting probability) and P_d (disconnecting probability). Each node periodically calculates a probability P that indicates if the node should join or leave the network. If the node is connected to the network and $P < P_d$ then the node leaves the network. If the node is not connected to the network and $P_d < P < P_d + P_c$, the node joins the network. If $P_c < P$, no action is taken. We used different values for P_c (0.25, 0.5 and 1) and P_d (0, 0.05 and 0.1) to evaluate our prototype on ad-hoc networks with high and low network churn. We ran the prototype for 20 minutes on each setup.

3.3.2.2 Experimental setup

A MANET environment was formed consisting of five devices: four laptops running Windows XP and a Samsung Galaxy S Captivate smartphone running Android OS 2.1. Two interconnector nodes were run on two separate devices. Each of the other devices ran an AppC node and a GWC node side by side. Since the AppC and the GWC

components have no direct communication, running both AppC and GWC nodes on the same device would not affect our performance evaluation.

3.3.3 Performance Evaluation

3.3.3.1 Metrics

We defined two metrics to evaluate the performance of our prototype: network load and delay. Network load indicates the total number of bytes sent and received by a node for the purpose of interconnection, and the messages exchanged between overlays. The delay is measured as the time difference between sending a request message and receiving an immediate response.

3.3.3.2 Performance Results

We measured the performance metrics on our prototype based on the evaluation scenario. Figure 3.5 (a-e) shows the comparison of the network load on four nodes (i.e. interconnector1, interconnector2, AppC1 and GWC1) with five setups, so that P_c and P_d were configured with (0.5, 0.05), (0.25, 0.1), (0.25, 0.05), (0.5, 0.1), and (1, 0) accordingly.

Figure 3.5 (f) shows the comparison of the network load on the interconnector node-1 (IC1) in different configurations. When $P_c = 1$ and $P_d = 0$, no nodes will leave the network, so the network load increases in a straight linear fashion. The network load always converges to higher values in other configurations, due to the operational procedures such as joining, leaving and message transfer. Node joining and leaving procedures generate the highest network overhead compared to other operations. We can extract this overhead in the IC nodes by comparing the load metrics measured from the

configuration with no node leave policy (i.e. $P_c = 1$, $P_d = 0$) and from other configurations. Figure 3.5 (f) indicates that the total network load after 20 minutes of execution is around 800kB, which is acceptable in MANETs when considering the values selected for leaving and joining probabilities.

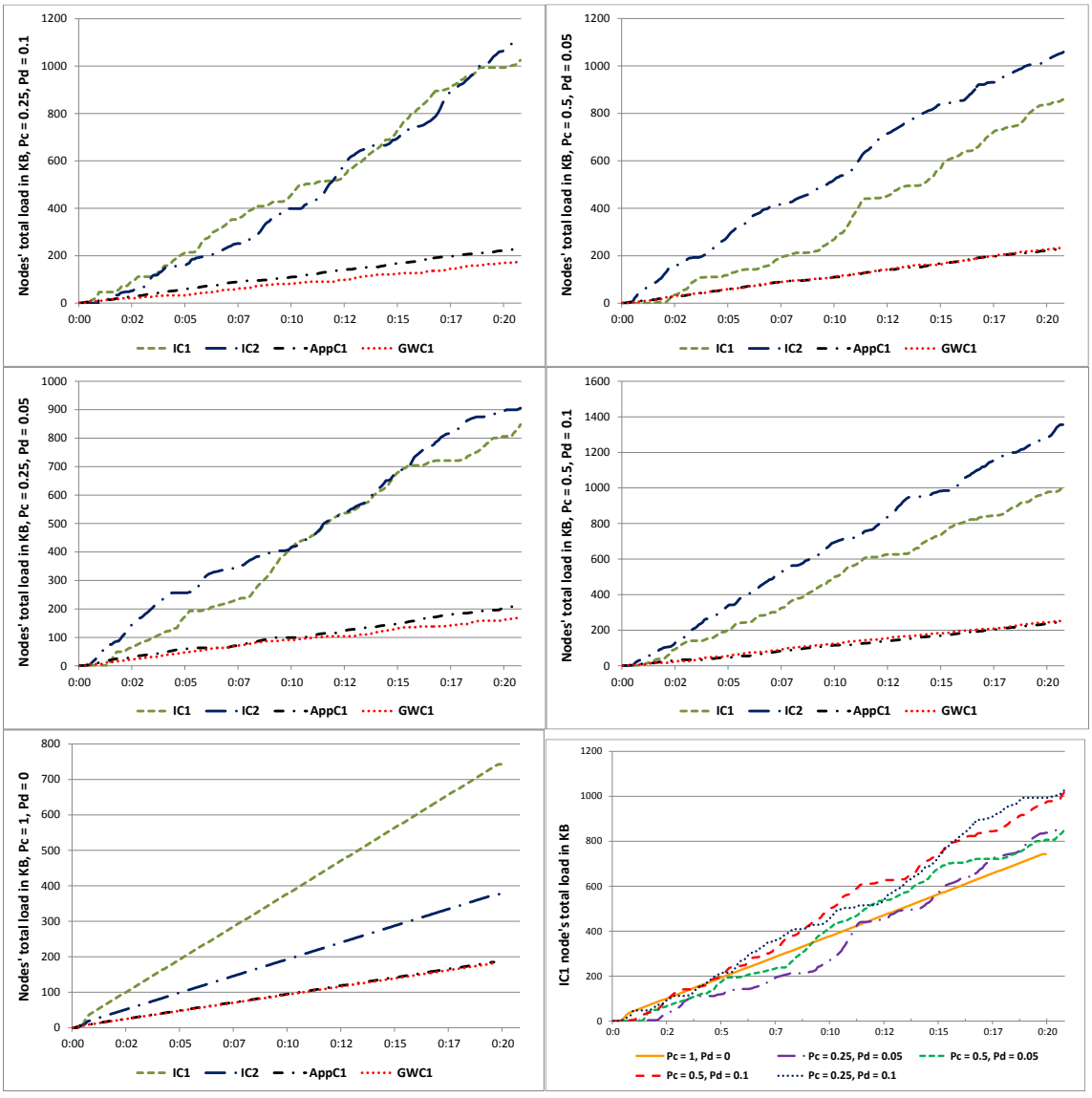

Figure 3.5 (a-e). Comparison of the total network load of interconnector (IC1 and IC2); AppC1 and GWC1 nodes measured in different configurations for churn generation. Figure 3.5 (f). Comparison of total network load on IC1 node, measured in different co

Figure 3.6 illustrates the delay metric measured for the prototype with the same configurations. The average delay between sending a request and receiving a response between overlays is around 1500 milliseconds, which is an acceptable delay for applications over MANETs.

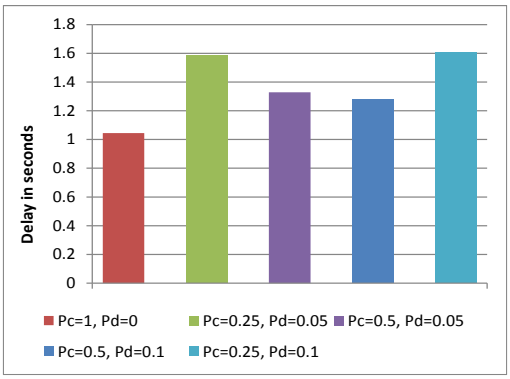

Figure 3.6. Comparison of delay measured in different configurations for churn generation

In summary, our results indicate that the network load is balanced among the interconnector nodes and it has linear growth over time. In addition, the network load and delay is acceptable for applications in MANET environment.

3.4 Meeting the Requirements

a) General Requirements

As we mentioned earlier, none of the entities defined in our architecture is permanently centralized, which satisfies our first general requirement. Our second requirement is also met because our architecture uses an application layer protocol, i.e. SIP, for its operation and it does not rely on lower layer protocols.

Our third requirement on scalability is also met because the introduced interconnector nodes are self-organized, i.e. they could be initiated or terminated by an OC. Therefore there is no single point of failure issue in the architecture.

Although we do not address mobility (the fourth requirement) directly in our architecture, our performance evaluation, shows that our architecture enables overlay interconnection in MANETs with high churn rate (i.e. nodes joining and leaving). High churn rate is a major issue caused by mobility.

Finally, our fifth requirement on supporting resource-constrained devices is not addressed in our architecture. However, since interconnector nodes carry the main overhead of our architecture, a potential approach is to restrict the devices that are involved in our operational procedures based on their capability (e.g. computation power or battery level). This ensures that the interconnection operations are carried out by resourceful devices.

b) Requirements on Overlay Interconnection Mechanism

Since our proposed architecture is not based on a specific overlay or middleware, our first requirement on interconnection architecture is met. The proposed operational procedures for intercommunication are devised so that they address our second and third requirement.

As we discussed earlier, SIP is a standard, lightweight protocol that is shown to be suitable for distributed architectures. Furthermore, SIP provides publish/subscribe message exchange and enables node discovery using SIP multicasting. Therefore it meets all our requirements on interconnection protocol.

3.5 Conclusion

In this chapter we proposed a decentralized architecture to interconnect P2P end-user application overlays and P2P gateway overlays running on top of MANETs. The

architecture provides a publish/subscribe information exchange model and gateway node discovery. We introduced interconnector nodes, which are responsible for message relay and address mapping between application and gateway overlay nodes. The proposed solution is independent of the overlay types, and it is scalable in terms of the number of overlays and the number of nodes in each overlay. We used SIP as a lightweight and standard protocol to provide interconnection mechanisms and as a subscription/notification service for event-based data delivery.

We implemented all the features of our architecture in a prototype application that assesses the publish/subscribe messaging between gateway and application overlays. We implemented an end-user application based on JXTA, and our gateway based on Open Chord middleware. We also used JAIN SIP to provide SIP stack services. We introduced performance metrics to measure the network overhead of our proposed architecture. Our performance results show that the network load and delay imposed by the interconnection architecture is acceptable for applications in MANETs with different churn rates.

Chapter 4: Service Discovery Architecture

Let us imagine that a swarm of robots with different capabilities from different organizations are employed to search a large urban disaster zone caused by a natural disaster or terrorist attack. The robots are directed to locate victims and provide basic rescue services (e.g. monitoring patient's health status, notifying medical staff or providing survival kits). These robots should be able to interact with each other, with humans and with sensors/actuators in the area through a MANET that is deployed over physical wireless networks (e.g. WLANs [20]).

In this dynamic environment, robots need to discover the available services and communicate with them to both increase their own awareness and to coordinate search and rescue tasks. For instance, a robot may need to look for a nearby robot with specific abilities such as internet connection or picture-taking. Robots may need to discover the sensors and actuators of nearby buildings to accurately report the status of those buildings or to activate actuators, which may be required to prevent further disasters. A pervasive service discovery mechanism is necessary to realize these scenarios.

Service discovery protocols (SDPs) make two major operations possible, service browsing and service name resolution. Service browsing lists all the service instances of a given service type, while service name resolution provides service contact information, such as a server's name or address, transport protocol, port, etc. This chapter proposes service discovery architecture for DRSs.

In this chapter, next section presents an overview of the existing service discovery architectures for MANETs and discusses their shortcomings in terms of DRSs. This is followed by the proposed architecture. The third section is devoted to its evaluation and validation. We conclude in the last section.

4.1 An Overview of Service Discovery Architectures for MANETs and their Shortcomings for DRSs

SDPs for MANETs should meet that general requirements discussed in Chapter 1, in order to be of practical use in DRSs. These SDPs can be categorized into two groups, directory-less approaches and distributed directory-based approaches. Directory-less approaches do not provide any directory structure to store service information, while distributed directory-based approaches provide a distributed directory structure that hosts the service information.

We review the directory-less approaches and the distributed directory-based approaches based on the general requirements. In this section we use the term provider for a node that wants its service information to be discovered by other nodes.

4.1.1 Directory-less SDPs for MANETs

In directory-less approaches a node searches the whole network to discover a requested service. In the simplest case the node floods a request to the network and each provider with the requested service sends a response back. This approach severely limits the scalability of an SDP because of the high overhead and packet loss caused by flood-based communications [65]. Therefore, the directory-less SDPs based on this approach do not comply with our third requirement.

As an example of flood-based directory-less SDPs, IETF has proposed Multicast DNS (mDNS) [12] that enables DNS-like operations on a local link. A local link is any group of hosts that are directly reachable without using a router or a gateway. Queries in mDNS are sent to a specific multicast IP address. Every node with DNS records is expected to operate as an mDNS server and to respond to multicast queries. IETF recommends using mDNS with another IETF proposal, DNS-based Service Discovery (DNS-SD) [66], to enable service discovery in ad hoc networks such as MANETs. DNS-SD defines the naming rules and the structure of DNS records for the purpose of service discovery.

In order to overcome the scalability issues, recent directory-less approaches have adopted two major ideas: proactiveness and scope limitation. In proactive SDPs [67], service providers advertise their service information and receivers cache that information for subsequent look ups. Scope limitation is a technique to optimize flooding with a forwarding scheme. The forwarding scheme can be as simple as limiting the number of hops in flooding to n, or it can be based on a selective approach [67]. We briefly discuss recent works and their shortcomings based on our general requirements. These approaches reduce the number of flooding requests in the network, especially when there are considerably more service requestors than service providers.

Advertisement-based Search Algorithm for Unstructured Peer-to-Peer (P2P) Systems (ASAP) [67] is a proactive directory-less approach for searching unstructured P2P networks. In ASAP, nodes periodically exchange their indexes, which are stored as Bloom filters [68]. In case of service discovery these indexes should be obtained from the service descriptions. For service discovery, a node first checks its local registr, if any match is found, it will know with a certain probability (based on the type of Bloom filter)

where the service is located. The node will then ask the service provider node to confirm the service information. The confirmation is required because of the false positive probability of Bloom filters. If the required service is not found in the local registry, the node will forward the request to its neighbors. ASAP is shown to have a much lower bandwidth consumption compared to flooding [67], and therefore it has addressed the scalability issue. However, in ASAP every node is responsible for caching the service information of the other nodes and must always be prepared to response to search requests. This property makes ASAP unsuitable for functioning on resource-constrained devices, our fifth requirement.

Mist-protocol [69] is a directory-less SDP designed for large, highly mobile networks. Mist has adopted proactive mechanisms similar to those of ASAP, but it has proposed a subscription-based forwarding scheme for propagating service information. Nodes in Mist subscribe to their neighbors to receive only service information that is in their interest. This forwarding scheme limits the propagation scope of service information, thus leading to lower bandwidth consumption. Furthermore, unlike ASAP, if a node does not find a requested service in its local registry, it will not forward the request to its neighbors (there is no fallback mechanism). The results provided in [69] show that Mist is scalable and it is suitable in highly mobile environment. However, similar to ASAP, every node in Mist is involved in the subscription and caching functions, which does not comply with our fifth requirement. It is worth mentioning that since Mist does not provide any fallback mechanism, and it is possible that an existing service could not be found during the service discovery process.

4.1.2 Distributed Directory-based SDPs for MANET

In distributed directory-based approaches, a set of nodes form a distributed directory which stores the providers' service information and responds to the discovery requests. Distributed directory-based approaches have been deemed suitable for medium and large-scale MANETs (our third requirement) and capable of supporting medium mobility (our fourth requirement) [65]. This makes them ideal for disaster response scenarios. Unfortunately, none of the existing SDPs meets all the requirements we have presented.

Some distributed directory-based SDPs enable global discovery while others do not. An SDP enables global discovery when it looks up requested service information in the whole network. SDPs that do not enable global discovery only search within a subset of the network. With the lack of global discovery, a node may never be able to discover a requested service. This limitation makes these SDPs unsuitable for emergency situations such as disaster response.

Jini [43] is an example of an SDP that does not enable global discovery. In Jini, each provider is responsible for publishing its service information to the directory nodes and there is no interaction among directory nodes. When a directory node receives a service discovery request, it only queries its own local storage.

We focus on SDPs that enable global discovery in the rest of this sub-section. Some of these SDPs rely on cross-layer approaches while others rely on application layer approaches. The former generally use the information provided by lower network layers (e.g. the routing layer) to optimize their service discovery or directory management

operations, while the latter rely solely on information and operations that are accessible in the application layer.

None of the SDPs that rely on cross-layer approaches meets our second requirement (i.e. being independent of lower layers). An example proposed by [70] aims to address issues such as scalability and mobility in MANETs by using a restricted flooding approach and ensuring that any node in the network has a corresponding directory node in its H hops vicinity. It also enables global discovery by allowing a directory node to forward a service discovery request that cannot be satisfied locally to other directory nodes, one of which will most likely have cached the requested information. It uses the routing protocol functionalities to provide limited broadcasting.

Another example is proposed in [71], in which the MANET nodes are grouped into different clusters. Each cluster has one cluster head (CH) and several backbone nodes (BBs) acting as service directory nodes. Global discovery is enabled by providing the communication between the CHs. To adapt to node mobility, a cluster weight metric is introduced based on the stability of the members of a cluster and the QoS parameters of the services in the cluster. The stability metric requires information from lower layer protocols (e.g. connectivity degrees and QoS parameters).

There are a few distributed directory-based proposals that function solely in the application layer. One example is the ontology-based clustering approach adopted in [72]. At the bottom level, clusters of devices are formed based on the similarity of the service information they provide. The next layer of clustering consists of groups of respective lower layer clusters, based on a more general semantic in the ontology tree.

Since that paper provides no performance analysis it is not clear how well the proposed approach deals with issues such as scalability and mobility. In fact, due to the complexity of forming multi-layer semantic clusters we doubt that the approach meets our requirement of scalability and mobility support.

SANDMAN [73] is another cluster-based SDP, and is focused on energy efficiency and discovery delay. Cluster nodes (CNs) register the service information in cluster head (CH) nodes and then go to sleep. They will wake up periodically to provide services, and if there is no request they will go into sleep mode again. Since SANDMAN does not provide any clustering algorithm it is not clear how it scales in terms of the number of nodes. It is also not clear how it deals with MANET node mobility.

Finally, it is important to point out that P2P middleware could be used to enable service discovery through their resource advertisement and search mechanisms. However, it has been observed that since existing P2P middleware such as JXTA [39] are designed for infrastructure networks, they do not address the bandwidth consumption and mobility issues in MANETs [74]. Agile Computing Middleware [74] is a P2P middleware specifically developed to address the requirements of tactical edge networks. Many of these requirements are similar to the requirements of DRSs. However, there are two issues when using the middleware in DRSs. First, the viability of the middleware in resource-constrained device environments is not discussed. Second, unlike in tactical edge networks, it is not possible to "force" all the different response teams to use a single P2P middleware. Therefore, using ubiquitous DNS for service discovery has a compelling advantage.

4.2 A distributed directory-based SDP Architecture

We propose a distributed directory-based SDP based on a clustering approach. Our contributions are twofold. The first contribution is an SDP that meets all the requirements of DRSs. To the best of our knowledge, none of the existing SDPs meets all of these requirements.

The second contribution is the set of semantic extensions we have made to the standard DNS protocol [75]. They enable the formation of service directories and the dynamic adaptation of these directories depending on network conditions. Using the DNS protocol for service discovery has several advantages over other SDPs in DRSs. First of all, DNS is a simple, mature and standard technology which is required by most network-based solutions. Second, service resolution in most SDPs returns the service address back as a host name, which then should be resolved to an actual network address using a name system (e.g. DNS). DNS provides both of these services using the same underlying structure.

There are existing proposals that have extended DNS to enable service discovery. However none of them can enable the formation and dynamic adaptation of service directories. For instance, in [76], DNS SVR is extended (as proposed in DNS-SD) to enable the discovery of multiple service directories organized by a company, but the issue of dynamic formation and adaptation of the directories is not discussed.

4.2.1 Architectural Principles

The first principle in our architecture is that we have two types of nodes, peers and super-peers. We group these nodes into clusters so that each cluster contains a super-peer and zero or more peers that are connected to the super-peer (Figure 4.1).

The second principle is that peers and super-peers may act as service providers or service requestors, but only super-peers act as service directories for their own clusters. Therefore, each super-peer stores the service information of all the peers that have registered to it, and it is responsible for handling the service requests of those peers. Furthermore, all super-peers must join the pre-defined multicast group to be discoverable by peers.

The third principle is that every node initially acts as a peer, and only peers with a capability higher than a pre-defined threshold (C) may change their type to super-peer. We call nodes with a capability higher than C capable peers and we call all the others incapable peers.

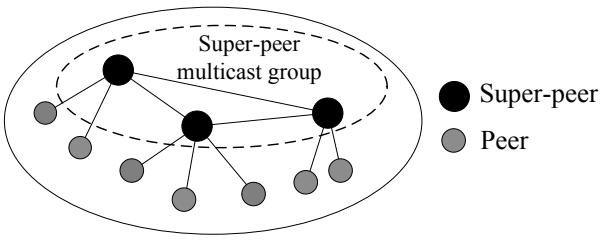

Figure 4.1. Overall Architecture

Capable peers and super-peers act as limited DNS servers. Each of these nodes hosts its capability as a DNS record named to the known multicast address. Furthermore, all the nodes, including the incapable peers, act as DNS clients to resolve DNS queries and register their service information when needed. We followed the IETF DNS-SD [66] recommendations for storing service information as DNS records.

DNS-SD uses three types of DNS resource records (RRs), SVR, PTR and TXT to store service information. An SVR record stores mapping information between a service instance name and the address where the service can be accessed. PTR records enable service browsing and TXT records store additional information that belongs to a service.

4.2.2 Operational Procedures

In this section we describe the service discovery and cluster management operations and how they are implemented based on DNS standard messages.

We use two types of DNS messages, DNS query and DNS update. When the opcode field in a DNS message is assigned to 0, it represents a standard DNS query message, while when it is assigned to 5 it represent a DNS update message. DNS query is used to retrieve DNS records and DNS update is used to send a request for updating the DNS records of a node.

To enable cluster management operations we extended the semantic of DNS TXT RR to store three new attributes: “node_type”, “capability”, and “number_of_registered_peers”. The “node_type” attribute can have a “peer” or “super-peer” value based on the type of node in the cluster. The value of the “capability” attribute indicates the current capability of the node that is dynamically updated. The “number_of_registered_peers” attribute is only hosted by super-peers and indicates the number of peers registered to a particular super-peer. Every capable peer and each super-peer should host the discussed TXT RR named to the pre-defined multicast address. We will discuss how this TXT RR is used in cluster management operations.

4.2.2.1 Service Discovery / Service Browsing

Figure 4.2 depicts the service discovery procedure; including both the service browsing and service name resolution operations. To perform service discovery/service browsing, a peer sends a DNS Query request to its registered super-peer looking for SVR/PTR RRs, and waits for the response(s). When the super-peer receives the DNS Query request, it responds directly to the requesting peer if the request can be satisfied from its own directory. Otherwise it forwards the request to the super-peer multicast group, and then forwards the responses it receives back to the requestor peer.

4.2.2.2 Cluster Management

a) Peer/Super-peer joining and peer leaving

Figure 4.3 depicts the operational procedure for peer/super-peer joining and peer leaving operations. A peer starts the joining process by multicasting a DNS Query request to the super-peer multicast group to discover the existing services and their capabilities (Capability Discovery request). A Capability Discovery request looks for TXT RRs that are named to the known multicast address. If no response is received after a time period Tt (round-trip in the network), the peer restarts the capability discovery procedure. If the peer is a capable peer and no response is received after repeating the capability discovery procedure three times, the peer joins the network as a super-peer by changing its node type, joining the super-peer multicast group and registering its service information in its own directory. If it is not a capable peer, it continues the capability discovery procedure until it receives a response.

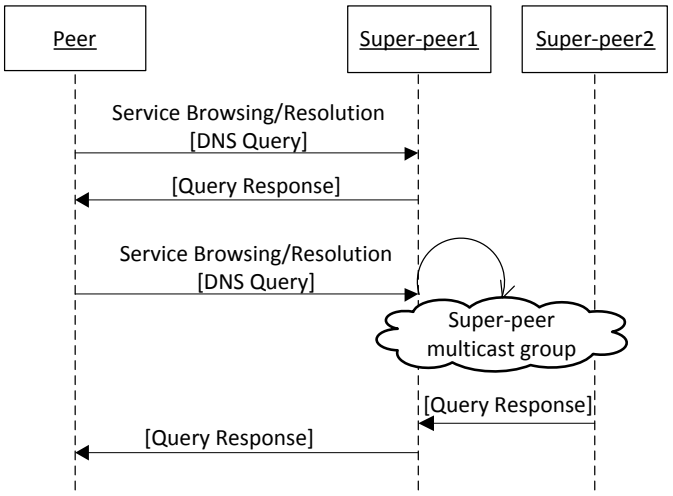

Figure 4.2. Service Discovery

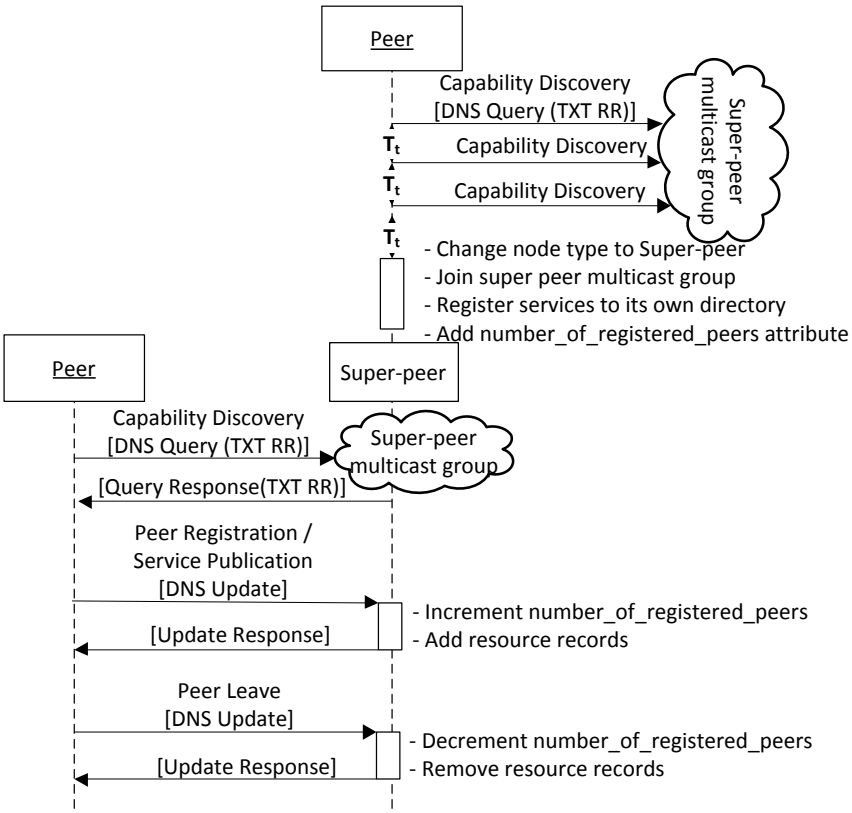

Figure 4.3. Peer/Super-peer joining and peer leaving

After the first super-peer has joined, other peers will be able to discover the existing super-peers, as illustrated in Figure 4.3. The joining peers will select the super-peer with

the highest value of capability attributes in the TXT records. To register its service information, the joining peer sends a DNS Update request including its DNS A and TXT records and all its service information records. When a peer wants to leave the network it sends a DNS Update request to delete all the registered records associated with that peer. The corresponding super-peer removes those records and decreases the number of its registered peers.

b) Super-peer merge

Figure 4.4 depicts the procedure for merging two super-peers. When the number of registered nodes of a super-peer becomes less than a pre-defined threshold (M), that super-peer multicasts a DNS Query request to the super-peer multicast group and collects responses from the available super-peers. It then selects the super-peer with the highest capability, with the optional constraint that the size of the merged cluster would not exceed the split threshold. Next, the super-peer sends a DNS Update message that includes all the records that belong to its registered peers and removes them from its own directory. Finally, it removes the "number_of_registered_peers" attribute from its TXT RR, leaves the super-peer multicast group and changes its node type to peer.

c) Super-peer splitting and leaving

Since super-peer splitting and leaving procedures are similar we present them in the same figure (Figure 4.5). When the number of registered peers of a super-peer exceeds a pre-defined threshold (S), or when a super-peer decides to leave, the super-peer first unicasts a DNS Query request to its registered capable peers asking for their TXT records. Upon receiving the responses, the super-peer selects the peer with the highest capability and sends a DNS Update that includes a modified TXT RR with the "node_type" attribute set

to “super-peer”. Upon receiving this request the selected peer will perform the super-peer joining tasks described in a). Finally, the splitting/leaving super-peer will transfer a portion/all of its hosted RRs to the new super-peer using a DNS Update request and remove these RRs from its own directory. The leaving super-peer removes its “number_of_registered_peers” attribute, changes its node type to “peer” and leaves the super-peer multicast group.

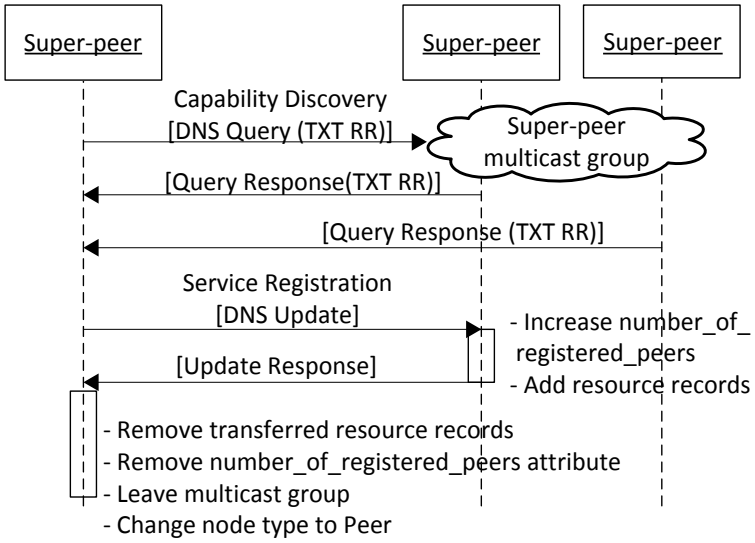

Figure 4.4. Super-peer merging

When a super-peer merges, splits or leaves (voluntarily or involuntarily), its registered peers are not notified until they try to update their service information or start a service discovery operation. The peer’s service information may have been transferred to another super-peer through a merging, splitting or leaving procedure. Therefore, when a peer contacts its super-peer and does not receive any response, the peer first sends a DNS Query request to the super-peer multicast group looking for its own A RR. If it receives a response it will use the new super-peer address for further operations, otherwise it will restart the peer joining procedure.

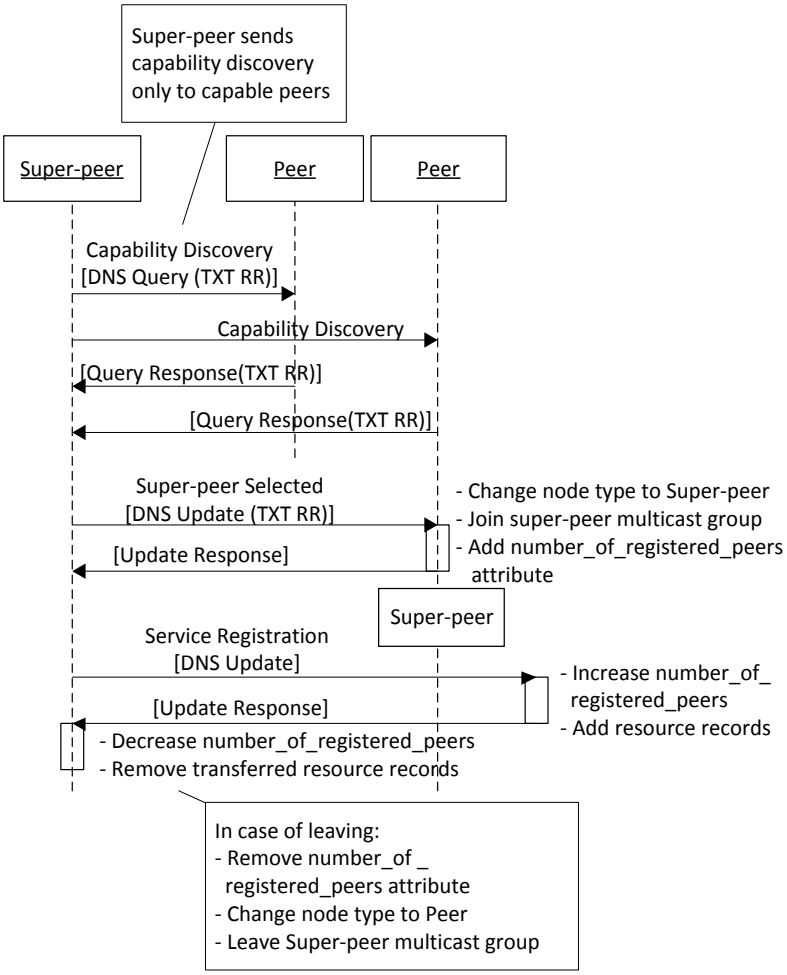

Figure 4.5. Super-peer splitting and leaving

In the worst-case scenario, the involuntary departure of super-peer nodes may cause the unavailability of some service information for a short time, or it may even cause multiple copies of a peer's RRs to be (temporarily) stored in two super-peers. However, since all RRs have a TTL attribute, these issues will phase out after a certain time has elapsed.

4.2.3 Adjustment of Thresholds

There are three thresholds in our architecture. C defines the minimum capability required for a node to be a capable node. M defines the minimum size of a cluster, i.e. when the size of a cluster is less than M, the super-peer of that cluster should start the merging

operation. Finally, S defines the maximum size of a cluster, i.e. when the number of nodes in a cluster is more than S, the super-peer of that cluster initiates a split operation. Based on the devised operational procedures, in general we expect that increasing S or increasing M would lead to fewer clusters in our architecture. This is because that by increasing S there would be less splitting and by increasing M clusters have to merge more frequently. Therefore, S and M are expected to indicate the level of distribution in our architecture. Although dynamic methods could be used to adjust these thresholds, for simplicity we assume that these values are set based on offline simulations.

4.3 Performance Evaluation

4.3.1 Simulation

To evaluate the performance of our architecture, we used simulation based on OPNET, a software tool for network modeling and simulation. The validation was done using OPNET V.12.0, a software tool for modeling and simulating communication networks and distributed systems [77]. OPNET provides a comprehensive and modular environment for user development, based on Finite State Machines (FSMs). The supported programming language is Proto-C, a combination of C, C++ and OPNET Event Simulation APIs.

We implemented our proposed architecture (here called CDNS) as well as the IETF service discovery proposal based on mDNS (here called mDNS). We compared the performance of the two approaches. We extended a built-in OPNET model for MANET nodes, in which each node uses an 802.11b wireless LAN, IP for the network layer with AODV as the routing protocol, and UDP for the transport layer. We further used the

disaster area mobility model proposed in [78] to simulate the node mobility. More details about the network and node models we used for this simulation is presented as an appendix.

4.3.1.1 Evaluation Scenario

For simulating the disaster area and node mobility we used the evaluation scenario proposed in [78]. In this scenario 150 nodes are distributed in a disaster area of size 350m x 200m. The distribution of nodes is based on the sub areas defined in the disaster area, such as incident location, ambulance parking point, etc. In order to be able to evaluate our architecture with a different number of nodes in the same scenario, we uniformly increased/decreased the number of nodes in all sub areas. We used BonnMotion [79], a mobility scenario generation tool, to generate the location of each node in intervals of 1 second. We implemented a mobility module in OPNET to set the location of nodes based on the generated mobility information. We evaluated our architecture with four different number of nodes: 50, 100, 150, and 200. It is worth noting that we evaluated our scenario with other values for the number of nodes but since the trend of the results remained the same, we only present our results for these four values.

The nodes have sequential identifiers (ID) (i.e. 1, 2, … , N). Each node hosts five unique service information records based on its ID. For instance, a node with ID = 0 hosts services with IDs 0 to 4, and a node with ID = 1 hosts services with IDs 5 to 9, etc. Each node periodically performs a service discovery operation every 10 to 20 seconds and requests a particular service information record hosted by one of the nodes in the network. We simulated each scenario 10 times with different random seed values, for 30 minutes each time.

For the CDNS, every node in the network picks a random value greater than 0 as its capability, and C is set to 0 in all the scenarios. The reason for setting C to 0 is to ensure a fair comparison between the CDNS and the mDNS. If C is set to a higher value, there would be a set of incapable peer nodes which only contribute to service discovery operations, and thus CDNS generates less overhead. Furthermore, M is set to 5 and S is set to 20 in all the scenarios. These two values are selected based on our knowledge about the number of nodes in different scenarios which covers a range of network sizes in our motivation scenario.

4.3.1.2 Performance Metrics

We defined three metrics to compare the performance of the two approaches: network load, delay and discovery success rate. We also defined three metrics to evaluate our proposal: cluster management load, peer load and super-peer load.

Network load measures the average amount of received traffic in the wireless cards of all the nodes at each second. We selected the total received traffic to measure the network load, as both approaches use multicasting. If we calculated the traffic sent, we could not consider the multiple recipients of each multicast message and therefore that figure would not present an accurate evaluation of the network traffic.

Delay measures the average end-to-end delay for all service discovery operations, calculated by the difference between the time a service discovery request is sent and when the first response is received. This metric compares the responsiveness of each simulated approach.

The discovery success rate measures the percentage of successful service discovery requests versus the total number of service discovery attempts. This metric indicates the average likelihood of success for each service discovery operation.

The cluster management load ratio measures what percentage of the traffic received in the application layer per second belongs to our cluster management operations. Peer load measures the amount of traffic that is received by the wireless card of a node when it is acting as a peer, while super-peer load measures the amount of traffic received by the wireless card of a node while it is acting as a super-peer.

4.3.1.3 Performance Results

Figure 4.6 (a) shows the comparison of the network load metric measured for the mDNS and the CDNS approaches for networks with 50 nodes and 100 nodes, and Figure 4.6 (b) shows the same comparison for networks with 150 and 200 nodes. As expected, the network load measured for the mDNS approach is higher than for the CDNS approach in all cases. We believe this is because in the mDNS protocol all the requests are broadcast to the network, while in CDNS a portion of requests are being responded to directly by super-peers.

Figure 4.6 (c) depicts the delay metric measured for networks with different sizes. It shows that the average end-to-end delay for service discovery operations in the mDNS approach is slightly less than with the CDNS approach. As mentioned above, when a request cannot be satisfied by a super-peer in CDNS, it will be multicast to the other super-peers. We believe that the higher delay in CDNS is due to these two-step operations, which do not occur in mDNS. However, the differences between the delays

provided by the two approaches are negligible (less than 30ms). Furthermore, it is compensated by a higher likelihood of successful service discovery, as presented below, which is the main objective of an SDP.

Figure 4.6 (d) shows the average likelihood of a successful service discovery operation for each method. This metric helps to evaluate the scalability of each method in terms of the number of nodes in the network. The CDNS performs much better than the mDNS, especially in larger networks. We believe the main reason is that the higher network load generated by mDNS results in more interference in the network and therefore less likelihood of the successful delivery of service discovery requests or responses.

Figure 4.6 (e) shows what percentage of the traffic received in the application layer is imposed by our cluster management operations. The figure indicates that the cluster management load forms a small portion of the total traffic in the application layer. We also notice that as the size of a network grows, the cluster management operations form a higher percentage of the total traffic, which we attribute to the higher density of nodes and more frequent changes in the clusters' structure.

Finally, Figure 4.6 (f) compares the peer load and super-peer load metrics. Since nodes in CDNS can change their type, we selected nodes that were peer/super-peer for most of the simulation time to measure peer load/super-peer load. We noticed that if we measure the same metrics in the application layer (instead of the wireless card), there would be a much greater difference between the traffic received by peers and super-peers. This difference is of course because super-peers, as directory nodes, are involved in service

discovery as well as cluster management operations. Therefore, Figure 4.6 (f) indicates that our requirement to function on resource-constrained devices is met by our approach.

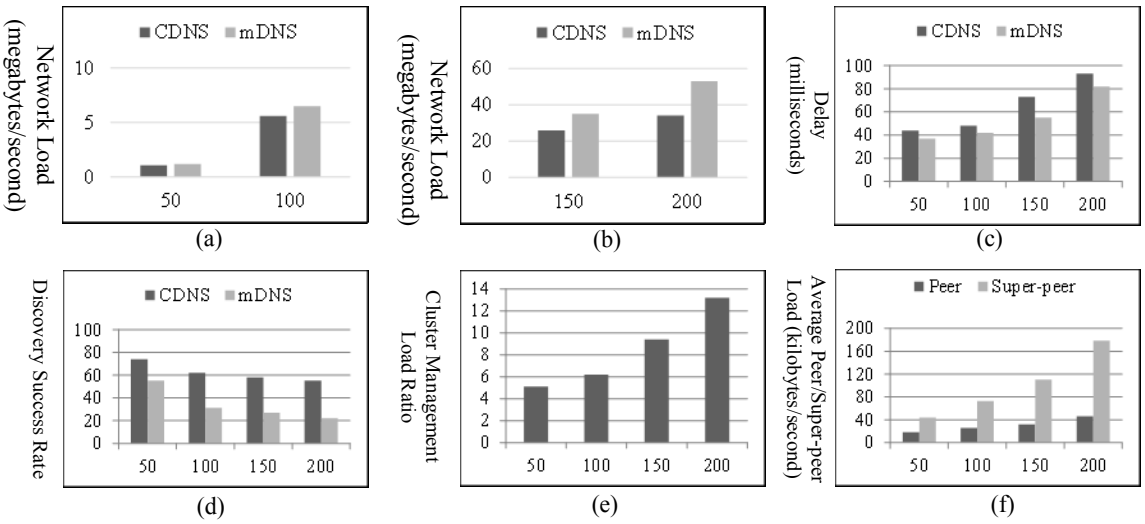

Figure 4.6. Measurements of our performance metrics (vertical axis) in networks with different numbers of nodes (horizontal axis). Labels are as follows: (a) and (b) Network load metric, (c) Delay metric, (d) Discovery success rate metric, (e) cluster management load ratio (f) Peer load vs. super-peer load.

To summarize, our performance results show that CDNS is more scalable in terms of the number of nodes in the network, because it generates less traffic and a higher service discovery success rate. Furthermore, the difference between the service discovery delays in the two approaches is around 30 milliseconds for the worst case, which is negligible. The evaluation of CDNS reveals two facts; first, our cluster management operations do not impose a huge overhead in the network. Second, the network load on a peer is about half of the network load on a super-peer, on average. Since we have used the capability of nodes for super-peer selection, this ensures that our architecture can enable the participation of less capable nodes (e.g. resource-constrained devices).

4.3.2 Prototype Implementation

In addition to the presented simulations, we implemented a prototype of our proposed architecture to show its practicality in the real MANET environment. In this section we present the software architecture for the prototype and our performance evaluation based on that.

4.3.2.1 Software Architecture

The software architecture for all the nodes is shown in Figure 4.7. The Joining/Leaving Management, Service Discovery Client, and DNS Client modules are active in all the nodes, including incapable peers.

The DNS Server module is an authoritative, DNS-only server that hosts peer/super-peer resource records and delivers query and update messages to the cluster management modules. The Capability Monitoring module frequently updates a node's capability based on the characteristics of its hosting device. The Super-peer Role Management module handles the "super-peer selection" message and changes a peer node to super-peer (or vice-versa, as needed). These three modules are active on all capable peers and super-peers.

The Load Balancing module is responsible for splitting/merging operations, and the Service Discovery Request Handler provides global service discovery by gathering the responses of other super-peers for a request that cannot be satisfied from its local directory. These two modules are only active in super-peers.

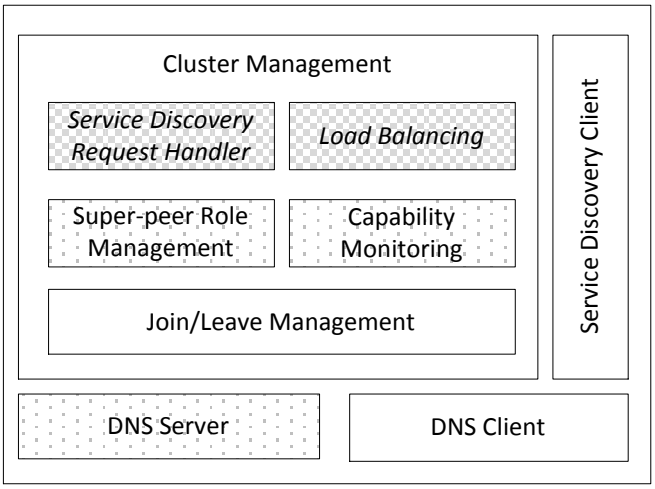

Figure 4.7. Peer/Super-peer software architecture

We implemented our architecture based on an open source implementation of DNS messages, dnsjava [80]. dnsjava has been used successfully in several projects that are listed on its website. We implemented a simple authoritative-only DNS server that supports DNS Query and DNS Update messages. The rest of our software architecture is implemented purely in java which enables the deployment of this prototype on any device with java runtime support.

4.3.2.2 Performance Evaluation

To evaluate the performance of our prototype, we defined a delay metric for each of the presented operational procedures. The delay metric measures the time difference between the start and end of each procedure. To have an accurate analysis of the delay we tested each operation 10 and calculated the average delay.

Our experimental setup consists of four laptops forming a MANET, with each laptop hosting three nodes. Our prototype provides a user interface that allows us to initialize the capability of each node and define when a node joins, leaves or performs a service

discovery request. We defined the thresholds for splitting and merging as 3 and 1, respectively. Thus, when a super-peer has more than 3 registered peers, it starts a splitting procedure, and a super-peer with less than 1 peer starts a merging procedure.

Our evaluation scenario was based on the random joining and leaving of different nodes to simulate high network churn in a typical MANET. Each peer also randomly performs service discovery operations while it is in the network.

Figure 4.8 depicts the average delay measured for each of the operational procedures. Peer Joining indicates the delay of a peer to join if a super-peer already exists, and Super-peer Joining indicates the joining delay for a capable peer that has not found any super-peer and therefore becomes a super-peer. Obviously, since in the latter case the joining peer needs to connect to a multicast group, the delay is much higher than that of "Peer Join".

Peer Leaving and Super-peer Leaving indicate the delays for a peer/super-peer leaving voluntarily. Since the super-peer voluntary leave deals with finding a registered peer and transferring the service information, it has a delay higher than that of Peer Leaving.

Local Service Resolution indicates the delay for performing a service resolution procedure when the responsible super-peer can satisfy the request. Global Service Resolution measures the delay of service resolution when the responsible super-peer has forwarded the request to the other super-peers and has gathered their responses. A similar scenario applies to Network Browsing since all the service instances in the network should be enumerated. It is clear the Local Service Resolution has less delay than the previous two cases because it does not include a multicast request.

Splitting indicates the delay for a splitting procedure, which is very similar to that of a Super-peer Leaving as described in the operational procedures. Finally, Merging indicates the delay for a super-peer to merge with another super-peer. Since the merge procedure requires a multicast request it has a higher average delay than the splitting procedure.

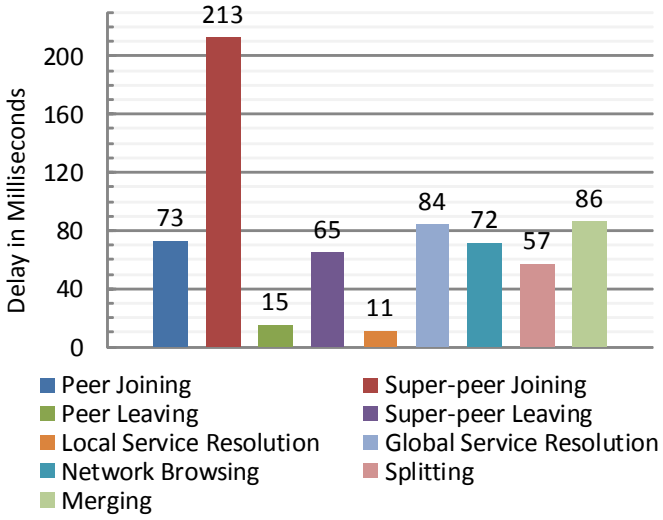

Figure 4.8. Delay of operational procedures

As depicted, the average delay for most of our proposed operational procedures is less than 100ms. The only exception is Super-peer Joining, which only happens when the first super-peer joins. In a large-scale MANET, by employing appropriate values for split and join thresholds, the frequency of this operation is much lower than for other procedures.

4.4 Meeting the Requirements

Our first general requirement on avoiding permanently centralized entity is satisfied because our proposed SDP uses on a dynamic clustering approach to form a distributed

service directory. Since our approach only uses an application layer protocol (DNS), our second requirement of being independent of the lower layers is also addressed.

To address our third and fourth general requirements on scalability and supporting mobility we devised lightweight splitting and merging operations as part of our clustering approach. We have run extensive simulations that evaluate the efficiency of our architecture in the networks with the required scalability and mobility support.

As we presented earlier, an incapable peer is never selected to be a super-peer. Incapable peers include the resource-constrained nodes in the network (e.g. sensors/actuators). Therefore, these nodes can go to sleep after registering their service information to their super-peers. This addresses our fifth general requirement on adequate functioning with resource-constrained devices.

4.5 Conclusion

Service discovery is a key element for the interoperability of heterogeneous applications and devices in DRSs. This chapter has derived a set of requirements based on disaster response scenarios and evaluated the state of the art based on the derived requirements.

To address the limitations of the state of the art we have proposed a distributed directory-based service discovery architecture. Our architecture works solely in the application layer (based on DNS protocol) and it employs a recent IETF proposal, DNS-SD, to store service information as DNS records.

IETF has also proposed a directory-less service discovery architecture for ad-hoc networks based on Multicast DNS (mDNS) and DNS-SD. We simulated the functioning of our architecture and of the IETF proposal using the OPNET simulation tool and

compared the performance results. The results reveal that our approach is more scalable in terms of the number of nodes because it generates less network traffic and has a higher success rate for service discovery operations. Furthermore, the results indicate that our cluster management operations impose a low level of traffic overhead in the application layer.

Chapter 5: Differentiated Quality of Service Architecture

In the disaster area, various teams are involved for different purposes and use different applications. For instance, paramedics aim to provide vital health services, firefighters rescue and evacuate people from dangerous buildings, extinguish fires and ensure they stay extinguished, and journalists are eager to report the news. Because of potential congestion and shortages of bandwidth it is possible that certain services may not operate properly. It is therefore imperative that life-critical services have the highest priority. For example, when a firefighter on the scene wants to start a communication, he/she should have a higher priority over news reporters.

When there are not enough resources available in the network, some of the existing communications that are of lower priority may be dropped to enable the requested communication. This will improve the reliability and the quality of the emergency services and potentially save lives. The same prioritization may apply between users of a given service according to their ranks in the hierarchy. For instance, in the case of a firefighter team, communications initiated by a captain would be considered as more important than other communications.

In order to establish such a prioritization scheme among overlays and users, a differentiated QoS model should be applied. This model ensures that network resources are allocated to the services/users with higher priority. The dynamic nature of MANETs and the wide variety of devices and technologies used by many different users are the

main challenges for such a QoS model. We propose a QoS model to provide differentiated QoS for DRSs and address the discussed challenges.

This chapter is organized as follows. Next section defines a set of requirements for potential solutions, taking into account the problem and its challenges. This is followed by an evaluation of the existing work with respect to these requirements. Second section is devoted to our proposed architecture which we discuss its performance evaluation and results in the third section before concluding in the last section.

5.1 Requirements and Related Work

5.1.1 Requirements

In addition to our general requirements, we identified three specific requirements that a differentiated QoS model should meet to be practically useful in DRSs. The first requirement is that the QoS model should support a two-level prioritization scheme, overlay-based and user-based within each overlay. The rationale behind this requirement has been illustrated as a scenario in the introduction.

Second, it should be possible to define an arbitrary number of prioritization levels, i.e., the QoS model should not assume a fixed set of prioritization levels/classes. This flexibility is needed because the configuration of overlays and user priorities is dependent upon the type of disaster and the disaster area.

Our last requirement is that the QoS model should reuse existing standard mechanisms for providing differentiated QoS (e.g. DiffServ [52]) as much as possible. In other words, it will be able to simply build on existing solutions.

5.1.2 Related Work

In this section we briefly review differentiated QoS models in MANETs and evaluate them in light of the proposed requirements.

As described in [52], DiffServ divides network flows into a set of service classes, and treats all packets on a per-hop basis according to which class the packets' flow belongs to. Intermediate nodes apply a per-hop behavior (PHB) to the packets of the same service class (DiffServ class). The differentiation among classes is enforced using variable buffer sizes, variable packet drop probability and packet scheduling. DiffServ has been proven to be suitable for MANET environments [81]. The number of service classes that can be meaningfully differentiated in MANETs ranges between two to four, depending on the type of network traffic [81]. However, DiffServ does not meet our first requirement, the ability to define an arbitrary number of prioritization levels. DiffServ makes this selective definition impossible because service classes should be statically assigned to each overlay/user prioritization level. Therefore, it is not possible to extend the prioritization scheme by adding overlay/user prioritization levels.

INSIGNIA [82] is a QoS model based on IP that provides adaptive services in MANETs. This model is similar to RSVP [83] except that it is organized to reduce the signaling overhead by using an in-band signaling protocol. Since this model is based on per-flow guarantees, it faces scalability issues in networks with a high number of nodes, such as DRSs. Therefore it does not satisfy the third general requirement.

The Flexible QoS Model (FQMM) [84] is another QoS model proposed for MANETs. This model provides service differentiation based on priority classes. It proposes that the

highest priority class should receive a per-flow provisioning similar to IntServ [51], while lower classes only receive per-class provisioning. In this model source nodes are responsible to mark the packets of the flows and condition the traffic. There are two issues with using FQMM in disaster response settings. First, the model combines two schemes in one, which makes it complex and raises the issue of scalability, just as with INSIGNIA. Second, since FQMM uses DiffServ classes, it faces the DiffServ limitation on defining an arbitrary number of prioritization levels.

The Complete and Efficient QoS Model for MANETs (CEQMM) [85] is similar to FQMM in the sense that it too combines the IntServ and DiffServ models for provisioning of QoS in MANETs. While the FQMM does not require a specific routing protocol, CEQMM is specifically based on the QoS Optimized Link State Routing (QOLSR) protocol [86]. Therefore, in addition to the problematic issues presented above for the FQMM, the CEQMM does not comply with our second general requirement on being independent of lower layers.

Service Differentiation in Stateless Wireless Ad Hoc Networks (SWAN) [87] is a stateless network model that aims to provide two prioritization levels for best effort traffic and real-time traffic. SWAN uses a local rate control mechanism for the best effort traffic. For real-time communications, a source-based admission control is used in which the source node probes the network toward the destination node and estimates the available bandwidth. SWAN also uses a dynamic regulation of real-time traffic based on explicit congestion notification (ECN) [88] in the face of network dynamics brought on by mobility or traffic overhead conditions. Although SWAN is well suited for the dynamic nature of MANETs, it has only focused on the prioritization of real-time traffic.

Therefore, it does not support the definition of an arbitrary number of prioritization levels (our first requirement).

5.2 Differentiated QoS Architecture

We propose a differentiated QoS architecture for DRSs that satisfies our proposed requirements. Our architecture provides three functionalities: admission control, mapping and enforcement. In the following sections we first discuss our architectural assumptions and principles, and then we describe each of the functionalities.

5.2.1 Architectural Assumptions and Principles

We assume that a set of priority levels are pre-assigned to overlays and users within the overlays. For instance, communications between firefighters are assigned higher priority than news agencies' communications, and within each firefighter team, the captain is assigned a higher priority than the other firefighters.

The first principle in our architecture is that we have two types of nodes, peers and super-peers. Each overlay consists of several peers and one super-peer that is selected from among the peers (see Figure 5.1). The selection criterion for super-peers is the capability of each node in the networks (e.g. processing power or battery level).

The second principle is that every overlay flow (i.e. communication between two peers in an overlay) must be first admitted by the super-peer of that overlay. In other words, the source node of a flow should first ask for admission for that flow from the super-peer of its overlay.

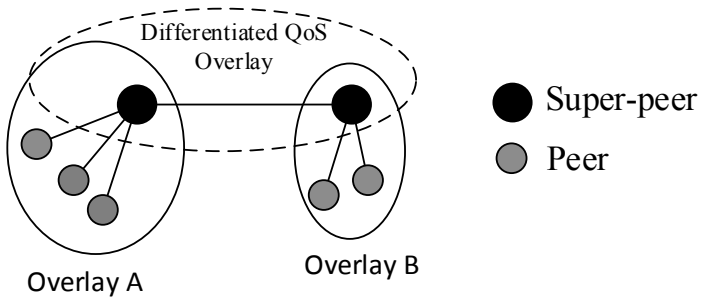

Figure 5.1. Overall Architecture

The third principle is that super-peers collaborate to ensure that the current total number of overlay flows in the network (N) does not exceed a value (M). N changes dynamically whenever a new flow is admitted or an existing flow is finished. M is also dynamically adjusted based on the success or failure of the flows which reflect the network conditions. Differentiated QoS Overlay (DQO) is formed for this required collaboration between super-peers.

The fourth principle is that when a super-peer admits a flow, it assigns a DiffServ class to that flow. The source peer must mark the packets of the flow based on the provided DiffServ class. The details of these functions will be discussed in the next sections.

5.2.2 Super-peer Selection and DQO Formation

We follow an approach similar to the one we proposed in [11] to select the most capable peer as the super-peer. In summary, when a peer joins an overlay it broadcasts a super-peer discovery request to the overlay. If it receives a response from a super-peer, the peer uses that super-peer for further interactions. If it does not receive a response after a period of time, it broadcast a capability discovery request which includes its capability. Any peer that receives a capability discovery request and has a higher capability than the

joining peer sends its response to the initial peer. The initial peer selects the peer with the highest capability and asks that peer to become its super-peer.

When a super-peer is selected it first broadcasts a super-peer discovery request in the network, which includes the priority that is pre-assigned to its overlay. Super-peers that receive this request will save this information and respond back with the current number of overlay flows in their corresponding overlays and their pre-assigned priority. This ensures that a recently selected super-peer is aware of the total number of overlay flows in the network (N). Furthermore, it ensures that all super-peers are aware of the existing overlays and their priorities in the network.

5.2.3 Admission Control Function

Admission control function is provided by the super-peers. When a peer wants to start an overlay flow it first sends an admission request (containing its priority) to its overlay's super-peer. The super-peer assesses if accepting this flow would cause the number of flows (N) to exceed the maximum number of flows allowed (M) in the network. If M would not be exceeded, it accepts the flow; otherwise the super-peer considers stopping an existing flow in an overlay with the lowest priority. If the requested flow belongs to the overlay with the lowest priority, the super-peer considers stopping the lowest priority existing flow in its own overlay. Note that this process is done locally (i.e. no message exchange is required) because first, each super-peer is aware of the priorities of the existing overlays and second, as we will describe later there is a mechanism that lets all super-peers be aware of the current number of flows in other overlays. A message exchange for admission control is provided in the illustrative scenario section.

When a flow is accepted, the super-peer adds that flow to the list of current flows in the overlay and the source peer starts the flow. If the flow is successfully received, the destination peer informs the super-peer by sending a successful flow notification. If the flow is not successfully received, the destination peer sends a fail flow notification to the super-peer. Upon receiving either notification, the super-peer removes this flow from the list of current flows in the overlay.

The collaboration between super-peers ensures that first, each super-peer is aware of the number flows in every other overlay, and second, each super-peer knows about the dynamic changes of M. Therefore, all changes to N (the summation of the number of flows in all of the overlays) and M are propagated throughout the network. To this end, each super-peer updates the other super-peers about changes in its current number of flows and changes to M by sending update flows notifications. In order to avoid issues with simultaneous changes of M, when a super-peer sends notification regarding M it will also indicate that M is increased or decreased, if applicable. Therefore when a super-peer receives the multiple notifications regarding changes of M, it can determine that those notifications are being sent simultaneously. Therefore, if k flow update notifications are received by a super-peer and those notifications have the same value for M and indicate that M is increased. In this case, the super-peer knows that these notifications are being sent simultaneously and will increase its value of M by k.

The number of flows in an overlay increases by one when a flow is accepted and it decreases by one when a flow is finished, either successfully or unsuccessfully. M is increased by one when a flow is successfully received and it is decreased by one when a flow fails. The rationale behind the dynamic changes of M is that when a flow is

successfully received, we assume that the network is capable of handling more flows, and thus M is increased, while when a flow fails we assume that the network conditions do not allow as many as M flows to exist, and thus M is decreased.

There is a race condition in our architecture between super-peers to admit new flows. The race condition raises from the fact that each super-peer uses its local values for N and M when admitting a flow. When two or more super-peers simultaneously admit flows, it is possible that total admitted flows in the network exceeds M. However, each super-peer will only become aware of this issue after it receives the flow update notifications of other super-peers.

5.2.4 Mapping and Enforcement Functions

When a super-peer admits a flow, it responds back to the source peer with a value indicating the DiffServ class assigned to the admitted flow. Therefore, the mapping from the priority of overlays and users in the network to DiffServ classes is done by the super-peers. Since super-peers are aware of the existing overlays in the network as well as their priorities, the mapping function tries to assign the best possible DiffServ class to the requested flow. The detailed implementation of the mapping function is not discussed here for the sake of simplicity.

Peers are responsible for marking the packets of accepted flows with the provided value associated to the DiffServ class. The enforcement function is thus carried out by the source peers, which eliminates the need for an intermediate node to mark the packets.

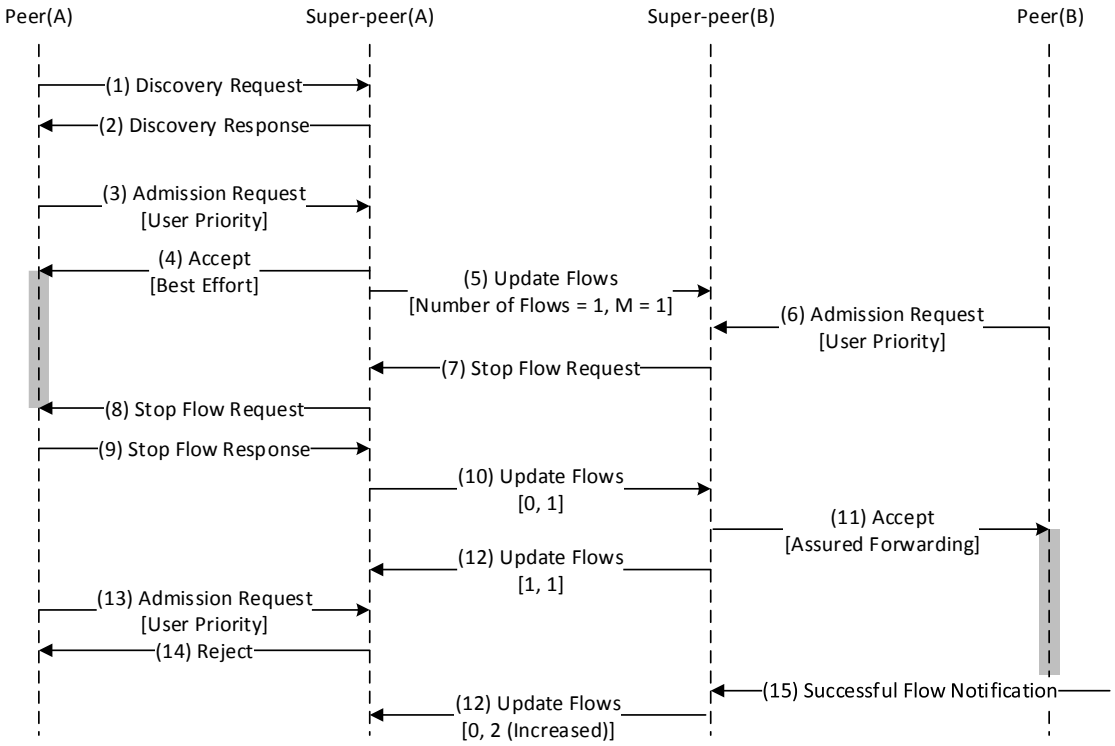

Figure 5.2. Illustrative Scenario

5.2.5 Illustrative Scenario

In this section we demonstrate a detailed message exchange in the admission control function using a sample scenario. Let us assume:

- Two different overlays, overlays A and B, exist in the network;
- Overlay B has a higher priority than overlay A;
- Each overlay has its own super-peer, super-peer(A) and super-peer(B), respectively;
- There is an existing peer in overlay B named peer(B);
- A new peer is joining overlay A, peer(A); and
- There is no existing flow in the network and the value of M is equal to 1.

Figure 5.2 illustrates the message exchanges between different nodes in our network. When peer(A) joins overlay A, it first sends a super-peer discovery message which is answered by super-peer(A) (messages 1-2). Now peer(A) wants to start a flow and sends an admission request to super-peer(A), which will be accepted because there are no existing flows in the network (messages 3-4) and M = 1. After admitting the flow, super-peer(A) updates other super-peers about the total number of flows in overlay A (message 5). While peer(A) is sending the admitted flow, peer(B) sends an admission request to start a flow. Since M = 1, and super-peer(B) knows that overlay A has a lower priority and an existing flow, it asks super-peer(A) to stop a flow (message 7). Accordingly super-peer(A) forwards this stop request to peer(A) and peer(A) immediately stops the flow and notifies super-peer(A) (messages 8-9). The notification is forwarded to super-peer(B) which makes it possible to admit the initial request of peer(B), and finally super-peer(B) informs the other super-peers about the newly admitted flow (messages 10-12).

Now if at this stage peer(A) send an admission request for a new flow, it will be rejected by super-peer(A) because M is still equal to 1 and a higher priority flow exists in the network (messages 13-14). The received flow message sent by peer(B) will eventually inform super-peer(B) that the flow has been successful (message 15).

5.3 Performance Evaluation

To evaluate the performance of our architecture, we performed a set of simulations using OPNET. We simulated a simple DRS that does not provide any QoS mechanism. We call this system non-QoS DRS. We applied our architecture to this system which is then called DQO DRS. We first evaluated how well our architecture enables service

differentiation in DQO DRS and then we compared the overall performance of the two systems.

For the implementation we reprogrammed the application layer of a built-in OPNET model for MANET nodes, in which each node uses an 802.11b wireless LAN, IP for the network layer with AODV as the routing protocol and UDP for the transport layer. We used a random waypoint model to simulate node mobility. More details about the network and node models we used for this simulation is presented as an appendix.

5.3.1 Evaluation Scenario

The simulation scenario for both systems is as follows: There are 30 nodes in the network, randomly distributed over an area of 1 kilometer square. Each node is assigned a random speed between 0 to 12 meters per second (to test low to medium mobility). There are three overlays in the scenario; overlay 1 is formed by nodes 1 to 10, overlay 2 is formed by nodes 11 to 20 and overlay 3 is formed by nodes 21 to 30.

For our architecture we assume that overlay 3 has a higher priority than overlay 2, which has a higher priority than overlay 1. We also assume that the fourth node of each overlay, i.e., nodes 4, 14 and 24 have a higher priority than other nodes within the same overlay. This scheme is used to validate the operation of our architecture in differentiating overlays and users within each overlay.

The nodes have sequential identifiers (IDs) (i.e. 1, 2, … , N). Each node periodically tries to send a new overlay flow to a random destination every 5 seconds. The duration of each flow is 4.5 seconds to ensure that only one flow is being sent by a node at a time. Each flow contains 1000 similar packets marked by a sequential packet identification number.

These packets are sent at a fixed rate. We simulated this scenario 10 times with different random seed values, for 30 minutes each time.

5.3.2 Performance Metric

To evaluate how effective our architecture differentiates overlays/users with different priorities, we have to compare how successful those overlay/users are in transmission of their flows. Therefore we introduce flow success ratio as the performance metric to measure the percentage of successful flows over the total number of flows started. A flow is considered to be successful when all of its packets are correctly received by the destination. There are various reasons for a flow not to be successful. The main reasons are congestion in the network and node mobility.

5.3.3 Performance Results

5.3.3.1 Evaluation of Service Differentiation

Figure 5.3 depicts a comparison between the flow success ratios measured for different overlays. As expected, overlay 3 with the highest priority achieves close to 15% more flow success ratio than overlay 1 which has the lowest priority. Even comparing overlay 2 and 3 reveals that using our architecture, overlay 3 receives slightly better service than overlay 2.

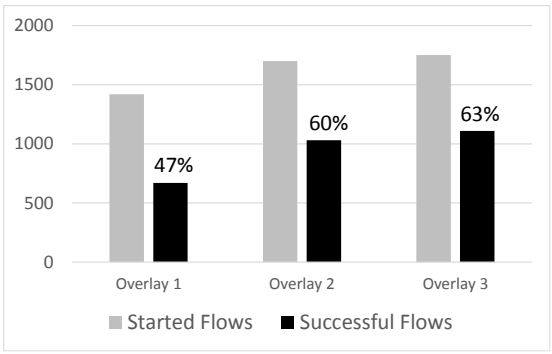

Figure 5.3. Comparison of the flow success ratios between overlays in DQO DRS

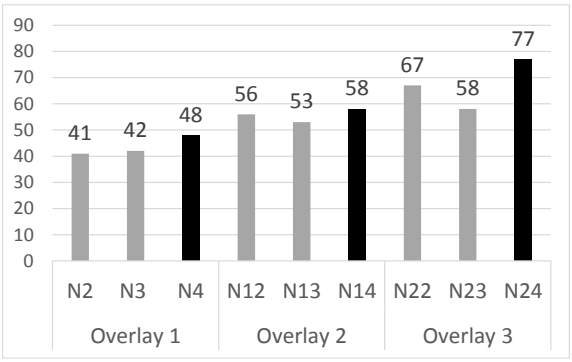

Figure 5.4. Comparison of the flow success ratios between different nodes in DQO DRS

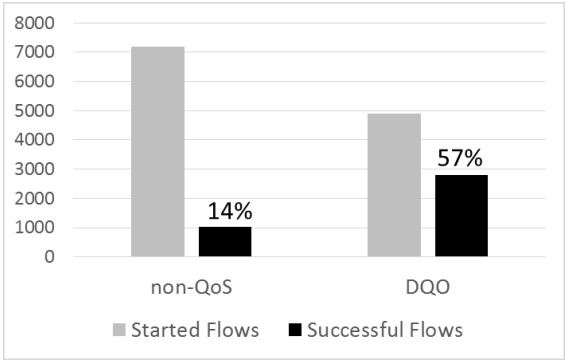

Figure 5.5. Comparison of the flow success ratios between the non-QoS and DQO DRSs

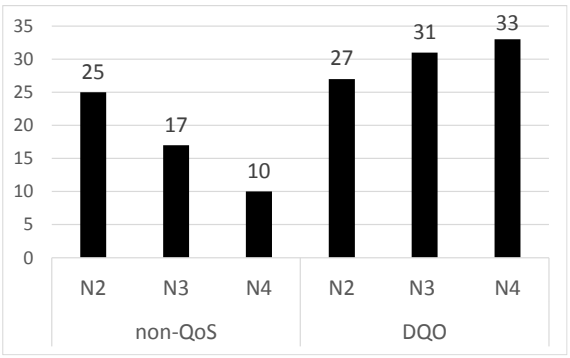

Figure 5.6. Comparison of the flow success ratios between the same set of nodes in non-QoS and DQO DRSs

Figure 5.4 shows the impact of our architecture on the level of service received by peers in the different overlays. We selected three nodes in each overlay including the nodes with the higher priority (i.e. nodes 4, 14, and 24) to compare the flow success ratio between peers in different overlays and between peers within the same overlay. The results in Figure 5.4 show that first, the flow success ratio for the peers in overlays with higher priority is higher than the peers in overlays with lower priority. Figure 5.4 also reveals that the peer with the highest priority within each overlay has a higher success ratio than its overlay peers.

5.3.3.2 Comparison with non-QoS DRS

Figure 5.5 shows the comparison of the flow success ratio metric in the non-QoS DRS and in the DQO DRS. For each system the total number of flows started in the network is shown versus the number of successful flows. As indicated, the success ratio for non-QoS DRS is only 14% while for DQO DRS it is about 57%. To ensure a fair comparison between the two methods we normalized the success ratio in DQO DRS based on the number of flows sent in the non-QoS DRS. After this normalization, the ratio of

successful flows in the DQO DRS over the total number of flows started in the non-QoS DRS is 39%, which is still a significant improvement.

We believe the main reason for this improvement is that our architecture functions in a way that addresses the two aforementioned issues that cause unsuccessful flows. Our architecture avoids congestion by limiting the maximum number of flows (M) in the network and it copes with network dynamics by adjusting M based on the feedback provided by the source nodes.

Finally, Figure 5.6 depicts a comparison of flow success ratios between three nodes in both architectures. Both were tested with the same scenarios (i.e. node location, speed, number of sent packets, etc.), which makes this comparison especially worthwhile. As in Figure 5.5, we normalized the percentage of the success rate, based on the number of flows started in the non-QoS DRS to have a fair comparison.

First, Figure 5.6 shows that when no QoS mechanism is in place, the quality of service received by different peers is very unpredictable. However, when our architecture is applied, each peer receives about the same level of quality of service. Second, although all three of these nodes are in overlay 1, which has the lowest priority among the three overlays, node 4 has a higher flow success ratio as expected, because it has a higher priority among the peers in overlay 1.

5.4 Meeting the Requirements

a) General Requirements

Since our proposed architecture is based on a self-organizing overlay, no permanently centralized entity is specified which satisfies our first requirement. Furthermore, as the

proposed overlay is formed on top of MANETs, our architecture is independent of lower layer protocols (our second requirement).

By distributing the admission control and mapping functions we aimed to address the third and fourth requirements on scalability and supporting mobility. Extensive simulations have been run to validate this claim as presented in the performance evaluation section.

To address our fifth requirement on supporting resource-constrained devices, in each overlay the most capable peer is selected as super-peer. Although this does not fully solve the issue, in practice, because of the heterogeneity of devices carried by responders, there is lower probability for a resource-constrained devices to be selected as super-peer.

b) Requirements on Differentiated QoS Model

The proposed architecture enables arbitrary prioritization scheme between overlays and between users within each overlay. This satisfies our first and second requirements. Furthermore, we have used DiffServ as a standard and commonly used mechanism to provide prioritization on overlay flows which addresses our third requirement.

5.5 Conclusion

Successful and well-organized communications in DRSs could save lives. This can be achieved through prioritization and admission control.

This chapter derives a set of requirements that potential solutions have to meet. We assessed the state of the art with respect to these requirements. None of the existing solutions meets the derived requirements and we therefore propose a novel architecture that provides three functions, admission control, mapping, and enforcement, using a

distributed self-organized overlay, DQO. Our architecture avoids network congestion by limiting the number of flows in the network and by dynamically adapting the admission process based on the feedback received from the existing network flows.

To evaluate our architecture, we ran extensive simulations. We simulated a simple DRS that does not provide any QoS mechanism (called non-QoS DRS) and then we applied our architecture on that system (called DQO DRS). We first evaluated the performance of our architecture at providing differentiated service to overlays and users. Then we compared the overall performance of the non-QoS DRS with that of the DQO DRS. Our results show that the proposed architecture at effectively enables service differentiation and improves communication between the network nodes.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

DRSs improve the situational awareness of responders and assist them by providing a wide range of applications for communication and collaboration. In this thesis we aimed to enhance three aspects of DRSs: interoperability, automation and prioritization. Interoperability enables the communication between different rescue teams which increases the efficiency of rescue operations. Automating rescue tasks allows responders to be dedicate to complex operations and it allows machines to operate without human intervention. Finally, prioritizing the access to resources in disaster area (e.g. network services) is critical to ensure that those resources are allocated to emergency services versus non-emergency services.

We have proposed application layer architectures for three services (with respect to our focus areas) in DRSs: overlay interconnection, service discovery and differentiated quality of services. Each of the proposed architectures conforms to a set of requirements that are derived from challenges raised from MANET environment and disaster response operations. We have reviewed the related work related to each service in light of the requirements. The following is a brief summary of each of our contributions.

- **Overlay Interconnection Architecture** ([10]): We proposed a distributed architecture that enables the interconnection between end-user application overlays and gateway overlays in a MANET environment. In this architecture,

interconnector nodes are responsible of address mapping and message relay between different overlays. We have selected SIP as a standard and lightweight protocol to enable the interconnection operational procedures and to provide a subscription/notification service for information exchange.

For proof of concept and to evaluate the performance of our architecture, we fully implemented the architecture as a prototype. In this prototype an end-user application overlay and a gateway overlay are developed based on JXTA and Open Chord middleware accordingly. Our performance results derived from the prototype indicate an acceptable network load overhead and interconnection delay in MANET environment.

- **Service Discovery Architecture** ([11]): We proposed a distributed directory-based service discovery architecture to enable service discovery in DRSs. Our architecture defines how service directories are formed and self-organized based on standard DNS protocol. In our architecture network nodes are categorized to different clusters. In each cluster there is a super-peer nodes which is responsible to store the service information of the peers in that cluster.

  We have simulated our architecture along with IETF directory-less service discovery proposal based on Multicast DNS (mDNS) and DNS-SD in OPNET. The performance result shows that our architecture is more scalable in terms of the number of nodes and the cluster management operations in our architecture impose a low level of traffic overhead in the application layer.

- **Differentiated QoS Architecture**:

We proposed a differentiated QoS architecture for DRSs that provides three functions, admission control, mapping, and enforcement. Admission control limits the number of flows in the network to avoid congestion. Mapping function is responsible to map the priority of overlay flows to DiffServ classes. Lastly, enforcement function marks the admitted overlay flow packets based on the DiffServ class provided by mapping function.

We simulated our architecture (called DQO DRS) and compared it with a simple DRS that does not provide any QoS mechanism. Our results show that our architecture effectively enables service differentiation and improves communication between the network nodes.

6.2 Future Work

Providing DRSs is a new and challenging research field which aims at addressing a wide range of problems. Recent research focuses include, providing end-user applications for specific rescue teams, providing general services to enable different applications or implementing realistic test beds to evaluate proposed solutions. The potential future research follow-ups for this thesis are twofold.

First, although we have provided performance evaluation for the proposed architectures, further performance comparison with the related works and in more realistic environment can be done. Such evaluation could guide us to enhance some of the design decisions in the architectures. For instance, by simulating our overlay interconnection architecture we can further validate our requirements on scalability and mobility support. For our service discovery architecture, we can extend our architecture to enable self-adaptation of the

threshold values (C, N, and M) using a learning process. In addition, we can further compare the performance of other existing approaches with our architecture.

Second, an integrated DRS architecture that includes our proposed architectures can be devised. Such architecture can even go beyond the areas targeted in this thesis by providing solutions for other DRSs issues such as security and reliability.

Bibliography

[1] A. S. Bahora *et al.*, "Integrated peer-to-peer applications for advanced emergency response systems. Part I. Concept of operations," IEEE Systems and Information Engineering Design Symposium, pp. 255- 260, 24-25 April 2003.

[2] K. Lorincz *et al.*, "Sensor networks for emergency response: challenges and opportunities," IEEE Pervasive Computing, vol.3, no.4, pp. 16- 23, Oct.-Dec. 2004.

[3] T. Gao *et al.*, "The Advanced Health and Disaster Aid Network: A Light-Weight Wireless Medical System for Triage," IEEE Transactions on Biomedical Circuits and Systems, vol.1, no.3, pp.203-216, Sept. 2007.

[4] A. Müller, A. S. Shirazi, F. Alt, A. Schmidt, "ZoneTrak: Design and Implementation of an Emergency Management Assistance System," Adjunct Proceedings of the Eigth International Conference on Pervasive Computing (Pervasive 2010), Springer Helsinki, Finland 2010.

[5] J. Casper, R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol.33, no.3, pp. 367- 385, June 2003.

[6] J. Liu and I. Chlamtac, "Mobile Ad-Hoc Networking with a View of 4G Wireless: Imperatives and Challenges", Mobile Ad Hoc Networking, chapter 1, Wiley-IEEE Press, July 2004.

[7] E. K. Lua, *et al.*, "A survey and comparison of peer-to-peer overlay network schemes," IEEE Communications Surveys & Tutorials, vol.7, no.2, pp. 72- 93, Second Quarter 2005.

[8] IDD-ESCAP. (2011), "ICT for Disaster Management: Real life examples," [Online]. Available: http://www.unapcict.org/ecohub/ict-for-disaster-management-real-life-examples.

[9] N. Aschenbruck *et al.*, "Modelling mobility in disaster area scenarios," Proceedings of 10th ACM IEEE Int. Symp. Model. Anal. Simul. Wirel. Mob. Syst. MSWIM, Chania, Greece, 2007.

[10] S. Hamedi, M. Hormati, R. Glitho, and F. Khendek, "Integrating wireless sensor networks and mobile ad hoc networks for an enhanced end-user experience," Kaleidoscope: Beyond the Internet? - Innovations for Future Networks and Services, 2010 ITU-T, pp.1-7, 13-15 Dec. 2010.

[11] M. Hormati, F. Belqasmi, R. Glitho, and F. Khendek, "A DNS Protocol-Based Service Discovery Architecture For Disaster Response Systems," Proceedings of IEEE symposium on Computers and Communications (ISCC), 2013.

[12] S. Cheshire and M. Krochmal, "Multicast DNS," IETF RFC 6762, February 2013.

[13] H. Hartenstein and K. P. Laberteaux, "A tutorial survey on vehicular ad hoc networks," IEEE Communications Magazine, vol.46, no.6, pp.164-171, June 2008.

[14] M. S. Corson, J. P. Macker, and G. H. Cirincione, "Internet-Based Mobile Ad Hoc Networking," IEEE Internet Computing, July–August 1999, pp.63-70.

[15] H. Luo *et al.*, "The Design and Evaluation of Unified Cellular and Ad-Hoc Networks," IEEE Transactions on Mobile Computing, vol.6, no.9, pp.1060-1074, Sept. 2007.

[16] H. Wu *et al.*, "Integrated cellular and ad hoc relaying systems: iCAR," IEEE Journal on Selected Areas in Communications, vol.19, no.10, pp.2105-2115, Oct 2001.

[17] Conti, M., "Body, Personal, and Local Ad Hoc Wireless Networks," M. Ilyas (Ed.), Handbook of Ad Hoc Networks, CRC Press, New York, 2003 (Chapter I).

[18] S. Ullah *et al.*, "A Comprehensive Survey of Wireless Body Area Networks," Journal of Medical Systems, Volume 36, Issue 3, pp.1065-1094, 2012.

[19] IEEE 802.15 working group for WPAN website, [online] Available: http://grouper.ieee.org/groups/802/15/.

[20] B. Crow *et al.*, "IEEE 802.11 Wireless Local Area Networks", IEEE Communications Magazine, vol.35, no.9, pp.116,126, Sep 1997.

[21] Napster, available at Internet: http://www.napster.com/.

[22] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," Proceedings of First International Conference on Peer-to-Peer Computing, pp.99-100, Aug 2001.

[23] Bittorrent, available at Internet: http://www.bittorrent.com/.

[24] Kazaa, available at Internet: http://www.kazaa.com/.

[25] Skype, available at Internet: http://www.skype.com/.

[26] J. Liebeherr, T. K. Beam, "HyperCast: A Protocol for Maintaining Multicast Group Members in a Logical Hypercube Topology," Proceedings of the First International COST264 Workshop on Networked Group Communication, p.72-89, November 17-20, 1999.

[27] I. Clarke *et al.*, "Freenet: A distributed anonymous information storage and retrieval system," ICSI Workshop on Design Issues in Anonymity and Unobservability, pp.46-66, 2000.

[28] Overnet, available at Internet: http://www.overnet.org/.

[29] K. Ponmozhi, R. S. Rajesh, "Applying P2P in MANETs for resource sharing," International Conference on Control, Automation, Communication and Energy Conservation (INCACEC) 2009, pp.1-5, 4-6 June 2009.

[30] I. Stoica *et al.*, "Chord: A scalable peer-to-peer lookup service for internet applications," SIGCOMM Comput. Commun. Rev. 31, 4 (August 2001), pp.149-160.

[31] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, pp.329-350, November 12-16, 2001.

[32] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-resilient wide-area location and routing," Technical Report UCB-CSD-01-1141, U. C. Berkeley, Apr. 2001.

[33] L. Yan, K. Sere, and X. Zhou, "Towards an integrated architecture for peer-to-peer and ad hoc overlay network applications," Proceedings of 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS) 2004, pp.312-318, 26-28 May 2004.

[34] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," IGCOMM Comput. Commun. Rev., vol. 32, no. 4, pp. 177-190, 2002.

[35] K. Jeong *et al.*, "RNet: A Hierarchical P2P Overlay Network for Improving Locality in a Mobile Environment," Fourth International Conference on Networked Computing and Advanced Information Management, NCM '08, vol.1, no., pp.623-630, 2-4 Sept. 2008.

[36] R. Winter, T. Zahn, and J. Schiller, "DynaMO: Applying Topological Locality to the Construction of Dynamic, Mobility-Aware Overlays," Technical Report B 03-04, Freie Universität Berlin, February 2004.

[37] T. Zahn, R. Winter, and J. Schiller, "Simple, efficient peer-to-peer overlay clustering in mobile, ad-hoc networks," Proceedings of the 12th IEEE Intl Conf. on Networks, pp.520-524, 2004.

[38] X. Shen *et al.*, "Handbook of Peer-to-Peer Networking," Springer, 2009.

[39] Li Gong, "JXTA: a network programming environment," IEEE Internet Computing, vol.5, no.3, pp.88,95, May/Jun 2001.

[40] Open Chord, available at Internet: http://open-chord.sourceforge.net/.

[41] Resilient Overlay Networks, available at Internet: http://nms.csail.mit.edu/ron/.

[42] E. Meshkova *et al.*, "A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks", Computer Networks, Volume 52, Issue 11, 8 August 2008, pp.2097-2128.

[43] K. Arnold *et al.*, "Jini Specification, first ed.," Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[44] E. Guttman *et al.*, "Service Location Protocol," Version 2, RFC 2165, June 1997.

[45] K. Lua *et al.*, "A survey and comparison of peer-to-peer overlay network schemes," IEEE Communications Surveys and Tutorials 7 (2) (2004) pp.72–93.

[46] S. Ratnasamy *et al.*, "A scalable content-addressable network," Proceedings of the ACM SIGCOMM, San Diego, CA, USA, August 2001, pp.161-172.

[47] M. Stokes, "Gnutella2 Standard," available at Internet: http://g2.trillinux.org/index.php?title=Main_Page

[48] T. Hargreaves, "The fasttrack protocol," available at Internet: http://cvs.berlios.de/cgi-bin/viewcvs.cgi/gift-fasttrack/giFT-FastTrack/PROTOCOL?view=markup&content-type=text%2Fvnd.viewcvs-markup&revision=HEAD

[49] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," Proceedings of INFOCOM, San Francisco, USA, March 2003, pp.177-180.

[50] E. Crawley et al., "A Framework for QoS-Based Routing in the Internet," IETF RFC 2386, Aug. 1998.

[51] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," IETF RFC 1633, June 1994.

[52] S. Blake *et al.*, "An Architecture for Differentiated Services," IETF RFC 2475, Dec. 1998.

[53] B. Davie *et al.*, "An Expedited Forwarding PHB (Per-Hop Behavior)," IETF RFC 3246, March 2002.

[54] J. Heinanen *et al.*, "Assured Forwarding PHB Group", IETF RFC 2597, June 1999.

[55] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless Sensor Networks: A Survey," IEEE Communication Magazine, August 2002.

[56] M.V. Pulgarin, R. Glitho, and A. Quintero, "An Overlay Gateway for the Integration of IP Multimedia Subsystem and Mobile Sink Based Wireless Sensor Networks," IEEE Vehicular Technology Conference (VTC)), Ottawa, Canada, Fall 2010.

[57] K. Kwak *et al.*, "An Overlay-Based Resource Monitoring Scheme for Social Applications in MANET", Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International, vol.1, no., pp.517-524, 20-24 July 2009.

[58] P. Costa et al., "The RUNES Middleware for Networked Embedded Systems and its Application in a Disaster Management Scenario", Fifth IEEE International Conference on Pervasive Computing and Communications IEEE Computer Society Washington, DC, USA 2007.

[59] L. Liquori *et al.*, "Synapse: A Scalable Protocol for Interconnecting Heterogeneous Overlay Networks," Crovella, M., Feeney, L.M., Rubenstein, D., Raghavan, S.V. (eds.) NETWORKING 2010, Lecture Notes in Computer Science (LNCS), vol. 6091, pp.67-82. Springer, Heidelberg (2010).

[60] L. Cheng, "Bridging Distributed Hash Tables in Wireless Ad-Hoc Networks," IEEE Global Telecommunications Conference, GLOBECOM 2007, pp.5159–5163, Los Alamitos (2007).

[61] D. Borsetti *et al.*, "Content Discovery in Heterogeneous Mobile Networks," Heterogeneous Wireless Access Networks: Architectures and Protocols, pp.419-441. Springer, Heidelberg (2009).

[62] L.G. Erice et *al.*, "Hierarchical P2P Systems," Kosch, H., Böszörményi, L., Hellwagner, H. (eds.) Euro-Par 2003, Lecture Notes in Computer Science (LNCS), vol. 2790, pp. 1230–1239. Springer, Heidelberg (2003).

[63] C. Fu, R. Glitho, and F. Khendek, "Signaling for Multimedia Conferencing in Stand Alone Mobile Ad Hoc Networks," IEEE Transactions on Mobile Computing, Vol. 8, No7, July 2009, pp.991-1005.

[64] J. Rosenberg *et al.*, "Session Initiation Protocol (SIP)," IETF RFC 3261, June 2002.

[65] A. N. Mian, R. Baldoni, and R. Beraldi, "A Survey of Service Discovery Protocols in Multihop Mobile Ad Hoc Networks," IEEE Pervasive Computing, vol.8, no.1, pp.66-74, Jan.-March 2009.

[66] J S. Cheshire and M. Krochmal, "DNS-Based Service Discovery," IETF RFC 6763, Feb. 2013.

[67] P. Gu, J. Wang, and H. Cai, "ASAP: An advertisement-based search algorithm for unstructured peer-topeer systems," International Conference on Parallel Processing (ICPP), September 10-14, page 8, Xian, China, 2007.

[68] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors", Communications of the ACM, 13(7), pp.422–426, 1970.

[69] M. Skjegstad *et al.*, "A protocol for robust and efficient service discovery in large, highly mobile radio networks," MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010, pp.456-463, Oct. 31 2010-Nov. 3 2010.

[70] F. Sailhan and V. Issarny, "Scalable Service Discovery for MANET," Proceedings of 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom'2005), Kauai Island, Hawaii 8–12 March 2005.

[71] R. Deepa, S. Swamynathan, "A Service Discovery Model for Mobile Ad Hoc Networks," International Conference on Recent Trends in Information, Telecommunication and Computing (ITC), pp.135-139, 12-13 March 2010.

[72] M. Klein and B. König-Ries, “Multi-layer clusters in ad hoc networks - an approach to service discovery,” Proceedings of 1st International Workshop on Peer-to-Peer Computing (Co-Located with Networking 2002), Pisa, Italy 2002, pp. 187–201.

[73] G. Schiele, C. Becker, and K. Rothermel, “Energy-efficient cluster-based service discovery for ubiquitous computing,” Proceedings of 11th ACM SIGOPS European Workshop, Leuven, Belgium, Sep. 2004.

[74] N. Suri *et al.*, “Peer-to-peer communications for tactical environments: Observations, requirements, and experiences,” IEEE Communications Magazine, vol.48, no.10, pp.60-69, October 2010.

[75] P. Mockapetris, “Domain names - concepts and facilities,” IETF RFC 1034, Nov. 1987.

[76] S. Pöhlsen, C. Buschmann, and C. Werner, “Integrating a Decentralized Web Service Discovery System into the Internet Infrastructure”, Sixth European Conference on Web Services, 2008.

[77] OPNET, available at Internet: http://www.opnet.com/.

[78] N. Aschenbruck, E. Gerhards-Padilla, and P. Martini, “Modeling mobility in disaster area scenarios”, Performance Evaluation, Volume 66, Issue 12, December 2009, Pages 773-790.

[79] N. Aschenbruck *et al.*, “BonnMotion - A Mobility Scenario Generation and Analysis Tool SIMUTools”, Proceedings of the 3rd International ICTS Conference on Simulation Tools and Techniques, Torremolinos, Malaga, Spain, 2010.

[80] B. Wellington. (2012), dnsjava (2.1.3) [Online]. Available: http://www.xbill.org/dnsjava/.

[81] T. K. Moseng and Ø. Kure, “DiffServ in Ad Hoc Networks. Presented at the EuroNGI Wireless and Mobility workshop,” Sitges, Spain, June 7-9, 2006.

[82] L. SoB, “INSIGNIA: An IP-based quality of service framework for mobile ad hoc networks,” Journal of Parallel and Distributed Computing, Special issue on Wireless and Mobile Computing and Communications, 2000, 60(4), pp.374-406.

[83] R. Braden et al., “Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification,” IETF RFC 2205, Sept. 1997.

[84] H. Xiao and W. K. G. Seah, “A Flexible Quality of Service Model for Mobile Ad Hoc Networks”, IEEE VTC2000-spring, Tokyo, Japan, 2000, pp.445-449.

[85] H. Badis and K. A. Agha, “CEQMM: a complete and efficient quality of service model for MANETs”, Proceedings of ACM Intl. w/s on Perf. Evaluation of wireless ad hoc, sensor & ubiquitous networks 2006, pp.25–32.

[86] P. Sinha, R. Sivakumar, and V. Bharghavan, “CEDAR: a core extraction distributed ad hoc routing algorithm,” Proceedings of the IEEE Infocom’99, Vol. 17(8), p. 1454-1465.

[87] G-S. Ahn *et al.*, “Support Service Differentiation for Real-Time and Best Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN),” IEEE Transactions on Mobile Computing Sept. 2002.

[88] K. Ramakrishnan, S. Floyd, and D. Black, “An Addition of Explicit Congestion Notification (ECN) to IP,” IETF RFC 3168, Sept. 2001.

Appendix: Simulation Models

This section presents more details about the provided OPNET-based simulations for evaluation of our proposed service discovery and differentiated quality of service architectures. We introduce the network and node models as well as other modules that have been used in the simulations.

1. Simulation of Service Discovery Architecture

Figure 1 depicts the network model we used for simulating our proposed service discovery architecture (CDNS) and IETF service discovery based on Multicast DNS (mDNS). In this model a MANET is formed by N nodes (here N=150) that are randomly distributed in an area of size 350m x 200m. An Rx Group Config component is used to indicate the distance threshold for radio communication of MANET nodes. Figure 2 depicts the assigned attributes of this node. As it is shown the distance threshold is set to 100 meters for all the nodes.

We modified a sample MANET node model provided by OPNET. In the node model, wireless LAN is used for MAC layer, IP for network layer and UDP for transport layer. We replaced the existing application layer process model with our own implementation of CDNS/mDNS process models (Figure 3). CDNSSDProc module is the process model we implemented based on our proposed architecture. For simulating mDNS approach we replaced CDNS process model with our implementation of mDNS, here is called mdnProc. Furthermore, mdnRequestGen process is used to generate periodic service discovery requests that should be sent by each node.

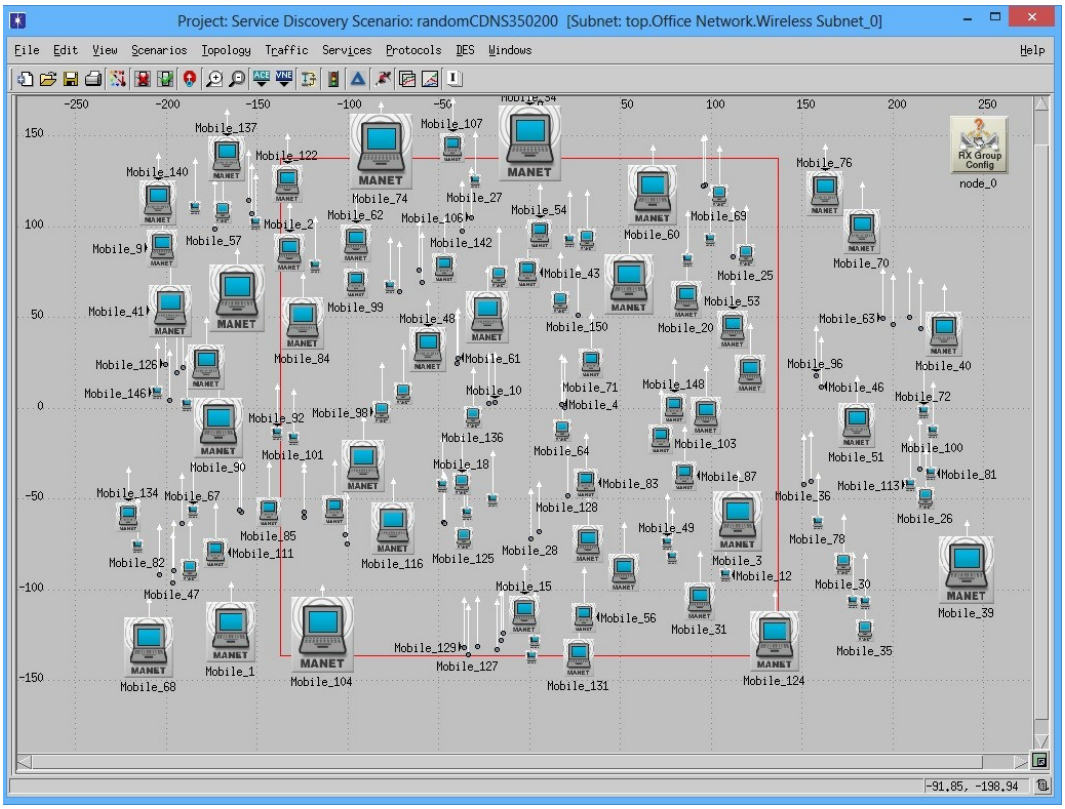

Figure 1. Network model for service discovery architecture

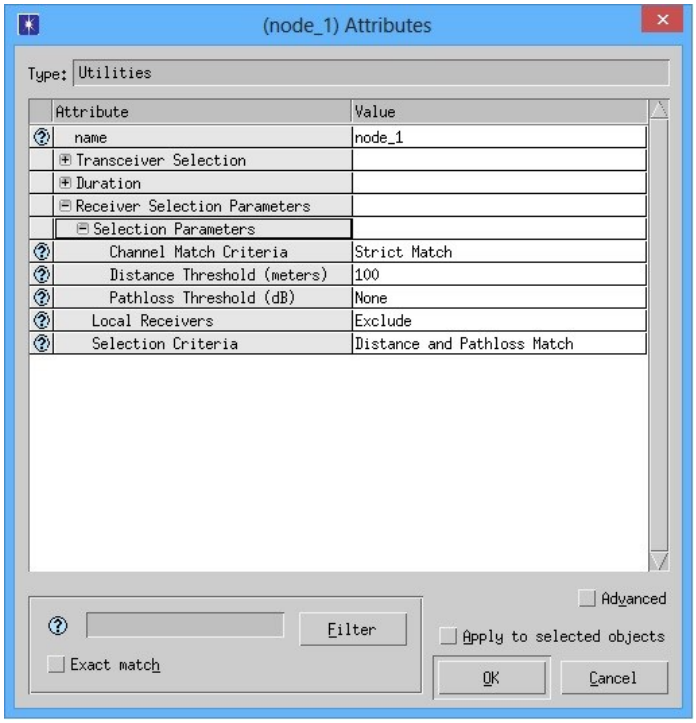

Figure 2. Rx Group Config module's attributes

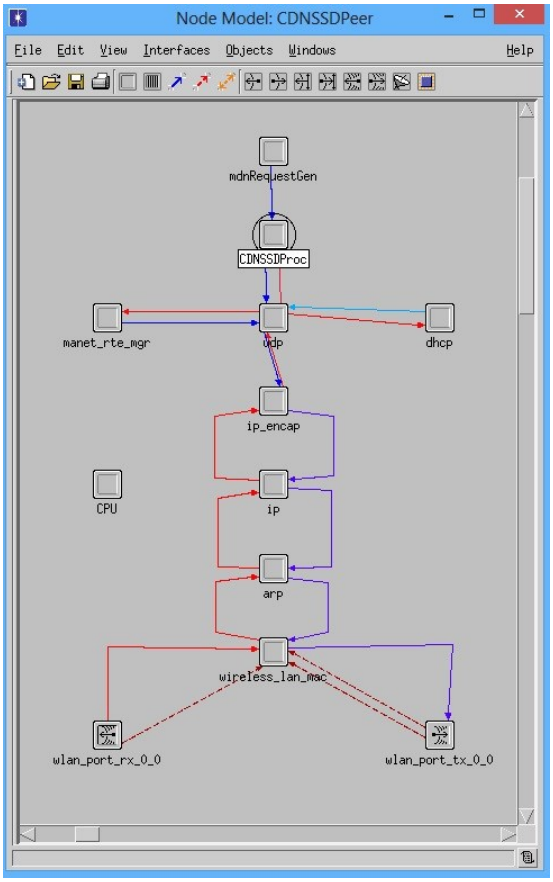

Figure 3. CDNS Node Model

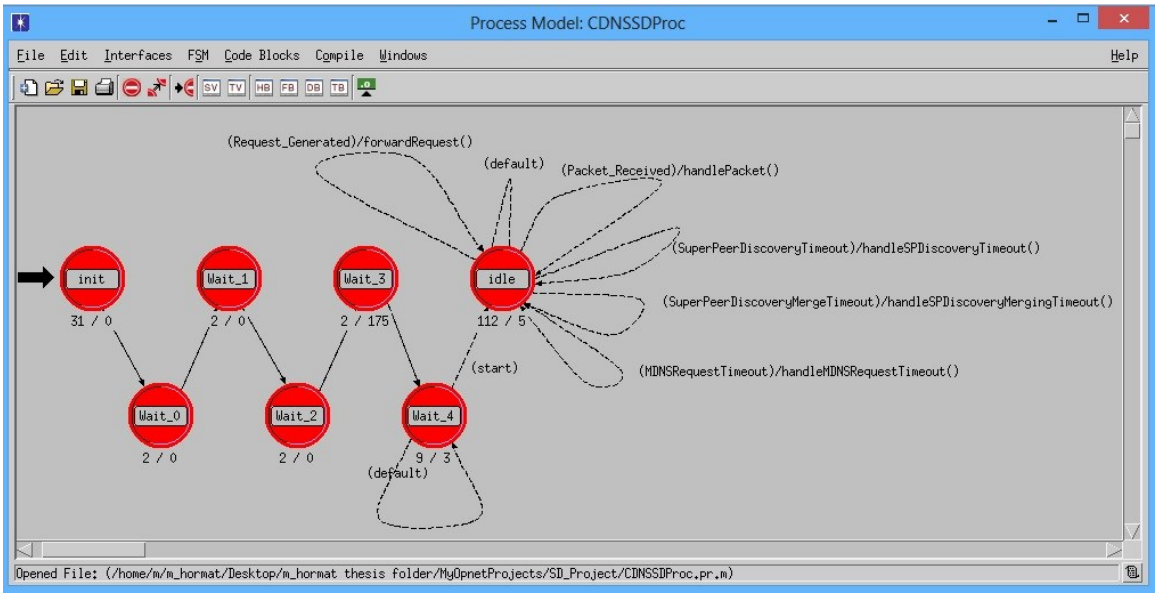

Figure 4. CDNS Process Model

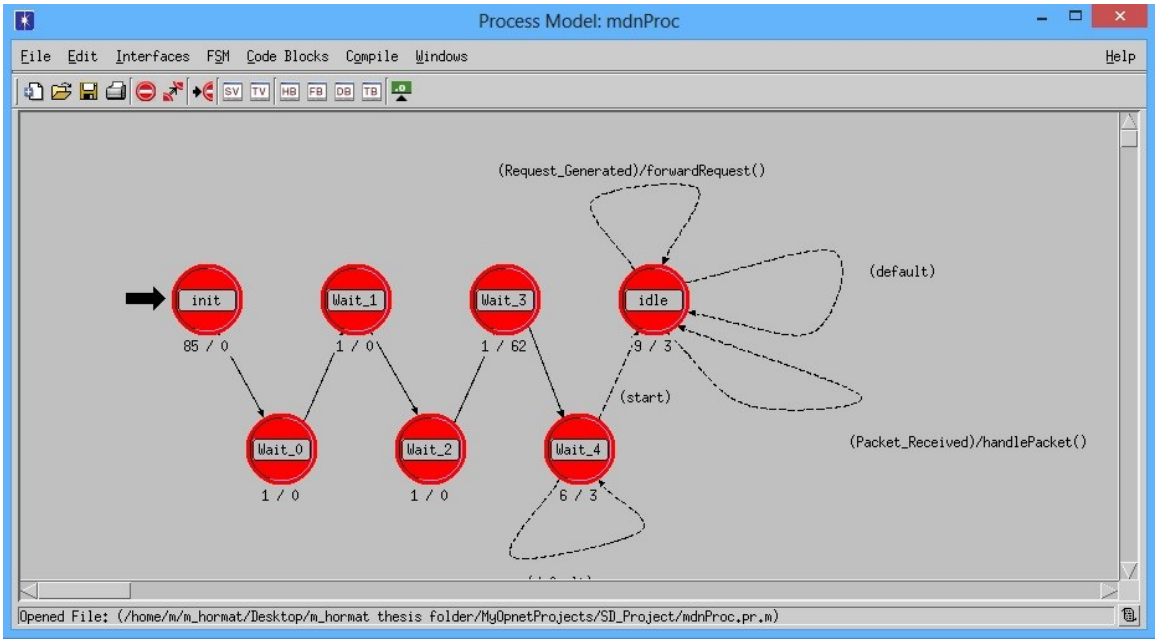

Figure 5. mDNS Process Model

Figures 4 and 5 present the process models for CDNS and mDNS architectures in the form of FSMs. In both models the process is started from *init* state and it needs to wait for five simulation cycles (*wait_0* to *wait_4*) to make sure that lower layer process models (e.g. UDP, IP) are ready to be used. The main part of our implementation is done in the *idle* state. As it is shown this state handles different simulated events such as receiving a new packer (*Packet_Recieved* event), or starting a service discovery upon the generation of a request (*Request_Generted* event). In addition, CDNS process deals with more events that enable clustering operations in our architecture.

Lastly, as we presented in Chapter 4, we used BonnMotion tool to generate the location of each node in intervals of 1 second based on the disaster area mobility model. Modification of location of nodes based on the generated values is implemented in *idle* state of CDNS/mDNS process model. Figure 6 presents the parameters we used to generate the mobility scenario. In this figure, *$x* and *$y* presents the size of the simulation

area, *$node* indicates the number of nodes and *$duration* indicates the duration for generated mobility scenario. The AREAS section, defines different type of sub-areas and obstacles in the simulated area.

```
my $bm = "../../bin/bm";
my $scenario    = "DA";
my $nodes       = "150";
my $x           = "350";
my $y           = "200";
my $groupchange = "0";
my $groupsize   = "1";
my $dist        = "3";
my $mindist     = "3";
my $circlevertices = "140";
my $factor      = "1";
my $duration    = "3000";
my $skip        = "5000";
my $seed        = "23";
my $maxpause    = "20";

################ AREAS #################

my $maxareas= "8";
my $IL1     = "25,5,125,5,125,100,25,100,25,50,125,50,0,15,15";
my $PWFTA1  = "133,25,180,25,180,75,133,75,133,50,180,50,1,39,37";
my $CCS1 = "220,5,300,5,300,40,220,40,220,20,310,20,2,15,0";
my $CCS2 = "220,46,300,46,300,80,220,80,220,60,310,60,2,15,0";
my $CCS3 = "220,86,300,86,300,120,220,120,220,100,310,100,2,15,0";
my $CCS4 = "220,126,300,126,300,160,220,160,220,140,310,140,2,15,0";
my $TEL  = "320,75,345,75,345,100,320,100,330,75,330,76,3,6,0";
my $APP1 = "320,5,345,5,345,50,320,50,5,0,345,0,325,5,325,50,4,30,28";
my $OBST1   = "25,150,100,150,100,200,25,200";

########################################

my $params = "-f $scenario DisasterArea -n $nodes -x $x -y $y -p $maxpause -
a $groupsize -g $circlevertices -r $dist -q $mindist -d $duration -e
$maxareas -i $skip -j $factor -b $TEL -b $CCS1 -b $CCS2 -b $CCS3 -b $CCS4 -b
$PWFTA1 -b $IL1 -b $APP1 -o $OBST1 -K -R$seed";

system "$bm $params";
```

Figure 6. Parameters for Mobility Generation using BonnMotion

2 Simulation of Differentiated QoS Architecture

Figure 7 presents the network model we used to simulate our differentiated QoS architecture (DQO) based on the simulation scenario that is discussed in Chapter 5. We also simulated the same scenario without applying any QoS mechanism (non-QoS) to

evaluate the efficiency of our approach. In this model N nodes (here N=30) are grouped as different overlays (here three overlays with 10 nodes in each overlay) and they are randomly scattered in an area of size 1 kilometer square. An Rx Group Config component is used which is the same as we previously discussed.

We have used OPNET Mobility Config module to enable random waypoint mobility model for all the MANET nodes. Figure 8 presents the attributes of this module as it is used in our simulations. Based on these properties, each node in the simulation starts to move after 10th second (*start time* attribute). In each step, a node moves through a randomly generated trajectory to a destination location. When the node reaches the destination it pauses for 10 seconds (*pause time* attribute) and then starts to move to a new location. The speed of each node is also selected randomly between 0 to 12 meter/second (*speed* attribute) as it was described is in the scenario.

Figure 9 presents the node model we used for the simulations. We modified a sample MANET node provided by OPNET by replacing the application layer process model with our own implementation of DQO and non-QoS process models. These process models provide algorithms that simulate sending and receiving of overlay flows. In case of DQO the process model also provides the functions of our differentiated QoS architecture.

Figures 10 and 11 show the FSMs provided for DQO and non-QoS process models respectively. The non-QoS process model is only responsible to periodically start a new overlay flow and to check if it is successfully received by the destination while DQO process model is also responsible to provide admission control, mapping and enforcement

functions. This difference is the reason for additional event handlings in the FSM of the DQO process model.

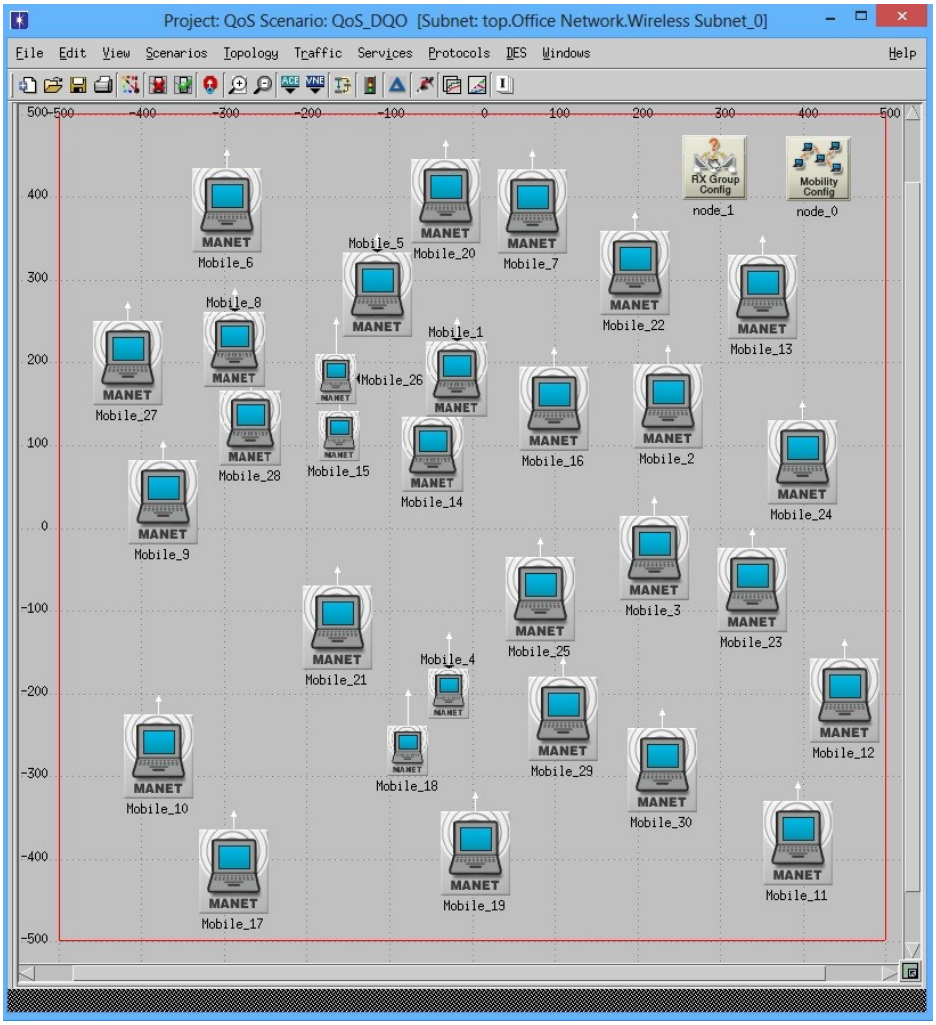

Figure 7. Network model for DQO

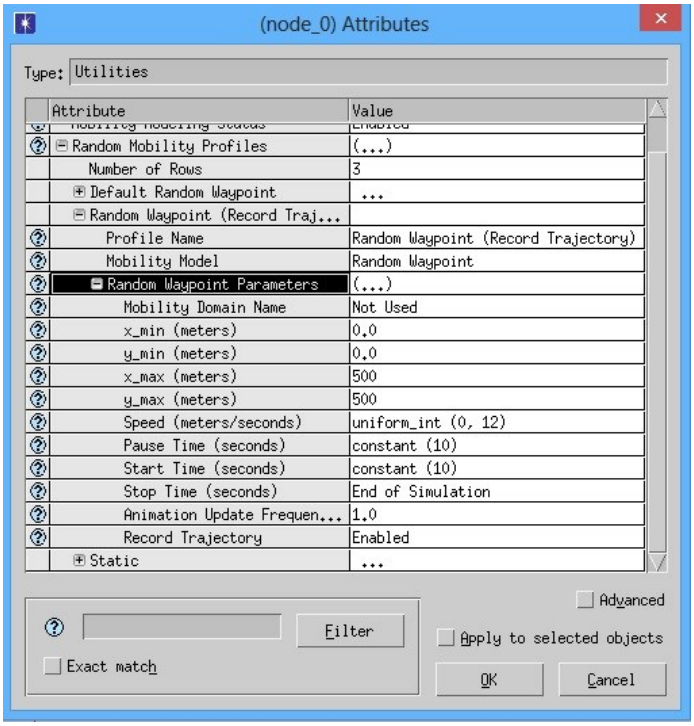

Figure 8. Mobility model module's attributes

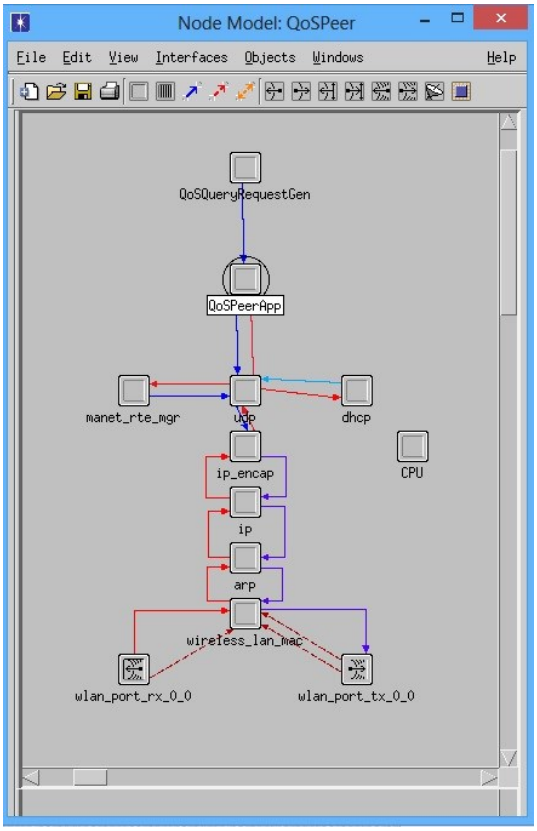

Figure 9. DQO Node Model

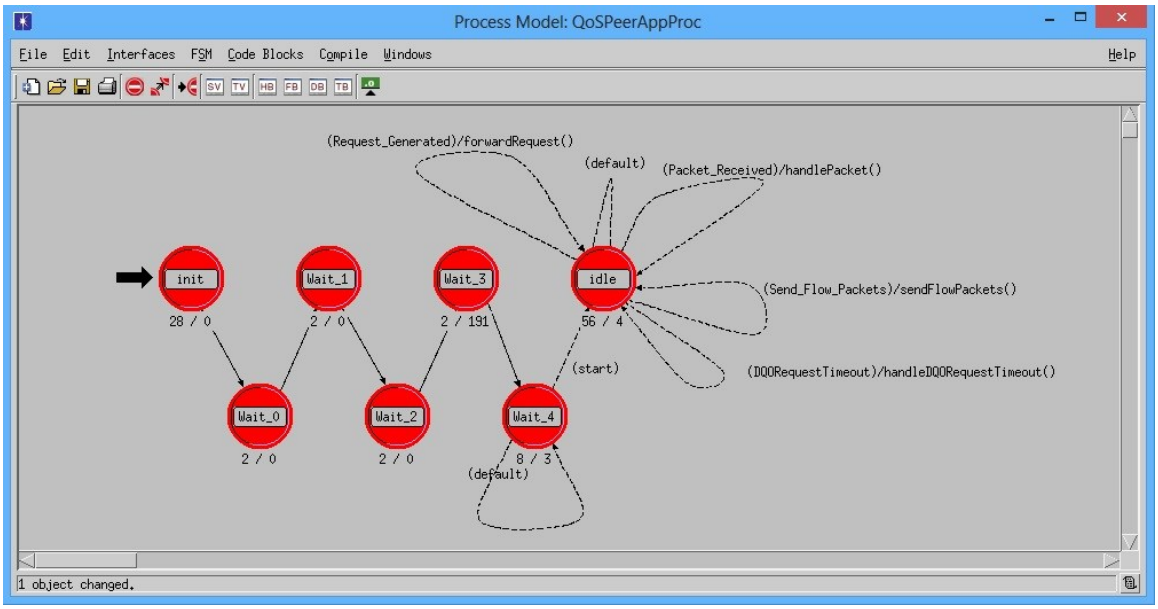

Figure 10. DQO Process Model

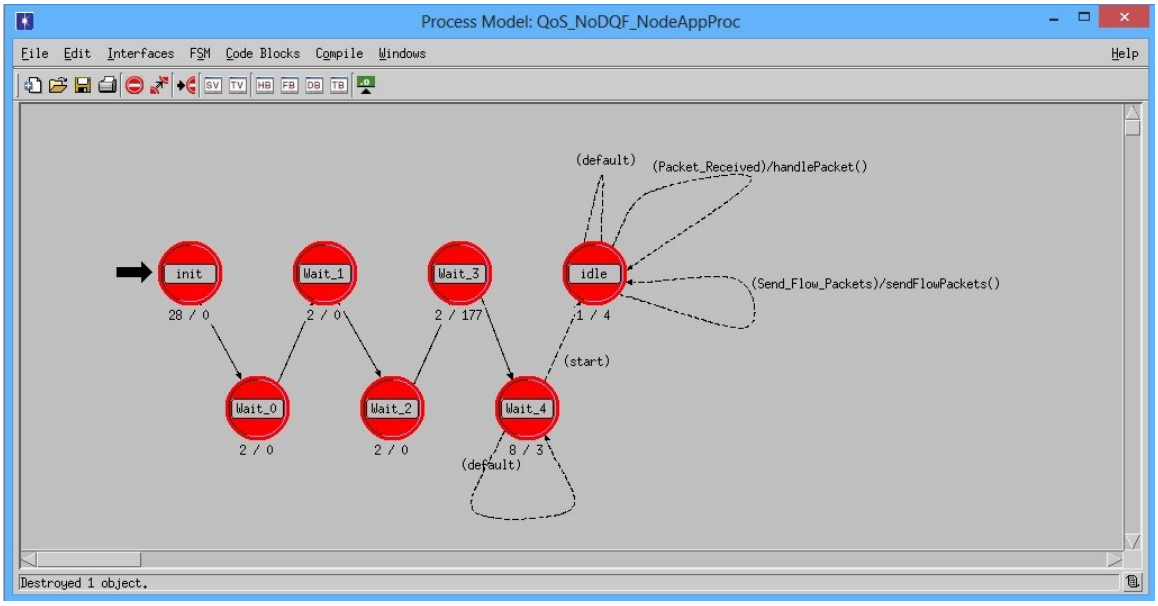

Figure 11. non-QoS Process Model