Distributed Control of Reconfigurable Mobile Network Agents for Resource Coordination

Stanley Aiyanyor Ogbeide

This thesis is submitted in partial fulfillment of the academic requirements
for the degree of
Masters of Science in Electrical Engineering
in the Faculty of Engineering and The Built Environment
University of Cape Town

February 2012

As the candidate's supervisor, I have approved this dissertation for submission.

Name: Dr. Alexandru Murgu

Signed: _____

Date: _____

Declaration

I hereby declare that this thesis is my own unaided work, both in conception and execution, and that apart from the normal guidance of my supervisor. I have received no assistance except as stated in the text document. Neither the substance nor any part of the thesis has been submitted in the past, or is being, or is to be submitted for a degree in this University or any other University.

I am now presenting the thesis for examination for the degree of MSc in Electrical Engineering. I also grant the University free license to reproduce the above thesis in whole or in part, for the purpose of research.

Stanley Aiyanyor OGBEIDE

_____ _____

Signature Date

This thesis is dedicated to my father, Late Gold Ogbeide, my mother, Mrs. Lucy Ogbeide and my wife, Mrs. Eghosa Sally Ogbeide who have been my constant source of inspiration. They have given me the drive and discipline to tackle any task with enthusiasm and determination.

Abstract

Considering the tremendous growth of internet applications and network resource federation proposed towards future open access network (FOAN), the need to analyze the robustness of the classical signaling mechanisms across multiple network operators cannot be over-emphasized. It is envisaged, there will be additional challenges in meeting the bandwidth requirements and network management. In addition, Next Generation Networks (NGN) applications are expected to be dominated by data traffic and most of its devices are capable of supporting interactive video applications. Video services come in burst as opposed to the streamline in voice. Higher bursty traffic can cause larger queue size, hence resulting in long packet delays and high packets drops, thus affecting resource utilization such as bandwidth, computational power, memory, etc.

The first objective of this project is to describe the networking environment based on the support for heterogeneity of network components. Afterwards, the thesis analyses the pros and cons of the classical network management approaches such as Simple Network Management Protocol (SNMP), OSI-SM Systems Management Functions and the Distributed Object Technologies (DOT) towards network resource coordination. The mobile agent (MA) paradigm is then presented as the new networking technology that allows direct access to network devices in a timely fashion while providing flexible network management solution.

Based on these assessments, a set of design requirements have been outlined to tailor the development and implementation of Reconfigurable Mobile Network Agents (RMNA) based on network resource coordination. This RMNA framework provides a distributed management interface to support the multi-network platform domain present in NGN. The performance of the designed prototype is experimentally evaluated using JADE platform, which demonstrates its ability to decentralize the traditional centralized management approaches and scale effectively as network size increases. This describes an interaction model that supports network-wide resource coordination in two different approaches:

i) Resource pooling for realizing cooperative communication transaction,

ii) Resource splitting for implementing diversity communication and reliability enhancement.

Acknowledgements

I would like to express my deep regard to God, the most high for giving me the courage, strength and power towards the completion of this thesis. Special thanks to Dr. Alexandru Murgu for providing me the opportunity to work under his kind supervision and guiding me throughout the course of my study. This work wouldn't have been possible without his direction, guidance, patience, and enthusiasm.

I would also like to thank all my current and past research associates at the University of Cape Town Communication Research Group (UCT-CRG) laboratory for their friendly assistance that have created an enjoyable atmosphere during my study at the University. It has been a genuine pleasure working with them.

I would like to thank the staffs in the Department of Electrical Engineering, University of Cape Town for their assistance and suggestions that made my UCT life comfortable. I am grateful to the University of Cape Town Postgraduate Funding Office for their generous scholarship support.

Furthermore, special thanks to my beloved wife, Eghosa Sally Ogbeide for enduring loneliness, love and affection, sacrifices and prayers, her continuous encouragement and support during hard times.

Lastly, I wish to thank my mom, brothers, and sisters for been there at the time of utmost needs. I knew I haven't been around as much as you all would have wanted but I will finally be soonest.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Thesis Background

Coordinating network resources such as bandwidth, buffer space, computational Power etc. are very crucial issues in network management as network growth is not commensurate with available resources at any time. Effective resource coordination will assist applications in obtaining the most appropriate services and supports quality of service (QoS) qualifiers. Users of mobile devices expect the highest quality and greatest features from network operators (NOs) with reduced cost. These constraints impose challenges on network operators to upgrade their technologies constantly. The various upgrades of technologies have introduced several challenges and difficulties towards the management of network applications. Over the past years, improvement of different network technologies has progressed in different directions, though they all seems alike, yet different protocols are deployed as shown in Figure 1.1.

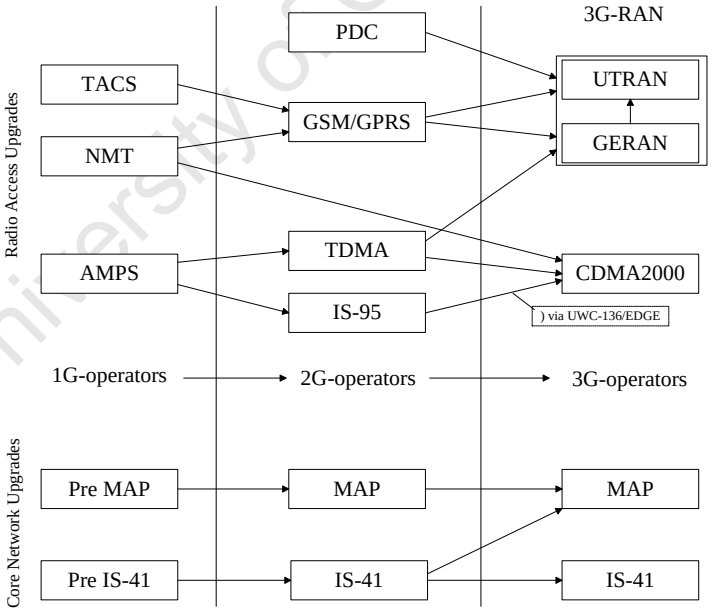

Figure 1.1. Network upgrades of operators [1].

These networks upgrades have successfully enhanced network performance as they provide specific packet data services (such as Voice over IP, video applications, etc.) over the mobile

network. While these solutions are acceptable in many situations, researchers still believe that this trend has inherent flaws. These protocols were originally for voice; improvement to accommodate data transfer accumulated various additional constraints, as they are inefficient with their usage of the available spectrum bandwidth hence causing high delays of data transmission. Data transfer comes in burst compared to constant streaming of voice.

In addition, the increasing demand for broad range of compute-intensive applications from network providers (NPs) has complicated today's telecommunication network technologies and architectures. Specifically the introduction of Tablets, Smartphones (such as iPhones, iPad, etc.) and other sophisticated next generation dongles that are usually plug-and play devices into any computer via the USB port to give users constant broadband access everywhere. These are evolved mobile devices. They have the capabilities of displaying high quality video content such as video conferencing, mobile TV, capturing video for video sharing, video blogging, video Twitter, and video broadcasting applications while maintaining mobility [2]. Such applications are widely referred to as the next generation applications due to the inability for the classical network technologies to support the required bit rate for video content. For example, a single low-resolution compressed video stream at about 384 kbps is roughly equivalent to about 63 voice users at 12.2 kbps with 50 percent activity factor [2]. This leads to unprecedented increase in traffic on the mobile networks; Cisco predicted in 2011, that mobile traffic would increase by a factor of 39 times between 2010 and 2015, and almost 66.4 percent of this traffic is expected to be video applications by 2015 [3]. If a proactive measure is not considered, video-based applications could easily swamp the capacity of a wireless network.

The current approach to these limitations is the deployment of much higher capacity networking technologies such as Mobile WiMAX and LTE. Mobile WiMAX (Worldwide Interoperability for Microwave Access) is a technology that provides mobile internet access. It is based on IEEE 802.16e and IEEE 802.16m standards which provides high-speed connection up to 70 Mbps over the area of 30 miles. There is no need for line of sight connection between subscriber terminals and the base station (BS) in WiMAX technology and it can support hundreds if not thousands of subscribers from a single BS [4]. LTE is the acronym for Long Term Evolution. It's based on the Third Generation Partnership Project (3GPP) a standard in the

mobile telecommunication network technology tree that produced the GSM/EDGE and UMTS/HSPA network technologies [5][6]. The LTE specification provides downlink/uplink peak rates of at least 100 Mbps and 50 Mbps respectively, with radio access network (RAN) round-trip times of less than 10ms.

These network technologies are expected to improve the end-user quality of experience (QoE) and to support the multimedia services via a simple architecture that provide low transmission cost which in turn result in low operating costs. Contrary to the general view and the exciting benefits of this broadband evolution, experience revealed that such technology upgrades does not entirely eradicate the challenges encountered by these video applications. Considering the rapid growth rate of mobile networking and applications, these solutions simply put it off for a while, as both the WiMAX and LTE technologies certainly reach their expandability limit thus, become saturated. Global mobile data traffic expected to increase 26-fold between 2010 and 2015. Mobile data traffic compound annual growth rate (CAGR) will increase by 92 percent from 2010 to 2015, reaching 6.3 Exabyte per month by 2015 [3].

In order to meet these requirements, researchers, engineers and other telecommunications standardization bodies are currently paying intensive attention towards the Future Open Networks (FON) introduced in early 2000 [7]. This field is beginning to gain standardization by the International Telecommunication Union (ITU) and the European Telecommunications Standards Institute (ETSI) as they incorporated the Open Service Environment (OSE) into the next generation networks (NGN) architecture [8]. This functionality in NGN enables the integration of different standardized Radio Access Technologies (RATs) such as GSM, UMTS, WiMAX and LTE, which traditionally belong to independent NPs into a single, converged, user-centric communication network. These network technologies is expected to exhibit different infrastructural capabilities either in terms of coverage, capacity, transmission rates, transmission delay, or transmission cost that mandates NPs to be cooperative in a competitive relationship for resources. These with no doubt will enhance global roaming, mobility, scalability and performance of the entire network as users of mobile devices will have unfettered access to different NPs. Hence, users can select, subscribe and activate network services based on their preferences (user driven networking) as depicted in Figure 1.2.

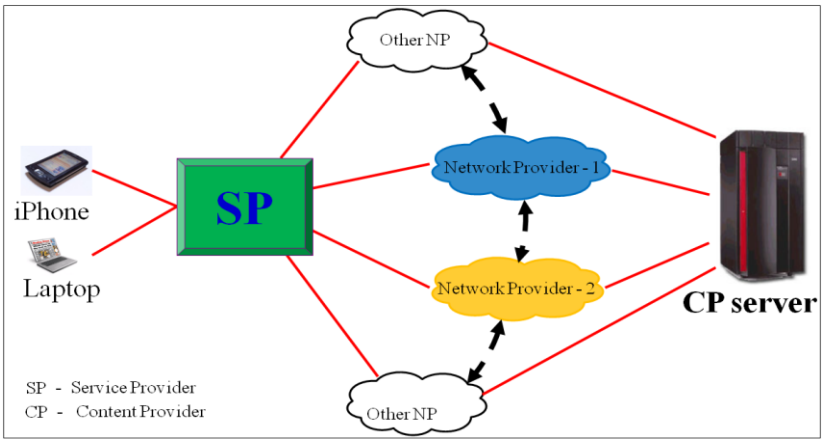

Figure 1.2. Cooperative networks internetworking strategy.

The service provider (SP) which can also be referred to as common access network operator acts as single point of contact between the mobile terminal and multiple network infrastructures. The SP is responsible for the definition of the service characteristics and the maintenance of the customer premises equipment, while the NPs provide the network infrastructure (i.e., high-speed network) [9]. The black arrow that transverse through the NPs domain indicate the capability of mobile terminal users to roam across multiple NPs domains in the ways that best reflects their preferences. The content provider (CP) stores and retrieves multimedia services such as voice, data and video and makes it accessible to all applications. Representing the various NPs domains as cloud instead of line or circle indicate that router connections are not restricted to a given network technology [10, p.272]. For simplicity, we represented users roaming ability with single arrow, there will be multiple entry and exit point of attachment between various NPs in real network scenario, this will enable load distribution, redundancy and efficiency.

While this integration of multiple wireless access networks can potentially provide extended coverage and additional value added services to mobile terminal users, it may often lead to inefficiency of the underlying network topology. These user's mobile terminals such as iPhone and laptop can only connect to one access network at a time as such imposing the need for appropriate interconnection strategy to neighboring host within the system while effectively utilizing the network resources and avoiding lost/delays of data [11]. Coordinating resources in such multi-network-provider domains that consist of multiple technologies is far from straightforward. It envisaged that such environment would require a greater degree of interactions

among the service components as gateway nodes should support the common authentication mechanisms [8], thus making network services provisioning more complex. Each of these interactions generates network traffic; higher bursty traffic can cause larger queue size, hence resulting in long packet delays and high dropping packets, which affect resource utilization such as bandwidth, computational power, memory, etc. [12]. Severe traffic congestion is also likely to occur during run-time if these systems architecture is not optimally tuned. These motivations compelled this project to understand the interactions of such system together in a fundamental way.

Considering the tremendous growth of internet applications and network resource federation proposed towards future open access network. The need to analyze the robustness of the classical signaling mechanisms across multiple network operators could not be over-emphasized. It envisaged there would be additional challenges in meeting the bandwidth requirements and network management. In traditional network management system, every NP has its own management station that periodically accesses the data collected by a set of software modules placed on the network devices, by using an appropriate protocol [9]. The Simple Network Management Protocol (SNMP) is currently the most widely adopted approach for the management of data networks [13]. SNMP usage is often in centralized network management environments [14]. For its centralized nature, static management capabilities and simple-minded model, the research communities have heavily criticized the approach [13]. Opening multiple entry and exit points to different NPs domains and enabling user-unfettered access during information exchange will cripple the traditional centralized management topologies when network size significantly increases [11]. Some of the major challenges with the traditional approach are:

- excessive processing load at manager;
- heavy usage of network bandwidth;
- information bottleneck at NMS when network grow significantly;
- lack of scalability.

A possible approach to these challenges is to enable and adopt an application that embraces the interoperability and proactive connectivity protocol. Mobile Agent programming technology has emerged as a flexible, proactive and complementary way to coordinate resources of distributed systems due to the increased flexibility in adapting to the dynamically changing requirements of such systems [9]. It is imperative to develop a reconfigurable mobile network agents (RMNA) framework as a middleware between the connectivity and application, which must have flexibility, adaptability and reconfigurable capabilities. RMNA are software entities that are distribute over the network components to perform monitoring functions in a local manner. Instead of sending multiple queries between the management station and the network manage nodes in order to analyze the performance and requirements as in SNMP, RMNA migrates onto the network locations (managed nodes) from the mobile user (MU) terminal to locally execute monitoring functions and report only the resource information results.

To implement a complete integrated system, software techniques are important in advance. This thesis aim at developing RMNA based on network resource coordination. The formalization of the planning and implementation phases of the RMNA development life cycle specifically focuses on JADE platform, and the concepts provided by it. JADE (Java Agent DEvelopment Framework) is a software environment to build agent systems over a well-known object-oriented language (JAVA) for the management of networked information resources in compliance with the IEEE Foundation for Intelligent Physical Agents (FIPA) specifications for interoperable multi-agent systems [15][16]. JADE was designed at the onset to focus on supporting agent communication using the FIPA agent standards [17]. JADE was very helpful in implementing the required services as it simplified by default all IEEE FIPA specifications and allows the creation of multiple containers in a single JAVA Virtual Machine (JVM). The decision to use JADE among other platform are:

- its open source nature,
- allowing easy extension with add-on modules;
- it is simple and friendly API;

- support agent mobility;
- facilitates the development of complete agent-based applications;
- its rich suite of graphical tools.

JADE platform has increasingly being used in agent development framework as it supports complex reasoning [18][19]. JADE enables the synchronization of software messages when several messages are sent and received from different agents in parallel [20].

1.2 Motivation and Problem Statement

An inherent characteristic of future wireless networks will be their support for heterogeneity, manifested by the presence of different access network with great variety of mobile terminal capabilities. Heavy pressure is been mounted on network operators (NOs) to effectively integrate voice, video, and web collaboration into the same conference. Also, NOs are expected to provide users the possibility to select, subscribe, and activate network service based on their preferences. One will expect the traditional decision to invest in infrastructure and technologies as a solution. However, provisioning of network applications such as internet protocol television (IPTV), virtual private network (VPN), and cloud computing is habitually across multiple network operators [21]. Additionally, with the wide array of network equipment, the choice of software or network, and uncertainty about such investment in a climate of already low and decreasing retail prices presents a challenging business case to network operators in stand-alone scenario. Partnership or in other words network cooperation offer the opportunity to share the significant capital/operational expenditure (CAPEX/OPEX) involved [22].

The major barriers to the effective deployment of such partnership lies on the separated communication technologies used by different network operators and different platforms that required management interface for support. The context of open access network (OAN) incorporated in next generation networks (NGN), where a common access operator unifies multiple network operators [21] as illustrated in Figure 1.2 will obviously overcome the associated complexities experience by the classical approach. Nevertheless, ensuring effective resource coordination and providing user's quality of experience in such multi-network platform

deem far from trivial task. A major challenge with the user unrestricted roaming ability would be intermittent network connection. The applications deployed in this environment should be robust enough to meet the changing network conditions.

To deliver system reliability in this environment, the need for efficient generalized solution that can monitor and manage heterogeneous network is very important. The key idea in this proposed scheme, is to manage network devices locally: a decision to directly delegate the management applications to the managed components (routers) in the form of reconfigurable mobile network agents (RMNA) which provide fast and flexible solutions to the dynamic network changes, distribution of network load and efficient automating of network managements. RMNA is a software module that travels to different nodes of the network, collecting status information and carrying out device control task. Mobile agent can reduce network traffic and easily support disconnected operations [23]. Mobile agents have the advantage of executing their monitoring and management tasks without administrator's intervention. Once administrator maintains connection during initiation of mobile agent request, the agent migrates to the specified host and continues its execution at the remote host until the task is accomplished. The agent only returns with the achieved result. This will prevent users from been frustrated where intermittent network connections are often inevitable.

1.3 Research Aim and Scope

The main target of this research work is to design a mechanism for coordinating and reserving resources in next generation networks (NGN). This will enhance the interaction and negotiating procedures between entities required to establish and maintain communication between different networks while ensuring end-to-end seamless connectivity.

While SNMP is extremely useful as a tool in network management, concerns are been expressed in its use for managing large networks. These deficiencies include: high-overhead to retrieve large blocks of data, heavy reliance on polling, unreliable trap delivery that can result in critical events being missed, limited security, and no ability to support manager to manager communications.

With the transition from centralized computing systems to the distributed scheme, system and network managers encounter a growing problem. Instead of a single, centrally managed system, systems have become widely distributed, with the importance and sophistication of each component increasing. Issues such as availability, performance, fault identification and diagnosis became greater challenges to system administrators. In the extreme case, the exponential growth of the Internet and FON presented a clear need for standard management approaches. Since SNMP needs to manage multiple devices manufactured by different vendors, the parameters that it can manage are quite limited, based on the standards defined. However, network devices are always very different from each other. Some are complex. As a result, users cannot define customization for management of particular elements.

Reconfigurable mobile network agent (RMNA) based network monitoring and management is a new paradigm to address these limitations incurred by SNMP. The concept of RMNA monitoring proposed in this project has increased flexibility in adapting to the dynamically changing requirements, and decentralizing the monitoring and management task in order to eliminate information bottleneck around network management station. RMNA can update its information base while executing other management task as oppose the mandatory communication between the SNMP manager and SNMP agent.

1.4 Thesis Contribution

Firstly, a broad overview of the need for network resource management is conducted. Findings include resource limitation in the classical telecommunication networking technologies and additional challenges await the next generation network deployments. Then examined some of the existing network management mechanisms, and conducted a detailed study on the pros and cons of these management schemes towards the next generation networking architecture. The SNMP is the most often used solutions for remote network management. Based on some of the challenges faced with the centralized nature of the SNMP as listed in Section 1.3, this project proposed and evaluated the use of reconfigurable mobile network agents (RMNA) scheme that allows direct access to network devices in a timely fashion as the flexible management solutions.

As revealed from survey, the major challenges with the existing SNMP scheme are centered

on the network management system (NMS). An NMS is the combination of hardware and software used to monitor and administer a network. It manages the network elements, also called managed devices. There are five major functions for device management, these includes:

- Faults;
- Accounting;
- Configuration;
- Performance;
- Security management.

Management tasks include:

- discovering network inventory;
- monitoring device health and status;
- providing alerts to conditions that affect system performance;
- identifying of problems, their source(s) and possible solution.

The network management station in SNMP holds the SNMP manger that periodically communicates with the SNMP agent located at the managed device. The tremendous growth in today's telecommunication networks introduced several complexities within the management station hence exposing it to issues like congestion, delays, and single point of failure (in the event of network storm, failure in one part affects the entire network).

The direction of this project is to manage the network devices locally; a decision that directly delegates the management applications to the managed components (routers) in the form of reconfigurable mobile network agents (RMNA). This describes an interaction model that support network-wide resource coordination in two different approaches:

- Resource pooling for realizing cooperative communication transaction;

- Resource splitting to enable implementation of diversity communication.

This new idea grants managed devices:

- the autonomy to maintain and coordinate activities within its location;
- decentralizing the functionalities of the network management station;

Therefore, each mobile subscriber has the opportunity of:

- implementing their own policy;
- service costs with quality of services according to their preferences and capacities in a more graceful manner.

SNMP requires continuous polling of network component variables due to SNMP Traps unreliability. Heavy polling of network component can:

- increase network traffic;
- impacts network resources;
- lead to excessive signaling overhead;
- impact the scalability of network management application.

The cooperative communication in this scheme allows RMNA signaling performance overhead introduced to be an exponential increase as network nodes or network provider's increases. Instead of having user agent traveling node by node and querying the network capabilities and performance, RMNA sends a synchronized replica of itself to all the nodes of the various network providers. This in turn minimizes RMNA response time that is very important parameter since the signaling mechanism must execute prior to real data transportation.

1.5 Thesis Outline

This chapter presents the thesis background information and briefly describes the need for

effective network resource coordination paradigm, with emphasis on mobile agent technology. The remainder of this thesis has the following structure:

Chapters 2: This chapter presents state-of-the-art work (literature review) on mobile networking and outlined a set of requirements for network resource management. The chapter comprises of two main parts:

- the first part introduces the main motivation behind the proposed mobile agent framework
- the second part presents a state-of-the-art report on network management approaches and outlined a set of design requirements to facilitate the coordination and reservation of network resources in next generation networks (NGN)

Chapter 3: This chapter incorporated the various requirements outlined in Chapter 2 into the designing phase of RMNA. The chapter further introduced JADE (RMNA design platform) and describes how the JADE platform supports these design requirements.

Chapter 4 presents a prototype mobile agent system. This is implemented using JADE platform to emulate internetworking of two NPs through WiMAX and LTE gateways. This chapter analyses how RMNA are created/executed and describes their communicative interactions. It further demonstrates how JADE is interfaced with a JavaSniffer tools that provides this scheme with appropriate evaluation technique.

Chapter 5: The main goal of this chapter is to evaluate the performance of the proposed RMNA framework via series of experiments. The thesis specifically considers the measurement of response time and signaling overhead incurred during RMNA monitoring functionalities. These evaluations describes how RMNA-based approach promises scalability solution when deployed for resource management operations.

Chapter 6 draws concluding remarks to this thesis finding. First it analyses the various contributions of this thesis and then provides certain directions for future research developments stemming from this thesis.

Chapter 2 Literature Review and Requirements

2.1 Telecommunication Network Characteristics

A network is a combination of devices and communications channels that follow a set of protocols and provide connectivity [24]. Typically, a network consists of several end nodes connected through channels/links via intermediate network devices. The connection of terminals, nodes and links together using appropriate protocols to enable communication between users of the terminals and network operators is referred to as telecommunications networking. These network components need collaboration to achieve functions such as routing, translation between different network protocols, and maintenance of connectivity between different physical networks. Networks may use circuit switching or packet switching. To create coordination and manageable environment, each terminal in the network must have a unique addressing scheme that ensures the routing of information or connections to their specified destination. The nodes such as routers or switches often distributed within the network (distances defined by system administrator) require the functionalities of the links to connect them together. This connection is built upon an underlying transmission network, which either physically or logically pushes the information across the link. Figure 2.1 illustrates the information exchange using the packet switching technique.

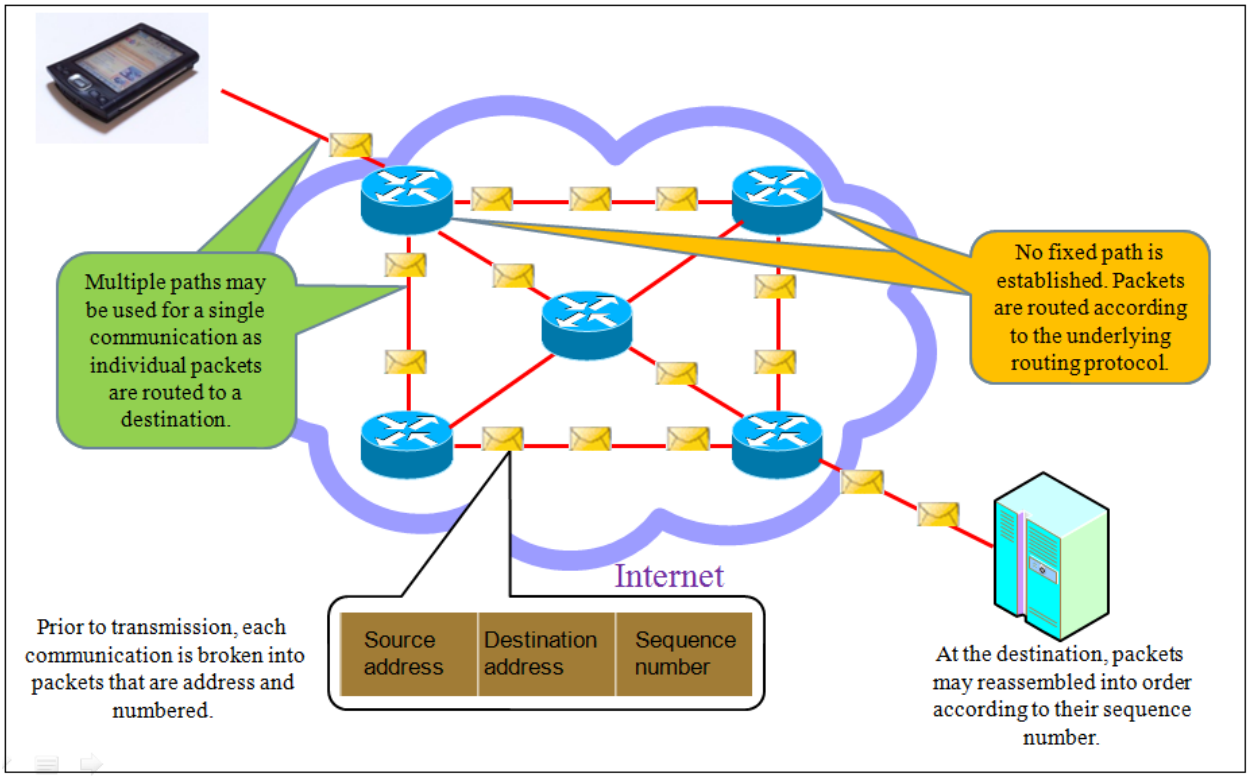

Figure 2.1. Data transmission [25].

Design Requirement 1 (DR-1): A unique addressing scheme is required to effectively create coordination and manageable networking environment

Communication systems are expanding both in physical terms, with the exponential growth in Internet usage and the mobile phone market, and in the range and complexity of services offered by providers [26]. Traditionally, the global access to Internet information limits one to a fixed point of attachment such as offices, schools, homes, or hospitals that provides the required infrastructures. However, today introduction of Tablets, Smartphone (such as iPhones, IPad, etc.) and other sophisticated next generation dongles that are usually plug-and-play devices into any computer via USB port to give users constant broadband access everywhere has change the perception of the Internet.

2.1.1 Mobile Networks

The emergence of mobility in computer networking has posed several challenges to the network management. The internet is becoming complex. The internet itself is a network of networks based on many underlying hardware technologies, but unified by an internetworking protocol standard, the Internet Protocol Suite, often also referred to as TCP/IP. Terminal and user mobility have great impact in the OSI stack. The Open Systems Interconnection (OSI) model is a product of the International Organization for Standardization. It is a prescription of characterizing and standardizing the functions of a communications system in terms of abstraction layers. Experience revealed that this enabled mobility complicates information exchange within the seven layers of OSI (from the Physical layer to the Application layer). The ownership of network hardware and software is usually based on either individual or a group of communication organizations. A network owned by individual is referred to as private networks while the networks owned by common carriers are called public networks [8, p.272]. Figure 2.2 depicts the organization of internet application software and hardware.

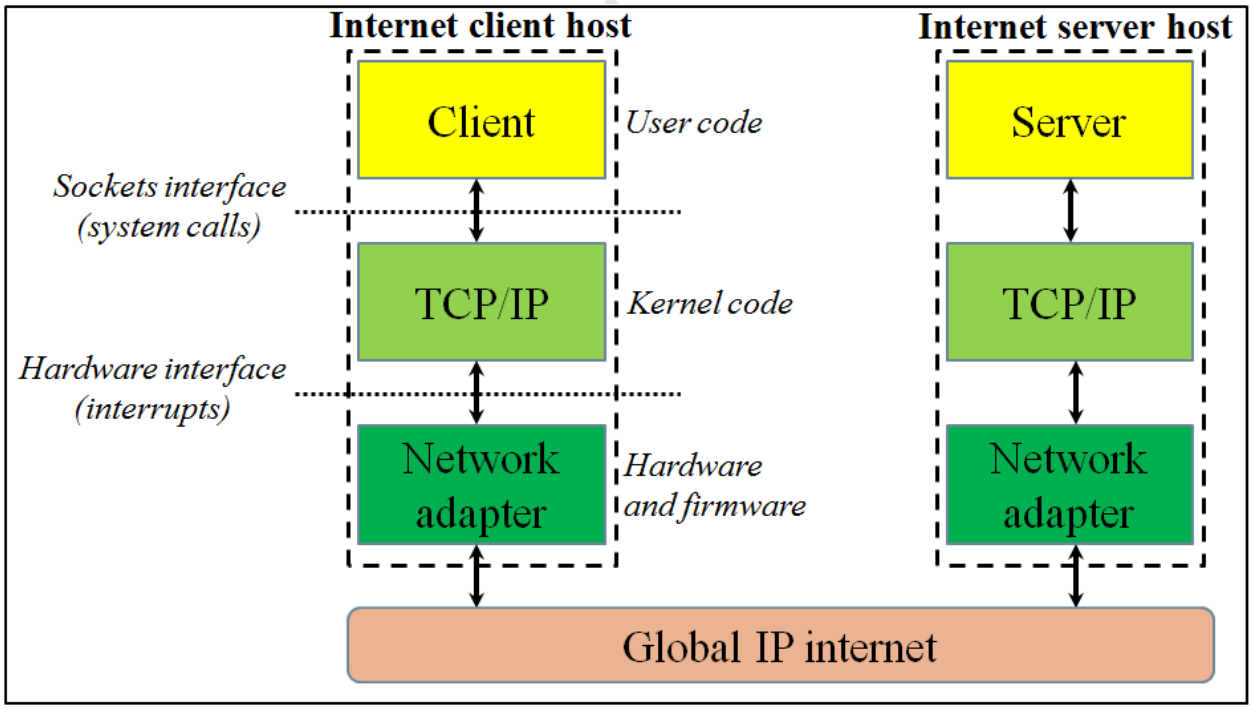

Figure 2.2. Organizing Internet Application.

The applications deployed in this environment should be robust enough to meet the changing network conditions due to number of competing systems running on different networks. To deliver system reliability, the need for efficient generalized solution that can monitor and manage heterogeneous network is very important. Network management is the process of maintaining and administering the network, including its intermediate nodes and links [27]. The management processes include:

- discovering network inventory;
- monitoring device health and status;
- providing alerts to conditions that affect system performance;
- identifying of problems, their source(s) and possible solution;
- providing network security.

Design Requirement 2 (DR-2): A generalized solution that can monitor and manage heterogeneous network is needed

The evolution of mobile network has offered so many global benefits including:

- users flexibility;
- cheaper implementation cost compared to traditional fixed networks;
- ideal for non-reachable places such as mountains, rural areas, and over the sea;
- granting Internet access anytime and anywhere.

2.1.2 Mobile Network Challenges

While the evolution of the mobile telephony system has offered huge success and benefits in the past, the mobile networking that appears to be similar with this operation inherent some flaws. In mobile telephony, the communication connectivity was often between end terminal users once connection is established (i.e., letting users to send and receive voice communication).

In contrast, provisioning of network applications such as internet protocol television (IPTV), virtual private network (VPN), and cloud computing is habitually across multiple network operators [28]. This implies the connectivity or communication is between machine or network components with little or no human interventions (i.e., equipped with customized software that combine computing). Specifying the interaction model becomes a complex configuration when the mobility is enabled since it involves Networks of Networks (Internetworking). Though users can define their preferences, they are not expected or interested in the internal operational activities, mandating network operators to hide network limitations such as intermittent connections and resource limitations from mobile terminal users. A major challenge with mobile network is intermittent network connection (maintaining connectivity while changing points of attachment). In a reliable mobile computing or networking environment, a user must not be interrupted or disrupted during information exchange when the point of attachment changes. In this case, re-establishment or rather reconnections of that user to new point of attachment need to be done automatically and inconspicuously. To effectively eliminate these limitations, the deployed application should be able to sense, interact, and adapt to network changes. Furthermore, the administrators require a network infrastructure to simultaneously provide lower costs, lower latency, and greater flexibility.

Design Requirement 3 (DR-3): Network operators must hide network resource limitations from mobile subscribers

2.2 Internetworking Strategies

Internetworking is the practice of connecting different networked computers or systems together through the use of gateways that provide a common method of routing information packets between the networks. The fundamental factor that necessitates interconnected networks is to connect disparate types of networking technology [29]. The gateway is the most often used network component to connect various networks. These interconnection gateways are in today's networking referred to as internet routers, and every router has its own model with a unique set of attributes. The Network layer (Layer 3) of the OSI model Implements the internetworking strategy. This is discussed in Section 2.3.1.

A number of network technologies currently exist to provide users with benefit services and each of this technology is designed to fit specific set of constraints. For example, a local area network (LAN) technology is a high-speed data network that covers a relatively small geographic area. It typically connects workstations, personal computers, printers, servers, and other devices. LANs offer computer users many advantages, including shared access to devices and applications, file exchange between connected users, and communication between users via electronic mail and other applications. In contrast, a wide area network (WAN) technology is a data communications network that covers a relatively broad geographic area and that often uses transmission facilities provided by common carriers, such as telephone companies. WAN technologies generally function at the lower three layers of the OSI reference model: the physical layer, the data link layer, and the network layer. No single network technology is best suited for all needs [10, p.270].

In this respect, multiple incompatible network technologies can be connected to create a large diverse networking. Routers are the basic computer hardware used to connecting these heterogeneous networks. A router is a special- purpose computer dedicated to the task of inter-connecting networks. Irrespective of the underlying technology, routers can connect different technologies that include different media, physical addressing schemes, or frame formats [10, p.273].

In this environment, the router acts as a routing switchboard that forward packets between the different network technologies. The router has a special element called routing table. A routing table is a data table stored in a router that lists the routes to particular network destinations, and in some cases, metrics (distances) associated with those routes. The routing table contains information about the topology of the network immediately around it; this provides a router with topology awareness. The construction of routing tables is the primary goal of routing protocols. When messages are forwarded from any of the segments within the network, the router first consults its routing table and then forwards the data to the destination segment according to its routing table. The router also implements the conversions of information received from different technologies as depicted in Figure 2.3.

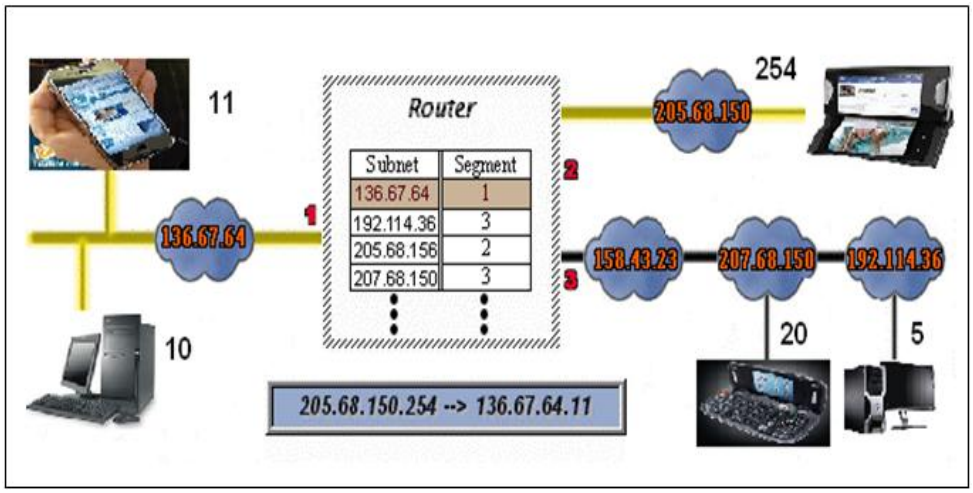

Figure 2.3. Router Internetworking [21].

In Figure 2.3, the router connects three different subnets that could be operating different technologies and belonging to different network providers. The scenario in subnet 3 is quite different. In this case, intermediate routers are used to connect the various networks together in order to provide terminal address 20 and 5 access to subnet 1 and 2. With this internetworking the network can scale rapidly to support end-users and applications. However, there are still some technical obstacles that need to be addressed to ensure a reliable and effective inter-connectivity. These include:

- Fault tolerance;
- Scalability;
- Quality of service (QoS);
- Security.

As internetworking evolves, the performance of existing users must not be affected. This internetworking depends on routers connected with redundant links between sources of messages to its destinations. Keeping at least one link between specified network nodes so that some or all of traffic between nodes is routed through becomes a major challenge. Failure of network component such as routers or links between routers can potentially bring down parts of the

network. The process of ensuring that messages can be instantly routed over a different link transparent to the users on either end can be referred to as *fault tolerance* and it's illustrated in Figure 2.4.

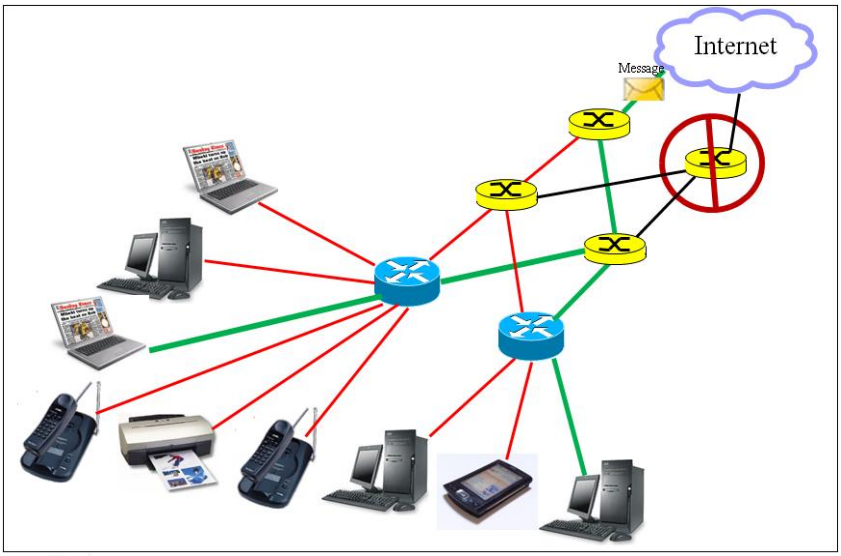

Figure 2.4. Fault tolerance [25].

Survivability in high capacity cooperative networks becomes very crucial [30]. As thousands of new users and service providers connect to the Internet on a daily basis, the ability of the network to support these new interconnections in a graceful manner is referred to as *scalability*. This depends on a hierarchical layered design for the underlying physical infrastructure and logical architecture. The operations at each layer enable users or service providers to be inserted without affecting the performance of the network. Technology developments are constantly increasing the message carrying capabilities and performance of the physical infrastructure components at every layer. These developments, along with new methods to identify and locate individual users within an internetwork, are enabling the Internet to provide users quality of experience.

Quality of service (QoS) guarantees are often important if the network capacity is insufficient, especially for real-time streaming multimedia applications such as video conferencing, mobile TV, capturing video for video sharing, video blogging, video Twitter, and video broadcasting applications while maintaining user's mobility. These applications often require fixed bit rate and

are delay sensitive. For these applications to adequately perform and be used widely, QoS must be quantified and managed.

Design Requirement 4 (DR-4): QoS must be quantified and managed

This project acknowledged security issue as another important concern when internetworking. Many works have and are currently focusing on providing security and ensuring privacy within the networks [31][32][33]; however, security is totally out of the scope of this research.

2.3 Cooperative Networks

Cooperative communications and networking represent a new paradigm, which involves both transmission and distributed processing, promising significant increase of capacity and diversity gain in wireless networks [34]. Cooperative networks are gaining an increasing interest in information and communications technologies since such networks can improve communication capability and provide a fertile environment for the development of context-aware services. As discussed in Section 2.2, different network technologies are designed to fit a specific set of constraints. Providing cooperation between these technologies such as LAN and WAN will improves the performance in terms of both area coverage and quality of service (QoS). In the same vein, the cooperation among nodes, as in the case of wireless sensor networks, allows a distributed space-time signal processing which enables environmental monitoring, localization techniques, distributed measurements, and others, with a reduced complexity or energy consumption per node [34].

The network layer provides infrastructure for delivering packets from source all the way to the destination, this may require many hops through intermediate routers. To achieve its goals the network layer must have knowledge about the topology of the communication subnet (gateway routers) and choose appropriate connectivity. It must carefully choose routers in other to avoid overloading some of the links and routers while others remain idle. When *source and destination* are in different networks as prescribed in CoNet, the set of feasible solutions can significantly increase; identifying the optimal solution becomes a complex optimization challenges since it

involves many parameters (such as bandwidth requirements, pricing scheme, quality of service etc.). In CoNet environment, NPs should have full internetworking strategy ("loosely collaborative" fashion) as such gateway routers must agree to forward information from source on one network to a specified destination on another network in other to achieve a universal services across heterogeneous networks [24].

Design Requirement 5 (DR-5): There is need for interoperability and proactive monitoring mechanism to complement the coordination of network resources in a distributed system

2.3.1 Network Layer

The network layer is the number three layer of the OSI layering model that provides:

- switching and routing technologies for packet forwarding;
- creating logical paths, known as virtual circuits, for transmitting data from node to node.

Routing and forwarding are functions of this layer, as well as addressing, internetworking, error handling, congestion control and packet sequencing. Other layers of the OSI layering model are Application layer, Presentation layer, Session layer, Transport layer, Data link layer and the Physical layer. Figure 2.5 depicts the structure of the OSI layering model.

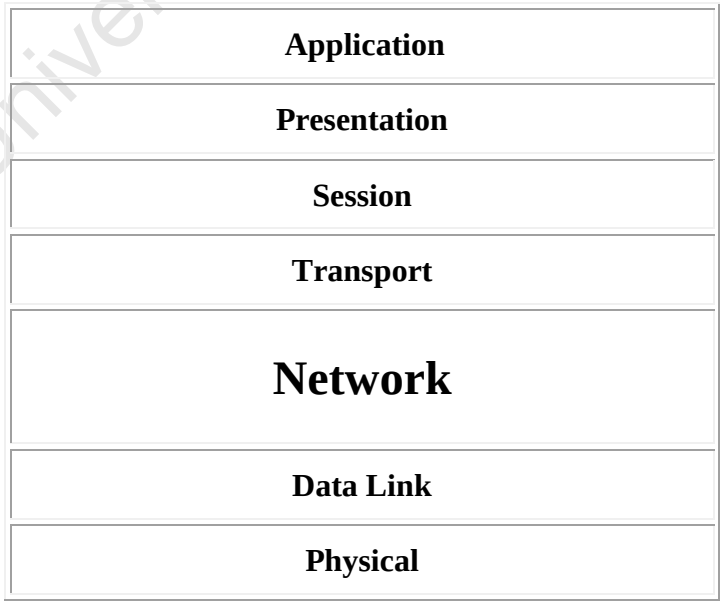

Figure 2.5. OSI layering model.

Considering the information provided in [35], the network layer serves as the backbone of the Internet. The premier network layer protocol is internet protocol (IP). The Domain Name System (DNS) removes the burden of remembering IP addresses by associating them with real names. The Dynamic Host Configuration Protocol (DHCP) eases the administration of IP addresses and is used extensively by network administrators and Internet service providers (ISPs). Routing protocols such as Open Shortest Path First (OSPF), Routing Information Protocol (RIP), and Border Gateway Protocol (BGP) provide information for network layer devices to direct data traffic to the intended destination.

With the routers causing information flow constriction (i.e., moving data flows off local subnets and onto the routed network, where the limitations of router performance increasingly led to bottlenecks), IT managers became increasingly reluctant to deploy new, enabling technologies, such as multicast-based applications and *middleware* [35].

2.3.2 Application Layer

Application layer is the top layer of the OSI (Open System Interconnection) server layer model. It provides services for an application program to ensure the possibility of different network application program to communicate effectively. The application layer is not the application itself that is doing the communication [36]. The service layer provides these services:

- makes sure that the other party is identified and can be reached;
- if appropriate, authenticates either the message sender or receiver or both;
- makes sure that necessary communication resources exist (for example, is there a modem in the sender's computer?);
- ensures agreement at both ends about error recovery procedures, data integrity, and privacy;
- determines protocol and data syntax rules at the application level.

Service layer is a conceptual layer within service provider architecture of the network. It aims at providing middleware that serves third-party value-added services and applications at a higher

application layer. The service layer provides capability servers owned by a telecommunication network SP, accessed through open and secure Application Programming Interfaces (APIs) by application layer servers owned by third-party content providers. The service layer also provides an interface to core networks at a lower resource layer [37].

2.4 Cooperative Connectivity

The context of open access network (OAN) incorporated in next generation networks (NGN), where a common access operator unifies multiple network operators will support mobile terminal users to effectively utilize their device functionalities. The widespread of Smartphones (such as iPhone, Tablets, iPads, etc.) and other sophisticated dongles that are usually equipped with multiple wireless interfaces is enabling users' connectivity to roam seamlessly across multiple network operators. To get meaningful, accurate, and timely information, network components must cooperate internally and externally to take full advantage of what is available. In this regards, mobile terminal users will benefit the connectivity opportunities provided by many infrastructure-based components, which tend to be ubiquitously available. However, ensuring connectivity between all the entities of a network in a consistent manner for any service as described in Section 2.3 is far from straightforward. This requires consistent support for device mobility, QoS, authentication, authorization, and accounting (AAA) etc. Other than application components, suitable abstractions and tools are needed to model and enable effective reliable global access to services of the content provider (CP). The connectivity layer provides cooperation across various realizations of networks, called cooperative connectivity, and shall be independent of the various network technologies used to link the nodes of the network together [38].

Design Requirement 6 (DR-6): Network components should cooperate internally and externally to take full advantage of available services

By separating access and transport, the CoNet architecture makes it transparent to the common and standardized transport infrastructure and hides the technology from the end user, while facilitating the most efficient usage of spectrum resources. Irrespective of the access technologies used by the various network operators, the CoNet architecture should support both

session continuation and simultaneous usage of several access networks. The user roaming ability across these different networks must also be automated that is with little or zero human intervention. This mandates the need for an effective user-network-interface (UNI) that provides user connectivity and search best suited network according to users preferences.

2.5 Network Management Approaches

In today's complex network of routers, switches, and servers, that emerged from heterogeneity support in next generation networks (NGN), optimizing network performance while guaranteeing quality of experience seems a major challenge to telecommunication systems. The various network applications such as cloud computing, Internet protocol television (IPTV), etc., are often executed across multiple network operators [39], which led to the need for network management to be robust enough in meeting the dynamic changing requirement of such systems. Management of service level agreement-sensitive application across multiple operator domains is far from straightforward. A mechanism such as network monitoring is needed for signaling processes and implementing the network management decisions. Network monitor allows surveying of the current state and behaviour of the network equipment and possibly responding to different events in the network more quickly to solve non-standard behaviour of the network devices [40]. The commonly used mechanisms and approaches in delivering the required network performance managements are described in the following Sections.

2.5.1 Simple Network Management Protocol (SNMP)

SNMP was introduced in 1988 [41] to meet the growing need for a standard for managing Internet Protocol (IP) devices. SNMP provides its users with a "simple" set of operations that allows these devices to be managed remotely [42]. Since its creation in 1988 as a short-term solution to manage elements in the growing Internet and other attached networks, SNMP has achieved widespread acceptance. SNMP is based on the manager/agent model and uses an SNMP Management Information Base (MIB) and a relatively small set of commands to exchange information [43]. SNMP consist of set of network protocol and has the following basic components:

- SNMP manager;
- SNMP agent;
- Managed SNMP devices;
- Management information base (MIB).

2.5.1.1 SNMP Manager

A manager or management system is a separate entity that is responsible to communicate with the SNMP agent implemented network devices. This is typically a computer that is used to run one or more network management systems (NMS) [44].

SNMP Manager's key functions are:

- querying agents;
- getting responses from agents;
- setting variables in agents;
- acknowledging asynchronous events from agents.

2.5.1.2 SNMP Agent

The agent is a program that is packaged within the network element. Enabling the agent to collect the management information database from the device locally and makes it available to the SNMP manager, when it is queried for. These agents could be standard (e.g. Net-SNMP) or specific to a vendor (e.g. HP insight agent) [44]

SNMP agent's key functions are:

- collecting management information about its local environment;
- storing and retrieving management information as defined in the MIB;
- signaling an event to the manager;

- acting as a proxy for some non–SNMP manageable network node.

2.5.1.3 Managed Devices

A managed device or the network element is a part of the network that requires some form of monitoring and management e.g. routers, switches, servers, workstations, printers, UPSs, etc.

The communication model of SNMP is shown in Figure 2.6.

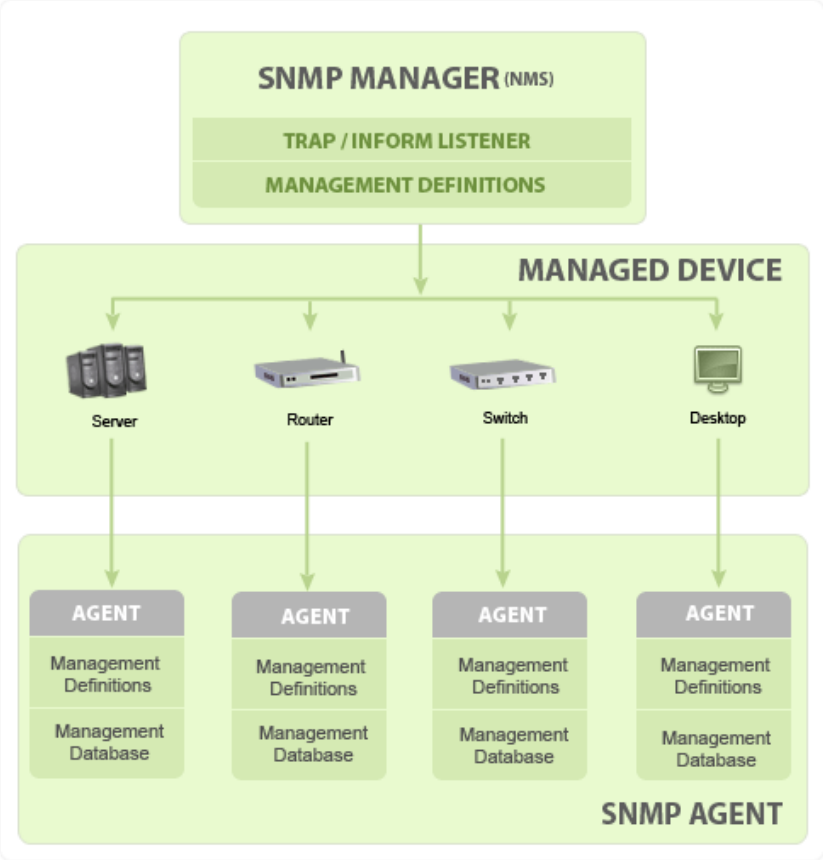

Figure 2.6. SNMP Communication Model [44].

SNMP usually is associated with managing routers, but it is important to understand that it can also apply to the management of many types of devices [42].

2.5.1.4 Management Information Base (MIB)

The Management Information Base (MIB) can be thought of as a database of managed objects that the agent tracks. Any sort of status or statistical information that can be accessed by

the NMS is defined in a MIB. The SMI provides a way to define managed objects, while the MIB is the definition (using the SMI syntax) of the objects themselves. Like a dictionary, which shows how to spell a word and then gives its meaning or definition, a MIB defines a textual name for a managed object and explains its meaning. The MIB entries are organized into a treelike hierarchy that enables to provide the MIB database into several independent branches [42]. The entire tree of MIBs is referred to as a namespace, which means that each managed object is referred to by a unique Object Identifier (OID). Depicted in Figure 2.7 is the structure of MIB tree diagram.

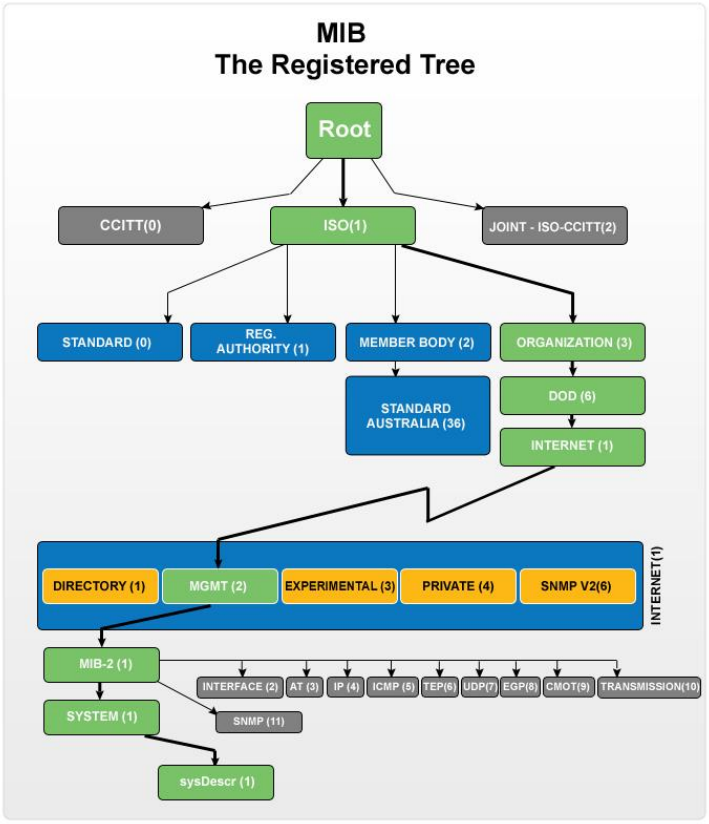

Figure 2.7. MIB tree [44].

2.5.1.5 SNMP Versions and Standards

As reported in RFC 1109, Report of the Second Ad Hoc Network Management Review Group [45], the requirements of the SNMP and the OSI network management frameworks were more different from anticipated. As such, the requirement for compatibility between the SMI/MIB and both frameworks was suspended. This action permitted the operational network

management framework, the SNMP, to respond to new operational needs in the Internet community by producing documents defining new MIB items.

The current SNMP versions and the IETF status of each are listed below:

- SNMP Version 1 (SNMPv1) is the current standard version of the SNMP protocol. It is defined in RFC 1157 and is a full IETF standard. SNMPv1 security is based on communities, which are nothing more than passwords: plain-text strings that allow any SNMP-based application that knows the strings to gain access to device's management information [42].

- SNMP Version 2 (SNMPv2) is often referred to as community string-based SNMPv2. This version of SNMP is technically called SNMPv2c. It is defined in RFC 1905, RFC 1906, and RFC 1907, and is an experimental IETF [42].

- SNMP Version 3 (SNMPv3) is an interoperable standards-based protocol for network management. SNMPv3 provides secure access to devices by a combination of authenticating and encrypting packets over the network. The security features provided in SNMPv3 are [46]:

    - ✓ Message integrity: ensuring that a packet has not been tampered with in-transit.
    - ✓ Authentication: determining the message is from a valid source.
    - ✓ Encryption: scrambling the contents of a packet prevent it from being seen by an unauthorized source.

2.5.1.6 SNMP Basic Commands

For gathering and changing management information in the network devices, SNMP uses five basic commands operations (GET, GET-NEXT, GET-RESPONSE, SET, and TRAP) to communicate between the SNMP manager and the SNMP agent. The GET and GET-NEXT messages allow the manager to request information for a specific variable [43].

The agent, upon receiving a GET or GET-NEXT message, will issue a GET-RESPONSE message to the SNMP manager with either the information requested or an error indication as to why the request cannot be processed. A SET message allows the SNMP manager to request a

change be made to the value of a specific variable in the case of an alarm remote that will operate a relay. The SNMP agent will then respond with a GET-RESPONSE message indicating the change has been made or an error indication as to why the change cannot be made. The SNMP TRAP message allows the agent to spontaneously inform the SNMP manager of an "important" event. A trap is a way for the agent to tell the NMS that something has happened. Traps are sent asynchronously, not in response to queries from the NMS. The NMS is further responsible for performing an action based upon the information it receives from the agent [24]. The relationship between the SNMP manager (NMS) and the SNMP agent is displayed in Figure 2.8.

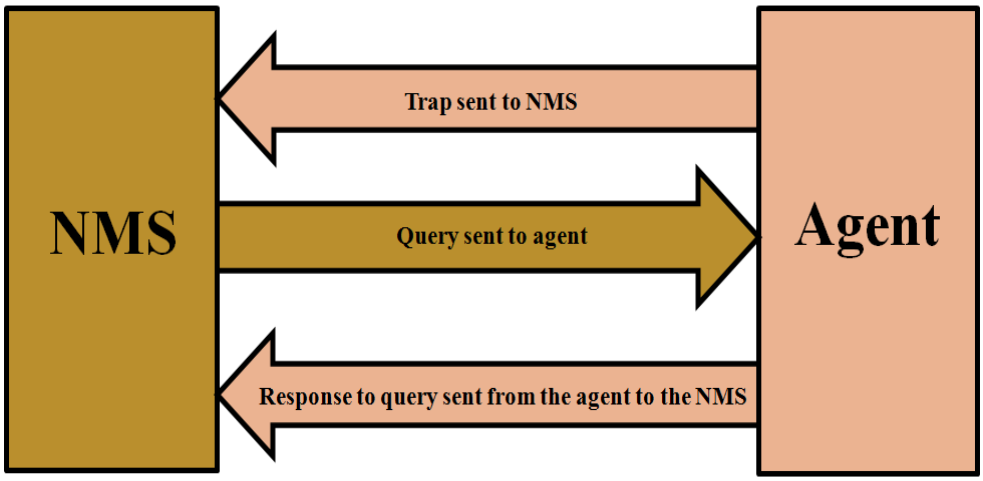

Figure 2.8. Communication platform between the SNMP agent and SNMP Manager.

The TRAP message is the only message capable of being initiated by an SNMP agent, it is the message used by DPS Remote Telemetry Units (RTUs) to report alarms [43].

2.5.1.7 Remote Network Monitoring (RMON)

Another aspect of network management is the network monitoring; that is, monitoring an entire network as opposed to individual routers, hosts, and other devices. Remote Network Monitoring (RMON) was developed to help in understanding how the network itself is functions, as well as how the individual devices on the network are affecting the network as a whole. It can be used to monitor not only LAN traffic, but WAN interfaces as well. Remote Monitoring Version 1 (RMONv1 or RMON) is defined in RFC 2819 [42]; an enhanced version of the standard, called RMON Version 2 (RMONv2) is defined in RFC 2021. RMONv1 provides the

network management station (NMS) with packet-level statistics about an entire LAN or WAN. NMS is a combination of hardware and software used to monitor and manages the network elements, also called managed devices. RMONv2 builds on RMONv1 by providing network- and application-level statistics. These statistics can be gathered in several ways. One way is to place an RMON probe on every network segment that is under investigation.

Design Requirement 7 (DR-7): Other than application components, suitable network monitors are required on every network segment that is under investigation

2.5.1.8 SNMP and UDP

SNMP uses the User Datagram Protocol (UDP) as the transport protocol for passing data between managers and agents [42]. UDP, defined in RFC 768, was chosen over the Transmission Control Protocol (TCP) because it is connectionless; that is, no end-to-end connection is made between the agent and the NMS when datagrams (packets) are sent back and forth. This aspect of UDP makes it unreliable, since there is no acknowledgment of lost datagrams at the protocol level.

The basis for any device to communicate on the Internet (e.g., Windows NT systems, UNIX servers, Cisco routers, etc.) relies on the protocol suite, which is the basis for all TCP/IP communication. This model is often referred to as a protocol stack, since each layer uses the information from the layer directly below it and provides a service to the layer directly above it as depicted in Figure 2.9.

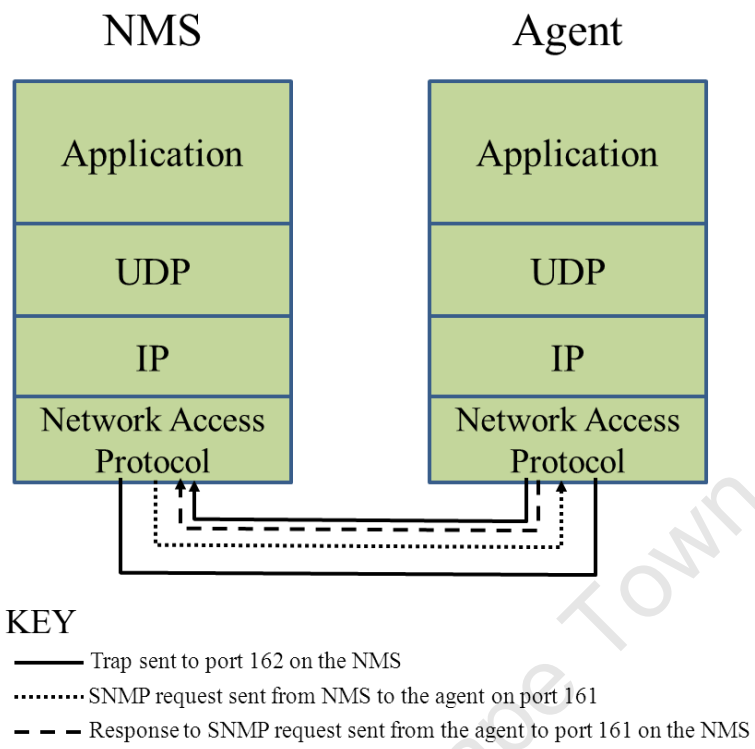

Figure 2.9. TCP/IP communication model and SNMP [42].

The following events occur in the protocol stack whenever an NMS or an agent wishes to perform an SNMP function (e.g., a request or trap):

- Application: First, the actual SNMP application (NMS or agent) decides what it is going to do. For example, it can send an SNMP request to an agent, send a response to an SNMP request (this would be sent from the agent), or send a trap to an NMS. The application layer provides services to an end user, such as an operator requesting status information for a port on an Ethernet switch.

- UDP: The next layer, UDP, allows two hosts to communicate with one another. The UDP header contains, among other things, the destination port of the device to which it is sending the request or trap. The destination port will either be 161 (query) or 162 (trap).

- IP: The IP layer tries to deliver the SNMP packet to its intended destination, as specified by its IP address.

- Medium Access Control (MAC): The final event that must occur for an SNMP packet to reach its destination is for it to be handed off to the physical network, where it can be routed to its final destination. The MAC layer is comprised of the actual hardware and device drivers that put the data onto a physical piece of wire, such as an Ethernet card. The MAC layer also is responsible for receiving packets from the physical network and sending them back up the protocol stack so they can be processed by the application layer (SNMP, in this case).

For a better understanding, let's consider everyday scenario. Assuming a student at the University of Cape Town need to send a letter to his supervisor in Pretoria informing him the progress made in the past weeks. The decision to send the progress report acts as *SNMP application*. The next phase after writing the report is to fill out the envelope with the supervisor address in Pretoria, which is equivalent to the function of the *UDP layer* that records the packet's destination port in the UDP header. At the post office, there is need to place a stamp on the envelope and drop it in the mailbox for the postmaster to pick up; this is equivalent to the *IP layer's* function. The postmaster picks up the envelope from the mailbox and route it to its final destination (supervisor address). The MAC layer of a computer network is equivalent to the mail trucks and airplanes that carry your letter on its way. On delivery of the letter to the supervisor, the supervisor reads the report and sends a feedback through the same procedure.

2.5.1.9 SNMP Limitations

While extremely useful as a tool in network management, concerns have been expressed in the use of SNMP for managing large networks [47]. These deficiencies include:

- high-overhead to retrieve large blocks of data;
- heavy reliance on polling;
- unreliable trap delivery that can result in critical events being missed;
- limited security, and no ability to support manager to manager communications.

The research communities for its centralized nature, static management capabilities and

simple-minded model have heavily criticized the approach [48]. Some of the major challenges with the SNMP approach are:

- excessive processing load at manager;
- heavy usage of network bandwidth;
- information bottleneck at NMS when network grows significantly;
- lack of scalability.

With the transition from centralized computing systems to the distributed scheme, system and network managers became faced with a growing problem. Instead of a single, centrally managed system, systems have become widely distributed, with the importance and sophistication of each component increasing. Issues such as availability, performance, fault identification and diagnosis became greater challenges to system administrators. In the extreme case, the exponential growth of the Internet presented a clear need for standard management approaches. Since SNMP needs to manage multiple devices manufactured by different vendors, the parameters that it can manage are quite limited, based on the standards defined. However, network devices are always very different from each other. Some are complex. Therefore, the user cannot define customization for management of particular elements (i.e. inability for users to incorporate their management preferences).

Design Requirement 8 (DR-8): Design of applications must be centered on user's preferences

2.5.2 OSI-SM Systems Management Functions (SMFs)

The OSI-SM defines management architecture with well-defined organizational, informational, communication and functional models [49]. This model also implements the similar SNMP manager and agent management role. A manager is an entity that controls the management process and makes decisions based on collected information, whereas the agents make available the management information to managers [50]. Similar to already analyzed SNMAP, the communication between the manager and the agents takes place via standardized management protocols. In general, the manager-agent paradigm can also be thought of as

client/server (CS) relationship, where the manager plays the role of the client and the agent the role of the server. The basic architecture of OS-SM is depicted in Figure 2.10.

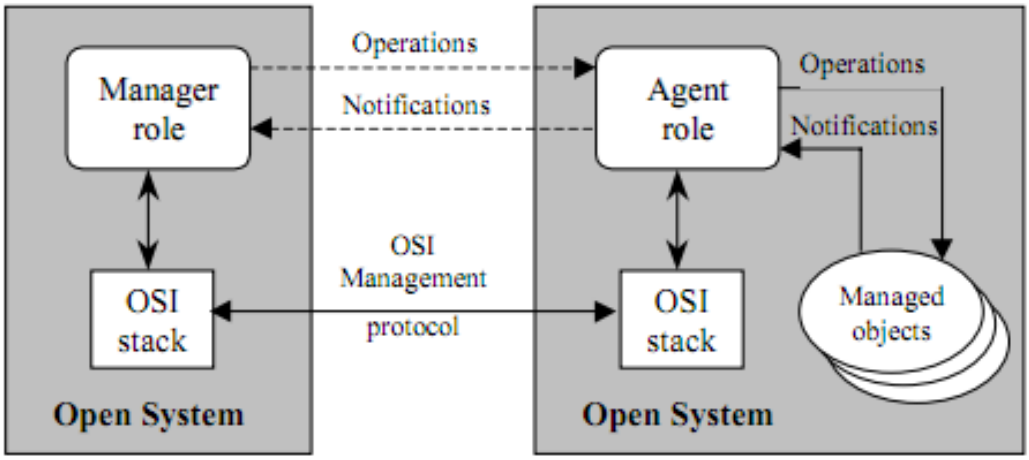

Figure 2.10. Basic OSI-SM architecture [51]

The exchange of management information between managers and agents is defined by a service, the Common Management Information Service (CMIS), and its protocol, the Common Management Information Protocol (CMIP) [52]. The CMIS provides management operation primitives that include M-GET to retrieve data, M-SET to modify data, M-ACTION to request the execution of an action, M-CREATE (M-DELETE) to request the creation (deletion) of an instance of a managed object, and M-CANCEL-GET to cancel an outstanding M-GET request [50]. Agents may report events about managed objects using M-EVENT-REPORT.

Both the SNMP and OSI-SM is a communication framework, where the management functions typically reside outside the network (in management stations and servers). These management functions interacts with network elements and devices via network protocols for management, in order to execute management tasks, including fault, configuration, accounting, performance, and security management, or, short (FCAPS). The ever-increasing demand for capacity and quality of service in wireless communication links has continued to compelled engineers, researchers, and telecommunication standardization bodies to innovate new design methodologies and concepts over wireless system and network with the aim of achieving a

successful transition to the NGN. It is envisaged, there will be additional challenges in meeting the bandwidth requirements and network management due to the proliferation of new information and communications technology (ICT) services and application [53]. In the past years, different methodologies have emerged to solve these challenges. One of such was the programming paradigm, Distributed Objects Technologies (DOT).

2.5.3 Distributed Objects Technologies (DOT)

Distributed Objects Technologies (DOT) defines an object-oriented paradigm, where objects can interact even if they do not reside on the same system. Interoperability of this paradigm and the legacy SNMP and the OSI-SM became a major concern for years until the introduction of gateways that seamlessly performs the conversion mechanism. CORBA and Java RMI are one of DOT representatives.

2.5.3.1 Common Object Request Broker Architecture (CORBA)

CORBA is a standard defined by the Object Management Group (OMG), which enables separate software components written in multiple programming languages and running on multiple computers to work together hence supporting multiple platforms. CORBA products provide a framework for the development and execution of distributed applications.

However, coordinating a distributed scheme is not a trivial task. The nature of certain application being distributed over multiple servers leaves administrator to choose. For instance, different administrators defined their data and provides certain access conditions such as “access remotely not storage” on their server. If the application to be accessed requires execution of such multiple servers, DOT seems a possible solution. The OMG has approached the problem of handling the interaction of distributed components by creating interface specifications, and not code. Distributed components of the system are able to describe their interfaces using the Interface Definition Language (IDL) and subsequently inter-operate through the underlying Object Request Broker (ORB). There is specifically IDL compiler that comes with CORBA implementations, which converts the user's IDL code into some language-specific generated code. A traditional compiler then compiles the generated code to create the linkable-object files for the application. Namely, the ORB provides the communication backbone through which

distributed components are able to interact. Depicted in Fig. 2.9 is the Usage of generated code in CORBA infrastructure.

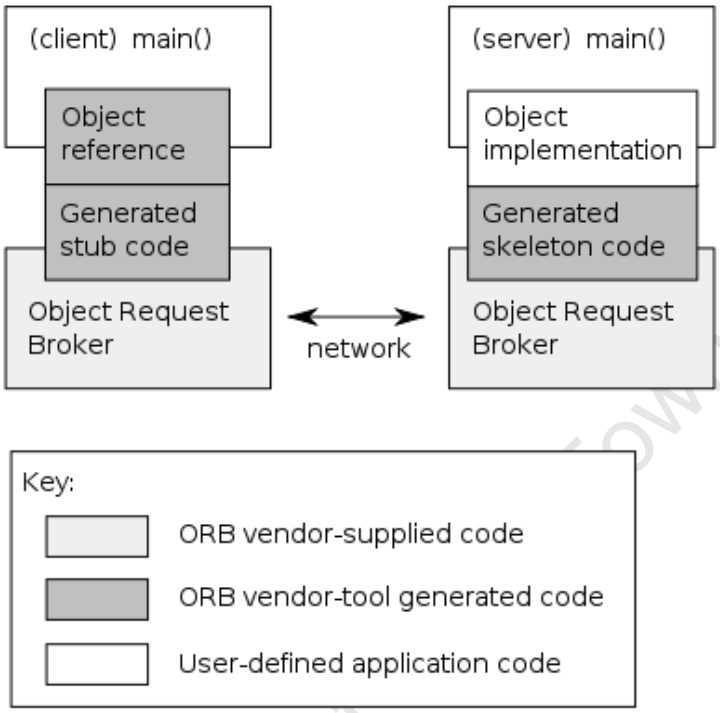

Figure 2.11. CORBA Infrastructure [54].

To perform requests or return replies, objects use a generic remote procedure call (RPC-like) request/response protocol, the General Inter-ORB Protocol (GIOP) or its TCP/IP mapping, the Internet Inter-ORB Protocol (IIOP).

2.5.3.2 Java Remote Method Invocation (Java RMI)

Another DOT representative is the Java Remote Method Invocation (RMI). Java RMI enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. RMI uses object serialization to marshal and un-marshal parameters and does not truncate types, supporting true object-oriented polymorphism [51]. The Java RMI mechanism also based on object-oriented RPC mechanism as it is in CORBA but differs from CORBA in the following aspects:

- CORBA is a language-independent standard and RMI is Java-centric system.
- CORBA includes many other mechanisms in its standard (such as a standard for TP monitors) none of which is part of Java RMI.
- CORBA rely on centralized information stores for such things as the name service, the trading service and implementation repositories, which represent performance bottleneck and single point of failure.
- Java (RMI) allows objects to invoke methods on remote objects using the same syntax as for local invocations.

Java RMI provides great flexibility as it allows methods to pass objects that a foreign virtual machine has never encountered before, allowing dynamic loading of new classes as required. Three processes participate in supporting remote method invocation.

- Client is the process that is invoking a method on a remote object.
- Server is the process that owns the remote object. The remote object is an ordinary object in the address space of the server process.
- Object Registry is a name server that relates objects with names. Objects register with the Object Registry. Once registered, one can use the Object Registry to obtain access to a remote object using the name of the object.

The relationship between client, server, and the object registry is depicted in Figure 2.12.

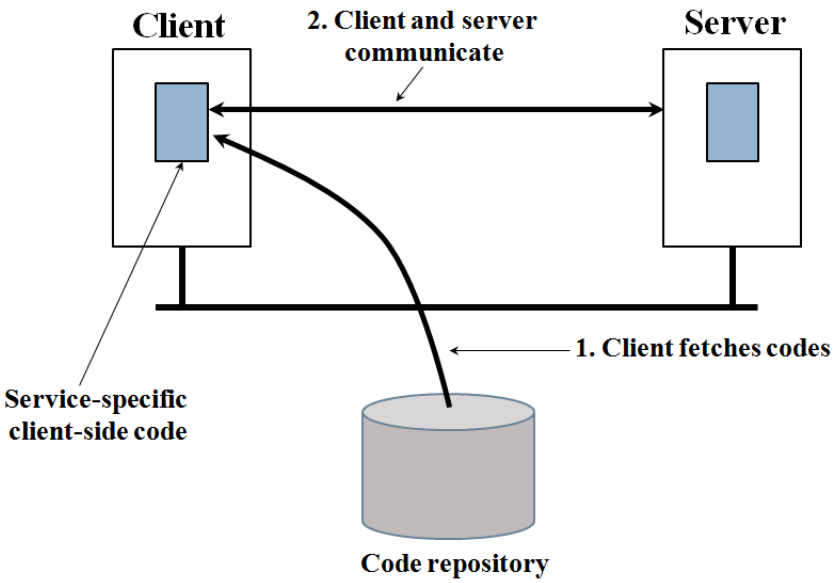

Figure 2.12. Client communication using Java RMI.

In the view of Future Internet activities in the research communities around the globe, the network management of the Future Internet is of major concern, as application requires greater self-management, more automation of the management, and easier use of management tools. The next generation of mobile communications systems is orientated towards the integration of available wireless access technologies, enabling user to roam seamlessly across heterogeneous networks in a way that best reflects her preferences, application requirements and terminal capabilities [56]. However, some of the outputs or internal states growing without the network bounds characterize these users' constant changes in location. Each of these users' interactions generates network traffic; higher bursty traffic can cause larger queue size, hence resulting in long packet delays and high dropping packets, thus affects resource utilization such as bandwidth, computational power, memory, etc. [57]. Severe traffic congestion is also likely to occur during run-time if these systems architecture are not modeled in advance.

Services that are currently provided by multiple specific network-centric architectures are unified toward a single, converged, user centric communication network [58]. Furthermore, [9] describes future wireless networks as evolution towards the traded resource service architecture (TRSA) model whereby network resources such as, bandwidth, buffer space, and computational power will be traded in the same manner as existing commodities. The high increase in the

number of new users and the appearance of new resource consuming applications rises the necessity of make an efficient use of the available network resource [59]. The operations are becoming complex. Additionally, the unpredictable behaviour of wireless channels over time provokes unpredictable loss of coverage impeding the MT (users) communication abilities temporarily. This means there is high probability of user disconnection during transmission if coverage is lost within certain time [60]. Hence, the process of node engagement needed to be highly emphasized. The reliability of network components such as routers and links affects not only the reliability of the schedule network task but also the quality and cost. The key way to achieving these challenges is the adoption of advanced control and management approaches such as the mobile agent technology.

2.5.4 Mobile Agent Systems

Mobile agent based network monitoring and management is a new paradigm to address these limitations incurred by the previously analyzed network management approaches. Mobile agent technology provides a solution to the flexible management of telecommunication systems [61]. Mobile agents are software programs that administrator often distributed over the network with the ability to sense the status of the network and act upon it using its knowledge base. This allows processes to migrate, in some cases split into multiple instances and executes on different machines that provides agent hosting capabilities, and return to their host. In addition, mobile agents roaming the Internet could search for information, find exiting deals on service efficiency, and interact with other agents that also roam networks (and converge at a point) or as well restrict itself to a particular machine. The concept of mobile agent monitoring decentralizes the monitoring and management task in order to eliminate information bottleneck around NMS. Mobile agent can update its information base while executing other management task as oppose the mandatory communication between the SNMP manager and SNMP agent and OSI-SM. The mobile agent is a software module that travels to different nodes of network, collecting status information and carrying out device control task. Mobile agent can reduce network traffic and easily support disconnected operations [61]. Described in Figure 2.13 is the network monitoring comparison between the traditional centralized management approach and the mobile agent (MA) paradigm.

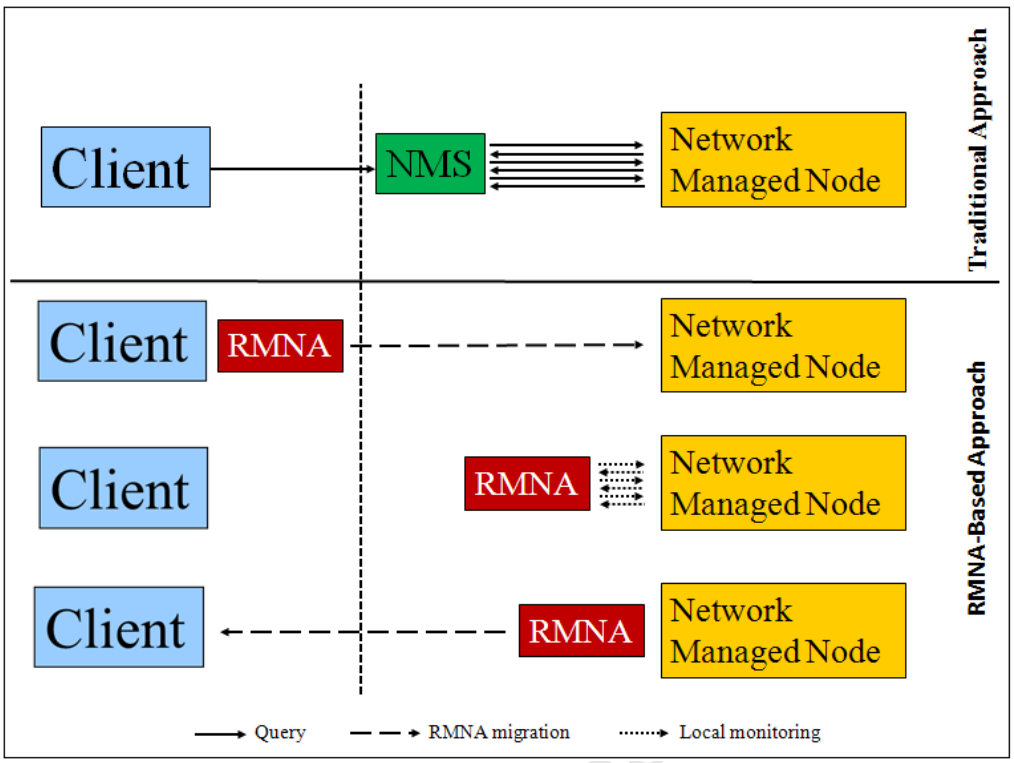

Figure 2.13. Network monitoring mechanisms (comparison).

One of the commercial application areas in which the added value of mobile agents is very high, is large-scale distributed or decentralized system integration with highly adaptive and dynamic business logic [62]. Existing solutions are generally centralized, pulling everything onto one platform, limiting the complexity and changes that can be handled. A decentralized agent approach divides and conquers complexity by pushing a large part of the business logic out onto source systems so that much monitoring and aggregation can be done on each. This distributes workload and increases robustness because the local processing can be performed independently of other systems, resulting in fewer and more relevant interactions with these systems, at a higher level of abstraction.

Mobile agent applications are expected to take an important role in the future wireless networking environment. However, supporting user personalization and the traditional mobile agent performance implications (overhead) are becoming critical issues. This thesis specifically focuses on implementing distributed control agent that enables constant lightweight agents to efficiently provide reduced signaling performance overhead as network size increases.

2.6 Chapter Summary

After a broad view on mobile network characterization, the following set of design requirements provides guidance to the proposed RMNA framework implementation.

Design Requirement 1 (DR-1): A unique addressing scheme is required to effectively create coordination and manageable networking environment

Design Requirement 2 (DR-2): A generalized solution that can monitor and manage heterogeneous network is needed

Design Requirement 3 (DR-3): Network operators must hide network resource limitations from mobile subscribers

Design Requirement 4 (DR-4): QoS must be quantified and managed

Design Requirement 5 (DR-5): There is need for interoperability and proactive monitoring mechanism to complement the coordination of network resources in a distributed system

Design Requirement 6 (DR-6): Network components should cooperate internally and externally to take full advantage of available services

Design Requirement 7 (DR-7): Other than application components, suitable network monitors are required on every network segment that is under investigation

Design Requirement 8 (DR-8): Design of applications must be centered on user's preferences

These design requirements are generated based on previous limitations with the existing network management scheme and the expected complications with future mobile networking. Through the use of coordination processes, the design requirements of this thesis will cater for the challenges faced with data distribution, unprecedented behaviour of network components and changes that occurs due to device mobility.

Chapter 3 Designing RMNA for Resource Coordination

An inherent characteristic of future mobile and wireless networking will be their support for heterogeneity, which requires network operators to have greater knowledge and increased training to cope with network management. It is envisaged that the management application will require the collection of large quantities of data from the networks: analyzing data before management activities can be initiated becomes essential [63]. To implement a complete integrated system, software agent techniques are important in advance. Agent technology has been the subject of extensive discussions and investigations within the scientific community for many years [64, p.1]. Regarding the network resource coordination processes, this project undertakes the following steps to enable a systematic flexible solution:

- Planning phase;
- Analysis phase;
- Design phase;
- Implementation phase.

3.1 Planning Phase

After a broad review on network resource management techniques described in previous Chapter, this project considers the deployment of mobile agents as suitable solution to the envisaged complexities that constitutes future mobile network environment. The process of deciding which management approach will best suit the coordination of heterogeneous network resources is known as network *planning*. Based on this research finding from Chapter 2, the advent of mobile agent for network resource management will enable quality of service (QoS), reduce hardware requirements, lower total cost of ownership (TCO) and increase user quality of experience.

3.2 Analysis Phase

This phase is to clarify the problem without any (or minimal) concerns about the solution. This is the task of analysis and it is generally an abstraction of the underlying development platform. This phase enable the adoption of another tool (JavaSniffer) that serve as performance visualizing tools to compliment the shortfalls of JADE default SnifferAgent, more details about JavaSniffer is discussed in Section 4.4.2. The analysis phase also provides the following functionalities:

- Mobile agents rules and behavioral diagram;
- consistently observes if the mobile agents are conforming to requirements;
- ensure the deployment diagram reflects the right direction.

3.3 Design Phase

The planning and analysis phase are considered as prerequisite that actually define and specify the Design phase. The design phase requires utmost attention, as it provides detailed information that is sufficient to have a relatively straightforward transition to the implementation. The implementation phase is considered as Chapter 4 of this thesis. Based on this research finding, the formalization of the design and implementation phases of network resource coordination processes specifically focuses on JADE platform as the mobile agent development and implementation framework, and the concepts provided by it. To create easy understanding for new JADE users, the design phase begins by describing the basic JADE features.

3.3.1 Introduction to JADE

Java Agent DEvelopment (JADE) framework has been developed by the Telecom Italia Lab (TILAB) in Italy since late 1998 [64, p.29]. JADE is an open source development framework devoted to multi-agent systems and applications. JADE is a completely distributed middleware system with flexible infrastructure allowing easy extension with add-on modules. The framework facilitates the development of complete agent-based applications by means of a run-time

environment implementing the life-cycle support features required by agents, the core logic of agents themselves, and a rich suite of graphical tools [64, p.1]. JADE is a software environment to build agent systems over a well-known object-oriented language (JAVA) for the management of networked information resources in compliance with the IEEE FIPA specifications, which provide interoperable multi-agent systems [65][66]. The JADE Agent Platform complies with FIPA specifications and includes all those mandatory components that manage the platform, which is the Agent Communication Channel (ACC), the Agent Management System (AMS), and the Directory Facilitator (DF) [67]. Foundation for Intelligent Physical Agents (FIPA) is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies [68]. The full FIPA communication model has been implemented and its components have been clearly distinct and fully integrated: interaction protocols, envelope, agent communication language (ACL), content languages, encoding schemes, ontology and, finally, transport protocols [67]. It is interesting to note that JADE contributed to widespread diffusion of the FIPA specifications by providing a set of software abstractions and tools that hid the specifications themselves; programmers could essentially implement according to the specifications without the need to study them [64, p.30].

Other than JADE compliance with FIPA specification, its component also offers beyond FIPA specification. For example, JADE provides a distributed, fault tolerant, container architecture, internal service architecture, persistent message delivery, semantic framework, security mechanisms, agent mobility support, web-service interaction, graphical interfaces, and much more [64, p.27]. These are some of the main benefits of choosing JADE platform in this project.

3.3.1.1 JADE Architecture

JADE platform is composed of agent containers that can be distributed over the network. Agents live in containers, which are the Java processes that provide the JADE run-time, and all the services needed for hosting and executing agents as depicted in Figure 3.1.

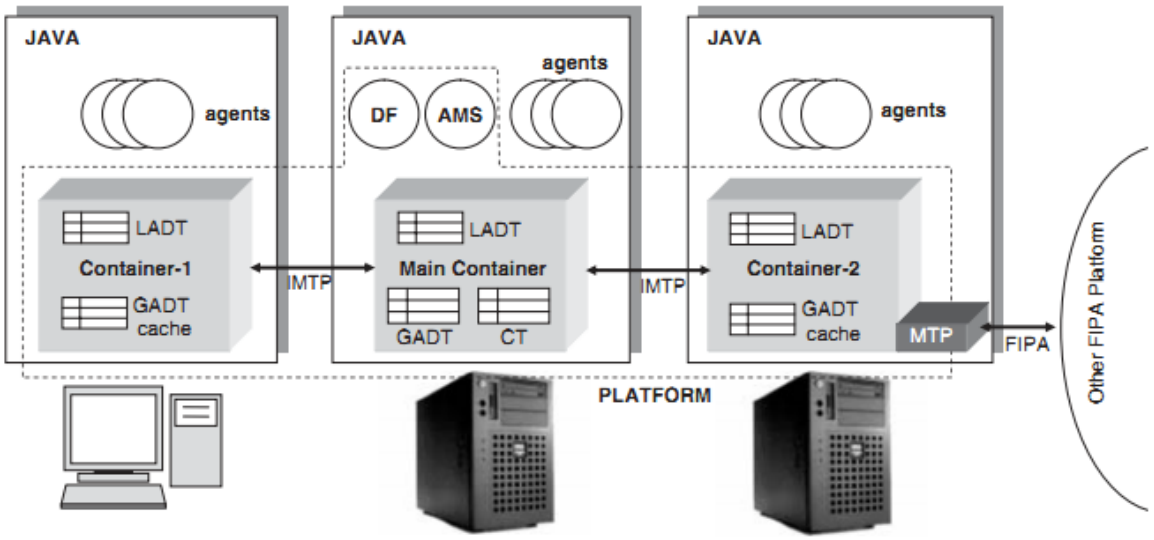

Figure 3.1. JADE Architecture [64, p.32].

The middle container is the special main-container that all other containers in a platform must register with [64, p.32]. The main-container has the following elements and functionalities:

- Container Table (CT) is the registry of the object reference and transport address of all container nodes composing the platform;
- Global Agent Descriptor Table (GADT) is the registry of all agents, their current status and locations within a platform;
- Local Agent Descriptor Table (LADT) is contained in all the containers and it's the first information search point;
- Agent Management System (AMS) is the agent that supervises the entire platform and every agent is required to register with it in order to obtain a valid agent identifier (AID);
- Directory Facilitator (DF) is the agent that implements the yellow pages services used by any agent wishing to register its services or search for other available services.

The main-container represents the bootstrap point of a platform and by default holds the AMS, DF, and Remote Monitoring Agent (RMA). Another important element of the platform is

the Global Agent Descriptor Table Cache (GADT Cache); this is a unique element present in all other sub-containers, which allows the sub-containers to store all information gathered from the main-container for local execution. This functionality prevents the main-container from being a system bottleneck.

3.3.1.2 JADE Platform

JADE is a software platform that provides basic middleware-layer functionalities, which are independent of the specific application. Each agent in JADE platform is identified by a globally unique name (the Agent-Identifier (AID), as defined by FIPA). It can join and leave a host platform at any time and can discover other agents through both white-page and yellow-page services (provided in JADE by AMS and the DF agents as defined also by the FIPA specifications). An agent can initiate communication with any other agent at any time it wishes and can equally be the object of an incoming communication at any time. These enable peer-to-peer system. Furthermore, JADE platform provide programmers with the following ready-to-use and easy-to-customize core functionalities [64, p.31]:

- A fully distributed system inhabited by agents, each running as a separate thread, potentially on different remote machines, and capable of transparently communicating with one another, i.e. the platform provides a unique location-independent application programming interface (API) that abstracts the underlying communication infrastructure.

- Full compliance with the FIPA specifications. The platform successfully participated in all FIPA interoperability events and was used as the middleware for many platforms in the Agentcities network (Agentcities). A great facilitator of this was active contribution by the JADE team to the FIPA standardization process.

- Efficient transport of asynchronous messages via a location-transparent API. The platform selects the best available means of communication and, when possible, avoids marshalling/unmarshalling Java objects. When crossing platform boundaries, messages are automatically transformed from JADE's own internal Java representation into proper FIPA-compliant syntaxes, encodings and transport protocols.

- Implementations of both white pages and yellow pages. Federated systems can be implemented to represent domains and sub-domains as a graph of federated directories.

- A simple, yet effective, agent life-cycle management. When agents are created they are automatically assigned a globally unique identifier and a transport address which are used to register with their platform's white page service. Simple APIs and graphical tools are also provided to both locally and remotely manage agent life cycles, i.e. create, suspend, resume, freeze, thaw, migrate, clone and kill.

- Support for agent mobility. Both agent code and, under certain restrictions, agent state can migrate between processes and machines. Agent migration is made transparent to communicating agents that can continue to interact even during the migration process.

- A subscription mechanism for agents, and even external applications, that wish to subscribe with a platform to be notified of all platform events, including life-cycle-related events and message exchange events.

- A set of graphical tools to support programmers when debugging and monitoring. These are particularly important and complex in multi-threaded, multi-process, multi-machine systems such as a typical JADE application. These enable conversations to be sniffed and emulated, and agent execution can be controlled remotely and introspected, including remote step-by-step debugging of agent execution.

- Support for ontologies and content languages. Ontology checking and content encoding is performed automatically by the platform with programmers able to select preferred content languages and ontologies (e.g. XML and RDF-based). One can also implement new content languages to fulfill specific application requirements.

- A library of interaction protocols which model typical patterns of communication oriented toward the achievement of one or more goal. Application-independent skeletons are available as a set of Java classes that can be customized with application-specific code. Interaction Protocols can also be represented and implemented as a set of concurrent finite state machines.

- Integration with various Web-based technologies including JSP, servlets, applets and Web service technology. The platform can also be easily configured to cross firewalls and use NAT systems.

- Support for J2ME platform and the wireless environment. The JADE run-time is available for the J2ME-CDC and -CLDC platforms through a uniform set of APIs covering both J2ME and J2SE environments.

- An in-process interface for launching/controlling a platform and its distributed components from an external application.

- An extensible kernel designed to allow programmers to extend platform functionality through the addition of kernel-level distributed services. This mechanism is inspired by the aspect-oriented programming approach where different aspects can be woven into application code and coordinated at kernel level.

3.3.1.3 JADE Packages

The JADE platform sources are organized into a hierarchy of Java packages and sub-packages, where each package, in principle, contains the set of classes and interfaces that implement specific functionality. The main packages are [64, p.37]:

- `jade.core` implements the kernel of JADE, the distributed run-time environment that supports the entire platform and its tools. It includes the fundamental `jade.core.Agent` class as well as all the basic run-time classes needed to implement agent containers. It also includes a set of sub-packages each implementing a specific kernel-level service. These are:

  – `jade.core.event` that implements the distributed event notification service. This allows subscribers to be notified of system events generated by the various distributed components of a platform;

  – `jade.core.management` that implements the distributed agent life-cycle management service;

  – `jade.core.messaging` that implements the message distribution service;

– `jade.core.mobility` that implements the agent mobility and cloning service, including the transfer of both the state and the code of an agent;

– `jade.core.nodeMonitoring` that enables containers to monitor each other and discover unreachable or dead containers;

– `jade.core.replication` that enables replication of the main container as a failover option in the event of serious faults in the original main container;

– `jade.core.behaviours` is a sub-package of `jade.core` that contains a hierarchy of the core application-independent behaviours. A JADE behaviour represents a task that an agent can carry out.

- `jade.content` and its sub-packages contain the collection of classes that support programmers in creating and manipulating complex content expressions according to a given content language and ontology. This includes all the coded mechanisms necessary for automatic conversion between the JADE internal representation format and the FIPA-compliant message content transmission format.

- `jade.gui` contains some general-purpose Java components and icons that can be useful to build Swing-based GUIs for JADE agents. The package provides several ready-to-use graphical components for displaying typical JADE abstractions, e.g. the AID, the **ACLMessage**, and the **AgentDescription**.

- `jade.imtp` contains the JADE IMTP (Internal Message Transport Protocol) implementations. In particular, the sub-package `jade.imtp.rmi` is the default IMTP of JADE, which based upon Java RMI (RMI).

- `jade.lang.acl` contains support for the FIPA Agent Communication Language (ACL) including the **ACLMessage** class, the parser, the encoder, and a helper class for representing templates of ACL messages.

- `jade.mtp` contains the set of Java interfaces that should be implemented by a JADE MTP.

It also contains two sub-packages with an implementation based upon the HTTP protocol (which is the default implementation) and one based upon the IIOP protocol.

- `jade.domain` contains the implementation of the AMS and DF agents, as specified by the FIPA standards, plus their JADE-specific extensions that will be described later. Each sub-package contains the classes representing the various entities of a predefined JADE ontology. Depicted in Table 3.1 are the predefined JADE ontologies.

Table 3.1 Predefined JADE ontologies.

Ontology	Package	Description
FIPA-Agent-Management	`jade.domain.FIPAAgentManagement`	Entities, exceptions and actions needed to interact with AMS and the DF, according to the FIPA specifications.
JADE-Agent-Management	`jade.domain.JADEAgentManagement`	JADE extensions to the FIPA-agent-management ontology.
JADE-Introspection	`jade.domain.introspection`	JADE extensions related to the monitoring of platform events.
JADE-Mobility	`jade.domain.mobility`	JADE extensions related to agent mobility.
JADE-Persistence	`jade.domain.persistence`	JADE extensions related to persistence.
DFApplet-Management	`jade.domain.DFGUIManagement`	Ontology used by the DF GUI to interact with the DF. It allows multiple GUIs of the same DF, which includes GUIs implemented as applets

- `jade.proto` contains the implementations of some general-purpose interaction protocols, including some of those specified by FIPA.

- `jade.tools` contains the implementation of all the JADE graphical tools

- `jade.util` contains several miscellaneous utility classes.

- `jade.wrapper` together with the `jade.core.Profile` and `jade.core.Runtime` classes provides support for the JADE in-process interface that allows external Java applications to use JADE as a library.

- FIPA is a package that includes the IDL (Interface Definition Language) module specified by FIPA for the IIOP-based MTP.

3.4 RMNA Framework

Reconfigurable mobile network agents (RMNA) are software entities that are distributed and instantiated over the network components to support applications in obtaining the best quality, reliability and most appropriate services within available resources. Based on coordination processes, RMNA acts as User-Network Interface (UNI) and Network-Network Interface (NNI) that continuously monitors internetworking connections and provides an abstract view of the transport network infrastructure to service provider (SP). This framework is to design a distributed control scheme that resides on specific network managed node (routers) as well as the user terminal with the aim of enabling effective resource coordination and network management processes. This will reduce the number of tasks to be executed by a particular mobile agent that is usually associated with an increase in mobile agent size hence hindering its migration time. The interaction between RMNA residing on different host nodes requires a unique addressing scheme to provide secure monitoring and management services (DR-1).

As mentioned in Section 3.3, this project considered JADE as RMNA development framework. The framework composes of agent containers that are hosted in Java Virtual Machine (JVM). Each JVM is basically a container of agents that provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same

host [67]. The communication architecture offers flexible and efficient messaging, where JADE creates and manages a queue of incoming ACL messages, private to each agent; agents can access their queue via a combination of several modes: blocking, polling, timeout and pattern matching based. Figure 3.2 illustrates the communication process. Two different modalities of controlling the JADE runtime system exist [69]:

- Multiple-container: Several containers (belonging to the same platform) can be executed in the local JVM. This modality is activated by means of the `createAgentContainer()` and `createMainContainer()` methods plus the classes included in the `jade.wrapper` package.

- Single-container: Only one container can be executed in the local JVM. This modality is activated by means of the `startUp()` and `shutDown()` methods

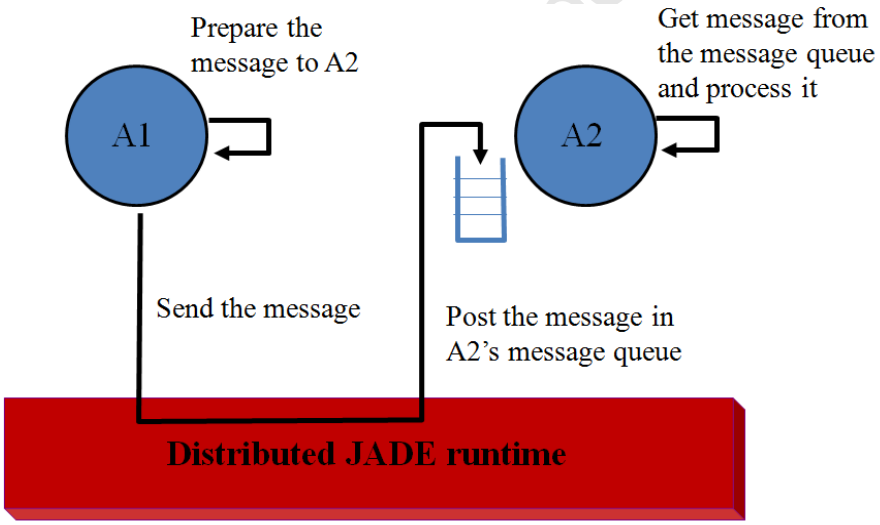

Figure 3.2. JADE messaging paradigm [64, p.66].

RMNA specifically implements the Multiple-container modality as its runtime system; however, a coordination process is required to achieve a reliable degree of acceptability and adaptability. This will enable all distributed mobile agents to have knowledge of where and how to locate and deliver queries to other agents within a particular execution space. The main component of JADE that facilitates this coordination functionality is the AMS and the DF (Section 3.3.1.1).

3.4.1 Case study Scenario

To effectively capture the potential functional requirements of RMNA resource coordination processes, a study case is presented to provide guidelines that aid in modeling application to suit heterogeneous networks. These guidelines enable the decision making when specifying the design processes.

The case study demonstrates a framework where network services (such as video streaming) is being provided to mobile users under the control of different stakeholders (service providers, network providers and content providers). This allows users the possibility of unrestrictedly selecting a wide range of network services from the networks, which provide different bandwidth requirements and various quality of service (QoS). However, these different stakeholders operate under different policies such as:

- bandwidth requirements;
- delays;
- computational power;
- pricing schemes.

The brokering process will be guided by user preferences, which need to be processed according to some customized logic. The demand for these parameters varies according to application types. To cope with this competitive and complicated networking environment, RMNA agents are designed to facilitate expertise brokering activities such that, mobile users (MU) can monitor the different networks components (i.e., links between routers) and provides the required monitoring services according to users specifications (DR-2).

In the context of this project, it is *assumed* that RMNA have direct access to the managed node (router) information. The information should be accessible; otherwise, RMNA will not be able to carry out the required task. Similarly, the RMNA sensor must be capable of perceiving the information. For simplicity, a managed node GUI that constantly provides update to the coordination entities (RMNA) is generated.

Certain rules were specified in advance and dynamically during runtime to enable effective network resource management in meeting with the unanticipated heterogeneous network changing environments. The designed RMNA agents are of different types and perform different tasks based on the network component it represents. The combinations of their various actions enable the RMNA agents to perform complex monitoring functions. A qualitative role and behaviour of RMNA interaction with the DF and the managed node agent is depicted in the block diagram of Figure 3.3.

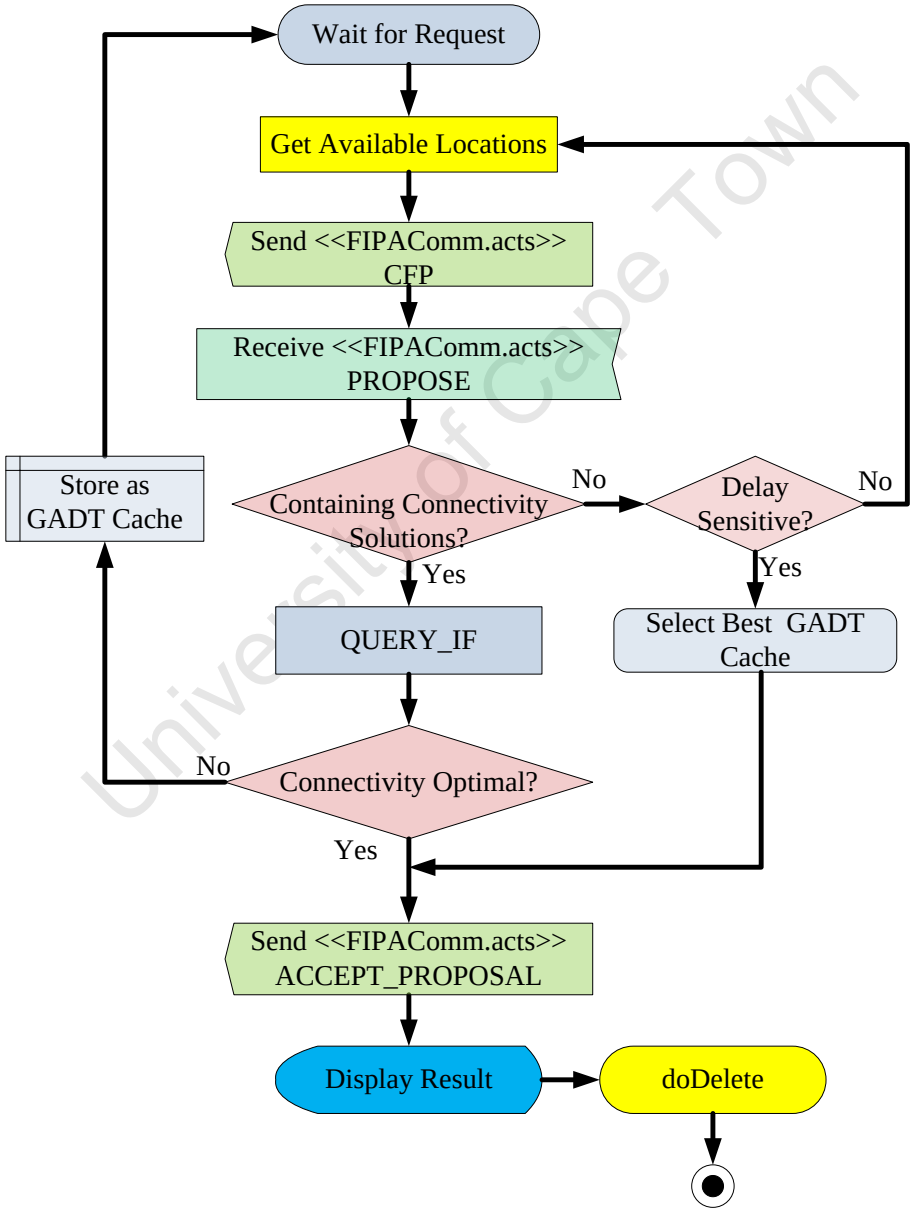

Figure 3.3 Flow Chart of RMNA.

The flow chart provides the guideline algorithm that specifies the design processes. Prior to the discussion of these guidelines, certain design assumptions are presented during the course of this study.

3.4.2 Design Assumptions

The design processes of this thesis is not implemented without certain assumptions since it incorporates features of the next generation networks (NGN). These assumptions include:

- Implementation platform specifically focusing on JADE.
- Networked computers being used to represent heterogeneous networks environment (i.e. network domains belonging to different providers).
- Windows and Linux operating system represents WiMAX and LTE technologies respectively.
- RMNA agents assume the network nodes of interest (i.e. gateway routers).
- The networking environment provides full internetworking capabilities, hence allowing the traffic flows (run between them in a loosely collaboration manner) as standardized by the ITU-T Y.2234 [8].
- RMNA have direct access to the managed node (routers) information;
- Mobile terminal agent is able to sense user's requirements.
- The organization structure of the guideline algorithm is static that implies there is no consideration for non-emergent behaviour outside the thesis algorithm during runtime.
- Security issues is solved and totally neglected.

3.4.3 FIPA Communicative acts

The FIPA agent communicative language (ACL) defines communication in terms of a function or action, called the communicative act or CA, performed by the act of communicating

[64, p.18]. FIPA defines ACL performatives to enable formal semantics that can be exploited to make an agent automatically take proper decisions when a message is received. Among the different FIPA ACL performatives, this design phase selects the following performatives with their descriptions outlined in Table 3.2.

Table 3.2 Selected FIPA Communicative acts.

FIPA Communicative Act	Description
REQUEST	The sender requests the receiver to perform some action. One important class of uses of the REQUEST act is to request the receiver to perform another communicative act
INFORM	The sender informs the receiver that a given proposition is achievable
CALL-FOR-PROPOSAL (CFP)	This is the computational request sent to receiver(s) for proposals to perform a given action
PROPOSE	The action of submitting a bid to perform a certain action, given certain preconditions
REFUSE	The action that signifies that an attempt occurred to bid for the received CFP but failed to perform the given task due to inability to meet the preconditions. This is usually communicated to the sender agent with reasons for refusal.
ACCEPT-PROPOSAL	The action of accepting a previously submitted proposal to perform an action (this decision may include an internal evaluation processes)

3.4.4 RMNA Characterization

The case study described in Section 3.3.2.1 constitute of four RMNA agent types. In this model, the different type of agents serve the purpose of collecting specific monitors (such as the available bandwidth and delays of the links connecting network nodes) from the set of nodes they represent. The key responsibility of these agents includes:

- The device manager agent (DM) acts as the agent that resides on user's mobile device and it maintains service requirements. The DM responds to users' needs and environments by creating a resource requester (RR).

- The Resource Requester (RR) is the agent that initiates the negotiation/reservation process with the network for resource access.

- The adaptive manager agent (AM) agent and the broker agent (BA) are agents that reside on the nodes of preference (i.e. the ingress and edge nodes) and provide detail information about the network elements they represent. These agents represent the network provider's domains.

The agent management system (AMS) and the directory facilitator (DF) are JADE default agents and assume the functionality of the service provider in this thesis. They are mandatory agent components specified by FIPA [64, p.34], which guarantee interoperability of agents and manage a platform.

Figure 3.3 describes the interaction of the resource requester (RR) created by the device manager (DM) with the visited networked nodes (i.e., the DF and AM). The DM based on users request creates a resource requester (RR) and initiates a task (i.e., providing the optimal connectivity details within heterogeneous networks). The startup of the flow chart in Figure 3.3 complies with JADE rules, which mandates every agent within a platform to register its services with the AMS in order to acquire a valid agent identifier (AID).

On arrival of RR to the service provider domain that host AMS and DF, it queries the DF the following communication request:

- information detailing the task required;

- its associated priority level;

- identity and location of the requested agents that can provide the services.

The DF dynamically communicates with the AMS to maintain knowledge about the location and services of each agent within the platform. The resource requester agent then sends a path message to the various agents that represent the networked nodes in the form of Call-For-Proposal (CFP) containing the characteristics of the network traffics (in this case, bandwidth requirement of network links). The role of the three decision points of the flow chart is given as follows:

- Containing Connectivity Solutions: This is the first aggregation process of the resource requester (RR). The RR evaluates the received bids from the AM(s) to determine if it contains the requested parameters (i.e., the network link information). If "Yes", the RR executes an internal action to QUERY-IF the link status conforms to user's preferences. When the result is "No" connectivity information is available in the PROPOSE message, the next decision point is considered.

- Delay Sensitive: At this point, the RR is concern with requested service priority. This is specified by the user in order to be assured of quality of experience. If the required service can tolerate delays, then RR will re-negotiate for network link status been a dynamic network environment otherwise a choice to select the best option from the GADT Cache is considered. See Section 3.3.1.1 for details about GADT Cache.

- Connectivity Optimal: This is the process whereby the resource requester then evaluates all AM bids to determine the optimal solution that will provide effective bandwidth utilization while providing user with quality of service. The computational task is assigned to the AM that promises to service the request within user specifications.

The result is then displayed and the RR is deleted from the network after task execution. This RR termination process helps in controlling the signaling overhead incurred by mobile agent system as its size increases.

3.4.4.1 RMNA Behaviour

RMNA specifies their actual responsibilities within the agent's "behaviour(s)". Behaviour represents a task that an agent can carry out and is implemented as an object of a class that extends `jade.core.behaviours.Behaviour` [64, p.57]. RMNA behaviour is then added to the agent by means of the `addBehaviour()` method of the `Agent` class. Two abstract methods are also implemented in each class that extends RMNA `behavior`:

- The `action()` method, which describes the operations to be performed when the behaviour is in execution.

- The `done()` method that returns a `boolean` value to indicate whether or not a behaviour has completed and is to be removed from the pool of behaviours an agent is executing.

Based on the case study described in Section 3.3.2.1 of Chapter 3, the scheduling of behaviours in RMNA is not pre-emptive (as for Java threads), but cooperative (i.e., collectively executing complex tasks). This implies that when a behaviour is scheduled for execution its `action()` method is called and runs until it returns, which reduces the RMNA complexity in terms of switching to different task execution. For simplicity, the JADE classes that implement the skeletons for commonly required types of task are implemented. These classes include:

- `OneShotBehaviour`: implementing an atomic task that runs once and terminates immediately. The `jade.core.behaviours.OneShotBehaviour` class already implements the `done()` method by returning `true` and can be conveniently extended to implement new one-shot behaviours.

- `CyclicBehaviour`: this implements an always active task, which is scheduled to perform the same operations. The `jade.core.behaviours.CyclicBehaviour` class already implements the `done()` method by returning `false` and can be conveniently extended to implement new cyclic behaviours.

- `TickerBehaviour`: implementing a task that periodically executes the same operations.

3.4.5 RMNA Coordination Processes

The reconfigurable aspect of this scheme enables the adaptability of the mobile network agents to respond to the unprecedented changes and requirements that occurs within different network operators/technologies. Reconfigurability of MA allows the following control attributes:

- specification of several reactive principles that can be necessary for different network conditions such as suspend, clone, etc
- conformation to existing networking principles
- consideration of new events that can instantly ensure tasks are rescheduled when failure or intermittent connectivity occurs during runtime (Example of such service is migrating)

For example, Figure 3.3 illustrates a study case of a system that comprise of two different network providers and each having four routers. The networks routers are IP configured as in the next generation networks (NGN) [8], hence the routers enables intra/inter network connections. In this scenario, a router connects to at least two other neighbours (one from its home network and the second connection to a visiting network). From each router point of view, identifying the optimal link connection become complex since it involves other metrics (such as bandwidth requirement, delay, computational power, etc.) from a different network.

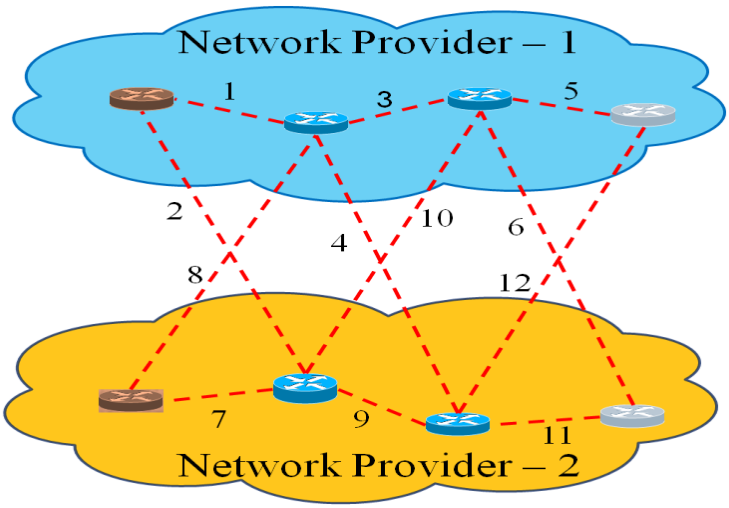

Figure 3.4. Reconfigurable reaction.

The deployed mobile agents monitor the links information and provide updates when needed. However, the dynamic nature of today networking environment poses a great challenge.

For example, if the optimal link was "3" at the time of request and the link "3" fails or saturate during the runtime, the reconfigurable ability of RMNA will enable the mobile agent to react by instantly providing details of link "4" in a timely fashion. To support this features, RMNA suspends its task and execute it on the visited router. JADE enables the suspend feature.

3.4.6 RMNA Interaction Mechanism

The communication between RMNAs is based on the JADE protocol. According to the FIPA specifications, the ACC that comprise of a Message Transport Service (MTS) is one of the three most important services that an agent platform is required to provide (the other two being the Agent Management Service and the Directory Facilitator). An MTS manages all message exchange within and between platforms [64, p.39]. To promote interoperability between different platforms (i.e. with non-JADE), JADE implements all the standard Message Transport Protocols (MTPs) defined by FIPA, where each MTP includes the definition of a transport protocol and a standard encoding of the message envelope. Table 3.2 illustrates the JADE MTPs.

Table 3.3 JADE MTPs available in public domain [64, p.40].

Transport protocol	Message encoding	Developer
HTTP and HTTPS	XML	Universitat Autonoma of barcelona (UAB), Spain
IIOP (Sun implementation)	CORBA IDL	JADE team
IIOP (ORBacus implementation)	CORBA IDL	Giovanni Rimassa, Universita di Parma, Italy
JMS	Java data structure	Edward Curry, University of Galway
Jabber XMPP	Java data structure	Universitat Politecnica of Valencia, Spain

HTTP is specifically selected for RMNA MTS in this project since it supports the following [64, p.40]:

- HTTP is the default MTP that is launched with the JADE main container ;
- Local port numbers for incoming and outgoing connections can be selected for stricter firewall configuration using the parameters `jade_mtp_http_port` and `jade_mtp_http_outPort;`
- A proxy can be configured and used for all outgoing connections invoking the parameters `jade_mtp_http_proxyHost and jade_mtp_http_proxyPort.`
- Performance can be fine-tuned through the use of persistent connections: instead of performing TCP handshaking for every message, connections can be cached and reused when messages are frequently exchanged between two different platforms. This is controlled with the parameters `jade_mtp_http_numKeepAlive` and `jade_mtp_http_timeout.`

3.4.7 RMNA Functionality

To achieve the required network resource monitoring and management control, a process is define using the standard Java API to coordinate the entities (RMNA) interactions throughout their lifecycle. The AMS and the DF earlier discussed in this chapter is the key enabler of this function. For efficiency, the lifecycle of RMNA requires the following functions to interwork to:

- *Device manager (DM)* creates a resource requester based on users preferences;
- *Resource requester initialization* is instantiated as needed and is responsible for the communicating of DM request to other network agents;
- *Task execution* describes the process of interactions;
- *Deployment function* defines RMNA migration process;
- *Resource requester termination* specifies the action to delete agent from the networks.

3.4.7.1 Device Manager (DM)

Device manager (DM) is a permanent single instance agent on the mobile device and has the responsibility to gather and maintain information about the physical device and its owner. This function is not formally addressed in this project, and it is assumed that different mobile device administrator/vendors control their device capability. The main target of this research is to ensure end-to-end seamless connectivity within heterogeneous networks, where the main players could be terminals with different capabilities operating in different networks. However, DM importance is analyzed been the main point of contact between the user and the network architecture. The agent circuitry diagram was analyzed in [12] and emphasis was laid on how agent uses its internal data to record information about the environment state and history as shown in Figure 3.4. DM functionalities include:

- sensing, predicting and responding to network resource limitations from a device point of view;
- creating the resource requester in order to schedule a performance intensive computation as User Network Interface (UNI) and initiate task.

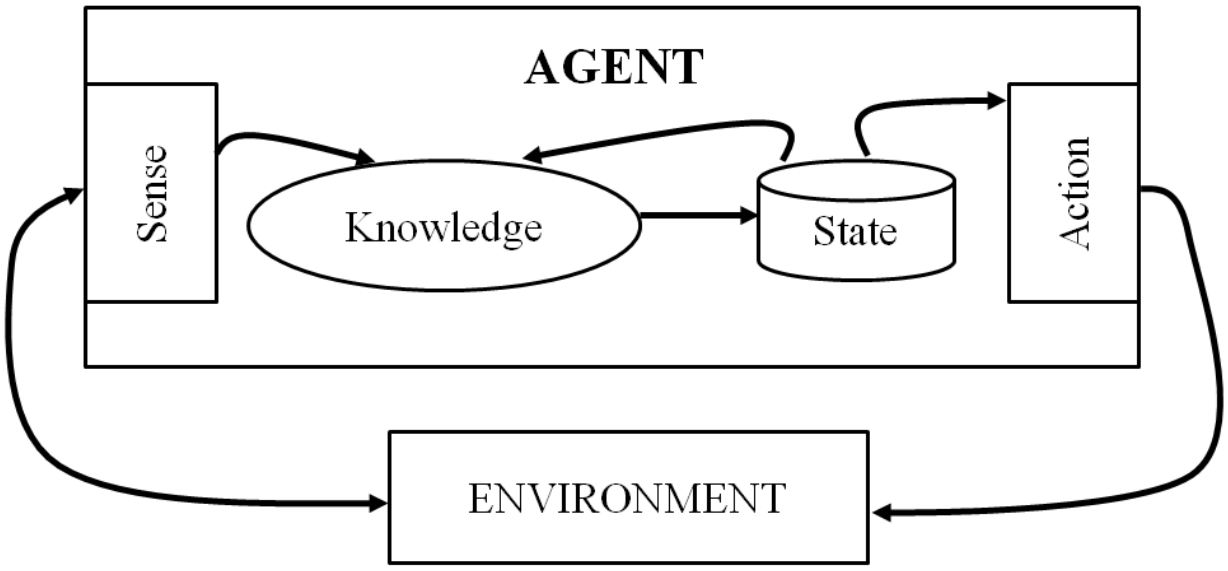

Figure 3.5 Agent state automation [12].

3.4.7.2 Resource Requester Initialization

Resource requester initialization entails the creation of ontological structures allowing for effective inter-communication amongst the network agents. This communication request includes information detailing the task required, its associated priority level and also the identity of the requesting device. Figure 3.5 illustrates the resource requester initialization process using UML sequence diagram.

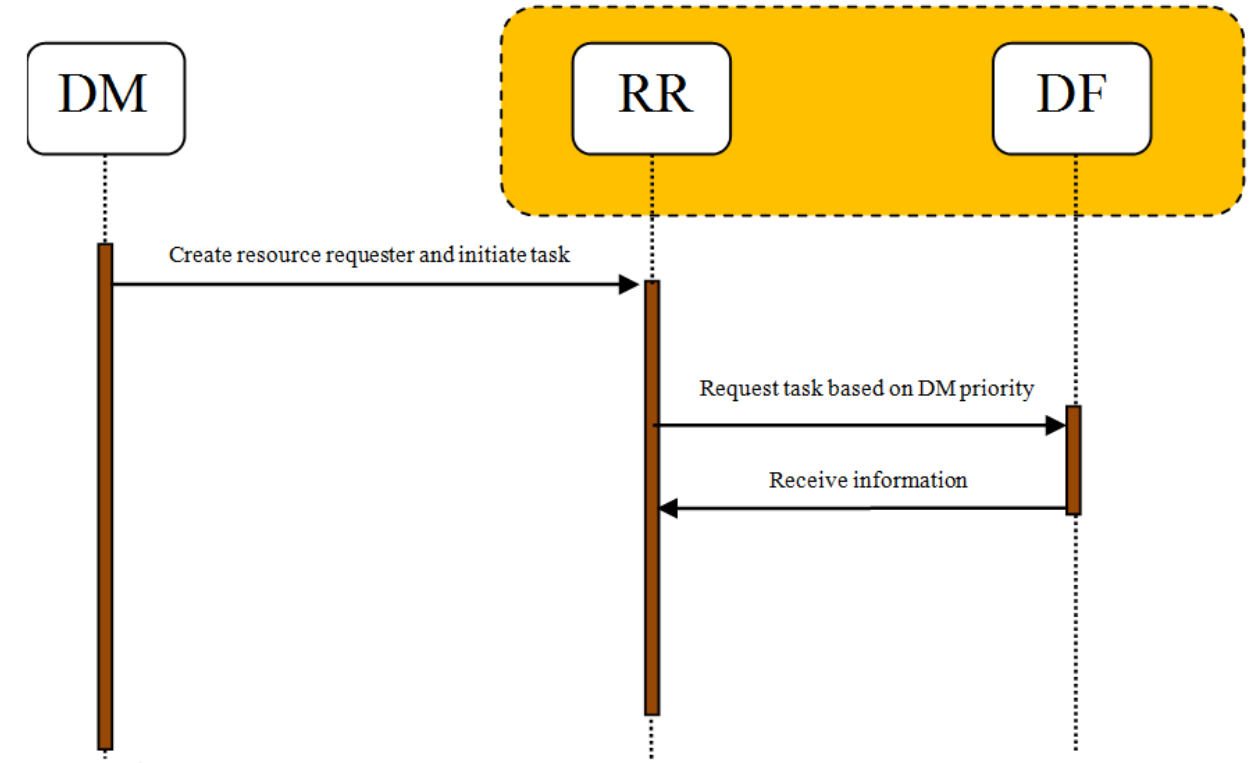

Figure 3.6. Initialization sequence diagram.

The resource requester interacts with the DF to request the possible execution space information based on DM preferences. As discussed earlier, the DF provides the yellow page information that enable agents to search for other available services. Upon receiving the request, the DF notifies the resource requester (RR) the required information (in this case, the location and services of the distributed network providers within the network architecture).

3.4.7.3 Task Execution

Task execution function is the process of scheduling performance intensive computation with the various agents residing within the distributed network provider domain as depicted in Figure 3.6. This includes satisfaction for the service level agreement (SLA), monitoring of network states and reservation of network resources based on DM demands. Using a sequence diagram, confined the illustration process to a single interaction communication with network provider, however the implementation chapter will elaborate the kind of replication and synchronization messages scheduled by the resource requester (RR).

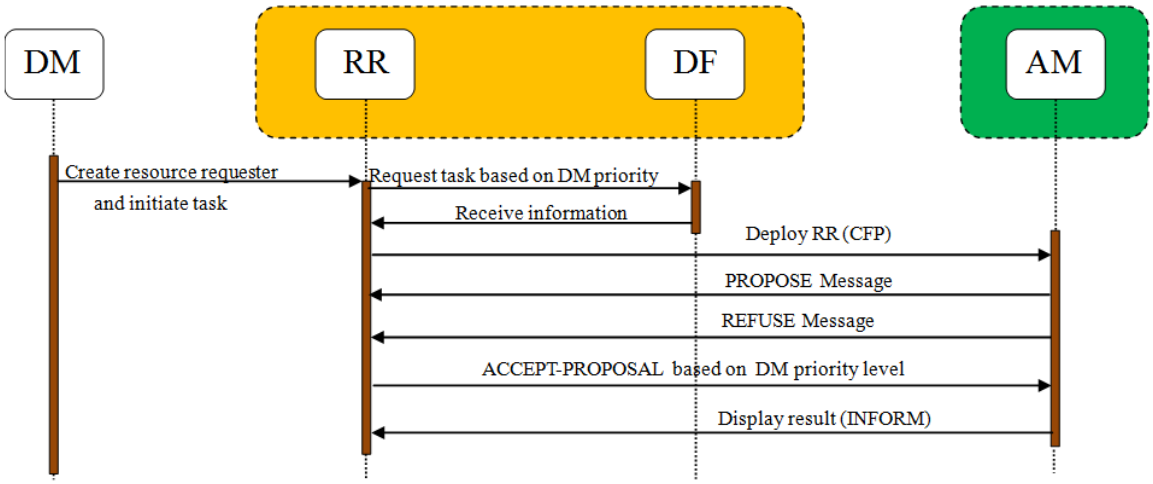

Figure 3.7. Task execution protocol.

Based on the information provided by the DF during initialization, the RR replicates itself (depending on the number of nodes) and deploys them as call for proposal (CFP) to the network provider nodes that host RMNA in anticipation to retrieve the network status (in this case, bandwidth associated with each links between routers). The PROPOSE or REFUSE ACL notifies the RR if the link the agent represent can support its requirement. This information enables the resource requester to send ACCEPT-PROPOSAL to the agent that best suit its requirement according to DM specification. Finally the INFORM notification is delivered to the originated resource requester, which provides SP the required information to schedule users task in terms of real data transportation.

3.4.7.4 Deployment Function

Deployment function specifies the ability for RMNA program to migrate or replicate (clone) itself across one or multiple network hosts. Calling the method `doMove(Location)`, will move the agent (RMNA) within the agent program passing in parameter to the new destination of the agent which must be a location object. As mentioned during initialization, to get the location object, the resource requester queries the DF about the necessary information, by sending a REQUEST with either `WhereIsAgentAction` or `QueryPlatformLocationsAction (package.jade.domain.FIPAAgentManagement)` as content message.

A `WhereIsAgentAction` allows one to obtain with the location of a given agent. It requires one parameter, the identifier of the agent that its location is been queried. This is provided via the method `setAgentIdentifier(AID)` of this class. Otherwise, a `QueryPlatformLocationsAction` allows one to obtain all available locations within a given platform. This enables the execution of management application as close as possible to the managed node. The deployment message consists of the main description of the component itself and in some cases contains the any other dependency to the components.

The resource requester (RR) is the only RMNA that observes the termination function. This process occurs after RR has successfully executed the required monitoring task as explained during execution protocol. In this case, the resource requester deletes from the network and all its data and coordination instance is cleaned up from the system.

3.5 Chapter Summary

In consideration for RMNA development based on network resource coordination, this chapter undertakes the first three characteristics measures to enable a systematic flexible solution (Sections 3.1 to 3.3). The measures are:

- Planning
- Analyses
- Design

The planning and parts of analysis phase were actually carried out in Chapter 2 as a prerequisite that provides the basis for this chapter. The planning phase provides detail information to whether the mobile agent-based option (among other network management approaches) is the most appropriate network monitoring solution. The literature review on network resource management techniques of this thesis considers the deployment of mobile agents as suitable solution to the envisaged complexities that mar future mobile network environment. Chapter 2 further provides a set of design requirements useful in terms of enabling some formal guidelines (i.e. a list of steps and deliverables). This is considered as the analyses phase, which aims to clarify the problem without any (or minimal) concerns about the solution.

Once the analysis has been carried out, a move is made to the design phase, which aims to specify the solution. The solution focuses on the JADE platform. The chapter is presented as follows:

- Analyzing the JADE architecture and its key features.
- Specifying RMNA interaction model and providing the different agent responsibilities.
- Presenting a list of design assumptions.
- Defining the FIPA ACL used throughout the implementation and evaluation phase.
- Based on the agent flow chart diagram produced, the agent resource interaction is described to determine how agents will interact with these resources.
- RMNA behaviours were also detail while highlighting the key functionalities of RMNA. Based on the responsibility describe in the case study scenario, the agent responsibilities are mapped to agent behaviours. Different types of responsibilities (including interactions) will require different types of agent behaviours to be specified.

Chapter 4 RMNA Platform Implementation

4.1 Implementation Overview

To effectively demonstrate the applicability of RMNA in network resource coordination process, a prototype mobile agent system was implemented using JADE platform to emulate internetworking of two NPs through WiMAX and LTE gateways. This scenario involves four different networked computers (running Windows and Linux operating systems) hosting the JADE containers used to represent heterogeneous networking environments as depicted in Figure 4.1. The developed RMNA test-bed allows the visualization of the core concepts, implementation and objectives of this research. These two different operating systems used in this project demonstrate that the RMNA scheme is network technologies transparent. The NP nodes of preference (gateways) are uploaded as RMNA and reside on these containers. RMNAs perform their monitoring tasks by migrating from one-networked computers to another carrying data about its state, which includes information obtained from previous tasks.

The development framework in this project is used to generate a Java Virtual Machine (JVM) that host the RMNA platform. JVM is necessary because the four networked computers that host RMNA platform contain processors that have non-uniform instruction sets (see Table 4.1). Compiling task for one machine will result to inability to execute the task on another machine with a different instruction set, hence, violating the premise that tasks be technologies transparent. Instead, tasks are compiled into the JVM's instruction set, which is uniform across all hardware platforms, ensuring that tasks are platform transparent.

The guides provided by the designed framework were described in chapter 3. The first implementation phase is to define the networking environment that would reflect the multiple NPs. The second phase is to create the communication paradigms (i.e., agent communication channels) between the various parts of the framework.

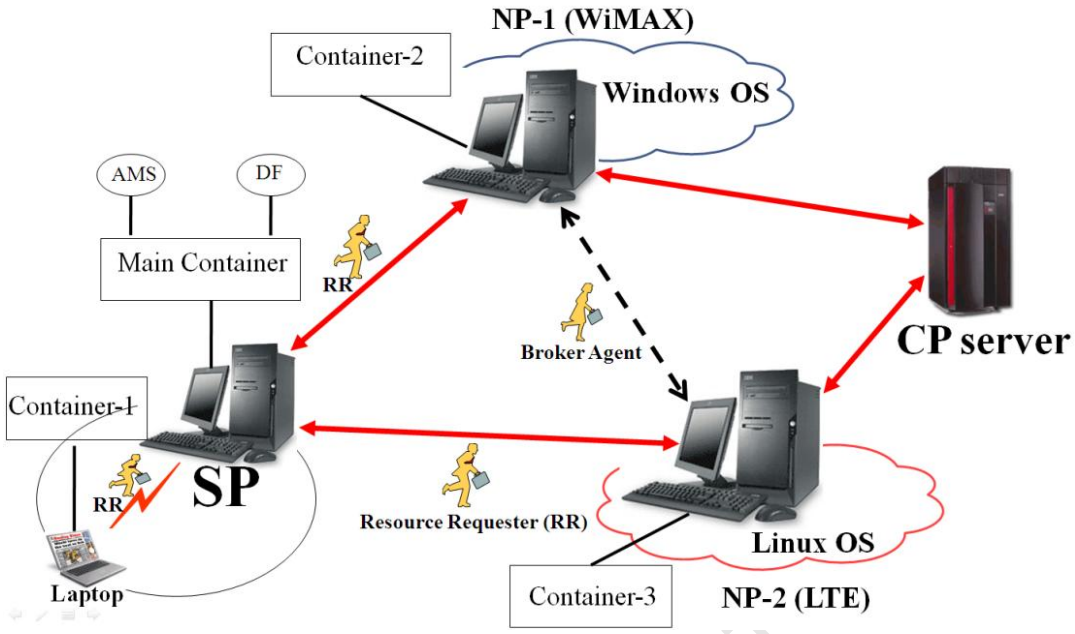

Figure 4.1. Implementation plan.

4.2 Prototype Installation and Configuration

The RMNA runtime and measurement environment involves a dedicated test-bed of four different networked computers (three workstations and a Laptop). The computer's hardware specifications are provided in Table 4.1.

Table 4.1. Hardware Description.

Unit	Processor	Memory	Operating System
1 (SP Computer)	Pentium[R] Dual-Core CPU E5400 @ 2.70GHz	1.99 GB of RAM	Microsoft Windows XP
1 (NP-1 computers)	Inter[R] Core[TM]2 CPU 6300 @ 1.86GHz	0.98 GB of RAM	Microsoft Windows XP
1 (NP-2 computers)	Inter[R] Core[TM]2 CPU 6300 @ 1.86GHz	0.98 GB of RAM	Linux (Ubuntu 9.04)
1 Laptop (client)	Inter(R) Core(TM) i5 2520M CPU @ 2.50GHz	4.0 GB of RAM	Microsoft Windows 7

Developing this prototype based on JADE platform requires the following software to be installed and executed:

- jadeAll.zip
- Java Development Kit (JDK)
- Java SE Runtime Environment (JRE)
- ANT Program
- Java Compiler

4.2.1 JADEAll.zip

The JADEAll.zip is an open source environment for peer-to-peer agent based applications and it composed of the four compressed files:

- JADE-bin-4.0.1.zip, which contains JADE already compiled and ready to be used (set of JAVA archive JAR files with all classes);
- JADE-doc-4.0.1.zip, which contains all the JADE documentation, included the Administrator's Guide and the Programmer's Guide;
- JADE-src-4.0.1.zip, which contains all JADE source code;
- JADE-examples-4.0.1.zip, which contains the source code of JADE implementation examples.

After uncompressing the archived files of the JADE version 4.0.1, a directory tree was generated, whose root is C:.\jade and a C:.\jade\lib subdirectory containing: jade.jar, jadeTools.jar, http.jar and iiop.jar files. The *.jar files is added to the CLASSPATH environment variable for the Windows operating system and the .bashrc of the Linux (Ubuntu 9.04) operating system.

4.2.2 Java Development Kit (JDK)/JRE

In order to build the JADE system, the JDK1.6.0_23 was selected for installation as its pre-built IDL stubs and Java parser classes are included with the JADE source distribution.

The JDK is a development environment for building applications, applets, and components using the Java programming language.

The Java Standard Edition Runtime Environment is necessary to run the programs written in the Java programming language and it contains:

- Java virtual machine (JVM);
- runtime class libraries;
- Java application launcher.

The JDK1.6.0_23 is freely available from Oracle and composed of the Java SE Runtime Environment.

4.2.3 ANT Program

The ANT Program can be downloaded from the website http://ant.apache.org/. It acts as compiler of JADE sources. The most important ant targets are described as follows:

- jade – compile the sources and create the *.class* files under the C:.\jade\classes subdirectory
- lib – generate the Java archive jar files under the C:.\jade\lib subdirectory
- doc – generate the javadoc documentation files under the C:.\jade\doc subdirectory
- examples – compile all the examples

4.2.4 Java Compiler

To facilitate the installation/execution tasks, a JCreator was installed to provide the basic functionalities of the Java compiler. JCreator is a powerful interactive development environment (IDE) for Java technologies that provides more power IDE than combination of ordinary IDEs.

One of the key features of the JCreator is the support for JDK tools that can be used to compile, debug, and run a project.

The *Project Settings* dialog box (Figure 4.2) allows the attachment of these tools to the project. If no project is specified, JCreator runs the default projects. When necessary, JCreator allows the administrator to easily create the tools for calling the JDK applications such as:

- Compiler;
- Ant builder;
- Interpreter;
- Applet viewer;
- Debugger.

JCreator provides the following advantages:

- easy management of projects using an interface that looks like the Microsoft® Visual Studio®;
- allows the administrator to define color schemes in XML for organizing the code;
- wrapping existing projects while using different JDK profiles;
- allowing project templates and assisting code development;
- class browser supports for project viewing;

- easy debugging and intuitive interface (no need for old DOS prompts!);
- enabling Ant and CVS integration for the management and exchanging code;
- saving valuable time on CLASSPATH configuration (JCreator runs the configuration);
- enabling self-customization of user interface;
- allows administrator to setup run-time environments in a way that best suit the application as an applet, either in JUnit environment, or in a DOS window.

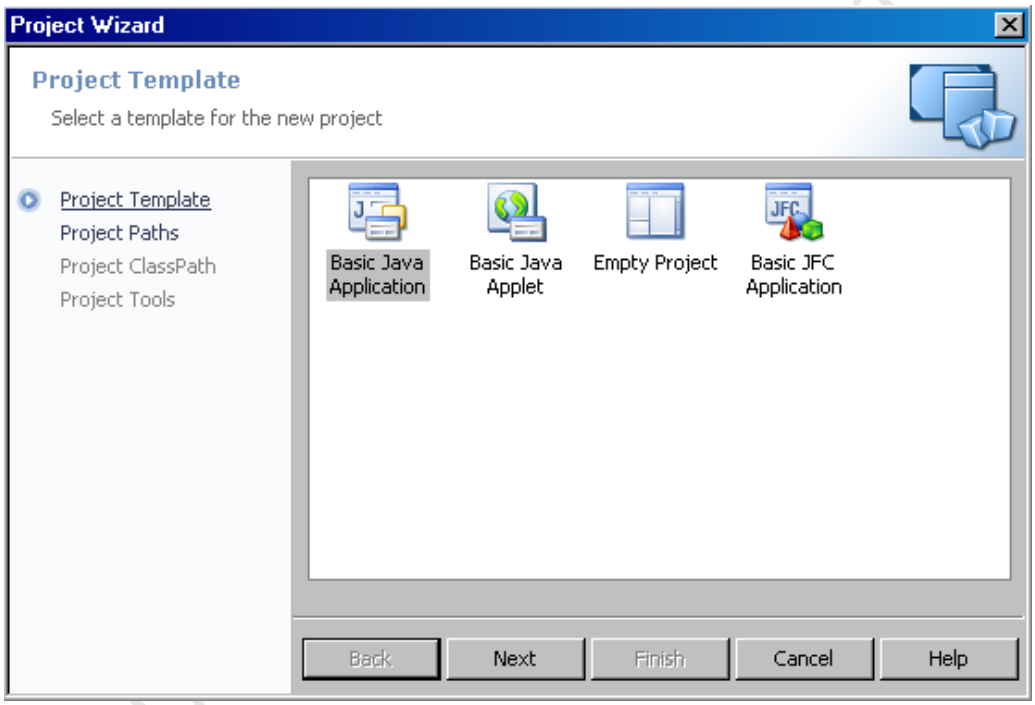

Figure 4.2. JCreator: Project Wizard.

4.3 Cooperative Network Environment

The undertaking of this thesis is to design and to develop a *middleware* between the network application and its connectivity with the aim of proactively performing monitoring functions and reserving network links in advance prior to real data communication. This will enable effective connectivity from great variety of terminals with different capabilities to access large array of varied heterogeneous networks services according to application requirement. Three sets of

JADE sub-containers were hosted by a main container to create RMNA platform as depicted in Figure 4.3.

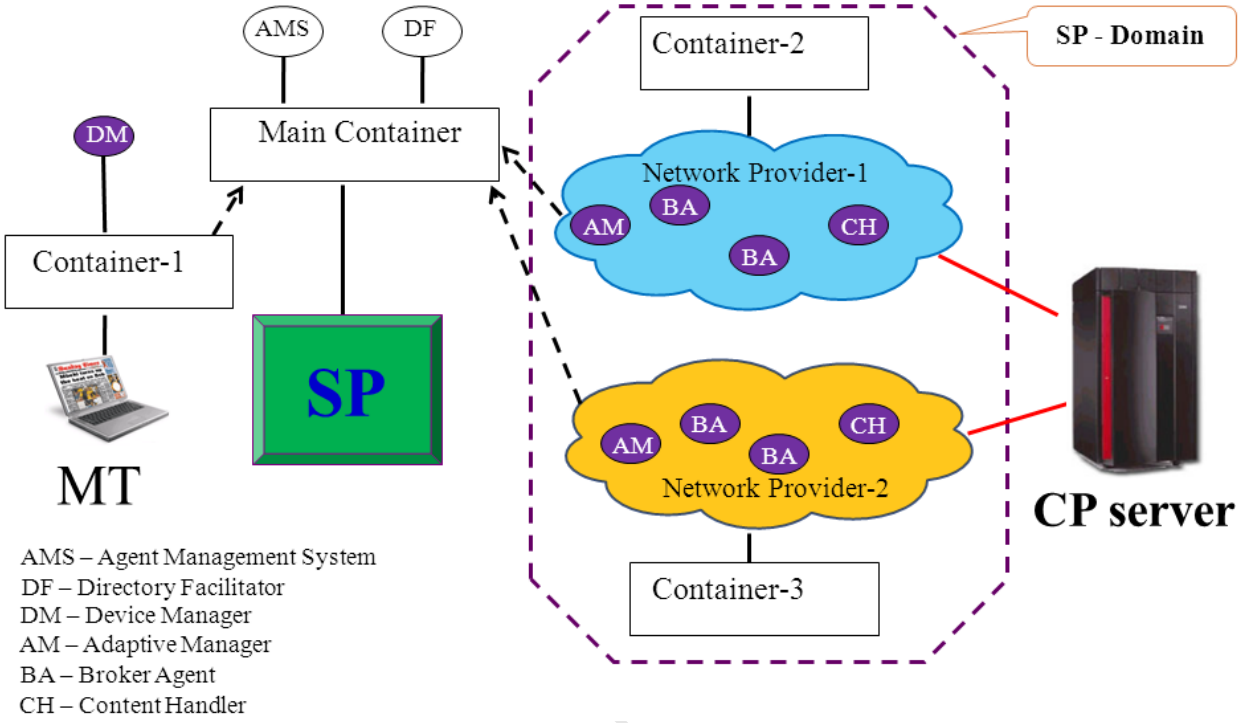

Figure 4.3. Distributed mobile agent environment.

The functionalities of the different containers are described as follows:

- Container-1, which assumes mobile terminal environment;
- Container-2, which assumes network provider 1 (NP-1) domain;
- Container-3, which assumes network provider 1 (NP-2) domain;
- Main-container, which provides platforms management services.

According to the case study of this project (Section 3.3.2.1), agents live in these containers and perform the task of monitoring and management of the nodes they represent. As described in chapter three, these containers are Java processes that provide the JADE run-time and all the services needed for hosting and executing agents. Specifically, this project assumes an

environment that comprise of two different network providers (NP) within one service provider (SP) domain accessing one content provider (CP). The two NP of case study are identical in terms of number of nodes (since the research focus is towards on mobile agent design and development for network resource coordination, the network topology can be undermined).

Reliance on mobile agent only for collecting network resource information have proven ineffective due to its performance implication incurred as it migrate [7]. The agent migration overheads worsen in a scenario that involves multi-hop mobility. However, this thesis distributed the network computational control to both mobile and static agents, which collectively perform the overall execution task (Figure 4.3).

The different case study of this project specifies different values of variables for the various links between the nodes using a graphical user interface (GUI) that is generated automatically by calling the `PathRMNAGui` class once the AM and BA are created. The final entity in this model is the content handler (CH). This agent resides at egress node and its responsibility is to create connectivity awareness between the NPs and CP server. Since data transfer process is outside the scope of this project, the focus is to coordinate resource of heterogeneous network (interoperable network providers), hence assumes the link operates at absolute full capacity.

The management process implemented by using RMNA platform is to create a cooperative interaction model that support network-wide resource coordination in two different phases:

1) Resource pooling for enabling the cooperative communication transaction;

2) Resource splitting for implementing diversity communication and reliability enhancement

JADE is very helpful in implementing a set of study cases. The platform provides an important element called the Global Agent Descriptor Table Cache (GADT Cache). This is a unique element present in all sub-containers which allows the sub-containers to store all information gathered from the main-container and previous task executed for local execution. This functionality prevent the main-container from been a system bottleneck.

4.4 Research Platform

To enable user personalization, a framework is needed to provide the required graphical user interface (GUI) that enabling the users to be described in terms of their preferences.

4.4.1 Building RMNA Platform

RMNA Platform is built on top JADE, the basis of JADE and its components is described in Chapter 3. To launch the RMNA platform (i.e. the main-container that acts as service provider and holds by default the AMS, DF, and the RMA), the following command is executed from the shell prompt of the computer that serve the functionality of the service provider (SP) as depicted in Figure 4.3.

```
C:.\> java jade.Boot -gui -container-name Service-Provider -platform-id RMNA-Platform
```

The output resulting by running this command together with the command-line option "`-gui`" is depicted in Figure 4.4.

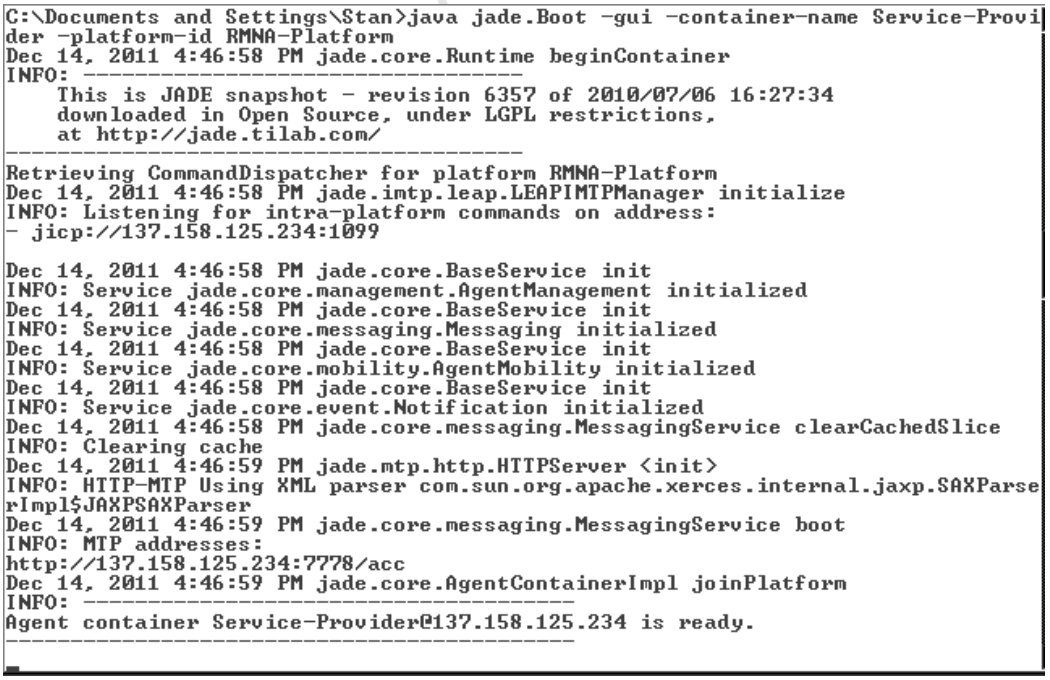

Figure 4.4. Standard JADE Start-up Output: RMNA-Platform.

The first part of this output prints the JADE disclaimer. The next part shows the host port where the main-container is listening to accept other containers joining the platform. The initialization of the kernel services (kernel services are JADE internal functionalities as described in Section 3.1.3 of this thesis) activated in the started platform follows. Since this instance of the JADE run-time is a main-container, an HTTP MTP starts by default and its local address printed (Section 3.4 details HTTP MTP functionalities). Finally, a notification indicates that a container called ‘main container’ is ready; this implies that the RMNA platform is now ready for use.

The command-line option “–gui” has the effect of launching the primary RMNA graphical interface. Figure 4.5 depicts the screenshot of RMNA Remote Agent Management graphical user interface (GUI).

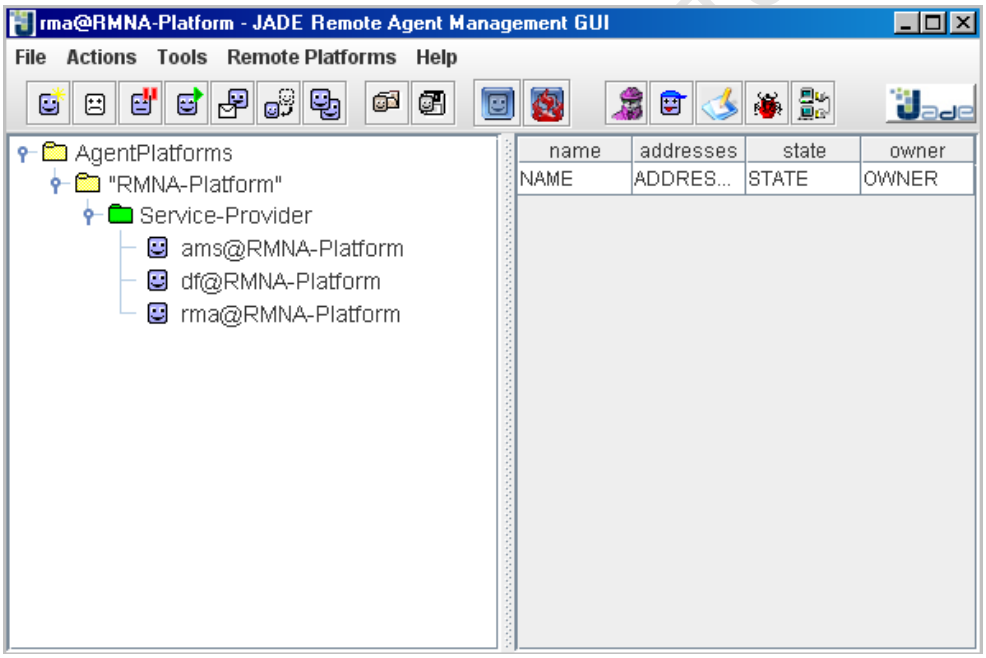

Figure 4.5. RMNA Platform GUI.

The Graphical User Interface (GUI) allows:

- remote management;
- monitoring and controlling of the status of agents (for example, stopping and restarting agents, killing of agent container and suspending or cloning agents);

- creation and starting of agent execution on the remote host are also possible from the GUI (Section 4.5.2).

With respect to Figure 4.3, additional agent containers were launched on the remaining three computers (remote hosts). They connect themselves with the main container of the Agent Platform, resulting in a *distributed system* that seems a single Agent Platform from the outside. There are two ways of starting JADE agents: directly from the command line (at container startup time) by means of the -agent option, or later on by means of the Management GUI. In respect to this research, only the resource requester (RR) agent starts from the command-line, while other agents started from the RMNA-Platform Management GUI (Section 4.5.2). The following commands were used in launching the sub-containers (i.e. Network Provider 1, Network Provider 2, and Mobile Terminal containers respectively):

```
C:.\> java jade.Boot –container –container-name Network-Provider-1 –host 137.158.125.234 –port 1099

C:./> java jade.Boot –container –container-name Network-Provider-2 –host 137.158.125.234 –port 1099

C:.\> java jade.Boot –container –container-name Mobile-Terminal –host 137.158.125.234 –port 1099 RR1:PathRequesterAgent("SP to CP")
```

A significant drawback encountered during the implementation of the resource coordination process on top of JADE platform is the RMNA performance evaluation. The major concern in this research is the response time of RMNA executions. The JADE default Sniffer agent only provides detail information about RMNA interactions after the first event. A click on the Sniffer tab will generate the default JADE Sniffer agent dialog display as shown in Figure 4.6. For this reason, the development of another tool (JavaSniffer) is therefore considered as appropriate evaluation technique that can be interfaced with the JADE.

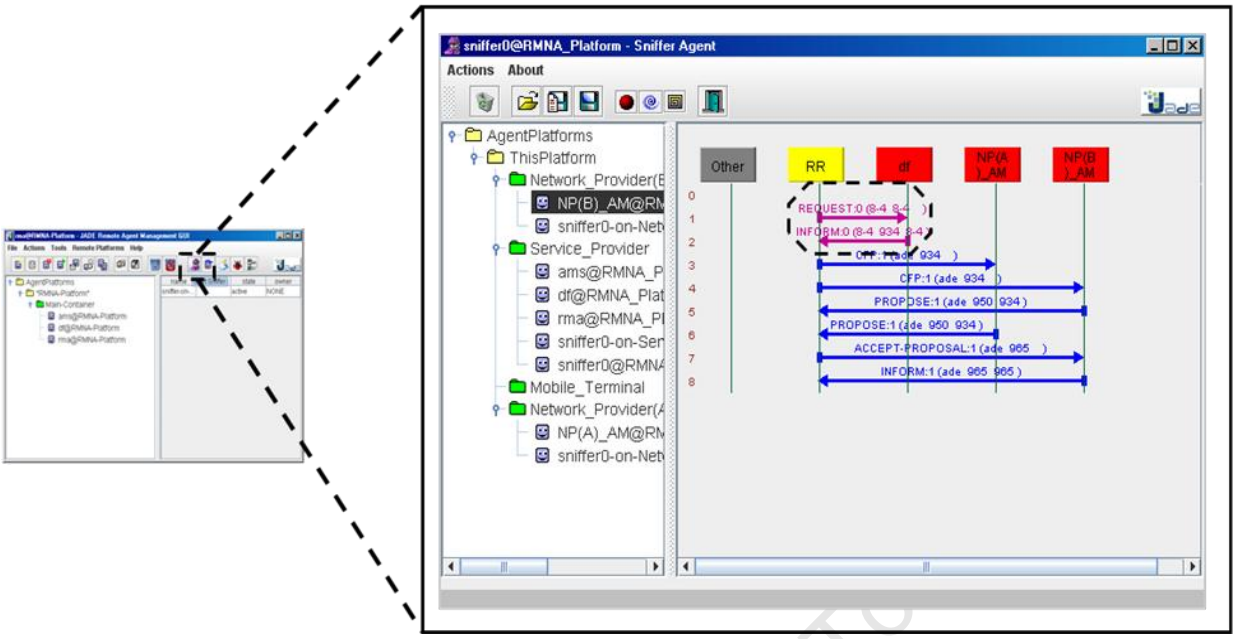

Figure 4.6. RMNA Platform: JADE Sniffer Agent Snapshot.

4.4.2 Incorporating JavaSniffer

JavaSniffer is developed by Rockwell Automation, Inc. as a standalone Java application that can remotely connect to running JADE systems and is intended as an alternative to JADE built-in sniffer. It is able to visualize messages as a low-level UML sequential diagram and provides high-level view via dynamically created traceable workflow diagrams. XML, Lisp, and BitEfficient message encodings are supported according to FIPA ACL specifications and SL, XML, JDL content languages are primarily supported. JavaSniffer offers:

- visualization of statistical information;
- message and agent filtering;
- automatic log file creation, etc.

A screenshot of JavaSniffer toolkit is presented in Figure 4.7. JavaSniffer is a freeware that can be downloaded from JADE website www.jade.tilab.com in the *community&developers/add-ons* section.

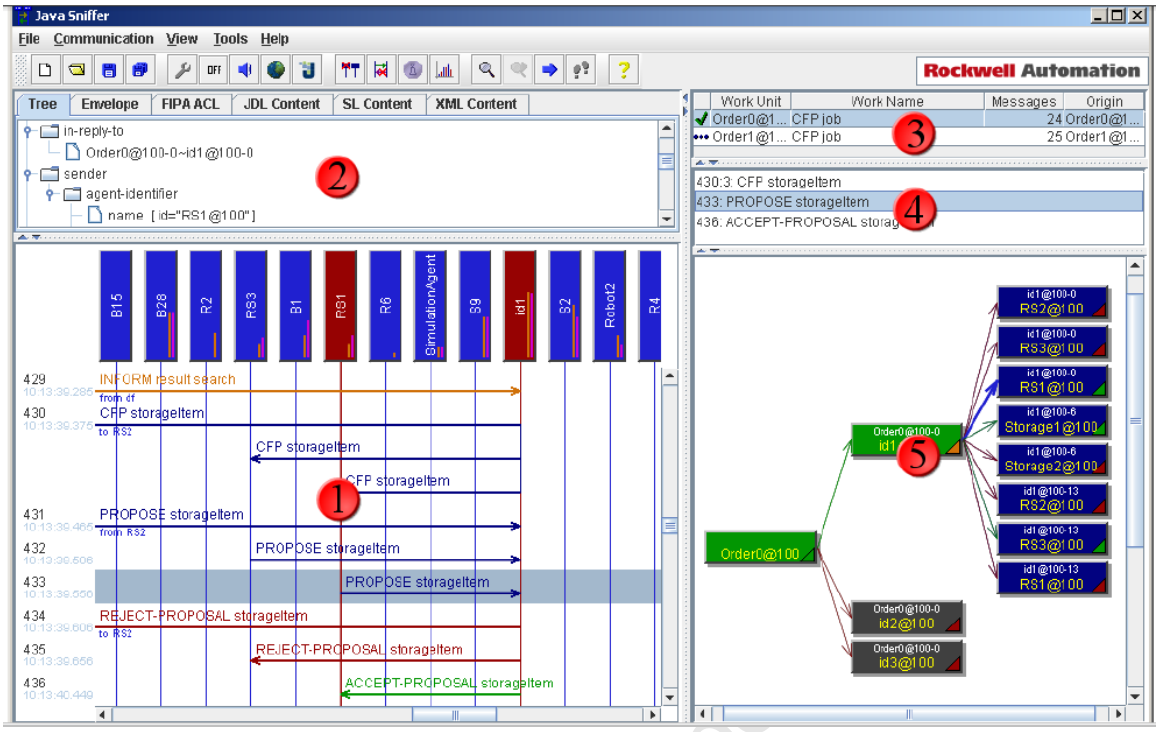

Figure 4.7. JavaSniffer GUI.

Similar to JADE, JavaSniffer requires a supporting software component to create a Java Virtual Machine (JVM), which also hosts the platform and the following Jar files (specified to the CLASSPATH) as follows:

- *JavaSniffer.jar*: Java Sniffer code is stored;
- *JavaUCS.jar*, *cipAgents2.jar*: Java CIP implementation is located;
- *JavaACSFIPA.jar*: communication languages are defined;
- *swing-layout-1.0.jar*: create professional cross platform layouts with Swing;
- *network.cfg*: network configuration is stored;
- *jcommon-1.0.0.jar*, *jfreechart-1.0.1.jar*: JFreeChart is used for graphs.

To enable a more detailed information and ability to support more RMNA messaging event, the memory of the JVM was increased to 512MB using the option `-Xmx512000000` from the

command shell. The description of the main component of the JavaSniffer application is depicted in Figure 4.7. The main window comprises of the window title, menu bar, tool bar with icons, and status line. One of the components of the tool bar is the Statistics dialog display tab that creates a GUI to analyze the number of interactive messages and performance monitoring overhead incurred by the RMNA (details in section 4.4.2.1). The JavaSniffer also provides the following five main parts for visualization of communication among agents (marked by numbers in Figure 4.7):

1) UML view which shows all messages as sequential UML diagram and the time stamp associated to every agent events

2) Message Detail view which visualizes all details of selected message

3) List of Plans which shows names of all plans

4) Workflow Detail view visualizes details of selected communication in Workflow view

5) Workflow view which visualizes how agents were contracted

4.4.2.1 Statistics Dialog

The statistics dialog display depicted in Figure 4.8 is a key functional tool of this project. It enables the evaluation and analyzing of various information about RMNA interactive communication within the JADE platform based on certain restrictions and its components are describe as follow:

- **Bind Agents to Graph Rows**: contains 6 different rows of various colours, each having its associated Edit button corresponding to the row. By pressing the Edit button to assign agents to the corresponding row, a subsequent dialog display containing all known agents to the system appears as shown in Figure 4.9. These enable filtering and selection process of which agent to analyze. Only selected agents will be considered in the computation. The tool also allows the possibility of selecting incoming, outgoing, and internal messages.

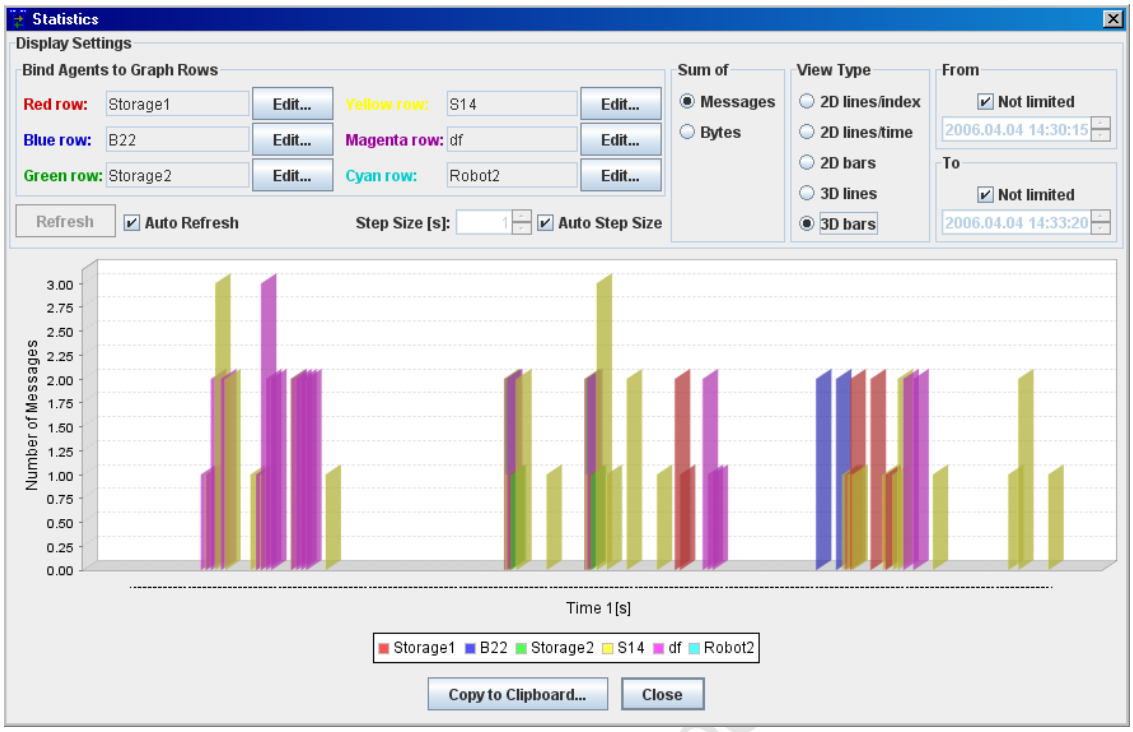

Figure 4.8. JavaSniffer: Statistics Dialog viewing 3D bars.

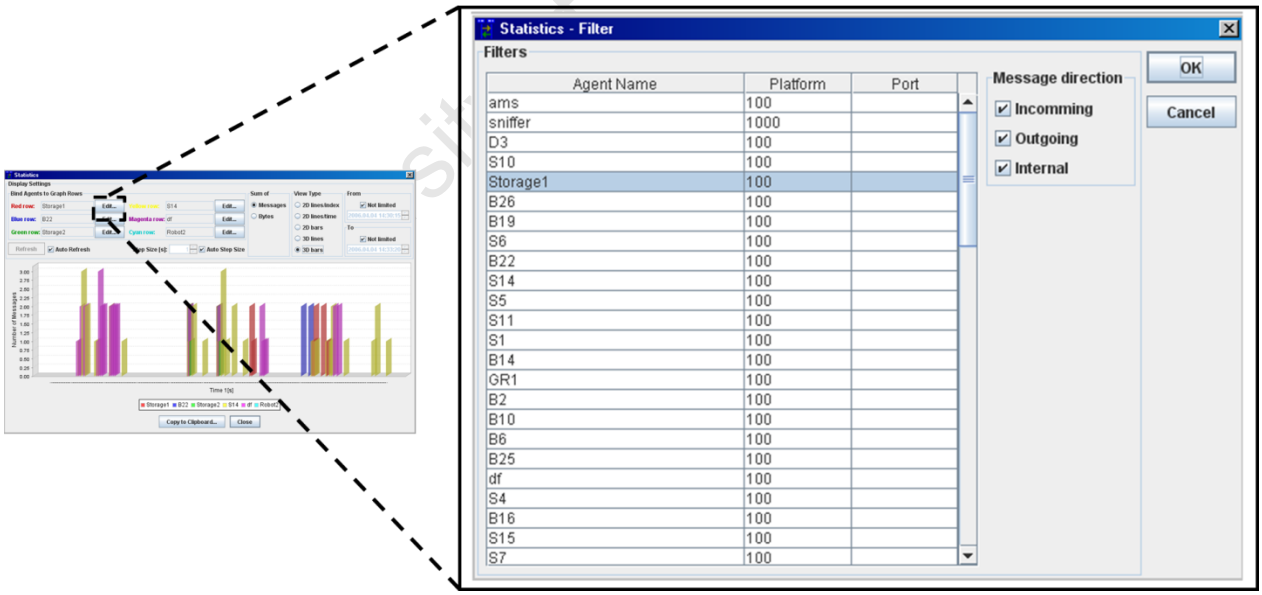

Figure 4.9. Statistics Dialog: Displaying Agent Filtering Process.

- **Sum of**: enables the Statistics Dialog to display on Y-axis in the graph either the number of messages for each agent communication or performance overhead in bytes associated with

the agent interactive communication within the system.

- **View type**: chooses among various types of graphs.
- **From**: unchecks "Not limited" to specify time limit for message arrival. All messages that arrived sooner are not considered.
- **To**: all messages that arrived later are not considered.
- **Autorefresh**: if checked the graph will be automatically refreshed based on changes in filter configuration.
- **Refresh**: manually trigger refresh of the graph.
- **Step size**: specifies time in seconds that one column in the graph will represent.
- **Auto step size**: step size is computed automatically based on “From” and “To” times so that all steps will be visible. The maximum number of steps (i.e., the number of columns in the graph) is restricted to 200.
- Copy to clipboard: specifies the size of a picture in pixels and a graph will be generated to fit this size and copied to the system clipboard.

4.4.2.2 Connecting JADE to JavaSniffer

As mentioned in Section 4.4.2, JavaSniffer is a standalone Java application that can be remotely connected to running JADE systems using a GUI known as JavaSniffer Welcome Wizard. Figure 4.10 depicts the JavaSniffer Welcome Wizard. By ticking the “**show this wizard on startup**” will automatically display the wizard each time the JavaSniffer is started otherwise the “**Help**” tab on the JavaSniffer main window as shown in Figure 4.7 provides an alternative link to displaying the Welcome Wizard.

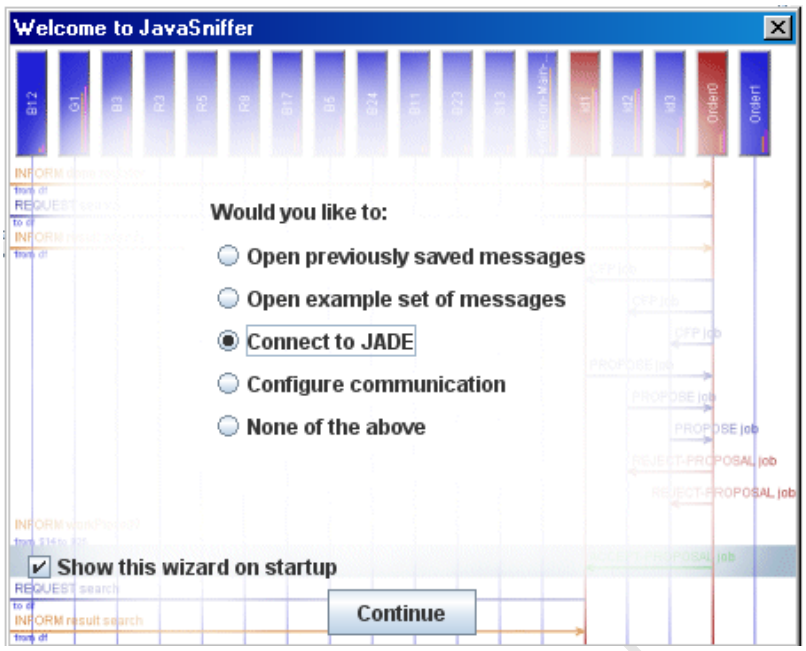

Figure 4.10. JavaSniffer: Welcome Wizard.

To enable successful connection to JADE, the JADE platform name (RMNA-Platform) as set in the command-line mode, the computer name, and the MTP http port number (see command-line output on Figure 4.4) were specified as depicted in Figure 4.11.

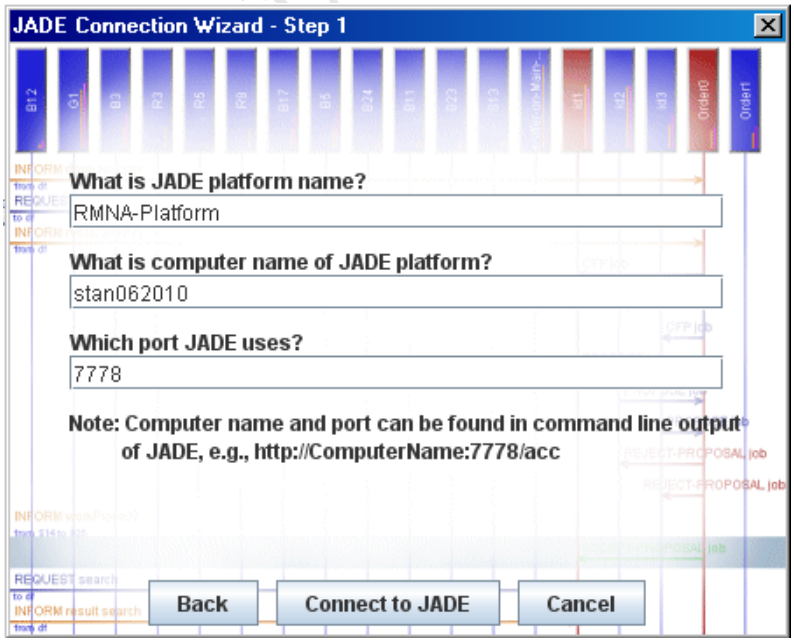

Figure 4.11. JavaSniffer: JADE Connection Wizard.

Having inserted the required parameters, then a click on the **Connect to JADE** tab will start receiving communicative messages from JADE platform (i.e., RMNA-Platform).

4.5 RMNA Setup

In previous Sections, the installation and setup of toolkit on the various networked computers required to implement this project were described. This Section is built upon the previous Sections. Building RMNA follows the procedural characteristics of creating a JADE agent. Defining a class that extends the `jade.core.Agent` class and implementing the `setup()` method will simply create an RMNA agent. The code shown in Figure 4.12 depicts the command. The RMNA full scripts can be found in the **software** sub-directory of the CD-Rom (F:\Stanley_CD\Software\JADE\JADE-src-4.0.1\jade\classes).

```
import jade.core.Agent;

public class PathRequesterAgent extends Agent {
// The node of the path to negotiate
  private String targetPathNode;
  // The list of known RMNA agents
  private AID[] RMNAAgents;
  // Put agent initializations here
    protected void setup() {
      // Printout a welcome message
      System.out.println("Hallo! Requester-agent "+getAID().getName()+" is ready.");
```

Figure 4.12. Creating RMNA

The JADE framework controls the birth of a new agent according to the following steps:

- agent constructor is executed;
- agent is given an identifier (AID);
- agent is registered with the AMS;
- agent is put in the ACTIVE state;
- `setup()` method is executed.

The agent thus created is compiled using the JCreator described in Section 4.2.4 to generate the agent class file.

4.5.1 Agent Class

The Agent class represents a common base class for user defined agents. This implies that JADE agent is simply an instance of a user defined Java class that extends the base Agent class. Therefore, the inheritance of features to accomplish basic interactions with the agent platform (registration, configuration, remote management, etc.) and a basic set of methods that can be called to implement the custom behaviour of the agent (e.g. send/receive messages, use standard interaction protocols, register with several domains, etc.).

4.5.2 Uploading RMNA Agents in the Containers

There are two ways of uploading the RMNA agent, either by using the Remote Monitoring Agent (RMA) or directly from the command-line when starting a container. As mentioned in Section 4.4.1, only the resource requester (RR) agent is started from the command-line. This command provides additional messages to the standard output as shown in Figure 4. 13.

```
C:.\> java jade.Boot -container -container-name Mobile-Terminal -
host 137.158.125.234 -port 1099 RR:PathRequesterAgent("SP to CP")
```

```
INFO: -------------------------------------
Agent container Mobile-Terminal@137.158.125.234 is ready.
---------------------------------------------
Hallo! Requester-agent RR@RMNA-Platform is ready.
Target connection is SP to CP
```

Figure 4.13. Start Agent from Command-line.

To upload the RMNA agent from the RMA, first right-click on the container to load the agent and then choose **Start New Agent** from the menu as shown in Figure 4.14.

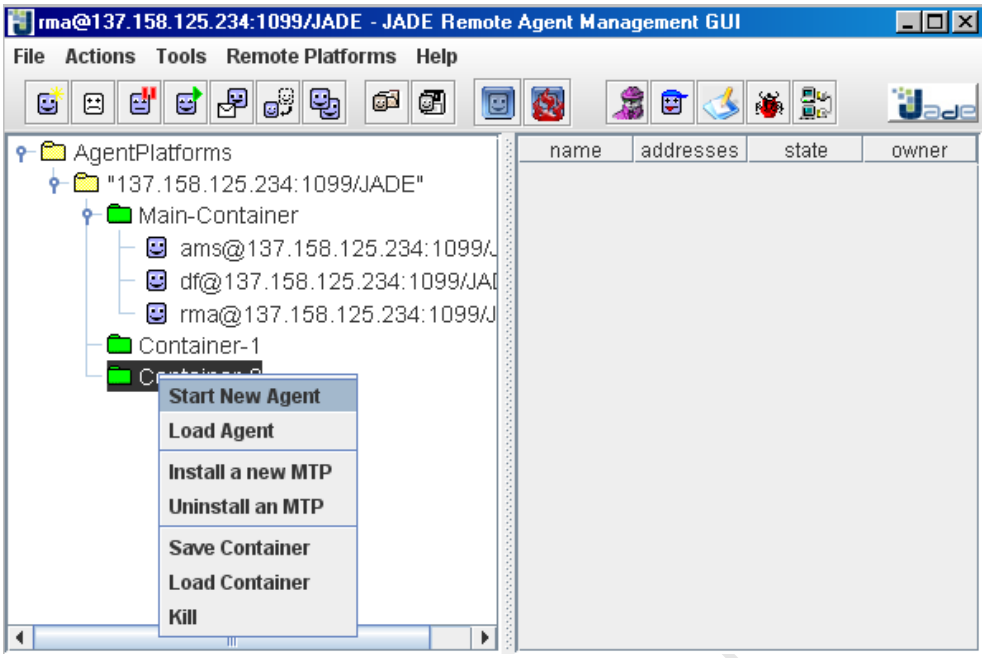

Figure 4.14. Start Agent from RMA GUI.

A click on the Start New Agent will pop up new window that allows the setting up of the agent identifier (AID) as depicted in Figure 4.15

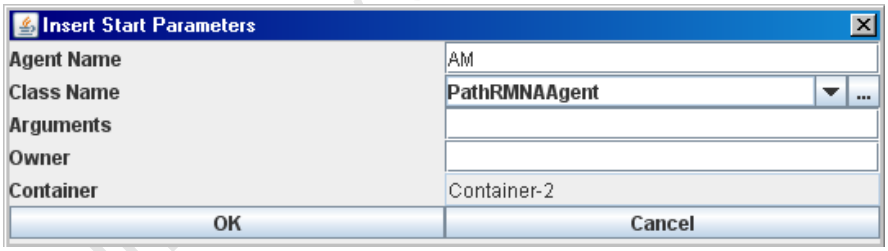

Figure 4.15. Agent Startup Window.

According to the setup and the design phase of this thesis, the AM and the BA are agents that resides on the managed node to perform network monitoring function in heterogeneous networks. Different value of variables (bandwidth and delay) for the various links between the nodes using a graphical user interface (GUI) that is generated automatically by calling the `PathRMNAGui` class once the AM and BA are created. Enter the various parameters as depicted in Figure 4.16.

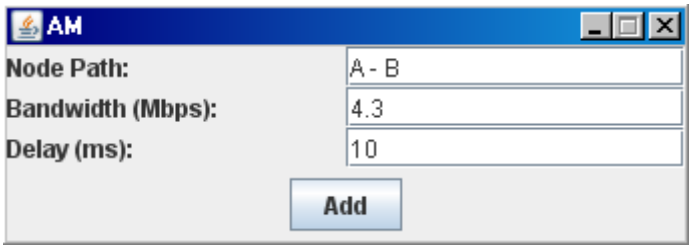

Figure 4.16. Setup Agent Parameter.

Having set the **class name** correctly, a click on the OK button of the agent parameter GUI will result to the appearance of the newly started agent in the container that hosts the agent as shown in Figure 4.17.

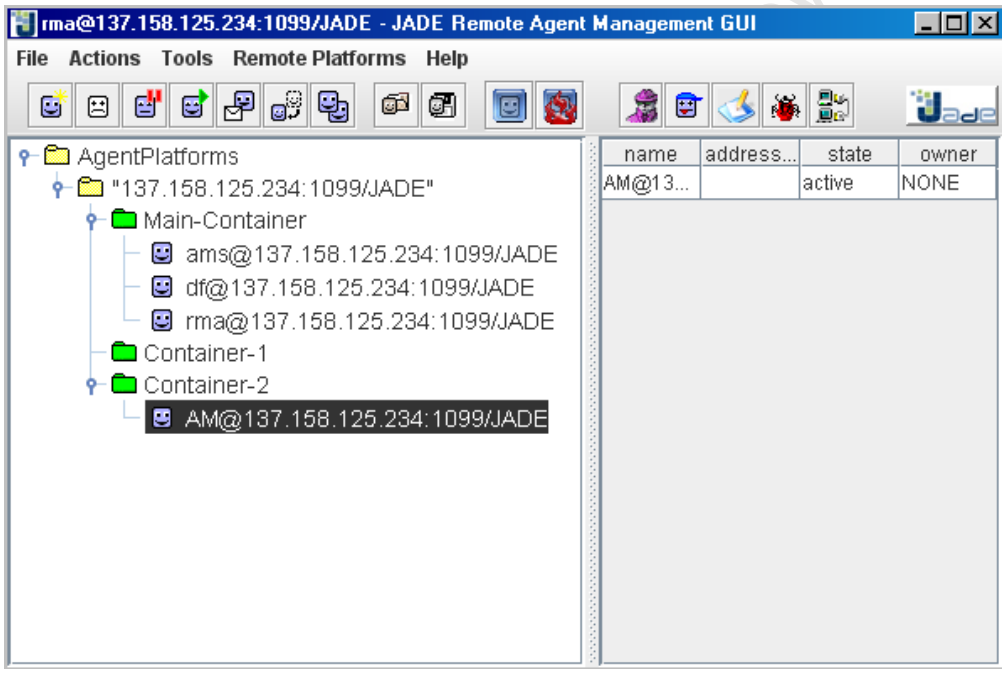

Figure 4.17. Management console showing started Agent.

In this case the name of the created agent is AM@137.158.125.234:1099/JADE. In JADE an agent identifier is represented as an instance of the `jade.core.AID` class. The `getAID()` method of the Agent class allows retrieval of the local agent identifier. The JADE AID object includes a globally unique name (GUID) plus a number of addresses. The AM is an agent living in a platform that displays its IP address 137.158.125.234.

4.6 Chapter Summary

This chapter implements RMNA agents based on the description of the design phase, which main aim is to provide coordination techniques to heterogeneous networks. The implementation framework incorporated variety of features that meet the special requirements of distributed management applications. The distributed control of this framework provides lightweight RMNA development that reduced its impact in terms of:

- Usage of network resources;
- managed devices;
- prompt execution of delay-sensitive applications.

The key features of this chapter are described as following:

- acknowledging the huge installation basis of JADE and JavaSniffer;
- variety of GUIs offering MIB functionalities, performance visualization, and high-level management operations were also implemented;
- implementation of the framework that allows users to initiates their preferences;
- providing a simple agent task that collectively executes complex interactions;
- interfacing JADE platform with JavaSniffer tool without affecting the performance statistics of RMNA;
- creation of RMNA agents both from the command line prompt and the GUI;
- defining agent class to suit different tasks;
- itineraries of travelling MAs may either by automatically built or manually assigned;
- GUI that allows the setting of agent AID and its associated network parameters.

Chapter 5 Experimental Performance Evaluation

Due to the challenges determined by the heterogeneity support in future mobile networks, the telecommunication providers have to continually express intensive attention towards mobile agent based approach for network resource coordination. The main focus of this thesis is to design and develop an RMNA-based framework for coordinating heterogeneous networks resources. This chapter aim is to evaluate the effectiveness of applying RMNA in addressing the bandwidth management problem (as the most sensitive network resource). The motivations behind this RMNA performance evaluation include:

- need to cope with the continuously increasing bandwidth demand for expanding ranges of computing-intensive applications;
- growing number of networks, services and content providers that resulted in complex service delivery environment.

These complex factors prevent the users from executing the benefits of their device capabilities when accessing/using heterogeneous networks and need special consideration in advancing the management capabilities.

The control/management transaction of RMNA is identified by its source code that describes the agent behaviour (Section 3.4.2). These source codes are compiled to generate different classes; RMNA in this context is considered an instance of user-defined Java classes. The JADE environment that serves the implementation platform of this project is a Java process. Java possesses are classes of programming languages that focus on taking into account the main characteristics of multi-agent systems. The next section describes features of class.

5.1 Class Generating Mechanism

Java classes contain *fields* and *methods*. A *field* describes the member of specific data and a *method* describes the functionality of member. Each *field* and *method* has an access level:

- `private`: accessible only in this class;

- `(package)`: accessible only in this package;
- `protected`: accessible only in this package and in all subclasses of this class;
- `public`: accessible everywhere this class is available.

Similarly, each class has one of two possible access levels:

- `(package)`: class objects is only declared and manipulated by code in this package;
- `public`: class objects is declared and manipulated by code in any package.

5.2 Performance Evaluation Metrics

To practically evaluate the resource management capability of RMNA, a number of experiments are carried out having two parameters as target:

- RMNA response time;
- signaling overhead of resource coordination.

The effect of RMNA response time and its associated signaling overhead is also investigated. This evaluation describes how RMNA-based approach promises scalability solution.

Notwithstanding this thesis considers bandwidth to be the most critical network resource to be managed due to the dynamic nature of services and the bandwidth changes occurring during runtime of a service which require some kind of bandwidth reservation mechanism. The project assumes that the need for bandwidth depends on different applications. For instance, a data transfer such as SMS will consider bandwidth utilization only in ensuring packet end-to-end delivery, while delays are tolerated. In contrast, the voice and video applications are real-time Quality of Service (QoS) processes that consider the bandwidth utilization as well as the packet end-to-end delays.

The measurement of RMNA response time when discovering the available bandwidth in order to enable a bandwidth reservation/cooperation schedule is a major interest. Timestamps are

taken using the `currentTimeMillis` method of the `java.lang.System` class for every RMNA communications (expressing the time scale in milliseconds).

5.2.1 RMNA Response Time

RMNA response time is defined as the time duration taken to complete and perform the network resource monitoring functions in the heterogeneous environment. The signaling processes are performed prior to the real data communication. Thus, measuring the response time is a critical parameter when considering the evaluations of the real-time network management.

To provide thorough evaluation, all the agent interactions and execution times are investigated. This approach provides the following benefits:

- Round trip time (RTT): this is the time taken for the resource requester agent to send a query and receive the acknowledgment. It enables the detailed understanding of the necessary communications and possible modifications for reducing the latency;
- time distribution: each phase of the task execution processes is analyzed in terms of resource requester agent migration time.

These response time measures will enable the successful evaluation of the real-time resource monitoring for the service provider at natural time scale of the services. The standard tolerable roundtrip time (RTT) in advanced GSM/EDGE networks is around 150ms [70].

5.2.2 RMNA Performance Signaling Overhead

While JADE simplifies the programming and increases the flexibility of resource monitoring activities in the heterogeneous networks, a Java Virtual Machine (JVM) would be needed for implementation, which results to performance overhead. As the mobile agents migrates from one networked computer to another, the signaling overhead increases. To quantify this signaling overhead, the JADE platform is interfaced with a JavaSniffer that provides the statistics dialog (as described in section 4.4.2).

The traditional performance implications incurred by the mobile agents have mandated

several projects to combine the functionalities of mobile agents with conventional centralized management system [58][9][7], which constitute single point of network failure and lack scalability.

The resource requester agent is a mobile agent that has the capability of migrating from one networked computer (i.e., networked node) to another while performing the required monitoring functions. As the number of messages or the number of visited nodes increases, the mobile agent size also increases, making the migration harder and leading to heavier network overhead. To evaluate how the mobile agent size affects its performance overhead, the incoming and outgoing messages of the RMNAs is investigated.

The network structure itself determines the frequency with which the agents collect data on the network and the specific routes. Thus, it is not possible to set a single value for such parameters, however, the right values can be decided after a fine tuning of the relevant network parameters. The framework of this project allows to directly delegating the management applications to the managed components (networked nodes) in the form of RMNA, therefore distributing the conventional centralized network management system.

5.3 Simulation Scenarios

RMNA performs its monitoring function by migrating from one networked computer to another. Before evaluating and presenting the RMNA coordination processes, this project first describes the networking environment depicted in Figure 5.1 and various assumptions for the different study cases are highlighted.

The scenario in Figure 5.1 involves two mobile users (iPhone and laptop) requesting video applications to run on the same service provider (SP). The SP delivers the service by invoking two different network providers (NPs). The two NPs are assumed to provide full internetworking capabilities, hence allowing the traffic flows (run between them in a loosely collaboration manner) as standardized by the ITU-T Y.2234 [8]. However, both networks manage their own link capacities and cost policies.

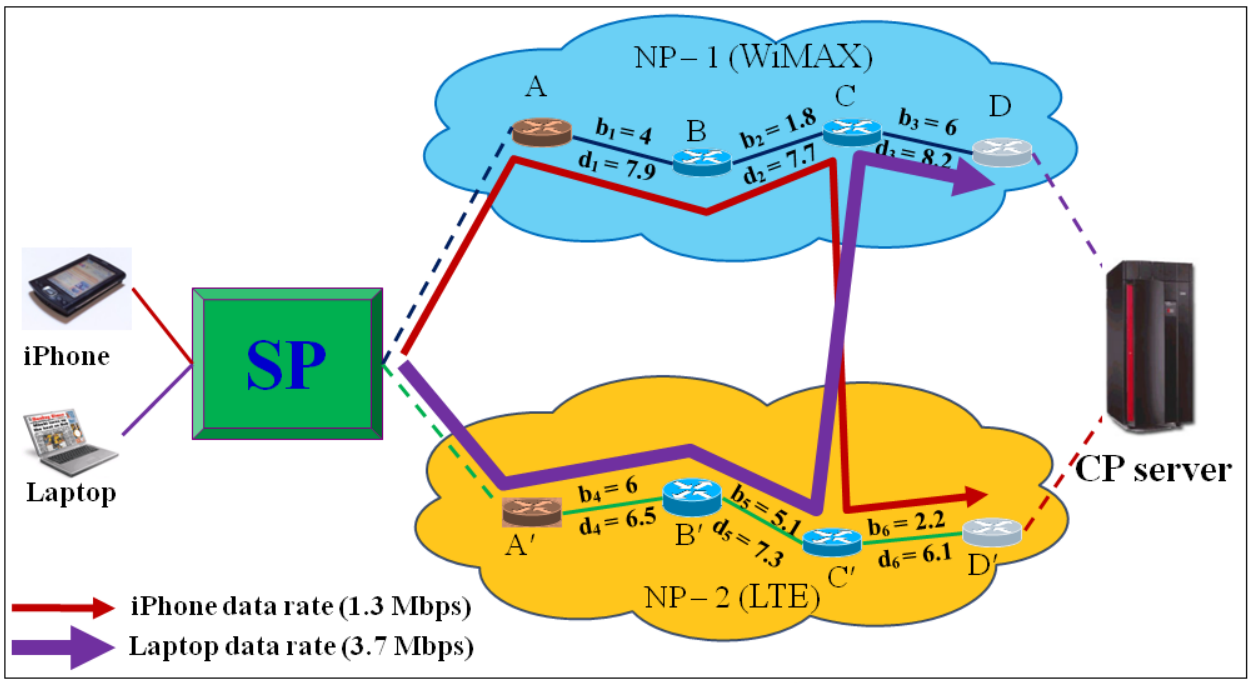

Figure 5.1. Generic study scenario.

The paths of this network model are subdivided into three segments for each network (links between nodes). The bandwidths of the various segments are assumed to be the available data rate in Mbps at a given time. The purple and red lines characterize two scenarios of this case study. Both scenarios indicate the mobile users (MU) connectivity request associated to their multimedia services. Users are not synchronized with each other due to their different device capabilities; hence networks treat the requests according to their own service level agreement (SLA).

The first case in purple illustrates a laptop-strength MU requiring a video application (such as a soccer video stream) from the SP, whose data rate should roughly correspond to 3.7 Mbps. In this case, none of the NPs infrastructures are capable to provide a complete end-to-end connectivity due to partial bandwidth limitations on their segments. Rather than turning down the service request, NP-2 completes its portion of the connection by supplementing bandwidth provided by NP-1 in order to compensate its shortfall.

The second case represented in red illustrates a handheld MU such as an iPhone requiring the same video stream, but with data rate of roughly 1.3 Mbps. In this case, both NPs can actually

provide the required bandwidth for this service. However, the NPs cooperation is beneficial in order to maximize the effectiveness of bandwidth utilization, such that the user connection request will not waste the bandwidth in order to satisfy the service request.

It is expected that these NPs will either be competing or cooperating to provide the SP with the best end-to-end connectivity configuration, by identifying the optimal selection of segments from each SP point of view. This becomes generally a complex issue since it involves several other parameters (delay, losses, node computational power, memory, cost, etc.). The RMNA framework coordinates decision making policy by migrating resource management agents to the different networked nodes and performing the real-time resource monitoring and control functions. Sending data across the network is out of this thesis scope.

5.3.1 RMNA Response Time Results

Based on the resource coordination process, the RMNA is distributed into the network in order to reduce the aggregation time of resource requester (RR) agent (i.e. the time to poll the legacy networked node bandwidth support). These agents are distributed to the network and reside with the networked nodes locally to provide the RR agent with the details regarding the management data each node can support.

The resource requester (RR) agent is instantiated by the device manager (DM) agent and sent into the network to negotiate the network resource (bandwidth) as the interface between the user and the network. The RR agent performs the required monitoring function locally by retrieving the necessary information from RMNA (i.e. AM agents) that resides on the networked nodes and only return with the achieved result. The RMNA paradigm is depicted in Figure 5.2.

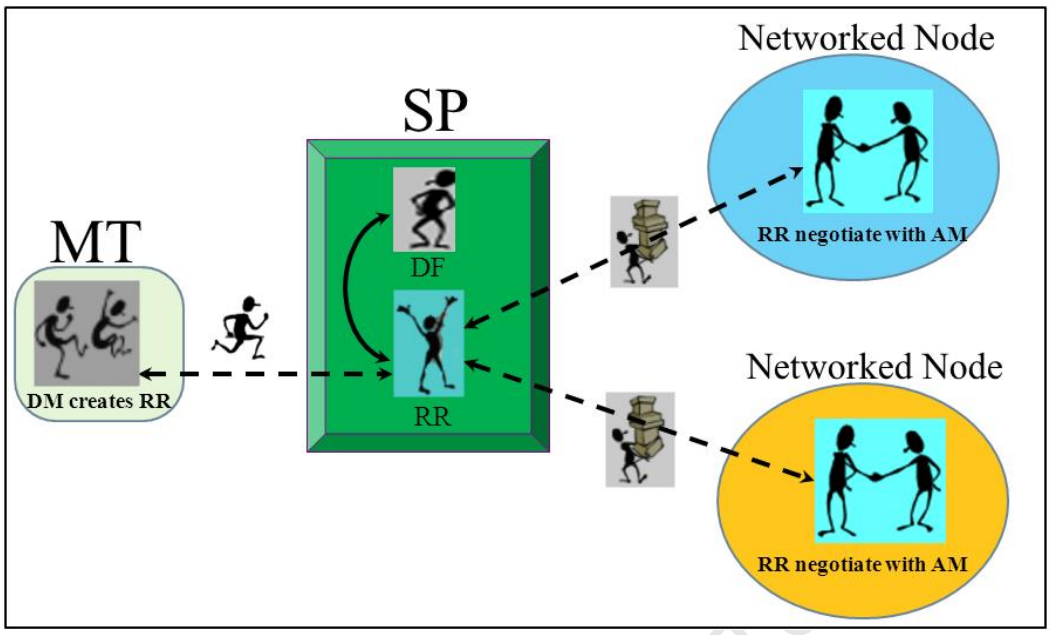

Figure 5.2. RMNA distributed cooperation paradigm.

The RMNA signaling schedule and its execution time scale for the two scenarios of Figure 5.1 are presented in Figure 5.3 and Figure 5.4 respectively.

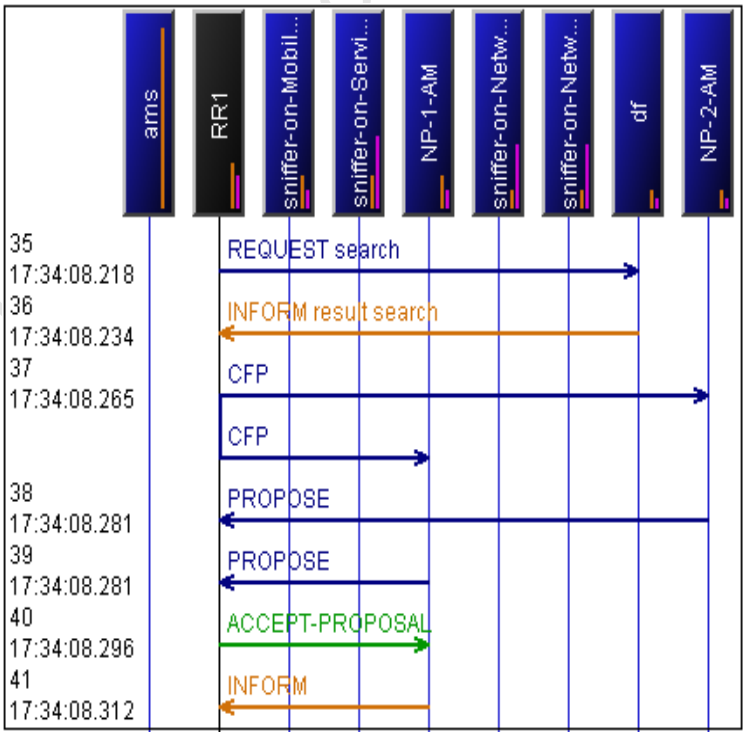

Figure 5.3. iPhone video streaming RMNA signaling.

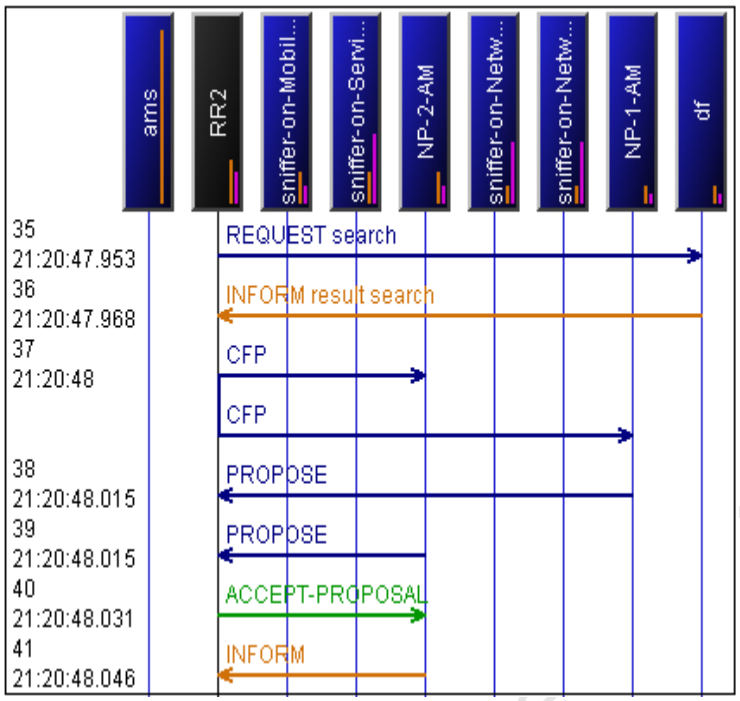

Figure 5.4. Laptop video streaming RMNA signaling.

Figure 5.3 and Figure 5.4 show the capabilities of RMNAs to locally perform resource monitoring functions according to user preferences. The user mobile terminal requires services from the network and creates a RR agent that instantiates its request by communicating the DF on the SP server. RR agent sends a "REQUEST" message asking for the list of all AM agents that can provide the required services based on users preferences. An "INFORM" message is received from DF notifying RR their various locations. RR agent then replicates itself and forward "CFP" message to both NPs agents (AM) based on the information gathered from DF. On arrival at the remote host (i.e. the NPs), the clone RR agents obtain the various AM agents bid in the form of "PROPOSE" message (containing what the link it represent can support) and return back to the original RR agent. The RR agent aggregates this information and send "ACCEPT-PROPOSAL" message about the most suitable connection. The final stage of the agent activity is to return back with the "INFORM" message from AM to the original RR agent.

A summary of the RMNA execution is automatically updated in the command shell of RR agent container confirming the reservation of that link before RR agent deletes itself from the network as shown in Figure 5.5 and Figure 5.6.

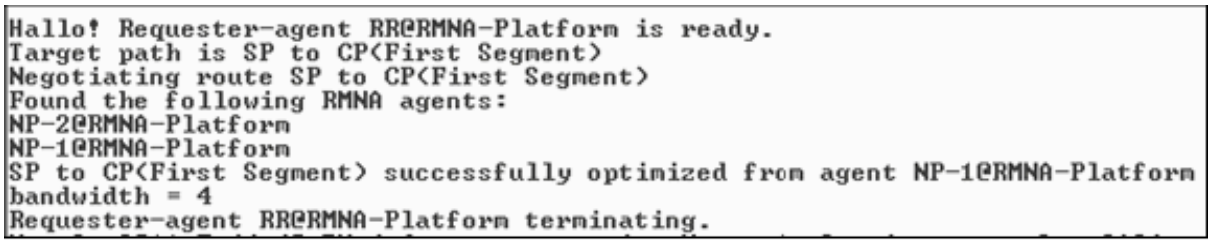

Figure 5.5. iPhone RR agent execution output.

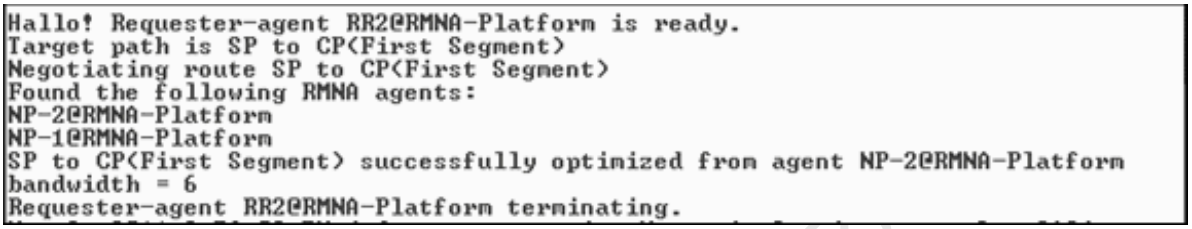

Figure 5.6. Laptop RR2 agent execution output.

The timestamp associated with every RMNA interactions can be seen from Figure 5.3 and Figure 5.4. Notice that the RR agent response time for the iPhone and Laptop has accumulated a total of 94ms and 93ms, respectively. The results are presented in Table 5.1.

Table 5.1. RR agent response time in ms.

	RR1 Aggregation Time (ms)	RR2 Aggregation Time (ms)	DF (ms)	NP-1 (WiMAX) AM (ms)	NP-2 (LTE) AM (ms)	Total Response Time (ms)
RR1	31 + 16 = 47		16	16 + 15 = 31	Synchronized with NP-1 (15)	94
RR2		32 + 15 = 47	15	Synchronized with NP-2 (16)	15 + 16 = 31	93

These results are certainly acceptable considering the voice latency measured via TeliaSonera's 4G LTE SIP calls to and from the UK was approximately 165ms on average, with peak measurements near 200ms [71]. Interestingly, the response time of RR agent for both NPs agents (AM) from different networked computer is the same for these scenarios (i.e. synchronized) as shown in Table 1. This implies the possibility of this scheme to scale effectively in coping with larger network (i.e. increase in number of NPs). This response time is an important factor when considering the successful evaluation of real-time performance of resource monitoring and control for service provider (SP) which varies according to different applications. For example, the typical range of tolerable response time for real time end-to-end gaming have been observed to be 75ms to 250ms [70].

5.3.2 RMNA Signaling Overhead Results

Signaling overhead has been a major concern for mobile agent paradigms. As mobile agent perform monitoring tasks across different network elements (specifically, distributed networks environment), its size increases. The initial size of a mobile agent also affects the agent performance since the larger the size, the more difficult the migration would become [14].

The results presented in this Section aim at controlling this mobile agent drawback by providing the mobile agent cloning technique. Since the management task of the resource requester (RR) agent is identical across the different network provider (NP) domain (i.e. analyzing the status of the networked nodes), the design phase of this thesis allows instant replication of the RR agent to minimize the deployment latency.

Figure 5.7 to Figure 5.10 evaluates the performance of the distributed control approach introduced in this thesis, which illustrates RMNA cumulative messaging counts and its performance overhead in bytes respectively.

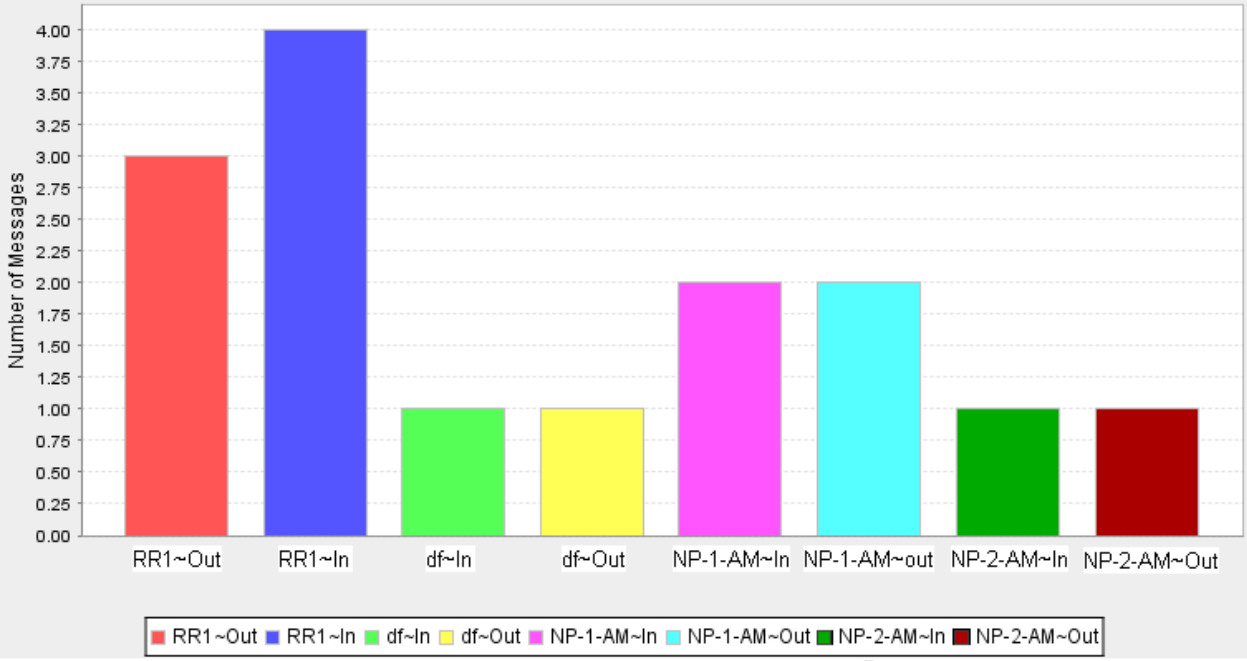

Figure 5.7. iPhone: cumulative agent messaging.

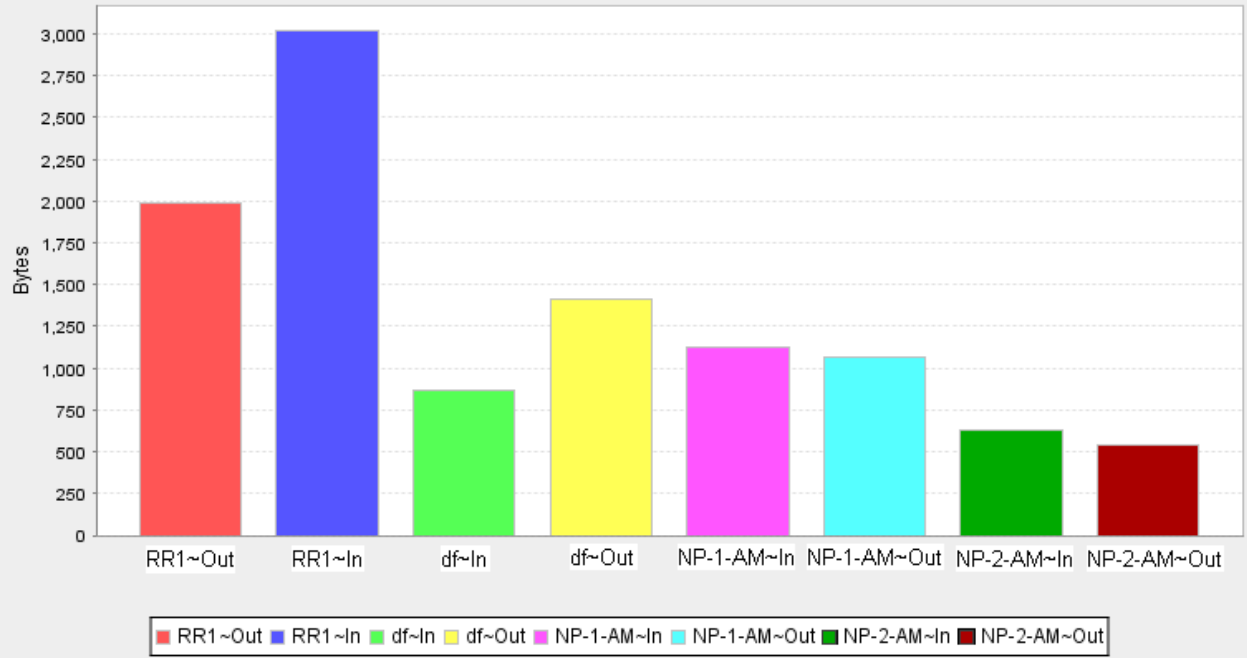

Figure 5.8. iPhone: agent monitoring overhead.

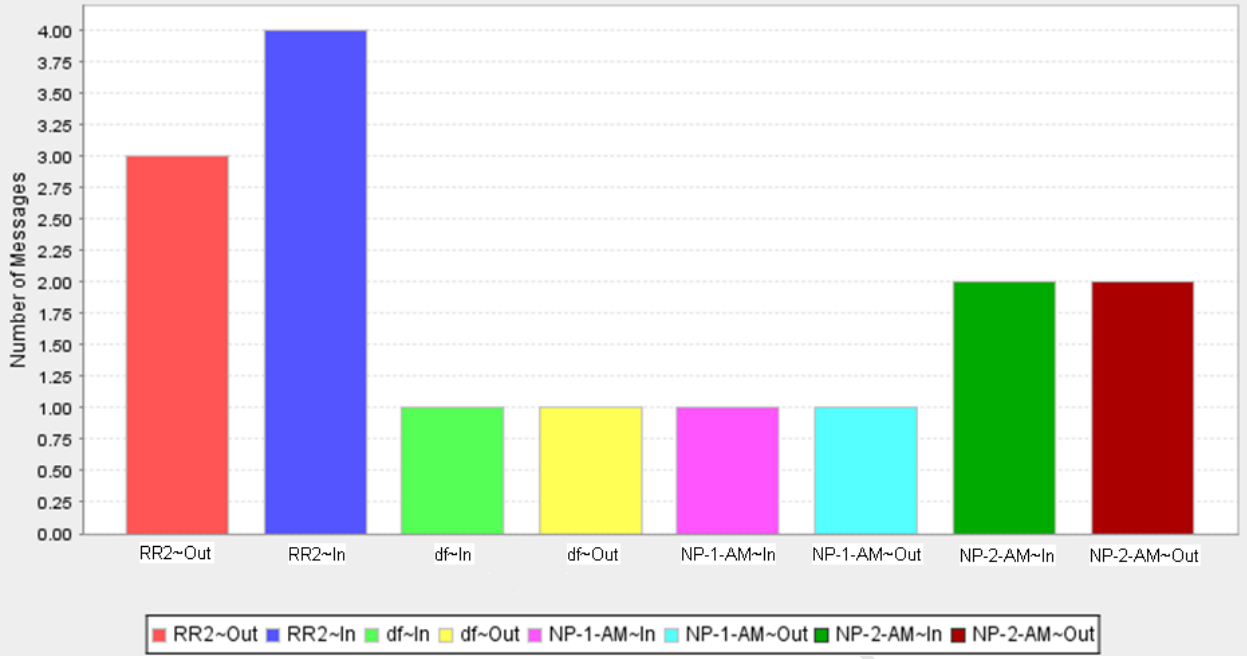

Figure 5.9. Laptop: cumulative agent messaging.

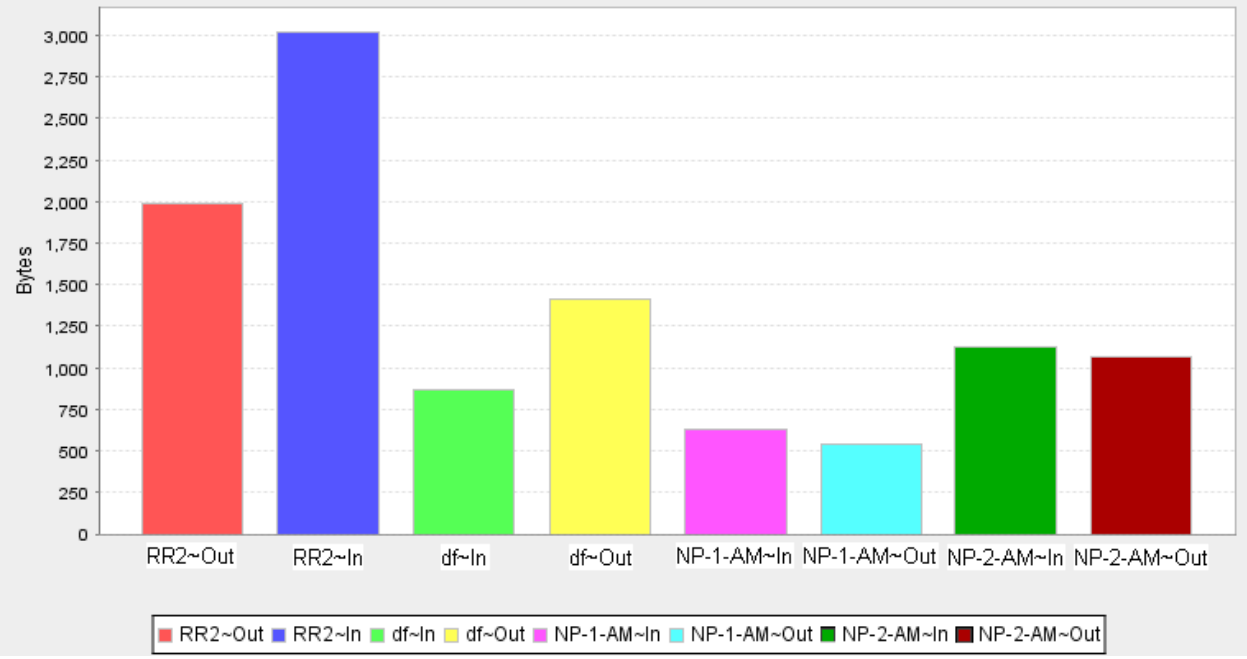

Figure 5.10. Laptop: agent monitoring overhead.

As a result of RMNA cooperative communications, the workload for each agent is low hence, enabling constant lightweight agent that reduces the signaling overhead incurred by this scheme.

It is noticed that the messaging counts for NP-1 agent (in and out) and NP-2 agents (in and out) were the same while their associated performance overhead differs for both scenarios. This justifies the local execution proposed in this scheme, as only the achieved results is communicated back to the original resource requester (RR) agent. From DF point of view, RR agent requires more information for possible execution; hence the signaling overhead associated with its messages took a different dimension. To enable reliability and efficiency, RR agent is designed to terminate after successful task execution to avoid its state size increment that can lead to poor monitoring performance.

5.3.3 RMNA Bandwidth Reservation Experimental Result

Bandwidth management is about making sure that enough bandwidth is available to meet traffic needs, otherwise, coordinating the traffic in some way to ensure that critical traffic gets through becomes necessary. As indicated in Figure 5.11, two different users require a video streaming application from the networks. Both users DM create a requester (RR agent and RR1 agent) and instantiates their negotiation for network resources (minimum available bandwidth) since network have enough resource capacity.

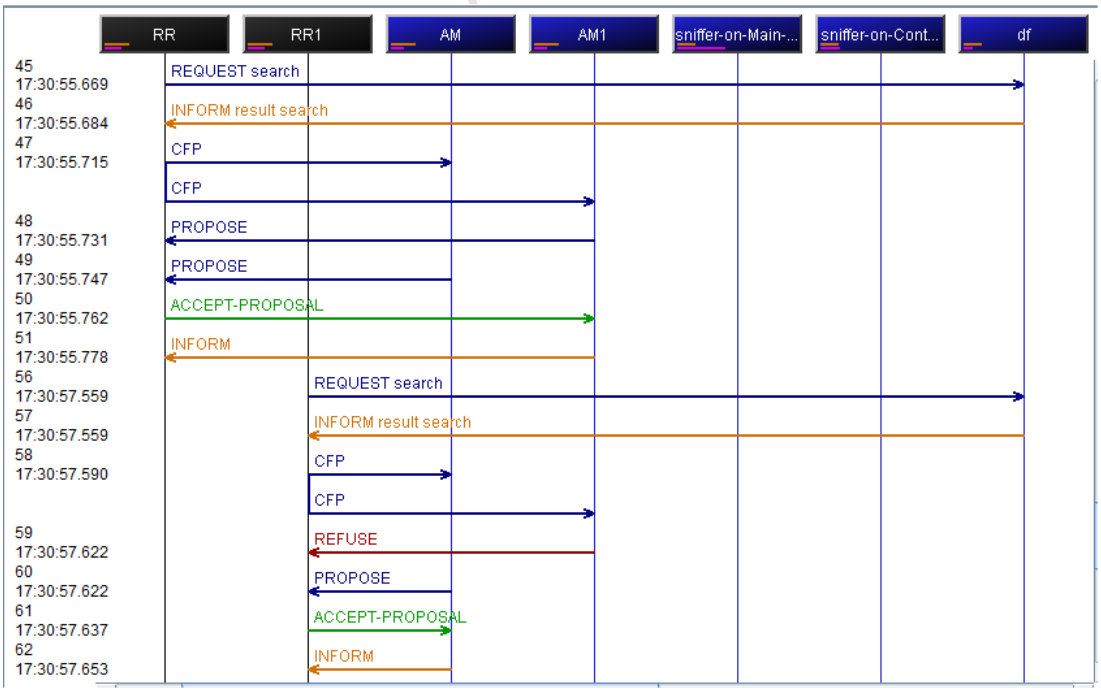

Figure 5.11. Link reservation technique.

RR agent then place a link reservation of that bandwidth amount through QoS. This reservation request is granted by AM1 and when the RR1 asks for the same link reservation, the AM1 then sends a REFUSE message to RR1 agent informing the agent that the bandwidth is no longer available; hence the RR1 then negotiate for another link within the network.

Finally, the experiment is repeated for various number of network providers that varies from 2, 4, 6 and 8 NPs. This observation is particularly important when measuring RMNA scalability (i.e. ability to gracefully cope with increase in the NPs). Table 5.2 presents the obtained result.

Table 5.2. Response Time as NP increases

Number of NP	RMNA Response Time (ms)	Linear Representation Response Time (ms)
2	109	109
4	140	218
6	187	327
8	219	436

The results displayed in Table 5.2, are also presented graphically in Figure 5.10. This set of experiments aims at investigating the effect of increase in network provider domain on the overall response time of RMNA agents, i.e. resource requester (RR) that sequentially visit an increased number of polled devices while obtaining a certain amount of data (network state information) from each agent and return to the original request initializer to deliver their collected data. The evaluated factors are the transport protocol, the network size and the volume of data obtained from each host.

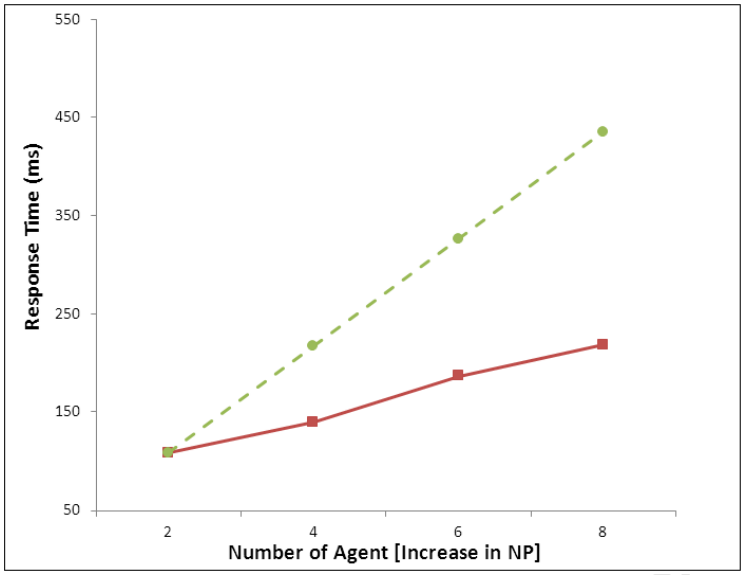

Figure 5.12. RMNA scaling ability.

As expected, the approach that reflects RMNA (red line) monitoring response time scales better than the linear (green line) representation. Despite this, it can be observed that RMNA scheme perform better when analyzing large number of nodes. The linear representation will certainly outperform RMNA scheme in terms of analyzing a single network provider nodes. This is attributed to the time required for RMNA to setup its first operations.

5.4 Chapter Summary

RMNA scheme provides a simple and flexible model to construct monitoring systems, by allowing dynamic creation, manipulation and integration of delegated monitoring objects. By taking advantage of the features of the JADE platform, RMNA then builds a dynamic and distributed management application, which enable intelligent monitoring systems. This provides a better scalability and performance than the centralized polling systems.

The observed performance overhead incurred by the resource requester agent to perform its monitoring functions between four distanced networked computers (Pentium[R] Dual-Core CPU E5400 @ 2.70GHz, Inter[R] Core[TM]2 CPU 6300 @ 1.86GHz X 2, and Inter(R) Core(TM) i5 2520M CPU @ 2.50GHz) was 5 KB (5,000 Bytes) for each application. This represents a good indication of the concrete applicability of the RMNA approach.

Chapter 6 Conclusions and Future Research Directions

The main objective of this thesis is to design and create a prototype implementation of Reconfigurable Mobile Network Agents (RMNA) based on network resource coordination. The developed prototype demonstrates how mobile agents can be distributed over the network and cooperatively providing a monitoring functions in large scale heterogeneous networks. To compliment this design approach, the thesis further describes the interfacing of different tools and methods in order to perform the desired network management tasks for the functional components of the future heterogeneous distributed system.

Given the emergence of the next generation networks (NGN), with network component such as routers being the key representatives of the open service environment (OSE), this thesis first examine in detail how the classical network management approaches will impact the resource coordination processes of the future mobile networks. Findings of these analyses generated a set of design requirements as presented in Chapter 2 that can be incorporated into the network management techniques to efficiently provide suitable management solutions.

This thesis presented a simple mobile agents approach that collectively execute a complex management tasks. A key aspect of this approach is that only the resource requester (RR) agents needed to transfer its byte codes while other management tasks where evenly distributed to different agents present within the RMNA platform. This aim towards providing a lightweight middleware to avoid performance signaling overhead associated with mobile agent system and reduces the management traffic required for future mobile networks and managed devices. In addition, it showed how the RMNA framework can exploit the advantage of the decoupled customer services and the service providers established by the NGN to enable applications to automatically adapt to changing network conditions and increase network reliability.

6.1 Main Thesis Contributions

The contributions relating to design and implementation phase of RMNA framework can be outlined as follows:

- JADE being the RMNA design and implementation platform is entirely written in Java, and Java supports the Java Virtual Machine (JVM) that provides a portable framework across all platforms, as such this framework addresses the requirement for integrated management;

- installation of JADE platform on different networked computers running both windows and Linux operating system;

- writing of Java codes to mimic (and possibly extend) the signaling and monitoring capabilities of the legacy systems (i.e. software codes), but with the added ability to enable mobility that makes RMNA suitable for mobile networks environment;

- designing of Graphical User Interface (GUI) that offers the functionalities of the Management Information Base (MIB);

- minimizing RMNA impact on networks and system resources in terms of performance overhead, the framework is characterized by a lightweight design mobile agent system that makes it particularly suitable for heterogeneous networks applications;

- Scalability issue have also been addressed, this covers a scenario where the number of network provider domain are increased to investigate the coping capability of RMNA;

- JADE provides a limited data visualization functions, the interfacing of the JADE platform with a stand-alone JavaSniffer in the thesis is the key enabler to the data capturing for performance evaluation.

6.2 Future Research Directions

The focus of this research on network resource coordination is to design and the development Reconfigurable Mobile Network Agents (RMNA) as execution platforms and applications for the management of heterogeneous distributed systems. However, time limitation hinders the ability of extending the RMNA paradigm to the last two segment of the study case scenario where the expected results will highlight the integration of these two different network computers and address the dynamic nature of mobile networking.

The design phase of Device Manager (DM) agent to sense environment and create a Resource Requester (RR) agent have received little attention. Sensing environment and automating the creation process of mobile agents can be very tedious and time consuming. This also requires advanced programming skills and detailed knowledge of the JADE platform. The introduction of new mobile agent management operation may include but not limited to the following undertaking:

- designing and writing the source code to reflect the mobile agent functionalities;
- compiling the source code requires significant amount of time for coding and debugging, expertise in programming will be an added advantage;
- possibly modify and re-compile the core JADE component classes in order to make possible the instantiation of the new mobile agent class.

The current experimental test-bed is simplified and dealt with signaling and monitoring function only. As direction for future research, it would be interesting to have distributed real network scenarios where data transportation and the implementation of network coordination processes in NGN technologies such as WiMAX and LTE will be the major concern. Achieving this will require additional time to enhance the familiarization with the features of NGN that has to be identified as part of the essential information provided by RMNA scheme.

The ideas presented in this thesis can be adapted to other platforms. Specifically, the concepts of agent behaviours, mobility, ontologies, interactions between agents, content language selection, etc, are all abstract concepts of the agent paradigm, and hence, can be adapted to other platforms which have the means to support such concepts.

Although the relationship of these experimental results to management applications may not be evidently verified, since it was not experimentally compared with any of the network management approaches analyzed in Chapter 2. But it sufficiently demonstrates the competence of the proposed framework against the limitations of centralized management approaches.

References

[1] International Telecommunication Union Question 18/2 Strategy for migration of mobile networks to IMT-2000 and beyond 3rd study period (2002 – 2006)

[2] O. Oyman et al, "Toward Enhanced Mobile Video Services over WiMAX and LTE" IEEE Communications Magazine August 2010

[3] www.cisco.com/go/vni

[4] http://www.freewimaxinfo.com/

[5] An Introduction to LTE". 3GPP LTE Encyclopedia. http://sites.google.com/site/lteencyclopedia/home. Retrieved 2010-12-03

[6] Long Term Evolution (LTE): A Technical Overview". Motorola. http://www.motorola.com/staticfiles/Business/Solutions/Industry%20Solutions/Service%20Providers/Wireless%20Operators/LTE/_Document/Static%20Files/6834_MotDoc_New.pdf. Retrieved 2010-07-03

[7] D. Chieng and I. Ho, "Brokering marketable network resources using autonomous agents" XVII World Telecommunications Congress/ISS2000, May 2000

[8] ITU-T Recommendation Y.2012, "Functional requirements and architecture of the NGN," Apr., 2010

[9] S. Papavassiliou, A. Puliafito, O. Tomarchio, and J. Ye. "Mobile Agent- Based Approach for Efficient Network Management and Resource Allocation: Framework and Applications" IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 20, NO. 4, MAY 2002

[10] D. E. Comer. "Computer networks and internets with internet applications" Third Edition, 2001.

[11] E. Chen, D. Sabaz and W. A. Gruver, "Wireless distributed systems with JADE" IEEE International Conference on Systems, Man and Cybernetics, 2004

[12] Y. Chen, Z. Xiang, Y. Dong, D. Lu. "Multi-Fractal characteristics of mobile node's traffic in wireless mesh network with aodv and dsdv routing protocols" © Springer Science+Business Media, LLC. 2009

[13] C. Bohoris, "Network performance management using network software agents" PhD thesis, June 2003

[14] I. Adhicandra, C. Pattinson, E. Shaghouei, "USING MOBILE AGENTS TO IMPROVE PERFORMANCE OF NETWORK MANAGEMENT OPERATIONS" ISBN: 1-9025-6009-4 © 2003 PGNet

[15] F. Bellifemine, F. Bergenti, G. Caire and A. Boggi, "Multi-Agent programming (Multiagent Systems, Artificial Societies, And Simulated Organizations, 2005)"

[16] C. Tao, S. Huang. "An extensible multi-agent based traffic simulation system" International Conference on Measuring Technology and Mechatronics Automation, IEEE 2009

[17] Alex L.G. Hayzelden, Rachel A. Bourne, "Agent Technology for Communication Infrastructures", ISBNs:0-471-49815-7 (Hardback); 0-470-84181-8 (Electronic)

[18] A. K. Castner, I. A. Chukhman, E. J. Colbert, M. E. Dale, B. Y. Lewis, D. R. Zaret, "An Agent-Supported Simulation Framework for Metric-Aware Dynamic Fidelity Modeling", SpringSim Vol. 1, ISBN 1-56555-313-6

[19] E. Grac e Mary Kanaga, Preethi S. H. Darius, M.L. Valarmathf, "An Efficient Multi-agent patient scheduling using market based coordination mechanism", 978-1-4244-471 1-4/09/$25 .00 IEEE IAMA 2009

[20] A. Saleem "Agent services for situation aware control of power systems with distributed generation" IEEE PES GENERAL MEETING 2009

[21] M. Hayashi, N. Matsumoto, K. Nishimura and H. Tanaka, "Design of network resource federation towards future open access networking" The 7th Advanced International Conference on Telecommunications, AICT 2011

[22] A. D. Little "The new reality of network cooperation" Telecom & Media Viewpoint 2010

[23] I. Satoh, "Building reusable mobile agents for network management", IEEE Transactions on systems, MAN, Cybernetics-Part C: Applications and reviews, VOL, 33, NO. 3, August 2003

[24] M. KONETY, R. DEV, D. PRABHAKAR, I. R. CHEN, "Mobile Components to Manage the Heterogeneous Internet" Proceedings Internet Society INET 99, http://www.isoc.org/inet99/proceedings/4m/4m_3.htm

[25] H:\Internetworking - communicating protocols and basic TCP IP.htm Retrieved 11/06/2011

[26] Alex L.G. Hayzelden, Rachel A. Bourne "Agent Technology for Communication Infrastructures" John Wiley & Sons Ltd © 2001 pp. 1

[27] S. Feit, SNMP: A Guide To Network Management, McGraw Hill, 1995

[28] M. Hayashi, N. Matsumoto, K. Nishimura and H. Tanaka, "Design of network resource federation towards future open access networking" The 7th Advanced International Conference on Telecommunications, AICT 2011

[29] http://en.wikipedia.org/wiki/Internetworking

[30] Qutaiba Ali, Salah Alabady, and Yehya Qasim "Applying Reliability Solutions to a Cooperative Network" International Arab Journal of e-Technology, Vol. 1, No. 2, June 2009

[31] G. Mapp, M. Aiash, A. Lasebae, "SECURITY MODELS FOR HETEROGENEOUS NETWORKING" Proceedings of the 2010 IEEE International Conference on Security and Cryptography (SECRYPT), Athens, 2010.

[32] K. Kimbler, "Service Interaction in Next Generation Networks: Challenges and Opportunities. Feature Interactions in Telecommunications and Software Systems". Sixths International Workshop on Feature Interactions in Telecommunications and Software Systems May 2000. 2000. ISBN 1586030655, 9781586030650.

[33] A. Conti, J. Wang, H. Shin, R. Annavajjala, and Moe Z.Win "Wireless Cooperative Networks" EURASIP Journal on Advances in Signal Processing, Volume 2008, Article ID 810149

[34] S. Saraf, http://www-scf.usc.edu/~ssaraf/EE555.pdf Retrieved 10/02/2011

[35] Data communication wide area networks computer networking, http://www.pulsewan.com/data101/layer3_switching_basics.htm Retrieved 30/11/2011

[36] http://searchnetworking.techtarget.com/definition/Application-layer Retrieved 30/11/2011

[37] K. Hana, Y. Seoa, Sungjune Y. James J. (Jong Hyuk) Parkb, T. Shonc, "Providing security vertical handoff in SARAH for heterogeneous networks" Journal of Network and Computer Applications Volume 34, Issue 6, November 2011, Pages 1903-1907

[38] M. Sourour B. Adel A. Tarek, "Ensuring security in depth based on heterogeneous network security technologies", Int. J. Inf. Secur. (2009) 8:233–246, © Springer-Verlag 2009

[39] M. Hayashi, N. Matsumoto, K. Nishimura and H. Tanaka, "Design of Network Resource Federation towards Future Open Access Networking" The Seventh Advanced International Conference on Telecommunications, AICT 2011

[40] M. Bartl, J. Hosek, T. Matocha, K. Molnar, L. Rucka, "Integration of Real Network Components into OPNET ModelerCo-simulation Process" WSEAS Transactions on Communications, Issue9, Volume 9, September 2010

[41] J. Case, M. Fedor, J. davin, Simple network management Protocol, RFC 1067, 1988

[42] Douglas R. Mauro and Kevin J. Schmidt, "Essential SNMP" ISBN: 0-596-00020-0 (http://docstore.mik.ua/orelly/networking_2ndEd/snmp/index.htm)

[43] Marshall Denhartog, "The Fast Track Introduction to SNMP Alarm Monitoring" available at http://www.dpstele.com/layers/l2/snmp_l2_tut_part1.php Retrieved 23/10/2011

[44] http://www.manageengine.com/networkmonitoring/what-is-snmp.html, Retrieved 27/12/2011

[45] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, IAB, August 1989

[46] http://www.cisco.com/en/US/docs/ios/12_0t/12_0t3/feature/guide/Snmp3.html Retrieved 21/01/2012

[47] http://www.cellsoft.de/telecom/snmp.htm Retrieved 06/10/2011

[48] C. BohorisNetwork performance management using network software agents" PhD thesis, June 2003

[49] Information Technology, Open Systems Interconnection, "Systems Management Overview", Sep. 1991.

[50] D. Gavalas, "Mobile Software Agents for Network Monitoring and Performance Management" Doctor of Philosophy Department of Electronic Systems Engineering University of Essex, June 2001

[51] Sloman M. (eds.), "Network and Distributed Systems Management", Addison-Wesley, 1994.

[52] Information Technology, Open Systems Interconnection, "Common Management Information Protocol (CMIP) – Part 1: Specification", ISO/IEC 9596, Nov. 1991.

[53] K.kazaura, K. Wakamori, and M. Matsumoto, "RoFSO: A Universal Platform for convergence of Fiber and Free-Space Optical Communication Networks", IEEE Communications Magazine, February 2010

[54] http://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture Retrieved 20/12/2012

[55] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html

[56] V. E. Zafeiris, E. A. Giakoumakis "An agent-based architecture for handover initiation and decision in 4G Networks" Proceedings of the Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM'05)

[57] Y. Chen, Z. Xiang, Y. Dong, D. Lu. "Multi-Fractal characteristics of mobile node's traffic in wireless mesh network with aodv and dsdv routing protocols" © Springer Science+Business Media, LLC. 2009

[58] H. Chao, S. Zeadally, Y. Chen,G. Martinez and R. Wang. "Next Generation Networks (NGNs). INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS Int. J. Commun. Syst. 2010; 23:691–693"

[59] P. Vilà, J.L. Marzo, E. Calle, L. Fàbreg, "Multi-Agent System Co-ordination in a Distributed Network Resource Management Scenario" 0-7803-9088-1/05/$20.00 (C)

2005 IEEE

[60] A. Suárez, M. La-Menza, E. M. Macías, and V. Sunderam, "Automatic resumption of streaming sessions over WiFi using JADE" IAENG International Journal of Computer Science, 33:1, IJCS_33_1_16, 2006

[61] I. Satoh, "Building reusable mobile agents for network management", IEEE Transactions on systems, MAN, Cybernetics-Part C: Applications and reviews, VOL, 33, NO. 3, August 2003

[62] M. Luck, P. McBurney, and C. Preist. "Agent Technology: Enabling Next Generation Computing," AgentLink, 2003, see: http://www.agentlink.org/admin/docs/2003/2003-48.pdf.

[63] A. Bieszczad, B. Pagurek and T. White, "Mobile agents for network management", IEEE Communications Surveys. http://www.comsoc.org/pubs/surveys Fourth Quarter 1998. Vol. 1 No. 1

[64] F. Bellifemine, G. Caire, D. Greenwood, "Developing multi-agent system with JADE", April, 2007

[65] F. Bellifemine, F. Bergenti, G. Caire and A. Boggi, "Multi-Agent programming (Multiagent Systems, Artificial Societies, And Simulated Organizations, 2005)"

[66] C. Tao, S. Huang. "An extensible multi-agent based traffic simulation system" International Conference on Measuring Technology and Mechatronics Automation, IEEE 2009

[67] http://jade.tilab.com/

[68] FIPA, Foundation for Intelligent Physical Agents, website: http://www.fipa.org

[69] http://jade.tilab.com/doc/api/jade/core/Runtime.html

[70] T. Blajic, D. Nogulic, M. Druzijanic, "Latency Improvement in 3G Long Term Evolution" http://www.ericsson.com/hr/about/events/archieve/2007/mipro_2007/mipro_1137.pdf, Retrieved 04/05/2012

[71] Epitiro Ltd. "LTE 'Real World' Performance Study", Broadband and Voice over LTE (VoLTE) Quality Analysis: TeliaSonera, Turku, Finland, http://www.genremobile.com/news/LTE%20Real%20World%20Performance%20Report-Finland.pdf, Retrieved 04/05/2012

Appendix A: Thesis Related Publications

Conference Papers

- S. Ogbeide, A. Murgu, "*Engineering Reconfigurable Mobile Network Agents Based on Resource Coordination*", In the proceedings of the Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2011, 4-7 September, East London, South Africa, 205

- S. Ogbiede, A. Murgu, "*Reconfigurable Mobile Agents Optimizing End-To-End Connectivity in Heterogeneous Networks*" In the proceedings of the Institute of Electrical and Electronics Engineers (IEEE) AFRICON 2011, 13-15 September, Victoria Falls, Livingstone, Zambia, ISSN: 2153-0025, Print ISBN: 978-1-61284-992-8

Appendix B: Design Platform

B. 1 Installing JADE

- The software relating to JADE can be downloaded from the JADE website at http://jade.tilab.com (Note: though JADE platform is an open source, it requires licence to regulate all use of the software).

- Download and unzip the `JadeAll.zip`. This contains the binaries, the source code, the documentation, and the examples. These four components can be downloaded separately. The four component `.zip` files are in this one `.zip` file (as `.zip` files).

- Extract the files into the directory `C:\Program Files\` (specifying directory is depending on administrator). Each of the four `.zip` files will extract into a `\jade folder` (immediately below `C:\Program Files\`). Extract all into the `C:\Program Files\` folder. Unzipping the first will create the folder `C:\Program Files\jade\`. Extracting the others into `C:\Program Files\` will add their contents also to `C:\Program Files\jade\`.

- Included in the lib directory are the five archive files containing the classes needed to run JADE. These files appears in the following format:

```
C:\Program Files\jade\lib\http.jar;

C:\Program Files\jade\lib\iiop.jar;

C:\Program Files\jade\lib\jade.jar;

C:\Program Files\jade\lib\jadeTools.jar

C:\Program Files\jade\lib\commons-codec\commons-codec-1.3.jar
```

- To launch the platform, the local Java `CLASSPATH` must be set first, i.e. the set of directories and Java archive files where the Java Virtual Machine will look for bytecode (i.e. the. class and. jar files).

- Also, to load an agent on the platform, its class file must also be reachable via the `CLASSPATH`.

For Windows Operating Systems:

- Click on Start, then right click My Computer and select Properties.
- Next, click the **Advanced** tab and click **Environment variables** (Note: **Advanced system settings** for Windows 7).
- `CLASSPATH` should be under **System variables**.
- Select `CLASSPATH` and click on **Edit**.
- This will pop up a dialogue box where you can type in the `.jar` files mentioned above at the end of the value in the Variable value text box (separated from the rest by a semicolon "`;`").
- Note that folder names are separate by semicolons and that folders are searched from left to right for a given class.

For Linux Operating Systems (Ubuntu 9.04):

- To set `CLASSPATH` for Java in Ubuntu, simply export `CLASSPATH`="your `CLASSPATH`" from either your `.bash_profile` or `.bashrc` script.
- `CLASSPATH` contains names of directory with colon "`:`" separated.
- Typing the command "`echo ${CLASSPATH}` will effectively enable the checking of the `CLASSPATH` value in Ubuntu.

B. 2 Directory structure and contributions

The directory structure of the unpacked JADE distribution is outlined below. The text coloured red indicates the generated class files that resulted from compilation of the RMNA software code.

```
jade\
    | -- Licence
    | -- classes
    | -- demo
    | -- doc\
    |       | -- index.html
    | -- lib\
    |       | -- http.jar
    |       | -- iiop.jar
    |       | -- jade.jar
    |       | -- jadeTools.jar
    |       | -- commons-codec\
    |               | -- commons-codec-1.3.jar
    | -- src\
    |       | -- demo
    |       | -- examples
    |       | -- FIPA
    |       | -- jade
    |               | -- demo
    |               | -- classes
```

```
|                   | -- PathRequesterAgent$1.class
|                   | -- PathRequesterAgent$Performer.class
|                   | -- PathRequesterAgent.class
|                   | -- PathRMNAAgent$1.class
|                   | -- PathRMNAAgent$BidRequestServer.class
|                   | -- PathRMNAAgent$ReservingServer.class
|                   | -- PathRMNAAgent.class
|                   | -- PathRMNAGui$1.class
|                   | -- PathRMNAGui$2.class
|                   | -- PathRMNAGui.class
```

Appendix C: Abbreviations

3GPP: Third Generation Partnership Project

AAA: Authentication, Authorization, and Accounting

ACC: Agent Communication Channel

ACL: Agent Communication Language

AID: Agent Identifier

AM: Adaptive Manager Agent

AMS: Agent Management System

APIs: Application Programming Interfaces

BA: Broker Agent

BGP: Border Gateway Protocol

BS: Base Station

CAGR: Compound Annual Growth Rate

CAPEX: Capital Expenditure

CH: Content Handler Agent

CMIP: Common Management Information Protocol

CMIS: Common Management Information Service

CoNet: Cooperative Networks

CORBA: Common Object Request Broker Architecture

CP: Content Provider

CS: Client/Server

CT: Container Table

DF: Directory Facilitator

DHCP: Dynamic Host Configuration Protocol

DM: Device Manager Agent

DNS: Domain Name System

DOT: Distributed Objects Technologies

EDGE: Enhanced Data for Global Evolution

ETSI: European Telecommunications Standards Institute

FCAPS: Fault, Configuration, Accounting, Performance, and Security

FIPA: Foundation for Intelligent Physical Agents

FOAN: Future Open Access Networks

FON: Future Open Network

GADT Cache: Global Agent Descriptor Table Cache

GADT: Global Agent Descriptor Table

GIOP: General Inter-ORB Protocol

GSM: Global System for Mobile

GUI: Graphical User Interface

HSPA: High Speed Packet Access

HTTP: Hypertext Transfer Protocol

ICT: Information and Communications Technology

IDE: Interactive Development Environment

IDL: Interface Definition Language

IEEE: Institute of Electrical and Electronics Engineering

IETF: Internet Engineering Task Force

IIOP: Internet Inter-ORB Protocol

IMTP: Internal Message Transport Protocol

IPTV: Internet Protocol Television

ISPs: Internet Service Providers

ITU: International Telecommunication Union

JADE: Java Agent DEvelopment

JDK: Java Development Kit

JRE: Java SE Runtime Environment

JVM: Java Virtual Machine

LADT: Local Agent Descriptor Table

LAN: Local Area Network

LTE: Long Term Evolution

MA: Mobile Agent

MIB: Management Information Base

MT: Mobile Terminal

MTP: Message Transport Protocol

MTS: Message Transport Service

MU: Mobile User

NGN: Next Generation Networks

NMS: Network Management Systems

NNI: Network Network Interface

NOs: Network Operators

NPs: Network Providers

OAN: Open Access Network

OID: Object Identifier

OMG: Object Management Group

OPEX: Operational Expenditure

OSE: Open Service Environment

OSI: Open System Interconnection

OSPF: Open Shortest Path First

QoE: Quality of Experience

QoS: Quality of Service

RAN: Radio Access Network

RATs: Radio Access Technologies

RFC: Request For Comment

RIP: Routing Information Protocol

RMA: Remote Monitoring Agent

RMI: Remote Method Invocation

RMNA: Reconfigurable Mobile Network Agents

RMON: Remote Network Monitoring

RPC: Remote Procedure Call

RR: Resource Requester

RTT: Round Trip Time

RTUs: DPS Remote Telemetry Units

SMFs: OSI-SM Systems Management Functions

SMI: Structure of Management Information

SNMP: Simple Network Management Protocol

SP: Service Provider

TCO: Total Cost of Ownership

TCP: Transmission Control Protocol

TILAB: Telecom Italia Lab

TRSA: Traded Resource Service Architecture

UDP: User Datagram Protocol

UMTS: Universal Mobile Telecommunications System

UNI: User Network Interface

USB: Universal Serial Bus

VPN: Virtual Private Network

WAN: Wide Area Network

WiMAX: Worldwide Interoperability for Microwave Access

END.