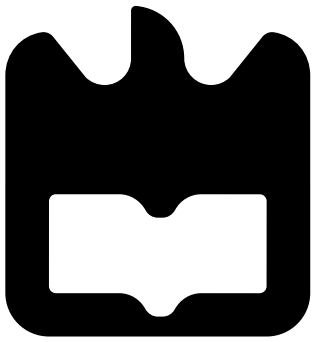
Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2015

Bojan Magušić

Visualization of Mobility and Multihoming in Wireless Networks

Visualização de Mobilidade e Multihoming em Redes sem Fios



Bojan Magušić

Visualization of Mobility and Multihoming in Wireless Networks

Visualização de Mobilidade e Multihoming em Redes sem Fios

"I am a lucky man. I have had a dream and it has come true,
and that is not a thing that happens often to men."
- *Sir Edmund Percival Hillary*

Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2015

Bojan Magušić

Visualization of Mobility and Multihoming in Wireless Networks

Visualização de Mobilidade e Multihoming em Redes sem Fios

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e de Telecomunicações, realizada sob a orientação científica de Doutora Susana Sargento, Professora Associada com Agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do co-orientador Lucas Guardalben, Investigador do Instituto de Telecomunicações.

o júri / the jury

presidente / president

Professor Doutor João Nuno Pimentel da Silva Matos
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Professora Doutora Susana Isabel Barreto de Miranda Sargento
Professora Associada com Agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro (orientadora)

Professor Doutor Sérgio Armindo Lopes Crisóstomo
Professor Auxiliar do Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto (arguente)

agradecimentos / acknowledgements

With great pleasure I dedicate the first paragraph to my sister and parents for their immense support.

Special appreciation is due to Nelson Capela for his guidance, professionalism and technical advices. The collaboration with Nelson was nothing short of a privilege and my gratitude has been earned by countless discussions about technical details which have always led to improvements of this work.

I would like to thank my supervisor Professor Susana Sargento for providing me the opportunity to work in the field I am interested in and for all her support, guidance and knowledge. Especially, I would like to thank her for accepting all the challenges that are included in being a supervisor of a mobility student

This paragraph I dedicate to my colleagues and friends: Marco Oliveira, Gonçalo Pessoa, Tiago Almeida, André Martins and Gonçalo Gomes. I was fortune to be part of such a great team.

At last, I would like to thank all the members of the Network and Architectures Group, at Institute of Telecommunications – Aveiro, for their questions and discussions which helped and provided objectivity of this work.

Resumo

Nos últimos anos, as pessoas têm-se tornado cada vez mais dependentes do uso de dispositivos móveis pessoais. Desta forma, tornou-se indispensável fornecer acesso à Internet aos dispositivos dos utilizadores, em qualquer lugar e em qualquer altura. Estes utilizadores estão em constante movimento, logo a mobilidade apresenta um fator chave nas comunicações nos dias de hoje. A utilização de *Host-multihoming* permite que os dispositivos dos utilizadores, que possuam várias interfaces de rede, passam estar ligados simultaneamente a várias redes de acesso. Com esta abordagem é possível aproveitar melhor os recursos que se encontram ao alcance do utilizador e fornecer uma ligação mais fiável. Com a integração de mobilidade e multihoming em redes sem fios é possível satisfazer as exigências cada vez mais elevadas dos utilizadores e oferecer uma melhor experiência de utilização. Com estes mecanismos implementados na rede é agora importante apresentar as suas características e funcionalidades. A forma mais intuitiva e interessante de perceber as mudanças que ocorrem na rede é através de uma plataforma de visualização. Para demonstrar as funcionalidades é necessário interagir com os nós da rede e despoletar remotamente eventos na rede. Estes eventos também devem ser visualizados de modo a perceber as mudanças que ocorrem na rede. O objetivo desta dissertação é desenvolver uma plataforma de visualização capaz de visualizar mobilidade e multihoming na rede, e interagir com a rede para despoletar remotamente eventos na mesma. Para atingir este objetivo, desenvolveu-se e integrou-se uma *framework* num protocolo de mobilidade com suporte para multihoming. De forma a demonstrar as funcionalidades da plataforma desenvolvida, implementaram-se duas *testbeds* em ambiente laboratorial, e de seguida realizaram-se testes para verificar as funcionalidades implementadas e o seu desempenho.

Abstract

In the last years we have witnessed that people are becoming increasingly dependent on the use of personal mobile devices. Providing Internet access to users' devices while maintaining Quality-of-service has become indispensable. The users are constantly moving, and mobility presents a key factor in today's communications. Host-multihoming allows end-user devices equipped with multiple network interfaces to be simultaneously connected to multiple access networks. This can optimally leverage the resources that are in the range of the end-user device and provide a greater sense of connection reliability. Implementing both mobility and multihoming in wireless networks can accommodate the increasing demands of the users and provide better user experience and network utilization. When these mechanisms are implemented in the network, it is important to present their features and demonstrate their functionalities. Visualization provides an intuitive and interesting way of understanding the changes that occur in the network. In order to demonstrate the functionalities, it is necessary to interact with the network nodes and remotely trigger network actions. These actions should also be visualized to understand the changes that happened in the network. The objective of this Dissertation is to develop a visualization platform able both to visualize mobility and multihoming in the network, and interact with the network to remotely trigger network actions. For this purpose, a framework has been developed and integrated in a mobility protocol with multihoming support. To demonstrate the functionalities of the developed platform, a testbed has been deployed in the laboratory environment and tests were performed to verify the implemented functionalities and evaluate its performance. The results show that the platform is able to visualize both mobility and multihoming in real-time. Also, the platform can interact with the network nodes in order to trigger network related actions, and visualize the result of these actions.

Contents

List of Figures

List of Tables

Acronyms

AMOD	Analysis Module
AP	Access Point
API	Application Programming Interface
BA	Binding Acknowledgement
BC	Binding Cache
BCE	Binding Cache Entry
BU	Binding Update
CMOD	Communication Module
CN	Correspondent Node
CoA	Care-of Address
DM	Demonstrator Manager
DMOD	Data Module
D-ITG	Distributed Internet Traffic Generator
ESSID	Extended Service Set Identification
FCE	Flow Cache Entry
FM	Flow Manager
FN	Foreign Network

GPS	Global Positioning System
GUI	Graphical User Interface
HA	Home Agent
HN	Home Network
HNP	Home Network Prefix
HoA	Home Address
IETF	Internet Engineering Task Force
IM	Information Manager
IMOD	Interaction Module
IP	Internet Protocol
IPv4	IP version 4
IPv6	IP version 6
JUNG	Java Universal Network/Graph
LMA	Local Mobility Anchor
LMD	Local Mobility Domain
MAC	Media Access Control
MANET	Mobile Ad-hoc Network
MAG	Mobile Access Gateway
MH	Multihoming
MIPv4	Mobile IPv4
MIPv6	Mobile IPv6
mMAG	Mobile MAG
MN	Mobile Node

MN-ID	MN Identifier
MN-HNP	MN Home Network Prefix
MNN	Mobile Network Node
MoViTo	Mobility Visualization Tool
NIS	Network Information Server
N-PMIPv6	Network PMIPv6
PBA	Proxy Binding Acknowledgement
PBU	Proxy Binding Update
PC	Personal Computer
PMIPv6	Proxy MIPv6
PoA	Point-of-Attachment
QoS	Quality-of-Service
RA	Router Advertisement
RS	Router Solicitation
RSS	Received Signal Strength
SCTP	Stream Control Transmission Protocol
SNAMP	Sensor Network Analysis and Management Platform
TCP	Transmission Control Protocol
TM	Terminal Manager
UCE	User Cache Entry
UDP	User Datagram Protocol
UIS	User Information Server
UTS	User Trigger Server

VANET	Vehicular Ad-hoc Network
VMOD	Visualization Module
WAVE	IEEE 802.11p
Wi-Fi	IEEE 802.11 a/b/g
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network

Chapter 1

Introduction

1.1 Motivation

The users have grown increasingly dependent on the use of personal mobile devices regardless of the location or time. This has been accompanied by the advances made in communication technologies, especially wireless technologies which have made it possible to provide network access not only at home and workplace, but also in rural areas where the wired infrastructure is difficult to place. With the evolution of the end-user devices and access networks, the paradigm of communication has been changing. Now, the users are in constant motion and in multiple access network environments. These environments are often heterogeneous and the users would greatly benefit from being able to use all available networks.

Providing Internet access to end-user devices while maintaining Quality-of-service (QoS) has become indispensable. In order to fulfil the ever more challenging demands of the users, various mechanisms are implemented in current networks. Mobility presents a key factor, since it allows the users to move freely while providing Internet access to the end-user devices. With mobility a seamless movement between different networks is allowed which presents an important requirement, since the users are constantly moving and expect that their devices are connected to the Internet regardless of their location. With multihoming it is possible to optimally leverage the resources that are in the range of the end-user device. The end-user devices are constantly evolving and are already equipped with multiple network interfaces. Multihoming allows multiple interfaces to be simultaneously connected to multiple access networks. This provides higher connection reliability since multiple network interfaces are connected, and disconnecting one interface will not leave the end-user

device disconnected, which is opposite to the situation when only one network interface is connected. Also, by using multihoming the end-user devices have the ability to optimize the use of resources that are in their range.

The implementation of both mobility and multihoming in the network provides a number of advantages. In order to present the advantages of mobility and multihoming, it becomes necessary to visualize their features and demonstrate their functionalities. On a daily basis the users are already exposed to visualization of various content, especially when interacting with web services. Visualization presents an interesting way of presenting the network information and changes that occur in the network. These changes can occur in real-time and affect an arbitrary number of network elements. The mechanisms implemented in the network and their functionalities are usually hidden from the end-users. Although end results of those mechanisms can be observed, they lack context and do not provide a clear understanding of the changes that leaded up to those events.

To visualize mobility and multihoming, a visualization platform is needed to present these mechanisms in an intuitive way. Since the users are familiar with accessing various web services and interacting with them, a good way to represent the network and related events would be using a visualization platform, which would represent data and a graphical user interface (GUI) which would allow user interaction. With the growing number of users in the network and the increasing complexity of mechanisms implemented, the network state is more complex to represent, and various network related events are more difficult to understand. This process can be simplified using a visualization platform that would represent the network state and changes that occur in real-time. Using a visualization platform is not only beneficial for displaying already implemented solutions and protocols, but can also be used for various applications. It can be used to monitor the network status or even determine the source of possible malfunctions. Also, should a network entity disconnect and leave the network, this change could easily be detected by using a visualization platform. In order to demonstrate the functionalities it is important to interact with the network nodes and trigger network related actions. A platform that has the ability to both visualize network events and trigger changes can present a powerful network tool. With the ability to interact with the network and trigger network actions, we have the ability to control events in the network from a central point, using only the terminal on which the platform is installed. Existing visualization platforms that present mobility or multihoming mostly focus on the visualization functionality and lack the ability to interact with the network. In order to visualize mobility, multihoming and

trigger network related actions, we need to be aware of the state of the entire network, as well as of every part of that network. With a large number of users and network entities this can present a complex problem and lead to scalability issues. When we add mobility in which the users can leave the network at any time and new users can connect to the network, to present the network topology can be a highly dynamic process.

1.2 Objectives and Contributions

In order to visualize and present both mobility and multihoming, a visualization platform is needed which will present mobility and multihoming in an intuitive way. This platform should also have the ability to interact with the network nodes and trigger network related actions. After the network related actions are triggered by the platform, the changes they cause in the network should also be made visible through the platform. The objectives of this Dissertation are the following:

- **Study the concepts related to mobility and multihoming:** in order to successfully present network related events, a clear understanding of mobility and multihoming concepts is required.
- **Study the existing protocol and implemented mechanisms:** get a clear understanding on how the existing protocol operates and is implemented, as well as about the current network mechanisms that are present in the network.
- **Development of the platform:** design and develop the platform which is able to visualize the network topology and events related to mobility and multihoming. This objective also includes finding the appropriate technology in which the platform will be developed and familiarization with that technology.
- **Integration with the mobility protocol with multihoming support:** develop entities to interact with the platform and integrate them in the existing mobility protocol with multihoming support.
- **Triggering of network related actions:** design and implement functionalities to interact with the network nodes and trigger changes in the network.
- **Design scenarios:** design of scenarios which will present the functionalities of the platform.

- **Evaluation of the developed platform:** using the developed platform, visualize the network and trigger network actions related to mobility and multihoming. This objective also includes the evaluation of the platform by performing experiments in a real network.

A paper entitled "Visualization of Mobility and Multihoming" is submitted to The 10th Conference on Telecommunications, Conftele 2015.

1.3 Document Organization

This Dissertation is organized as follows:

- **Chapter 1:** presents the Dissertation contextualization, including motivation, the proposed objectives, contributions and organization of the document.
- **Chapter 2:** emphasises the importance of data visualization and presents the state of the art on visualization frameworks and platforms. Then it explains the concepts related to mobility and multihoming.
- **Chapter 3:** presents the several scenarios related to mobility and multihoming. It also presents the multihoming framework used as base in this Dissertation, and the developed entities which were integrated in the framework, in order to provide visualization and interaction functionalities.
- **Chapter 4:** introduces the platform developed in this Dissertation. This chapter describes the challenges when developing such a platform, presents the architecture of the developed platform, describes the operation method and implementation. This chapter also explains the interaction process between the platform and the developed framework which is integrated in a mobility protocol with multihoming support.
- **Chapter 5:** describes the testbed used to verify the functionalities and performance of the developed platform. This chapter also presents the results obtained in the laboratory environment in a wireless network and a vehicular ad-hoc network.
- **Chapter 6:** presents the conclusion of the work done in this Dissertation and suggests possible improvements in the form of future work.

Chapter 2

State of the Art

2.1 Introduction

To accomplish the stated objectives, it is important to get an overview of the relevant current work done in the area of this Dissertation. This chapter emphasizes the importance of using visualizations and provides an overview of the frameworks and software tools used to visualize data. In addition, several relevant visualization platforms are presented that have the ability to visualize network data. In order to understand the displayed visualization, it is necessary to comprehend the main concepts related to both mobility and multihoming. For this purpose, an explanation of mobility and multihoming is provided, and their functionalities described. This chapter is organized in the following way:

Section 2.2 explains the motivation for data visualization and presents the advantages. In order to visualize network data, an overview is provided of current visualization frameworks and software tools. This section also describes the functionalities and abilities of the relevant visualization platforms.

Section 2.3 presents mobility and describes the main concepts found in current mobility protocols.

Section 2.4 introduces multihoming, describes its applications, explains benefits of using multihoming and provides an overview of the current approaches related to multihoming.

Finally, section 2.5 provides the chapter considerations with a summary of this chapter.

2.2 Visualization

Visualizations are the graphic presentations of data - portrays meant to reveal complex information [1]. They are useful to leverage the perceptual abilities of humans, in order to find features in network structure and data. However, this process is inherently difficult and requires exploration strategy [2]. This section states the reasons for data visualization and provides an overview of the relevant visualization frameworks, software tools and platforms.

2.2.1 Visualizing Data

The human visual system is very well built for visual analysis and understanding visualizations. Large amount of data is coming to the human brain through the eyes. The optic nerve sends data to our brain very quickly, with an estimated transmission speed of about 9 Mb/sec [3]. Then this data is processed by the brain, using very sophisticated techniques, such as edge detection, shape recognition and pattern matching. The brains ability for pattern matching is the key for rapidly processing visually presented information, since typically the messages that are hidden in data are represented in patterns.

Although the brain is able to effectively process visualizations, this does not present a sufficient reason for visualizing data related to the network. Importance of data visualization can be seen in the following example. Figure 2.1 depicts the *Anscombe's Quartet* [4], four datasets that were constructed by *Francis Anscombe*. When these datasets are analysed, they have nearly identical simple statistical characteristics: the mean, the sample variance, the correlation and the regression. However when these datasets are visualized graphically, their visualization shows a clear difference between them. This example emphasises the importance of using visualizations. The visualization allows for much better and faster comprehension of data, when compared to analytical comparison of numerical data.

The graphical display of data is superior when compared to simple textual representation of information. Data visualization presents an interesting way of understanding information. It is the graphical presentation of abstract information used for two purposes: data analysis and communication [5]. It is important to visualize data in order to discover and understand their meaning. Visualizations can be used to portray a variety of information. The focus of this Dissertation is visualization of network data related to mobility and multihoming. Data regarding mobility and multihoming is abstract, since it describes things that are not physical. Although data visualization usually features re-

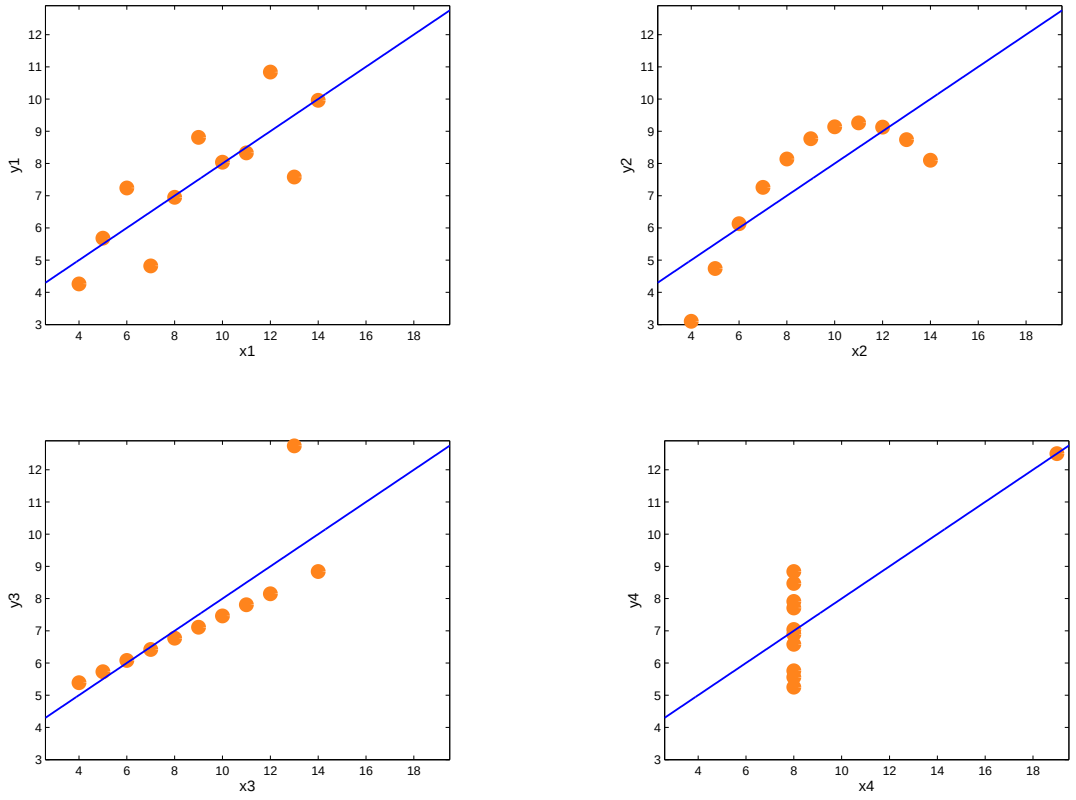


Figure 2.1: Anscombe's Quartet

lationships between quantitative values, it can also be used to display relationships that are not quantitative in nature. This corresponds to the situation when network data is presented and visualized.

Network data usually consists of groups of entities and connections between these entities. This data provides important information about the network, but without the proper visualization needed to understand this data, many informations might go unnoticed or unrecognised. Nowadays, many standard or default software tools provide the ability to present data in the form of charts and graphs. These tools create simple data presentations, but lack the ability to interact with the data and understand its meaning.

Visualization is an attractive way to understand behaviours of complex systems and networks. It allows the information to be presented in an intuitive and user-friendly way. For this purpose, visualization platforms have been used to present various networks and their application. By using visualization platforms, information about the network can be visualized in real-time. The networks could be either dynamic with a large number of

nodes and edge attributes which change over time, or static with only a few fixed nodes with edge attributes which do not change over time. By using visualization platforms to display data, important insights can be gained through the display of the network in visual form [6]. The visualization of networks presents many challenges [7] that need to be taken into consideration. The network topology can change rapidly in unpredictable ways. Therefore, scalability issues [8] must be considered when developing visualization platforms as the size of the network can increase significantly over time. When developing a visualization platform, it is important to identify the purpose of the platform and the main characteristics. Their use can be versatile and complicated as well as specific and with only a few features. Many of the existing platforms use the traditional node-link graph representation to display networks, which has proven to be intuitive and provides a clear overview of the relevant information.

2.2.2 Visualization Frameworks and Software Tools

Nowadays, there exists a variety of software tools and frameworks which can be used to visualize data. Since the focus of this Dissertation is visualizing mobility and multihoming, of interest are frameworks and software tools used to visualize network data.

NetDraw

NetDraw [9] is a standalone program that operates on the Windows platform and is developed for visualizing network data. It offers a variety of features such as node attributes, multiple relations, relations based on values, multiple layouts, social network analysis, display options and presentation of 2-mode data. The network is visualized using *NetDraw*s own data format, called the *VNA* format. The *VNA* format is a specific data format which allows network data to be stored together with the attributes of the nodes, as well as information on how that data should be visualized. The attributes assigned to nodes define multiple characteristics such as color, shape and the size of nodes. The visualization presented in *NetDraw* is in the form of a node-link graph, where the *VNA* format determines the presentation of the nodes based on the assigned characteristics. The *VNA* format relies on textual data which is processed and visualized, rather then relying on numeric code. This presents a significant advantage of using *NetDraw*, since a file containing network data can be created using any textual editor or word processor. Also, *NetDraw* has the ability to read multiple data formats besides the *VNA* format, which

makes it very flexible. It can be used independently or as part of the *UCINET*, a software package used primarily for social networks analysis. When *NetDraw* is used independently, node attributes can be customized to correspond to specific network entities and network types. After the node attributes are specified, the network data is saved together with the layout information, such as colors or spatial coordinates. Then based on the network data and layout information, the visualization is presented with the option to export it in a variety of formats. Figure 2.2 [9] shows the visualization displayed in *NetDraw*. The visualization presented is fairly simple, although *NetDraw* has the ability to handle multiple relations at the same time, which allows more complex visualization.

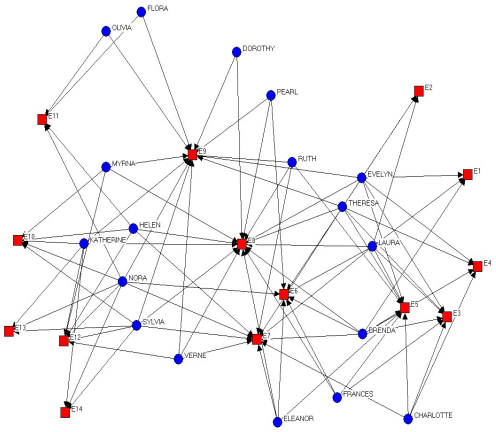


Figure 2.2: Visualization in NetDraw [9]

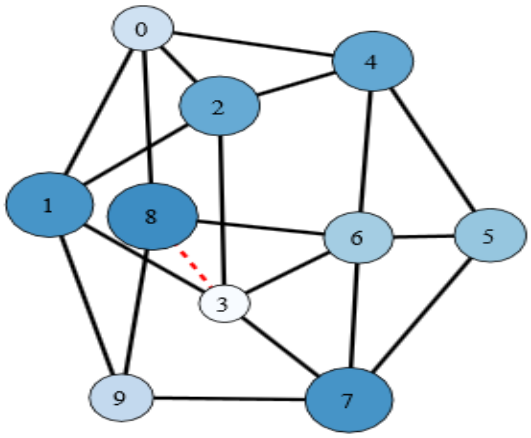


Figure 2.3: Visualization in NetworkX [10]

NetworkX

NetworkX is a Python language package for exploration and analysis of networks and network algorithms [10]. The initial development of *NetworkX* aimed to build an open-source tool base that could easily grow in a multidisciplinary environment with users and developers. The goal was to provide an open-source network analysis tool that is well-tested, well-documented and that could easily span research application domains. The structure of a network or graph in *NetworkX* is encoded in the edges between the nodes. The core package provides basic network data structures for representing many types of networks or graphs. Since *NetworkX* is written in Python, every nodes in the graph can be an arbitrary Python object and arbitrary objects can be associated to edges. This presents a powerful advantage of *NetworkX*, since the network structure can be integrated with custom objects and data structures. This flexibility makes *NetworkX* versatile for

representing networks from many different scientific fields. However, since the nodes are Python objects, they have the same restriction as Python dictionaries: they must be hash-able objects. Figure 2.3 [10] shows the visualization in *NetworkX*. The visualization can be displayed by reading various graph formats, and then saving the visualization as various graph formats. Once a network is represented as a *NetworkX* object, the network structure can be analysed using implemented graph algorithms. The graph algorithms that are implemented in *NetworkX* include finding the shortest path, clustering coefficients, degree distributions, spectral measures and communities [10]. *NetworkX* can also serve as a platform to develop and test new network algorithms. In addition to the graph algorithms, *NetworkX* includes functions for computing network statistics and metrics, such as diameter and betweenness centrality.

Pajek

Pajek [11] is a standalone program for analysis and visualization of large networks. It is freely available and operates on the Windows platform. The word "pajek" in the Slovene language means "spider". *Pajek* can be used to visualize large networks with some thousands or even million nodes. The implementation is done in Delphi (Pascal) and is developed based on the experience gained in the development of graph data structures. The main motivation for the development of *Pajek* was the observation that several sources of large networks already exist, in machine-readable form. *Pajek* is intended to provide tools for analysis and visualization of a variety of networks: collaboration networks, Internet networks, telecommunication networks, diffusion networks and even data mining (2-mode networks). The network is presented in the form of a node-link graph, where the nodes present network entities, and links the connections between these entities. Figure 2.4 [12] shows the visualization in *Pajek*. The program provides the ability to find clusters in a network, shrink vertices in clusters and show relations among them, extract vertices that belong to the same clusters and show them separately, possibly with parts of the context. Besides undirected and directed networks, *Pajek* also supports 2-mode networks and networks changing over time. In *Pajek* the analysis and visualization is performed using six data types. These data types specify the network and the entities of which it is composed. In *Pajek*, several algorithms are implemented which mainly provide different ways of identifying interesting substructures in a given network. These algorithms provide ability for finding the shortest path, maximum flow, critical paths, extracting sub-network, shrinking clusters in network and reordering. In addition, *Pajek* has efficient algorithms implemented

specially for analysis of large networks. Although it was developed primarily for analysis of large networks, *Pajek* can be used for visualizing small networks. It also contains some data analysis procedures with higher order time complexities which can therefore be used only on smaller networks, or selecting parts of large networks. The visualization of networks in *Pajek* is done by identifying the interesting substructures in the network, and then presenting the visualization of these substructures in separate views in order to visualize the network in details. In *Pajek* special emphasis is on the automatic generation of network layouts, for which purpose several algorithms for automatic graph drawing are implemented. These algorithms were modified and extended to enable additional options such as drawing with constraints and drawing in 3D space. *Pajek* also provided tools that allow manual editing of graph layouts.

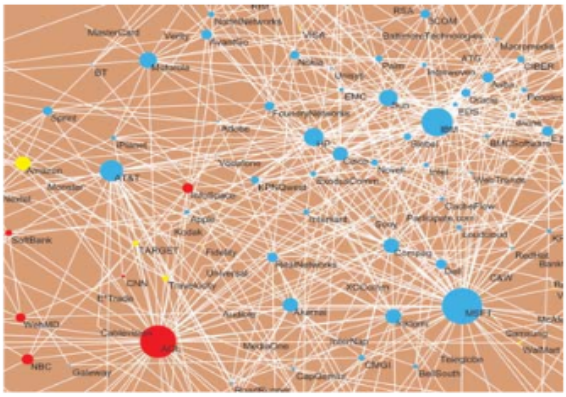


Figure 2.4: Visualization in Pajek [12]

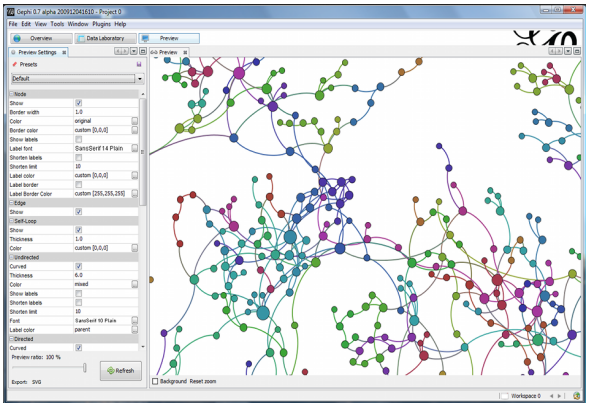


Figure 2.5: Visualization in Gephi [13]

Gephi

Gephi [14] is an open-source software for graph and network analysis. It consists of multiple modules which can import, visualize, spatialize, filter, manipulate and export all types of networks. The main entity in *Gephi* is the visualization module, which uses a special 3D render engine to visualize graphs in real-time. This technique uses the PCs' graphic card while leaving the processor free for other computing. It is built on a multi-task model and can take advantage of multiprocessors in order to visualize large networks. *Gephi* allows for the nodes in the network to be customized, so instead of a classical shape, the node design can be a panel or picture. In order to visualize the network, *Gephi* runs highly configurable layout algorithms in real-time on the window that is displaying the graph. This allows adjusting the settings of a selected algorithm in real-time. Several algorithms can run

simultaneously in *Gephi*, in separate workspaces without blocking the user interface. The user interface is structured into separate workspaces, where in each workspace a different task can be performed. *Gephi* provides software extensibility, allowing an algorithm, tool or filter to be added easily to the program. The filters are used in *Gephi* to select a group of nodes with the same properties such as thresholds and range. Figure 2.5 [13] shows the visualization in *Gephi*. The networks that are displayed with the visualization module can be exported in a vector graphics format. The network displayed should be clear and readable, for this purpose *Gephi* takes into consideration fonts and labels. Any data attribute can be associated to a node in the graph. Labels of these attributes are shown in the visualization module, by the text module without overlapping. The ability to explore networks that change over time has been incorporated into *Gephi* from the beginning. The architecture supports graphs whose structure or content varies over time by adding a time component, which allows for previous network state to be retrieved. This allows the dynamic networks to be visualized as movie sequences. The architecture of *Gephi* is interoperable and the dynamic module can get network data from either a compatible graph file or from external data sources. The data source can send network data to the dynamic module at any time, and the result will be immediately visualized by the visualization module.

JUNG

Java Universal Network/Graph (JUNG) is a free, open-source software library that provides a common and extendible language for the manipulation, analysis and visualization of data that can be presented as a graph or network [15]. It is written in the Java programming language, allowing the applications based on JUNG to take advantage of extensive built-in capabilities of the Java Application Programming Interface (API). The features of JUNG include the following [15]:

- Supporting a variety of representations of entities and their relations, including undirected and directed graphs, graphs that contain more than one type of vertex or edge, graphs with parallel edges.

- Basic mechanisms for annotating graphs, entities and relations between entities. These mechanisms support the creation of complex tools that can visualize and examine the relations between entities, as well as additional information attached to each entity and relation.

- Implementation of various algorithms from graph theory, exploratory data analysis, social network analysis and machine learning. This includes procedures for decomposition, optimization, vertex clustering, statistical analysis, calculation of network distances, flows and ranking measures.
- Visualization framework which allows the development of complex tools including provided layout and rendering algorithms, or using the framework to create custom algorithms.
- Custom mechanisms for extracting subsets of a network, allowing representation of certain parts of the network rather then the entire network.

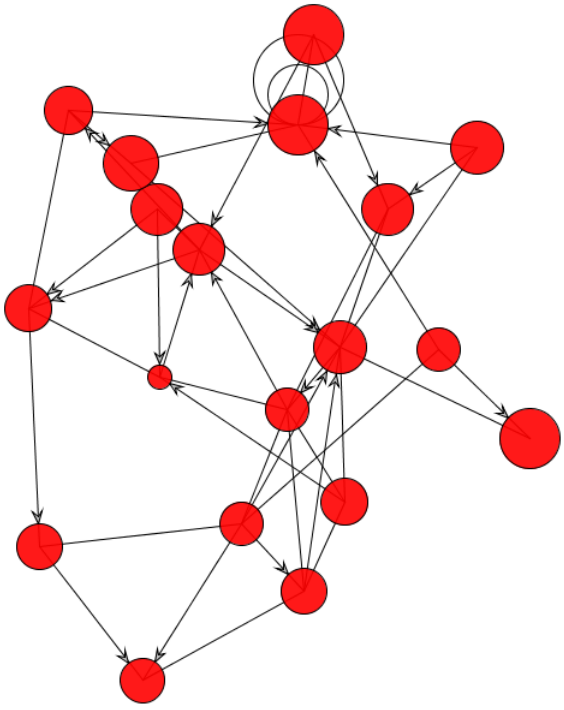


Figure 2.6: Visualization in JUNG

The stated features make JUNG a good candidate for use in development of platforms for exploratory data analysis on relational data sets. Figure 2.6 shows the visualization in JUNG. JUNG is not a standalone tool, but rather a library that can be used for development of complex tools [16] with graphical user interface (GUI). It can be used to build tools that are network oriented or provide capabilities to existing networks. Building

complex tools with JUNG requires knowledge of programming in Java. As a library, JUNG can be used to develop complex tools with specific abilities, rather then using generic network tools. When developing an application based on JUNG, the Java API can be used to assign arbitrary data attributes to any node in the network. The Java API provides a wide choice of options and offers good flexibility when using JUNG as a library. It allows specific layout and filtering mechanism to be easily implemented. JUNG can be used to visualize both small and large networks, making its use versatile.

2.2.3 Visualization Platforms

Network visualization is an interesting topic and has been widely researched in the area of wireless networks. Clarke et al. propose *NiCE* [17], a cross platform which provides setup, runtime, analysis and visualization environment for mobile ad-hoc networks (MANETs). The platform *NiCE* uses a common data model and format called *Network Data Model and Format*. This format adds to the existing data model features like mobility, discrete events and necessary network devices. Yang et al. in [18] propose SNAMP, a multi-view and multi-sniffer visualization platform for wireless sensor networks (WSNs). The data emitted by individual sensor nodes is collected by a multi-sniffer data collection network and passed to a multi-view visualization mechanism. It has the ability to indicate network topology, sensing data, network performance, and allows developers adding application specific visualization functions with the purpose to facilitate the research and development of various WSNs. Bi in [19] proposes an integrated visualization system for debugging, monitoring and control of WSNs. It is developed in Java using the Eclipse platform with the ability to integrate plugins developed in other programming languages. The visualization is displayed in a user-friendly GUI which displays the acquired real-time data of WSNs. Spanakis et al. [20] present a graphical-oriented real-time visualization tool for vehicular ad-hoc network (VANET) connectivity graphs called *VIVAGr*. The *VIVAGr* takes into account a variety of parameters that affect the shape and characteristics of a VANET, such as wireless range, mobility models, road-network topology, market penetration radio and exhibited interference. It is a portable, multi-platform and modular tool including various independent modules that could be easily modified according to specific needs and requirements. Barberis et al. in [21] propose *ELVS*, an open-source toolbox for simulation and analysis of VANETs. *ELVS* provides an extensible GUI that shows the vehicle positions as well as experiment-specific data in map, using symbol, charts and other visualization tools. Zeng et al. [22] present a visual analytics framework able to visualize

and explore mobility-related factors in a public transportation system, using multiple visualization modules. These visualization modules provide an isochrone map view, isotime flow map view and OD-pair journey view. The isotime flow map view is a novel visualization strategy, which linearises a flow map in a parallel isotime layout. This novel strategy allows presenting clear and smooth pathways from the origin to the destination, while visualizing and comparing various mobility-related factors along the routes. The framework also implements a visual strategy called mobility wheel, for examining temporal variation with all the contributing mobility information.

Since our focus is on visualization in wireless networks, below are presented the visualization platforms regarding wireless networks which require additional attention.

MANET Viewer III

MANET Viewer III [23] is 3D visualization system for MANETs. It presents an improvement of the *MANET Viewer II* [24] which had an inefficient communication method and inaccurate synchronization among nodes. Also, the visualization in *MANET Viewer II* only supported 2D visualization, opposite to the *MANET Viewer III* which supports 3D visualization. *MANET Viewer III* has the ability to visualize the network topology, packet flows and various information by using laptops Wireless Local Area Network (WLAN) cards. The system consists of four components: log generation, log collection, log integration and visualization. The log generation generates logs needed for the visualization. The log collection collects the log data of each node in the MANET to a monitor node. In the log integration part, the collected log data in the monitor node is converted into an animation table and finally visualized in the visualization part using the animation table. Visualization of the network topology uses two modes: manual mode and Global Positioning System (GPS) mode. Figure 2.7 shows the manual mode and Figure 2.8 shows the GPS mode of the *MANET Viewer III*.

The packet flow animation in *MANET Viewer III* is displayed using a sphere for unicast packets and a cube for broadcast packets. Although the *MANET Viewer III* has the ability to visualize the network, it does not offer the ability to interact with the network nodes and trigger network related actions. Also, the platform does not visualize multihoming and would need to be modified to do this, since each WLAN card would be visualized as an independent user. The platform is able to visualize mobility but requires a time lag between the monitor node and the other nodes. This time lag is dependent on the number of hops and increases as the number of hops increase. An advantage of *MANET*

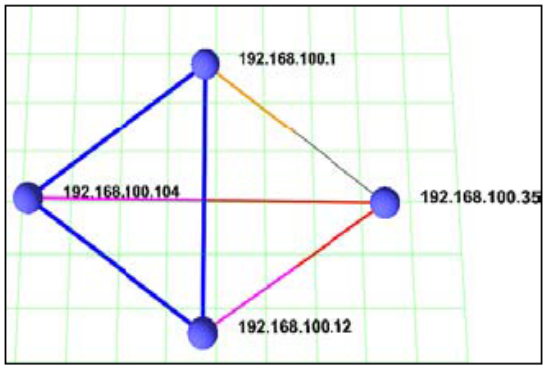


Figure 2.7: Manual mode (3D) [23]

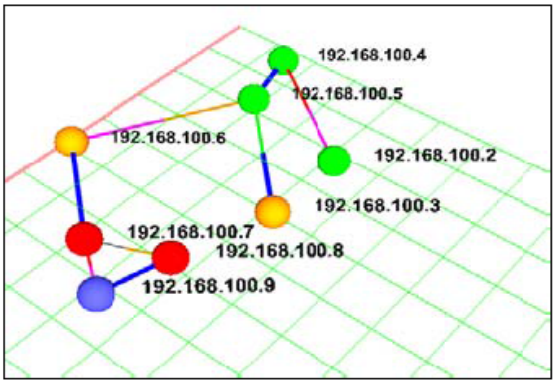


Figure 2.8: GPS mode (3D) [23]

Viewer III is that with the GPS mode, it takes into account the real-time position of the nodes. The traffic animated by the platform is limited to being labelled only as broadcast or as unicast packets. Since the platform is developed to visualize MANETs, each network entity is visualized as a simple node without the ability to clearly differentiate the nodes based on their form. The only ability of *MANET Viewer III* to group nodes with certain attributes is by assigning different colors to them. In [23] test were performed, but only for a small number of nodes, and they do not show how the *MANET Viewer III* operates in situations when there is a large number of nodes in the network, or when the network topology changes rapidly.

MoViTo

Mobility Visualization Tool (MoViTo) [25] is a generic visualization tool developed mainly to facilitate the observation and understanding of mobility patterns in wireless networks. It is composed of generic modules and components which can be adapted to visualize different activities: users' mobility, ad-hoc network topology, exchanged messages, signal quality in wireless systems and others. The system consists of three main modules. The first one is the collector component which extracts the location of the access points and devices, as well as the trace generated or the signal strength. This information is saved in two files: *Network topology* and *Trace*. The second module prepares the data for visualization and manages the history of the network topology and the trace changes. The third component is the GUI component which shows the mobility graph and time-space statistics. In [25], MoViTo was implemented in the context of users' mobility for

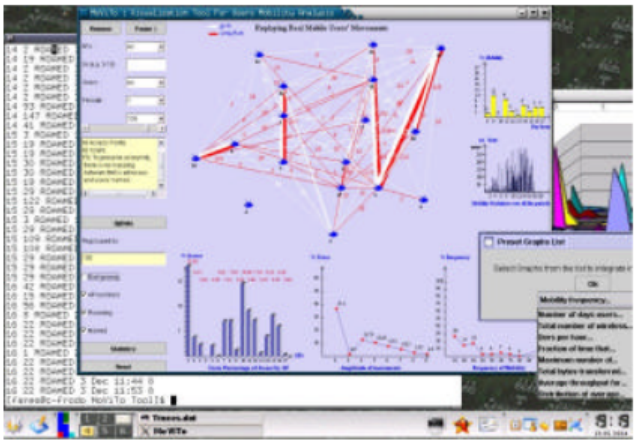


Figure 2.9: Mobility Visualization Tool (MoViTo)

the purpose of studying users' mobility patterns. In this context the nodes represent the Wi-Fi Access Points (APs) with their real coordinates, and the links represent the real users' movement which provides a dynamic vision of mobility. The links are assigned two characteristic values: color which represents the move from and to AP, and thickness which represents the frequency of movement. Figure 2.9 shows the visualization of users' mobility using MoViTo.

MoViTo presents an interesting tool for mobility analysis. The main advantage of MoViTo is the ability to be applied to several contexts related to users' mobility. The display presents mobility of the nodes in 3D environment using node coordinates, as well as providing time-space statistics in the form of graphs. Using the GUI component the context of the study is specified, including the definition of the geographical area, the period of the analysis and the grouping of the nodes. Apart from the users' mobility, MoViTo does not present additional network features, such as multihoming. The nodes use *shell* scripts which communicate with a *syslog* server that collects the events gathered by the Wi-Fi APs. Figure 2.9 shows the ability of the MoViTo to visualize users' mobility in situation with a large number of users and APs. MoViTo presents an interesting network tool designed for analysis of wireless mobility which can be easily adapted to several contexts of mobility.

1.5D Dynamic Network Visualization

In [26], the authors propose an interesting visualization design called *1.5D dynamic network visualization*, which is based on the egocentric data reduction of the dynamic network.

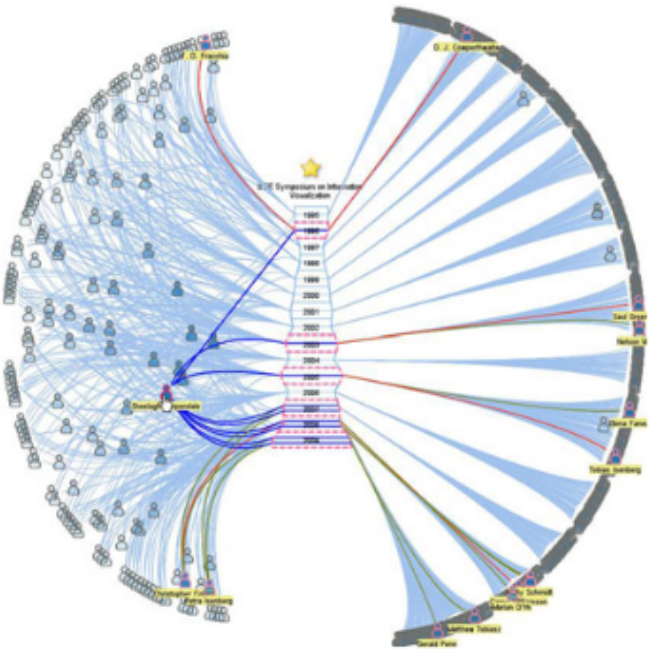


Figure 2.10: 1.5D Dynamic Network Visualization [26]

This design is well suited for dynamic networks that exhibit time-varying relationships as well as node and edge attributes that change over time. Figure 2.10 shows an example of the *1.5D dynamic network visualization* [26]. As shown, the key visual metaphors are the temporal trend glyph in the center which replaces the representation of the focus node, and the glyph's affiliated multiple edges carrying temporal information. Similarity to the node-link graph is related to all non-focus nodes and the edges among them. This trade-off results that the visualization inherits the intuitiveness of a graph representation, while accommodating both topological and temporal information in 2D view space.

General network data needs to be transformed into a format suitable for *1.5D dynamic network visualization*. The raw dynamic network data is represented by a time-varying graph, spanning a specific time period. The graph consists of a set of nodes and a set of edges. Each node is associated with a time set which defines the active time period of the node (edge). The time set can be composed of multiple time intervals for continuous dynamic networks or multiple time points for discrete dynamic networks. Based on this network data, the egocentric dynamic network central to the focus node is defined by a discrete sub-graph of the original graph. This visualization allows grouping of events by inserting a node representing the group of events as the central trend glyph.

The time dimension is encoded into one dimension of the view space, along the trend

glyph, but no strict layout mapping on non-focus nodes is imposed. This means that 1.5D freedom is provided for a visually aesthetic network layout, therefore, the name of the approach. To calculate and accomplish *1.5D dynamic network visualization* is non trivial, and requires an optimized algorithm to calculate the layout for the egocentric dynamic network. This visualization targets a subset of dynamic network analysis tasks, that take one network node as the focus, and requires looking at only the dynamic network central to the focus node [26]. This network is refereed to as egocentric dynamic network.

There are some major challenges in applying the *1.5D dynamic network visualization*. First challenge is related to the shape of the trend glyph in the central, which is non-trivial and can lead to overlapping of the nodes. Since the network can have thousands of entities, the algorithm that calculates the layout can be quite slow and the final drawing can be too cluttered to be understood. This is especially highlighted when the *1.5D dynamic network visualization* has too many edges crossing the central trend glyph. In the 1.5D design, the time independent edges will sometimes pass through the central trend glyph. The interaction is limited to only a few customized interactions which include time navigation, which visualizes the graph in predetermined time slots, rather in the moment when the network changes occur.

2.2.4 Considerations on Visualization Platforms

Existing visualization platforms provide advanced features, but their abilities are mostly constrained to only a subset of predefined and custom features. This makes their use restricted to only certain scenarios. The platform that meets the objectives stated in this Dissertation needs to be more flexible than that. For this purpose, instead of using the existing visualization platforms, a software library is used to develop the visualization and interaction platform.

Choosing a software library instead of a existing platform allows for a great amount of flexibility. In addition, it provides the ability to develop customized and specific functionalities corresponding to the need of the objective stated in this Dissertation. Although choosing a software library allows the development of customized features, it also has certain limitations. The time required to develop the platform in a software library is greater than the time required to make modifications to an existing platform. Furthermore, developing a platform using a software library requires previous knowledge and background in the programming language in which the platform is developed.

For the development of the platform, the software library chosen is JUNG. JUNG is

written in Java programming language and provides the ability to develop a customized and highly flexible platform. Other libraries stated in this chapter do not provide the extensive amount of capabilities as JUNG, and therefore, do not meet the requirements stated in the objectives of this Dissertation.

2.3 Mobility

In the current Internet [27], a session to an end-user device is uniquely identified by a 4-tuple. This 4-tuple consists of the IP addresses and TCP/UDP ports of the source and destination network nodes. The source represents the network node who originates the traffic data, and the destination to the network node for which that data is intended. When the end-user device changes its PoA to the Internet, it gets a new IP address and the previous session is broken. In this situation, no consistent and portable identity is attached to the end-user device except the IP address. Traditional routing mechanisms [28] rely on the assumption that the end user device will not change its PoA to the Internet. When that device connects to a new PoA to receive data, it needs to obtain a new IP address and appropriate network mask. The nodes need to communicate despite the node changing its IP address every time it connects to a new PoA. Users can frequently change their location and change their PoA to the Internet. If previous sessions are broken during this process, the QoS is decreased. It is important to provide the ability for the users when changing their PoA to the Internet to retain previous sessions. Without support for mobility data destined to the end-user device, it is not able to reach that device when it changes its PoA to the Internet. Current mobility protocols make this possible, however this situation does not corresponds to providing mobility for the whole network and its users. Network mobility provides the ability to support mobility of an entire network between different PoA. In this section an overview will be made of the relevant mobility protocols that provide user and network mobility support. This overview will include the explanation of the terminology used and the description of the operation method for each mobility protocol presented in this section.

2.3.1 MIPv6

The first protocol introduced is the Mobile Internet Protocol version 6 (MIPv6) proposed by IETF. The MIPv6 allows network nodes to remain reachable while changing their PoA to the Internet. It is suitable for mobility in both homogeneous media, as well as in

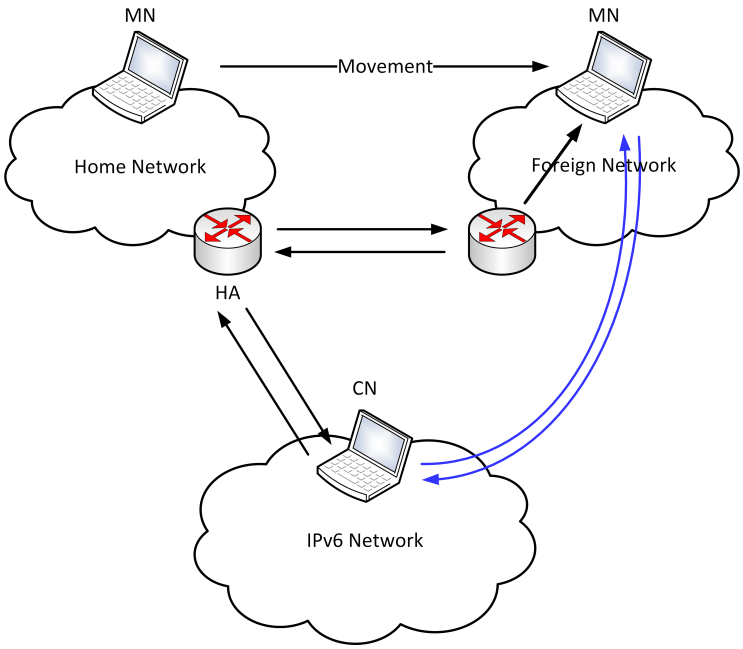


Figure 2.11: Operation method of the MIPv6

heterogeneous media. MIPv6 allows the network node to move from one PoA to another without changing the *home address* [29] of that node. Packets can be sent to the node using the *home address* regardless of the current PoA to which that node is connected. This mobility protocol allows the network node after changing its PoA to communicate with other mobile or stationary network nodes. MIPv6 was developed based on the experience gained from development of mobility support in IPv4 [30] and because of the limitations present in MIPv4 [31] protocol. Primary limitation of the MIPv4 was the insufficient number of addresses which is highlighted nowadays with the ever-increasing number of end-user devices. MIPv6 uses IPv6 which solves the problem of the short number of available addresses while providing better QoS than MIPv4. Figure 2.11 shows the operation method of MIPv6.

MIPv6 uses a specific terminology [29] which is explained as follows:

- **Mobile Node (MN):** network node capable of changing its PoA from one link to another.

- **Home address (HoA):** IPv6 address assigned to MN and used as permanent address of that node. One MN can have multiple home addresses, for instance when there are multiple home prefixes on the home link.

- **Correspondent Node (CN):** mobile or stationary network node that is communicating with the MN.
- **Care-of Address (CoA):** IP address associated to the MN at its current PoA.
- **Home Agent (HA):** router on the MNs home link which registered the MN with its current CoA.
- **Binding:** association of the HoA of a MN with its CoA for a certain period of time.
- **Binding Update (BU):** used by the MN to notify a CN or HA of the MN of its current address.
- **Binding Acknowledgement (BA):** used to confirm the receipt of a BU, if the confirmation was requested in the BU.
- **Binding Cache (BC):** cache of bindings for other MNs maintained by HAs and CNs. Each entry contains information about the MN which includes its: HA, CoA and other related information [29].
- **Router Solicitation (RS):** used by the host and sent to local routers in order to obtain information about local routing [32].
- **Router Advertisement (RA):** sent by the router as response to the received RS.
- **Home Network (HN):** network to which the MN belongs. This network is associated with the network link of the HA.
- **Foreign Network (FN):** current network in which the MN is located and which is different from the HN.

The MN can freely move between access points and change its PoA to the network. The operation method of MIPv6 is based on the following mechanisms:

- **Discovery:** the routers of the FN send RA messages to which the MN responds with the RS message.
- **Registration:** After the MN obtains a new CoA, it informs its HA by sending a BU. The information from the BU is stored in the BC of the HA, and used to forward packets intended for the MN.

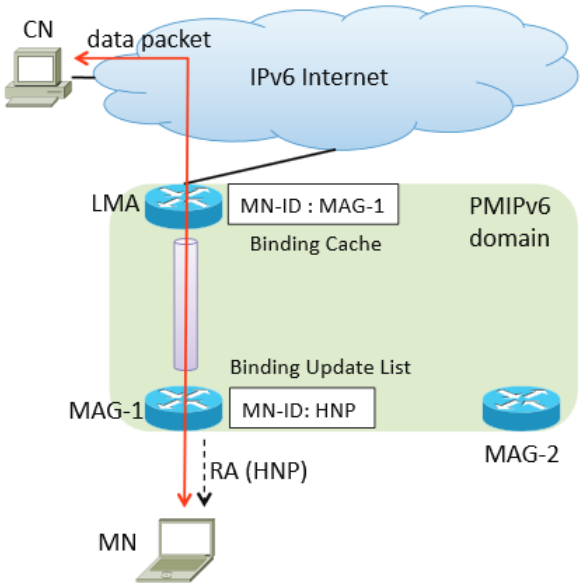


Figure 2.12: Operation method of the PMIPv6 [33]

- **Tunneling:** The HA responds to the BU, by sending a BA to the MN, and creating a tunnel to the corresponding CoA.

The MN when connecting to a FN, informs its HA. The HA intercepts the packets, for the MN and tunnels them towards the MN. The MIPv6 allows end-user mobility, but does not provide mobility of the entire network. Also, MIPv6 requires the interaction by all MNs, which for large networks could present a problem.

2.3.2 PMIPv6

Proxy MIPv6 (PMIPv6) is a network-based localized mobility management protocol standardized by the IETF. PMIPv6 provides host mobility in an area called the PMIPv6 domain. In PMIPv6 the location of the MN is managed by the network, without involving the MN in the mobility signalling [33].

PMIPv6 uses a specific terminology [34] which is explained as follows [32]:

- **Local Mobility Anchor (LMA):** the central entity through which the entire traffic is routed. It maintains a set of routes for each MN connected to the network.

- **Local Mobility Domain (LMD):** the network that is PMIP-enabled. In contains one LMA and multiple MAGs.
- **Mobile Access Gateway (MAG):** performs the mobility related signalling for the MN. It is usually the access router (first hop router) for the MN.
- **Binding Cache (BC):** a cache maintained by the LMA that contains the BCE entries.
- **Binding Cache Entry (BCE):** an entry in the LMA's BC. Every BCE entry contains the fields MN-ID, MAG proxy-CoA and the MN-prefix.
- **Proxy Binding Update (PBU):** a signalling packet sent to the LMA from the MAG, which contains information about a new MN. Every PBU contains the fields MN-ID, MAG address (proxy-CoA) and handoff indicator to indicate if the MN-attachment is a new one or handoff from another MAG.
- **Proxy Binding Acknowledgement (PBA):** packet sent to the MAG as response to a PBU sent by the LMA. The PBA contains the MN-ID, the MAG address and the prefix assigned to the MN.
- **Proxy care-of address (proxy-CoA):** IP address of the public interface of the MAG. The proxy-CoA is the tunnel endpoint address on the MAG.
- **Mobile Node Identifier(MN-ID):** identifier which uniquely differentiates the MN.
- **Home Network Prefix (MN-HNP):** prefix that is assigned to the MN by the LMA.

Figure 2.12 [33] shows the operation method of PMIPv6. The handover procedure in PMIPv6 is performed in the following way [33]:

- When the MN attaches to one MAG, the MAG detects the attachment and sends the PBU to the LMA. When the LMA receives the PBU, it assigns a MN-HNP to the MN and creates a BCE that binds the MN-HNP to the proxy-CoA. Then the LMA sends the PBA including the MN-HNP to the MAG.
- When the MAG receives the PBA, it sets up a tunnel to the LMA and adds a default route over the tunnel to the LMA. Then the MAG sends the RA containing the MN-HNP prefix, to the MN.

- When the MN receives the RA, it configures its IP address. Once the IP address is configured, the MN is ready to receive and send packets.

- When the data packets destined to the MN first reach the LMA, the LMA checks its internal cache in order to obtain the position of the MN. Then, the LMA sends the packets to the corresponding MAG, which removes the outer header and forwards the packets to the MN.

- When the MN sends packets to the CN, the packets are first sent to the corresponding MAG. When the MAG receives the packets, it tunnels the packets to the LMA. Then, the LMA removes the outer header and routes the packets to the CN.

- Should the MN leave the network, the MAG detects that the MN has moved away from its access link. Then the MAG sends a de-registration PBU to the LMA, with the lifetime value set to zero. When the LMA receives the PBU with the lifetime value set to zero, it sends the PBA to the MAG and deletes the MN's BCE.

PMIPv6 provides stable handovers, since the signalling messages do not traverse the wireless link between the MN and the access router. It also provides host mobility without any modifications on the MN, allowing any end-user device to connect to the network. However, PMIPv6 is a host mobility protocol and does not provide network mobility. Also, it only allows the end-user devices to connect to fixed access points. The PMIPv6 is the protocol that is implemented into an existing framework that is used as a starting point in this Dissertation.

2.3.3 N-PMIPv6

N-PMIPv6 was developed as an extension of the PMIPv6 in order to support network mobility. In addition to the LMA and MAG, the N-PMIPv6 introduces the mobile MAG (mMAG). The mMAG presents a MAG that is not fixed, but can move freely through the network. The MN connected to the mMAG is refered to as Mobile Network Node (MNN). The mMAG is responsible for detecting the MNN's movement, performing mobility management operations on behalf of MNNs, and managing binding update list as a MAG does [35].

Figure 2.13 [33] shows the operation method of N-PMIPv6. The registration and handover procedure in N-PMIPv6 are performed in the following way [33] [35]:

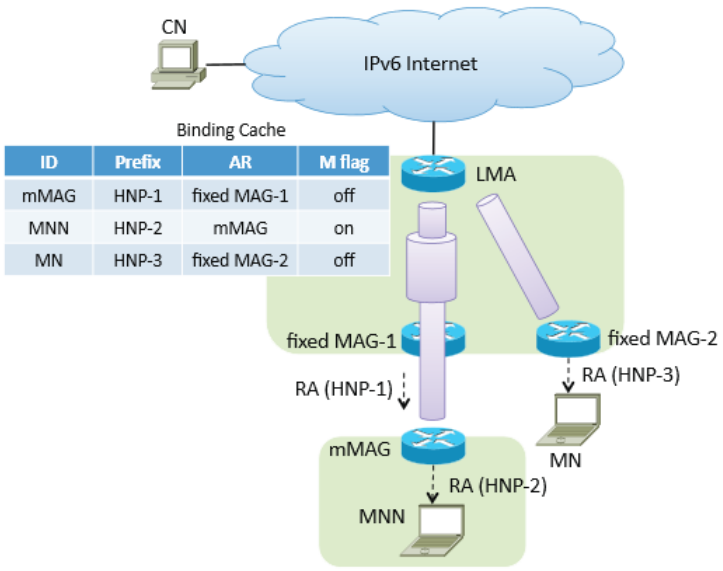


Figure 2.13: Operation method of the N-PMIPv6 [33]

- In the situation when a mMAG with a MNN attaches to the fixed MAG, the MAG sends the PBU containing the mMAG-ID to the LMA.
- When the LMA receives the PBU, it creates the BCE and assigns the HNP to the mMAG. Then, the LMA sends the PBA to the MAG.
- When the MAG receives the PBA, it sends the RA containing the HNP to the mMAG.
- When the mMAG receives the RA, it sends the PBU containing the MNN-ID to the LMA.
- When the LMA receives the PBU, it assigns the HNP prefix to the MNN and creates the BCE. N-PMIPv6 adds a new field, the *M flag*, to the BCE. The *M flag* is set to indicate that the MNN is connected to the network. Then, the LMA sends the PBA containing the HNP prefix to the mMAG
- When the mMAG receives the PBA, it sends the RA containing the HNP to the MNN.
- When the data packets destined to the MNN first reach the LMA, the LMA finds the MNNs BCE. Since the *M flag* is set, the LMA searches for the mMAG BCE. Then,

the LMA encapsulates the packet for tunneling to the mMAG and encapsulates it again for tunneling to the MAG. The packet is sent from the LMA to the MAG. The MAG decapsulates the packet and forwards it to the mMAG. Then, the mMAG retrieves the original packet and forwards it to the MNN.

- When the mMAG moves to another fixed MAG, the procedure executed is the same as in the initial registration. In this process, only the AR fields in the BCE are updated, while the ones of the mMAG BCE and MNN BCE remain unchanged.

N-PMIPv6 supports both host and network mobility. However, N-PMIPv6 has large tunneling overhead because it uses multiple encapsulations. Also, information about every entity needs to be kept on the LMA, which cause the local communication to have very large overhead.

2.3.4 Considerations on Mobility

This Dissertation will develop a platform visualization for wireless access networks that support mobility, with both PMIPv6 in single-hop wireless networks, and N-PMIPv6 in multi-hop wireless networks (vehicular networks). Both these approaches will then be used to get the mobility information required for the developed platform.

2.4 Multihoming

In a classical network, the end user device is connected to the network only with one network interface, which corresponds to the situation most users are familiar with. By adopting multihoming (MH), devices equipped with more than one network interface can be simultaneously connected to multiple access networks. Multihoming can be either site or host based, depending if the network end-node is either a host or a site. Site-multihoming [36] is the process of an end-site, such as an enterprise network or campus network, to be simultaneously connected to multiple Internet service providers. The focus of this Dissertation is on host based multihoming.

Nowadays, it is becoming a norm to have multi-interface enabled devices, also known as multi-homed devices [37] [38]. When a end-user device that is equipped with multiple network interfaces is connected to the network, this device is referred to as a host and if it is using multihoming, the correct term is host-multihoming. A set of advantages can be achieved when adopting multihoming in the end-user devices and networks. Advantages of

multihoming are increased connection reliability, available resources and QoS. One of the advantages is better resource management, which as a result, decreases the economic cost. In order to successfully apply host-multihoming, the percentage of traffic sent through each network interface is an important parameter. In order to fully take advantage of multihoming and improve network performance, the percentage of traffic sent through each interface should be optimal. Host-multihoming is especially useful in vertical handovers, when the connection migrates from one access point or technology to another. Let us consider that a user is connected to a Wi-Fi network and is downloading a file to his laptop or making a VoIP telephone call. If the user is moving, without overlapping Wi-Fi networks, the download will be interrupted or the call terminated even after the device travels a short distance. Although Wi-Fi offers mobility with high data rates at low cost, cellular technologies can keep downloads and calls from being interrupted at lower data rates but with a significantly higher cost.

2.4.1 SCTP

One of the first protocols to have multihoming capabilities is the Stream Control Transmission Protocol (SCTP). SCTP is a transport layer protocol developed by the IETF with multihoming incorporated into its design. It enabled two hosts to establish a logical connection over more than one network interface and uniquely identify the connection by separating IP addresses. Although SCTP has multihoming ability, it uses it only when the IP address to which it is sending data, becomes unreachable. This can be seen as a backup service in which the SCTP will try sending data to a secondary IP address. Since SCTP can employ retransmission in the scenario when an IP address becomes unreachable, it will try sending the data to a secondary IP address and the retransmited data will be sent to an alternative location. The use of SCTP is limited by the issues and problems regarding handover management which concerns mobility, concurrent multipath transfer and cross-layer activities [39].

2.4.2 Shim6

In [40] the authors describe a multihoming solution for IPv6 that relies on a sublayer inside the IP layer, called Shim6 sub-layer. The Shim6 protocol is standardized by the IETF and provides multihoming support to end-hosts. In order to do so, a Shim6 context is established for each pairs of communicating hosts. After a Shim6 context has

been established, the protocol exchanges the locators associated with a pair of *upper-layer identifiers*. At any given time, a pair of source and destination locators is used for one direction, and a possibly different pair of locators is used for the other one. In addition to providing host-multihoming, Shim6 also includes some interesting features such as context forking and context recovery.

2.4.3 Proxy-based Multihoming

The idea of proxy-based multihoming is placing a proxy-server in the network that is aware of the multiple network interface, with which the end-user device is connected to the network [37] [38]. This approach aims to minimize the traffic sent through the network and increase the overall network performance. Since only the proxy-server is required, adopting proxy-based multihoming minimizes the adoption cost.

In this approach a proxy server is placed in the network that is aware of the multiple network interfaces with which the end-user device is connected to the network. This technique relies on accurately estimating the capacity of each connected interface, and sending traffic to the end-user device accordingly to this estimation. Deploying a proxy server minimizes the amount of traffic generated by the updates sent from the communication server, therefore reducing the amount of information sent through the network. The advantages of using proxy-based multihoming are that it allows end-user devices to use scheduling techniques as well as efficient solutions to estimate the characteristics of the connected interfaces. However, in proxy-based multihoming, multiple end-user devices could be contending for the same proxy server. In order to prevent the proxy server becoming a bottleneck [38], it is important to optimally place the proxy server.

The framework used as a starting point in this Dissertation is based on this approach. However, the framework that is used has several additional functionalities that are explained in the next chapter.

2.4.4 Considerations on Multihoming

This Dissertation will develop a platform visualization for wireless access networks that support multihoming based on both PMIPv6 extensions in single-hop wireless networks, and N-PMIPv6 extensions in multi-hop wireless networks (vehicular networks), both complemented by the proxy-based multihoming. Both these approaches will then be used to get the multihoming information required for the developed platform.

2.5 Chapter Considerations

The current chapter emphasized the reasons to visualize data, and provided an overview of the relevant visualization frameworks and software tools. In this chapter, several current visualization platforms were introduced and presented together with a description of their functionalities.

With the importance that visualization has in everyday life, it is not surprising that it is being used more and more in the academic community. There are several studies that present visualization platforms used to display mobility and other network mechanisms. However, most of these platforms focus only on visualizing and rarely provide abilities to interact with the network. To the best of our knowledge, a platform that is able to visualize both mobility and multihoming, and trigger network related actions does not yet exist. While developing such a platform, it was important to get a clear understanding of the main concepts related to both mobility and multihoming.

The existing mobility protocols provide the end-users the ability to perform seamless handovers while moving between different access networks. This feature has made mobility a key factor in todays communications. Nowadays, most of the end-user devices are equipped with multiple network interfaces. Multihoming allows those devices to be simultaneously connected to multiple access networks. Adopting host based multihoming provides a number of advantages for those devices, such as allowing for better resource management and network utilization. Implementing both mobility and multihoming in the network provides a number of advantages, but makes the situation in the network more difficult to comprehend. For this purpose, a visualization platform can be used to visualize the implemented mechanisms in a intuitive way.

After the main concepts related to mobility and multihoming were explained in this chapter, the next chapter will introduce the visualization and interaction framework that is integrated in a mobility protocol with multihoming support.

Chapter 3

Visualization and Interaction Framework

3.1 Introduction

Mobility has proven to be a key factor in current communications, since the users are accustomed of moving freely while still being connected to the network and having Internet access. Since in networks where mobility is a characteristic, the users can frequently change their location, which can cause the network topology to become highly dynamic and extremely unstable. Adding multihoming features to the network increases the complexity of the network representation, since it adds more information to the network entities that need to be presented. In order to present these concepts in an intuitive way, a platform is needed that is able to visualize the network topology in real-time together with the relevant information. In this Dissertation the aim is to develop a platform capable of visualizing and also triggering network actions related to multihoming and mobility. For this purpose, along with the platform, a visualization and interaction framework is proposed and developed in this Dissertation. This chapter introduces the developed framework and provides an overview of its functionalities. The visualization and interaction framework is integrated in a mobility protocol with multihoming support. Exchanging relevant information between the platform and the framework is crucial in visualizing network data and triggering network changes. This is not a trivial process, since the developed framework should trigger network actions in a transparent way, and network data visualized should correspond to the actual situation in the network.

This chapter is organized in the following way. Section 3.2 describes the several sce-

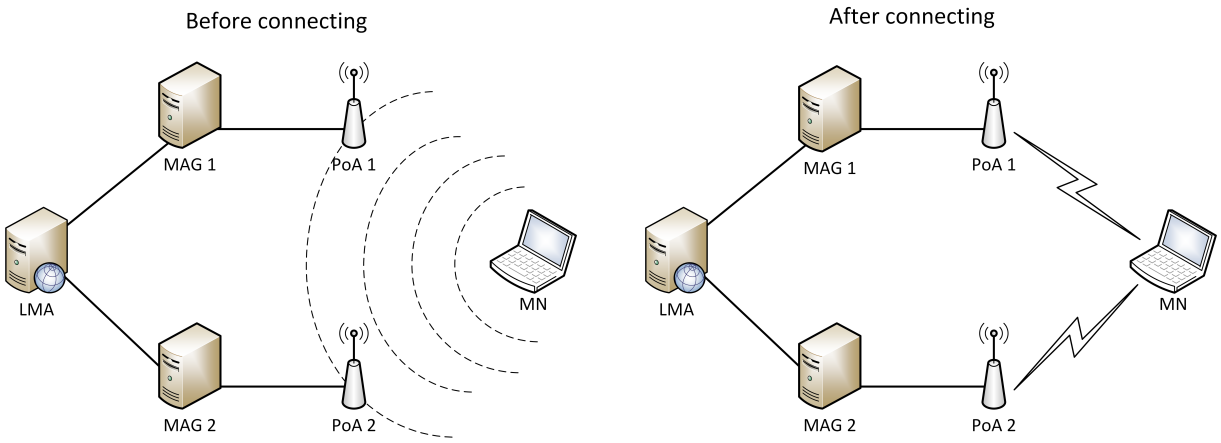


Figure 3.1: MN connecting to the network

narios related to mobility and multihoming that need to be taken into account. Section 3.3 describes the multihoming framework used as a starting point in this Dissertation. Section 3.4 describes the modifications done to the multihoming framework described in Section 3.3, in order to provide visualization and interaction capabilities. This section, also introduces the entities that are developed and integrated in the multihoming framework. Section 3.5 summarizes this chapter and provides chapter considerations.

3.2 Mobility and Multihoming Scenarios

This section describes the relevant scenarios related to both mobility and multihoming. It is important to analyse these scenarios to get a clear understanding on how they affect the network.

The first scenario presented corresponds to the basic scenario when a new MN enters the network and tries to connect simultaneously with multiple network interfaces. This implies that the MN is equipped with multiple network interfaces. Figure 3.1 depicts the described scenario. When the MN tries to connect to the network, the MAG for the corresponding Point-of-Attachment (PoA) to which the MN is trying to connect will send a PBU to the LMA. The LMA, after receiving the PBU, will verify if the MN has permission to connect to the network. If the user has permission, the LMA will create a BCE entry containing information about the user and the connection. After this process, a PBA is sent to the MAG that originated the PBU and the connection will be established. The depicted scenario does not take into account the situation when the MN tries to connect

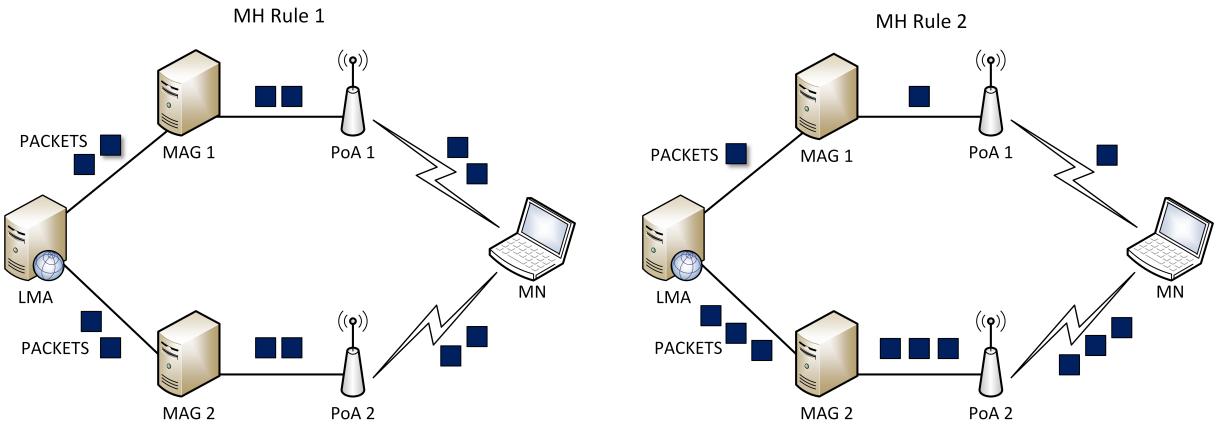


Figure 3.2: Optimization of data sent to a MN

to a mMAG. In this case the same rule applies as in the situation when the MN tries to connect to a MAG. When detecting that a MN is trying to connect, the mMAG sends the PBU to the LMA. The LMA receives the PBU, creates the BCE entry and returns the PBA to the mMAG. Upon receiving the PBA, the mMAG sends the RA to the MN and the connection is established. Since, the users are constantly moving and are in range of new access networks, the scenario of a new MN entering the network can occur frequently.

The next scenario describes the situation when data intended for the user is sent to a MN, that is connected to the network with more than one network interface. In order to take full advantage of multihoming, it is important to properly manage the percentage of user data sent through each connected interface. The percentage for each connected interface should be determined taking into account real-time information about the terminal, connection and network state [37]. The conditions in wireless networks are variable and can cause the percentage of the user data sent through each interface to change. When the LMA detects data being sent to the MN, it determines the percentage of data that should be sent through each connected interface. In the case when one of the parameters needed to determine the multihoming rule changes, the LMA will redetermine the percentage that needs to be sent through each connected interface. Figure 3.2 depicts the described scenario. Initially, we can see that the percentage of data transmitted to the MN, through each connected interface is evenly matched. When the conditions in the network change, the LMA determines that more data should be routed through PoA2 and less traffic through PoA1. This could correspond to the situation, when multiple MNs connect to the PoA1 increasing its load. This will cause the LMA to determine the new percentages of user data that need to be sent through each connected interface. In the case when the MN is connected to a

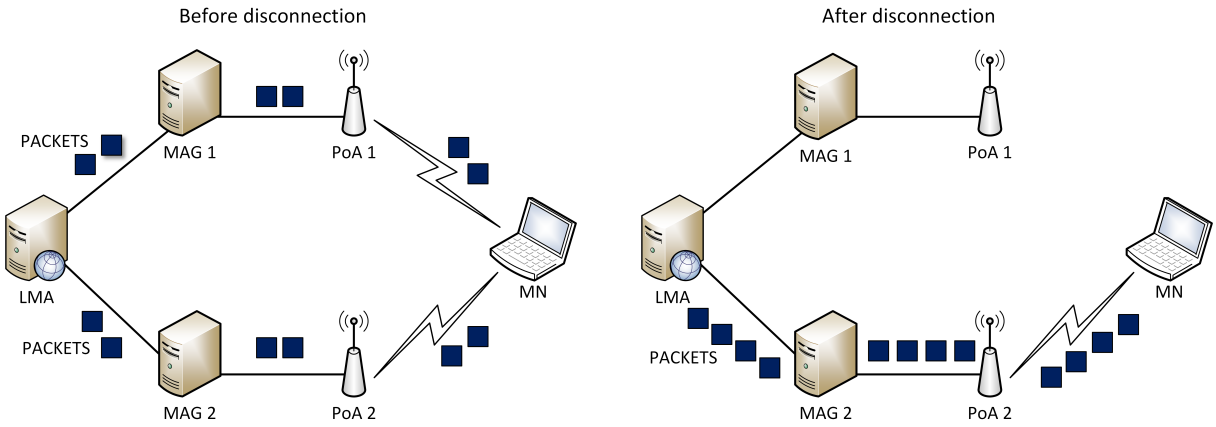


Figure 3.3: Redirecting the data sent to a MN

mMAG, the process is the same as if the MN is connected to a MAG. The conditions in wireless networks can vary and change frequently. This means that the percentage of user data sent through each connected interface, should be optimal in real-time.

In the previous scenario, the percentage of user data routed through one PoA was decreased due to the increased load of that PoA. In the following scenario depicted in Figure 3.3, the situation is analysed when a network interface of a MN disconnects while data is still being sent. This could be caused by either hardware malfunction of the MN simply moving out of the range of the PoA. In this situation, when the network interface disconnects, the LMA will detect the disconnection and determine the percentage of user data that needs to be sent through the remaining interfaces, that are still connected. If, after the interface disconnected, the MN remains connected with only one network interface, then all user data will be routed to that interface.

The last scenario described in this section corresponds to the situation when the MN leaves the network, or disconnects all interfaces from the network. This scenario is depicted in Figure 3.4. As we can see the MN is connected to the network with two network interfaces. When one network interface disconnects, the LMA detects the interface that disconnected. After verifying that the MN is still connected to the network with at least one network interface, the LMA will simply update the BCE entry for that MN, and delete information about the interface that disconnected. In the situation when all network interfaces are disconnected, the LMA will after verifying that the MN has no remaining interfaces connected, delete the BCE entry for that MN. Since, the users are constantly moving between different access networks, this process will occur when the MN leaves the network, or simply moves out of the range of the PoA.

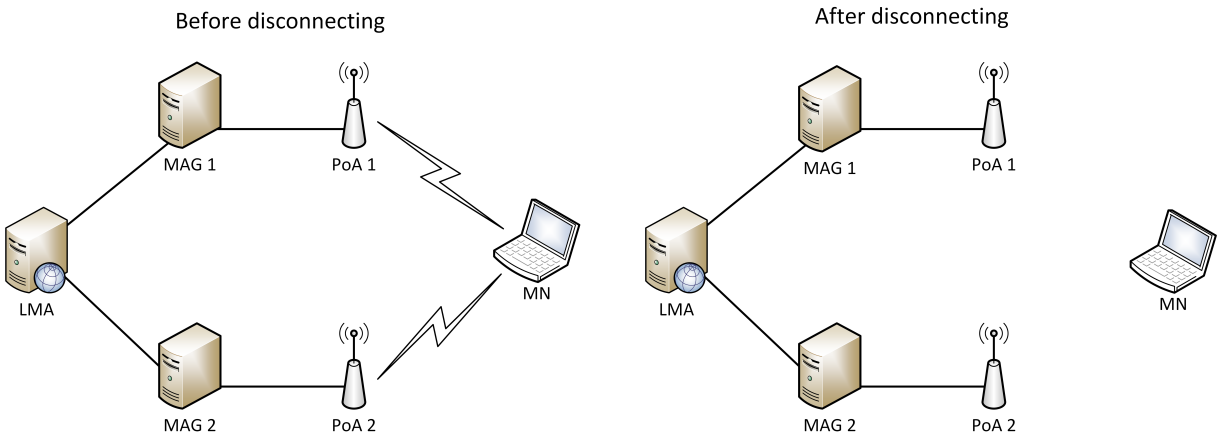


Figure 3.4: MN leaving the network

3.3 Multihoming and Mobility Framework

In this section we introduce the multihoming framework used as a starting point in this Dissertation. This framework proposed by [37] is based on the mobility protocol PMIPv6 [34] and has been modified to provide multihoming support. The framework optimizes the use and performance of available access networks, and its approach is based on an algorithm which is able to determine the best traffic allocated through different available network interfaces.

In order to provide multihoming ability, despite the limitations of the PMIPv6 protocol, a set of entities is proposed and developed [37]. These entities provide multihoming functionalities while taking into account changes and variations of environment characteristics. The algorithm that determines the multihoming rule is provided with all the necessary information by the entities of the framework. Then, the algorithm determines the percentage of the traffic that should be allocated through each connected interface. The entities providing multihoming abilities are also responsible for the management of all network interfaces of the end-user terminals, the respective data flows and the context information (traffic information, access networks state, wireless characteristics of each network interface and information about the end-user terminal). The architecture and description of the framework is provided below in order to get a clear understanding on how these entities provide multihoming abilities and communicate with each other.

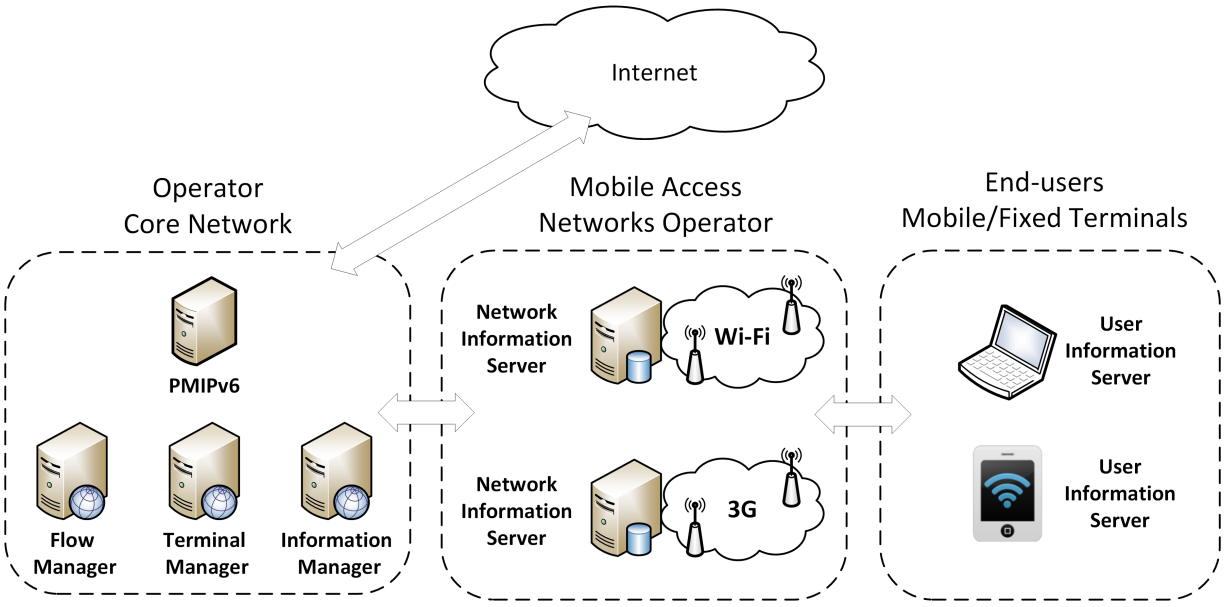


Figure 3.5: General multihoming architecture [37]

3.3.1 Architecture

In [37], the authors developed a multihoming architecture able to take advantage of heterogeneous access networks and optimize their performance. Figure 3.5 [37] shows the general architecture that provides multihoming functionalities. It is composed of three main network segments: the *Operator Core Network*, the *Mobile Access Networks Operator* and the *End-Users Mobile/Fixed Terminals*.

The *Operator Core Network* contains mainly the entities that provide multihoming functionalities, and which determine the multihoming rule that is applied. The *Mobile Access Networks Operator* contains the PoAs to which the end-user terminals can connect, and they can be of different access technologies. The *Mobile Access Networks Operator* connects the *Operator Core Network* with the *End-users Mobile/Fixed Terminals*. In the *Mobile Access Networks Operator* segment, the PMIPv6 is also present and running as a MAG. The third segment is the *End-users Mobile/Fixed Terminals*, which provides information about the end-user terminal and the characteristics of its connection with PoAs.

In [37], a base implementation of the mobility protocol [41] was used and extended to provide multihoming functionalities. These improvements consist on implementing the ability to associate each interface to a specific end-user terminal, since PMIPv6 does not

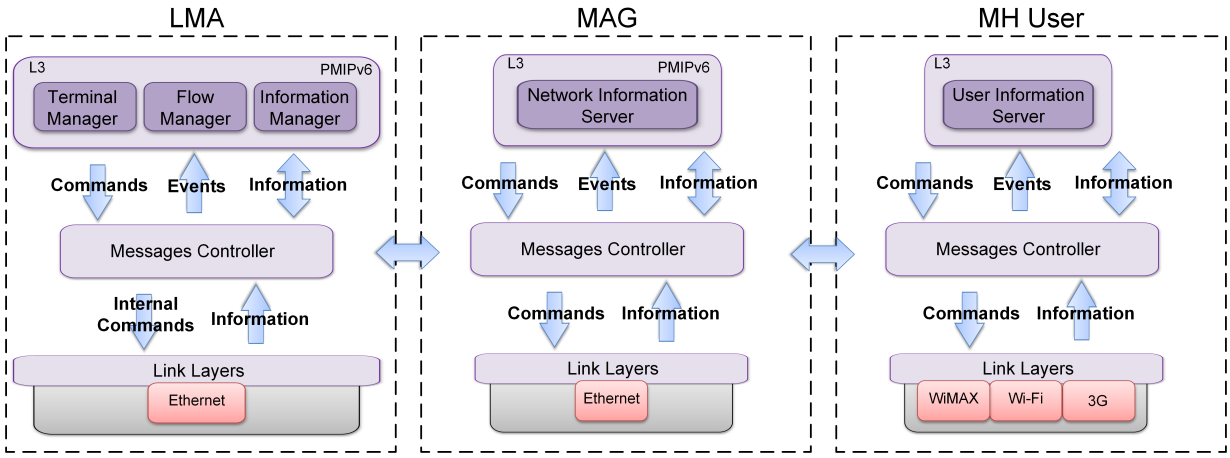


Figure 3.6: Initial multihoming framework [37]

have the ability to associate multiple network interfaces to a single end-user terminal. Furthermore, PMIPv6 does not provide the ability to decide through which route the data traffic will be sent to the end-user terminal. For this purpose, in [37] an intelligent process is implemented to manage through which router traffic data is sent to the end-user terminal. Also, a mechanism is implemented to obtain information about the end-user terminal, the connection, network state and the traffic flow characteristics. For this purpose, a set of entities was developed and implemented. These entities are shown in Figure 3.6 [37] and present the multihoming framework used as a starting point in this Dissertation. The developed entities that provide multihoming functionalities are described below.

3.3.2 Flow Manager

The first entity introduced is the Flow Manager (FM). It is located in the *Operator Core Network*. The FM entity is responsible for defining and applying the multihoming rule. This entity detects when the LMA receives traffic data for a certain end-user terminal, and verifies the destination address of the packet, which identifies the end-user terminal to which the traffic is sent. Then, the FM checks if a Flow Cache Entry (FCE) exists for that end-user terminal. If it does not exist, the FM creates a new FCE entry with the necessary information based on which the multihoming rule is determined. In the case when the end-user device already has an FCE entry, then the FM entity simply updates the information in the corresponding FCE and applies the multihoming rule. The FCE contains all necessary information [37] related to all flows associated to a certain end-user

terminal.

This information is divided into general information about the end-user terminal and information about every flow for that terminal. General information contains: the user identifier, flow identifier, number of existing flows, MH rule, MH rule update time.

In order to distinguish several flows, the FCE also contains information about each flow: the flow prefix, the source address, the target address, the source port, the destination port, the time without receiving any packet, the flow rate, the mean packets size, the number of packets that were received, the transport protocol and the preferred PoA. Based on this information, it is possible to uniquely distinguish several flows corresponding to the same user. Given the information regarding the preferred PoA, the FM entity has the ability to apply flow preferences. In that case, the traffic is sent to the preferred PoA and it is not considered as multihoming traffic. Should the users' end terminal disconnect from the preferred PoA, the FM entity will determine the multihoming rule, and the traffic will be considered as multihoming traffic.

To clarify how the FM entity applies the multihoming rule we consider the following scenario. The end-user terminal is equipped with two network interfaces where each interface is connected to a different PoA. When traffic data is sent to the end-user terminal, the FM checks if a FCE exists for the destination address. If the FCE exists, it is simply updated; if not, a new FCE entry is created for that terminal and the necessary information is stored. Then, the multihoming rule is determined by the FM and applied. Should the terminal for some reason disconnect its network interface from one PoA, the FM will determine and apply the new multihoming rule which will route all the traffic through the other PoA. The described scenario provides a better understanding about the purpose of the FM entity and the way it operates.

3.3.3 Terminal Manager

In order to support multihoming functionalities, a connection identifier is defined that is related to the user and not to the MAC address of the users' end device. This identifier relates the terminal device with the MAC address of each network interface that the terminal is equipped with. For this purpose, the Terminal Manager (TM) entity is introduced. The TM entity is located in the *Operator Core Network*. Using a list that performs the relation between the end-user terminal and the interface MAC address, the TM associates the various connections to a specific terminal. This list that relates the MAC address to the terminal also acts as an access permission list. When an end-user terminal is trying to con-

nect, the TM entity verifies that the user has permission to connect to the network. Should the user have permission to connect, the corresponding MAG for that end-user terminal will send an PBU message to the LMA with the regarding information. After receiving the PBU message from the MAG, the LMA will check if the interface corresponds to an already existing user. If the user already exists, the TM will update the end-user terminal information. If a new user is trying to connect, the LMA will create a new BCE entry containing information regarding the network interface and connection. In this case the TM will add a new entry in the UCE for the end-user terminal, containing the information about the end-user terminal and the interfaces.

After this process, a PBA message is sent to the MAG which originated the PBU message and the connection is established. Should for any reason the network interface disconnect from the PoA, the corresponding MAG will detect the disconnected interface and inform the LMA. When the LMA receives this notification, it will delete the BCE of the disconnected interface. If the terminal is connected to the PoAs with at least one interface, the TM will just update the end-user terminal information. Otherwise, the UCE entry for that end-user terminal entry will be deleted.

3.3.4 Information Manager

The Information Manager (IM) is located in the *Mobile Access Networks Operator* and is responsible for obtaining information about the environment that is needed to determine the multihoming rule. This information contains the characteristics of the PoAs where the end-user device is connected, the mean packet size, the real PoA capacity, and the information about the wireless connection between the PoA and the end-user terminal, such as the available bandwidth, the packet loss and the achieved throughput. In order for the FM to determine the multihoming rule, it is necessary to obtain up to date information about the environment. For this reason, every time the FM needs to determine the multihoming rule, it sends a request to the IM to obtain this information. The IM then informs the NIS about the request from the FM.

3.3.5 Network Information Server

The Network Information Server (NIS) entity is integrated in the *Mobile Access Networks Operator* segment and interacts with both the IM and the UIS entity. The NIS entity is responsible for providing information needed for the FM to determine the multihoming

rule. This information is related to the respective PoA and wireless characteristics of the end-user terminal. In order to obtain information about the actual capacity of the PoA, an estimation process is performed that is based on an extended version of Wbest [42]. This process determines the packet loss, the achieved throughput, the achieved bandwidth and the achieved capacity.

3.3.6 User Information Server

The User Information Server (UIS) is located in the *End-users Mobile/Fixed Terminals* segment of the network. The UIS communicates with the NIS entity and provides it information gathered through the enabled measurement process: interface name, received signal strength (RSS), interface quality level, packet loss, load of the PoA and achieved throughput. The measurement process enabled by the UIS is based on the extended version of Wbest [42], similar to the NIS entity.

3.4 Performed Modifications

The MH framework explained in the previous section provides both mobility and multihoming support. In order to provide visualization and interaction capabilities, a set of entities is proposed and developed in this Dissertation. These entities are able to interact with the visualization platform, and are integrated in the framework described in Section 3.3. The implementation of these entities and the way they operate, need not to interfere with other functionalities of the framework, but rather extend the existing functionalities of the framework. The proposed entities operate independently, but communicate with other entities of the framework in order to exchange information.

In order to integrate functionalities related to scenarios described in Section 3.2, several modifications need to be made to the existing implementation [37]. In this section we describe the modifications done to the framework and the entities that are developed. In order to visualize mobility and multihoming, information about the network is required. The framework used as basis provides mobility and multihoming support, but does not provide information required for visualization and neither does it trigger network related actions. For that reason the framework is modified. Before explaining the operation method of the entities that are developed, we first need to define the required information. The information required for the visualization process is the following:

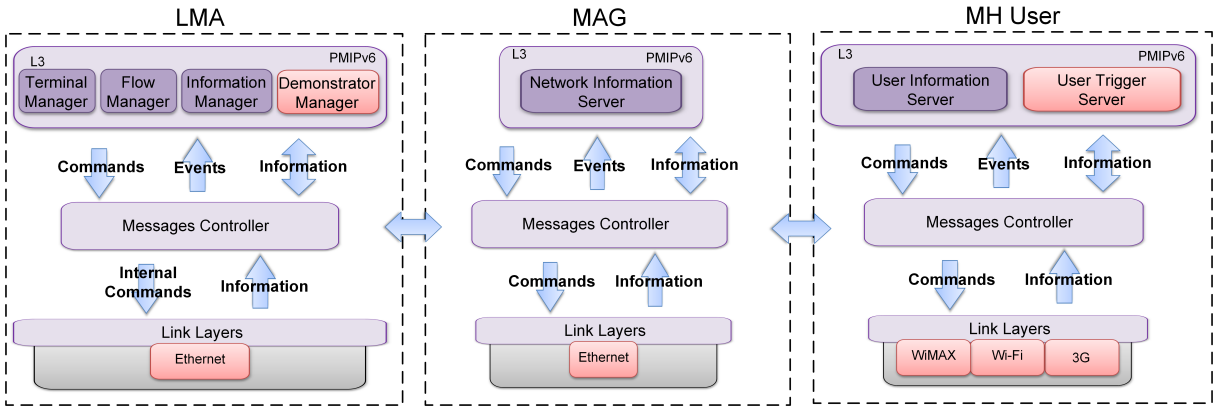


Figure 3.7: Visualization and interaction framework

- Network entities: identifier of every network entity in the network.
- Network topology: connections between the entities in the network
- Traffic flows: information about every traffic flow in the network.

In this Dissertation two additional entities were developed and integrated in the multihoming framework [37]. These entities provide the needed visualization and interaction functionalities to the existing framework. In this section, the developed entities are introduced together with a description of their functionalities.

3.4.1 Demonstrator Manager

The Demonstrator Manager (DM) can be considered as the main entity developed for the visualization and triggering framework. The DM entity is responsible for providing the platform the necessary information about the network nodes, network topology and existing traffic flows. As seen in Figure 3.7 [37], this entity is located on the LMA side. It can communicate with the TM, the FM and the IM entities in order to gather the information required for the visualization. When the information needed for the visualization is requested from the DM entity, the DM provides that information to the platform. The information contains data about the entire network: the identifiers of all network nodes, the connections between those nodes and the information about traffic flows. Whenever there is a change in the network, the DM will detect the change and inform the platform. This change can be either a MN disconnecting or connecting a network interface, or a MN entering or leaving the network. Also, when the LMA detects a traffic flow, the FM entity

will check if there is a FCE entry for the destination address of the packets corresponding to the traffic flow. If it exists, the FM will update the FCE entry, or it will create a new FCE entry and store the necessary information in that FCE. After this process, the FM will send this information to the DM entity which in turn will provide this information to the platform. The DM entity is also involved in the triggering process. Each request to trigger a network related action is first sent to the DM entity. The DM verifies the received request and sends it to the UTS entity of the corresponding network node which will trigger the network action.

3.4.2 User Trigger Server

The other entity developed in this Dissertation and integrated in the visualization and interaction framework is the User Trigger Server (UTS). The UTS entity is responsible for the triggering process, and it is located on the MH user side, which is shown in Figure 3.7. The main responsibility of the UTS entity is to perform physical actions related to the process of triggering network related actions. When a network related action needs to be triggered, the platform sends a request to the DM entity. The DM entity, when receiving the request, determines the specified network node and forwards the request to the UTS entity of that node. After receiving the request, the UTS entity determines the type of network action needed to be triggered and then executes the action corresponding to that event. To provide information about the end-user terminal, the UTS also communicates with the NIS entity.

3.5 Chapter Considerations

The current chapter presented the visualization and interaction framework developed in this Dissertation. The framework is based on an existing framework that provides both mobility and multihoming support.

After a careful study of the multihoming framework used as base, it was necessary to extend it, in order to provide both visualization and interaction abilities. These abilities needed to be integrated into the framework without interfering with any other functionality of the framework. This would not be possible if the additional features were implemented into the existing entities of the framework. Therefore, two separate entities, the DM and the UTS are developed and presented in this chapter. These entities are implemented in

the framework and operate independently, without affecting the other functionalities of the framework used as starting point.

The visualization process requires information about the entire network. This task is the responsibility of the DM, which can communicate with other entities of the framework, in order to obtain the necessary information.

The interaction process requires network related actions to be triggered. For this purpose, actions corresponding to specific events need to be executed on the network nodes. This task is the responsibility of the UTS entity. The UTS entity determines the type of network action that is triggered, and executes the corresponding action on the MN side.

The DM and UTS entities operate independently, but can communicate with each other and other entities of the framework. The DM knows the position of every node in the network, and every request is first sent to the DM, which then forwards this request to the specified network nodes. The DM communicates with the UTS, and sends its requests to trigger network related actions.

Although the developed framework has visualization and interaction abilities, it needs to interact with a platform which specifies the network actions that need to be triggered, which indicates the network nodes that are involved in this process and which displays the visualization. The next chapter presents and describes this visualization and interaction platform.

Chapter 4

Visualization and Interaction Platform

4.1 Introduction

In order to present the advantages of mobility and multihoming, a platform is needed to present these concepts in an intuitive way and to visualize the network topology together with the relevant information. This chapter presents the platform developed in this Dissertation, which has the ability to visualize both mobility and multihoming as well as to trigger network related actions. To achieve this is not trivial, because every information presented needs to correspond to the actual situation in the network, and every action triggered needs to be transparent. In addition to these requirements, the platform needs to operate in a real network and in real-time. For this purpose, the platform interacts with the visualization and interaction framework described in the previous chapter. This chapter gives an overview of the developed platform and describes its operation method. Using the platform to trigger network related actions provides the ability to demonstrate mobility and multihoming on desired network entities for demonstration purposes. In order to accomplish this, specific methods are developed and implemented in order to trigger network related actions.

This chapter is organized as follows. Section 4.2 states the challenges related to the development of a visualization and interaction platform. Section 4.3 describes the implementation of the platform. Section 4.4 introduces the architecture of the developed platform and gives an overview of its main entities. Section 4.5 describes the operation method of the platform. Section 4.6 describes the interaction process between the platform and the visualization and interaction framework. Section 4.7 provides the chapter considerations and summarizes this chapter.

4.2 Challenges

When developing a platform to visualize multihoming, mobility and trigger network related actions, there are several challenges that need to be addressed. Solving these challenges is important because it guarantees the reliability of the developed platform. The challenges of developing a visualization and interaction platform in this Dissertation are the following:

- **Visualization:** it is important to present the network topology that corresponds to the actual situation in the network. This implies visualizing every entity in the network as well as connections that exist between those entities. In addition, the platform should display multihoming and mobility in an intuitive way.

- **Scalability:** the functionality of the platform should be independent of the number of network entities that are presented. Due to the unstable network topology and highly dynamic behaviour the network could have, the platform needs to function correctly even if the network topology changes drastically in a short period of time.

- **Reliability:** the user interacting with the platform should have confidence that it will work reliably without presenting unexpected behaviour. This means that the platform should not terminate its work even if there is an unexpected action registered or unknown information received.

- **Interactivity:** in addition to visualization of mobility and multihoming, the platform should also provide features to trigger network related actions. Triggering these actions should not exclude the ability to use the platform to perform other actions or functionalities.

- **Transparency:** every action triggered by the platform should be transparent and made visible through the platform. Network entities that are visualized in the platform should correspond to the state of the network in real-time.

- **Portability:** not knowing on which operating system or computer the platform should operate is one of the challenges. The developed platform should be capable of operating on multiple operating systems.

- **Information exchange:** the platform requires information, about the network for the visualization process. This data is sent through the network in network byte

order which corresponds to big-endian. If the receiving side operates also in big-endian there are no compatibility problems. The challenge is when the information is exchanged between machines which use a different order to store information. This should then be indicated because it would require the information to be converted before usage or storage.

- **Identifier:** it is important to uniquely identify each network entity. Assigning a identifier to a network entity should uniquely differentiate that entity from every other in the network.

- **Interoperability:** the platform should operate independently but needs to exchange information with the visualization and interaction framework. This action should not interfere with other operations of the framework.

4.3 Implementation

This section describes the implementation of the platform developed in this Dissertation. The implementation is done using JUNG [43], a free open-source library that is introduced in section 2.2. The reason for choosing JUNG is that it solves many challenges that are stated in the previous section.

JUNG is written in the Java programming language and provides the ability to use the extensive capabilities of the Java API when developing the platform. The advantage of choosing a software library rather then compromising with the features of an existing platform, is the ability to develop a customized platform that meets the specific needs of this Dissertation's objective.

A major advantage of using JUNG is that it is written in Java and, when compiled, it can run on all platforms that support the Java Virtual Machine. The Java Virtual Machine exists for all the major operating systems, including Linux, Windows and Mac OS, and therefore, makes the platform able to run on multiple platforms. This also solves the challenge of portability. Since JUNG is written in Java, the data that is sent is in the network byte order, which corresponds to big-endian. If the side receiving the data operates in big-endian, this could present a problem. For this purpose, the developed platform sends the information as a sequence of characters. This sequence is read character by character at the receiving side. Since each character that is read is equal to one byte, this makes the information sent as a sequence of characters independent of the little or

big-endian order. The information sent to the platform is also in the form of a sequence of characters, making it possible to exchange data between machines that use different order to store information. This also solves the challenge of interoperability between the developed platform and the interaction and visualization framework described in the previous chapter. By using the extensive Java API, the platform has implemented mechanisms that will ensure that the platform operates even if there is unknown information received, which solves the challenge of reliability. Although the Java programming language allows a great number of advantages, it also has certain requirements. The end-user terminal on which the platform is running should have the version *Java 8* or higher. If the end-user terminal has a earlier version than *Java 8*, it simply needs to be updated to version 8 or higher.

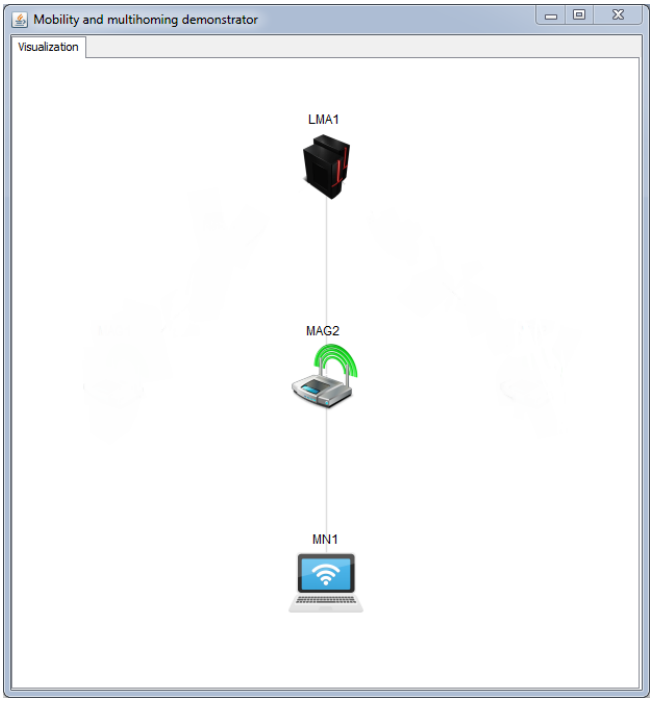


Figure 4.1: Initial view in the platform

In order to present mobility and multihoming, it is important to agree in which way the network data is going to be displayed. A network usually consists of a set of entities and known relationships between these entities. Therefore, it is practical to present network data as a node-link graph, with vertices representing network entities and edges representing their relationships. Figure 4.2 shows the basic representation of data in JUNG. As we can see, the representation corresponds to a node-link graph. However, in the network we can have multiple types of entities which make that network a multi-modal network, and

different type of relationships between the entities. In this case the basic node-link graph representation provided by JUNG, does not provide a clear overview of the different entities in the network. In order to clearly distinguish different network entities, modifications are performed to the basic implementation of JUNG. These modifications make it possible to display network nodes as icons, where network entities of the same type are represented by the same icon. Figure 4.3 shows the modified node representation where network nodes are displayed as icons. This representation allows to clearly distinguish different network entities, while keeping the simplicity of the node-link graph representation. The network entities shown in Figure 4.3 are the LMA, the MAG and the MN. In the situation when among the network entities is a mMAG, it is presented using a icon that is different from the icons that represent other network entities.

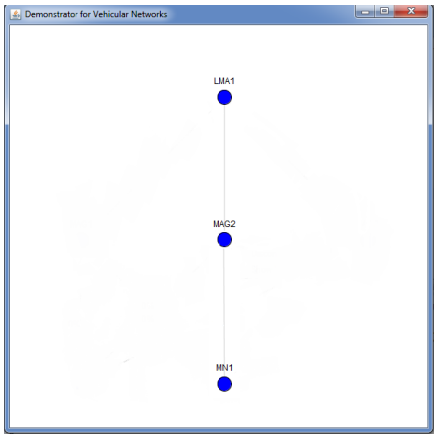


Figure 4.2: Basic node representation

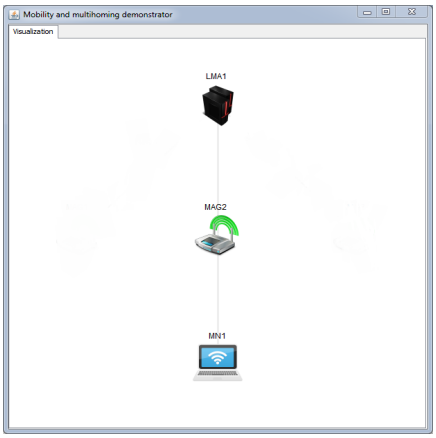


Figure 4.3: Modified node representation

The network topology needs to be displayed taking into account both large networks that have a large number of nodes, as well as small networks with only a few nodes. The existing layout mechanisms in JUNG are not well suited to provide a clear network overview, when the network topology changes rapidly over time. Therefore, a layout algorithm was developed and implemented in JUNG that dynamically adjusts the positions of the network nodes, based on the number of entities in the network and the connections between those entities. Figure 4.4 and Figure 4.5 show the implemented layout algorithm in the platform. As we can see, the algorithm displays the network entities in a hierarchical order, where the LMAs are displayed over the MAGs, followed by the mMAGs and finally by the MNs. In Figure 4.4 we see the initial layout of the network entities. Figure 4.5 shows the network layout when a second MN connects to the network. As we can see, the layout changed dynamically and adjusted the node positions of both MNs to provide a better overview of

the network. Although in this case the algorithm only adjusted the node positions of the MNs, it has the ability to adjust the node position of any network entity depending on the situation in the network.

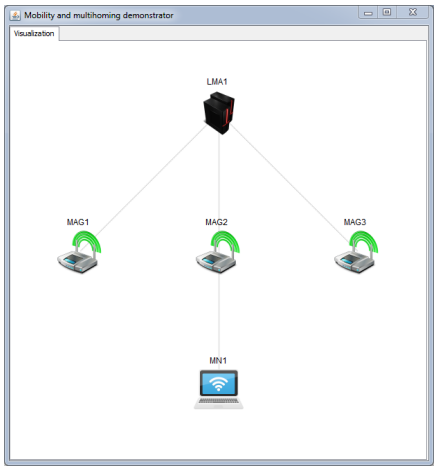


Figure 4.4: Layout with one MN

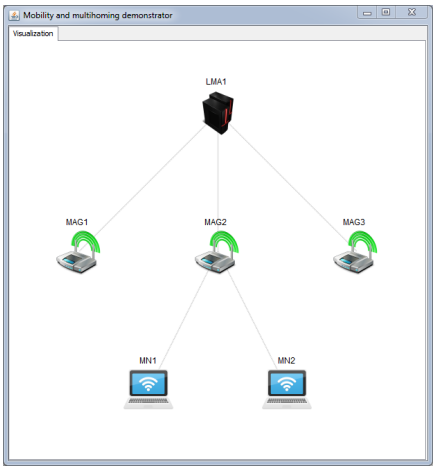


Figure 4.5: Layout with two MNs

An important feature of the platform is the ability to interact with the network nodes and trigger network related actions. The basic implementation of JUNG provides only a few simple interaction features, such as selecting and labelling network nodes. Therefore, a custom plug-in is developed and implemented that provides interaction features to the existing implementation of JUNG. The plug-in allows a certain network entity to be selected in the platform, after which a pop-up menu is displayed, presenting the interaction options. This allows for a great amount of flexibility, since interaction methods can be added and removed easily in the plug-in.

4.4 Architecture

Before explaining the operation method of the platform, it is important to present the overall architecture and entities of which it is composed. Figure 4.6 presents the architecture of the developed platform. The platform consists of multiple entities. Each entity provides a different functionality and can communicate with other entities to accomplish its task. More detailed explanation about the entities is given below.

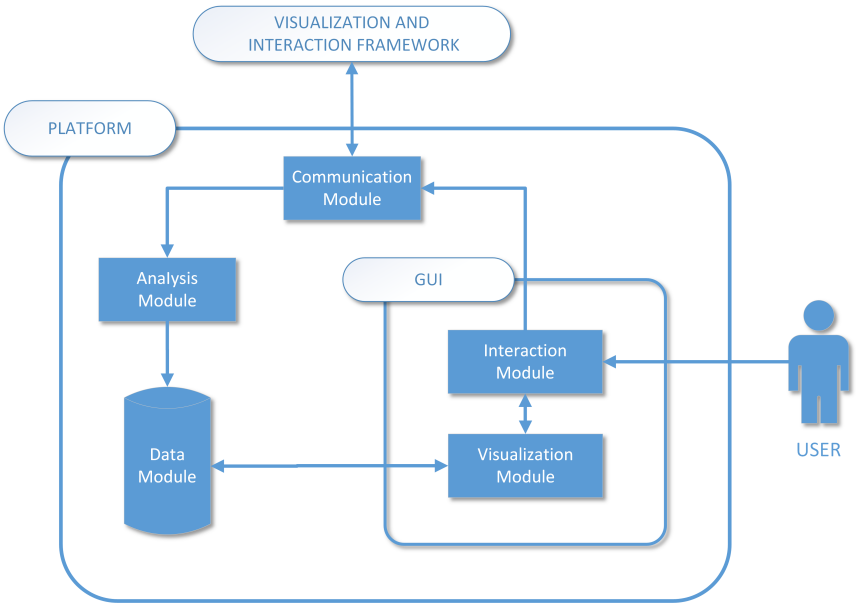


Figure 4.6: Architecture of the developed platform

4.4.1 Communication Module

The platform needs to be able to communicate with the interaction and visualization framework explained in the previous chapter. The Communication Module (CMOD) is responsible for sending and receiving information from the visualization and interaction framework explained in the previous chapter. The CMOD entity has the ability to communicate with the AMOD and IMOD entities. When information is received, the CMOD entity forwards this information directly to the AMOD. In the case when information needs to be sent, the IMOD forwards this information to the CMOD entity which then sends it to the DM entity. The CMOD entity does not make any changes to the information it receives or sends. Since the information can be sent or received at any time, we have to take into account the situation when information is both sent and received in the same moment. In this case the CMOD entity first stores the received information into a buffer and then sends the information it received from the IMOD. After the information is sent the CMOD entity retrieves the information it stored in the buffer, forwards this information to the AMOD and empties the buffer in which the information was stored.

4.4.2 Analysis Module

In order to determine the content of the received information, an intelligence process is needed to analyse the received information. This is the primary function of the Analysis Module (AMOD). The AMOD entity is able to communicate with the CMOD from which it receives the information, and the DMOD to which it sends the information after it is analysed. When the information is received from the CMOD entity, AMOD analyses its content, determines what type of network data is contained in the message and extracts the corresponding data. After this process, the extracted data is sent to the DMOD. Without the AMOD it would not be possible to analyse the received information and extract information from it.

4.4.3 Data Module

After the AMOD entity processes the received message, the extracted data is forwarded to the Data Module (DMOD). The DMOD stores this information into internal data structures which contain information about the entire network. Should the DMOD entity receive information about an entity for which an internal structure does not exist, the DMOD entity will create the internal data structure for this network entity and store information about it in corresponding structure. If the received information is about a network entity for which an internal data structure already exists, the DMOD entity will simply update the internal data structure with the most recent information. After each update or after a new data structure is created, the DMOD will inform the VMOD entity and make the data structures available in order for the VMOD to access them.

4.4.4 Visualization Module

As shown in the Figure 4.6 a sub-part of the platform is the GUI which allows the user to interact with the platform. The GUI is also responsible for presenting the visualization of mobility and multihoming. The GUI can be considered as the main entity of the developed platform, since it is directly responsible for visualization and network interaction. The platforms' GUI is comprised of two parts. The first part is the Visualization module (VMOD). The VMOD entity is responsible for visualizing the network data stored in the DMOD and presenting that data to the user. Every time the data structures are updated, the VMOD verifies the change that occurred, updates the visualization and visualizes this change. The VMOD entity visualizes the network data by presenting all the entities present

Table 4.1: Messages required for the visualization

Message header	Message type
GETNODES	Information about the network nodes
GETCONNS	Connections between the network nodes
GETFLOWS	Initial traffic flow information
ADD_MN	Network interface has connected
DEL_MN	Network interface has disconnected
ADD_FR	Information about the traffic flows
DEL_FR	Traffic flow has terminated

in the network and the connections between them.

4.4.5 Interaction Module

The second part of the platforms' GUI is the Interaction module (IMOD). The IMOD entity allows user interaction with the platform in order to trigger network related actions. The triggering and interaction is done in the following way. Using the IMOD the user selects a preferred network entity and chooses between various options which will trigger different network actions. After an option has been selected, a request is constructed that indicates the network entity which was selected and the network related actions that need to be triggered. This message is then forwarded to the CMOD entity which sends it to the visualization and interaction framework.

4.5 Operation Method

This section describes the operation method of the platform developed in this Dissertation. The platform communicates with the visualization and interaction framework explained in the previous chapter. The communication is done in the form of exchanging messages sent as sequence of characters. Each message contains a specific header which determines the type of message. Table 4.1 and Table 4.2 show the message headers corresponding to each type.

Messages that are exchanged are divided into two types:

- messages required for the visualization.
- messages required to trigger network related actions.

Table 4.2: Messages required to trigger network related actions

Message header	Message type
SCANREQ	Request to perform scan of available networks
SCANRSP	Information about the available networks
DISCONN	Request to disconnect a network interface
CONNECT	Request to connect a network interface
STARTFL	Request to start a traffic flow
QUERYFL	Confirmation the traffic flow can be started

After the platform is initialized, a UDP packet is sent to the DM entity located on the LMA, containing a message with the *GETNODES* header (Table 4.1). This message indicates that information about the entire network should be sent to the source address of the received packet. After the DM entity receives the packet, it checks the message header contained in the packet. Based on the *GETNODES* header, the DM determines that the message received corresponds to the initial request sent after the platform is initialized. This request indicates that information about the entire network needs to be sent to the platform.

The information sent about the network includes the following data:

- **Identifier of every entity:** contains the unique identifier of every entity in the network and type of each entity.
- **Connections between entities:** for every entity the identifier is sent paired with the identifier of every entity connected to that entity.
- **Traffic flow information:** if there are traffic flows present in the network this information will be sent to the platform.

Based on this information, three messages are constructed, each with a specific header: the identifiers of the network nodes together with the *GETNODES* header; connections between the network nodes with the *GETCONNS* header; information about the traffic flows with the *GETFLOWS* header. These messages are then sent in packets to the platform that originated the initial request.

The platform receives this packet using the CMOD module. The information contained in the packet is then analysed using the AMOD module, which based on the specific *GETNODES*, *GETCONNS* and *GETFLOWS* headers, determines that the received information is the response to the initial request and contains information about the entire

network. This information is then easily extracted using the specific headers in the message and stored in the DMOD, after which the VMOD and IMOD are initialized, and the visualization process presents this data using the VMOD.

The network can be subject to a large number of changes which can occur at any point and effect an arbitrary number of network entities. For the visualization and triggering process to work properly, the information presented in the platform should be accurate and correspond to the actual situation in the network. This implies that the platform should be informed about every change that happens in the network. For this reason, after its initialization, the DM entity detects any changes that occur in the network and sends a message to the platform informing it about the changes that occurred. The messages that are sent only contain information about the network change that occurred, and about the entities that were involved. These messages are short and introduce less overhead in the network, opposite to the situation when after, every network change, information about the entire network is sent to the platform. In a large network with a large amount of nodes, this could present a problem. Since different changes can occur in the network, it is important to define the changes for which the DM is going to notify the platform.

These are the following:

- Network entity has connected a network interface.
- Network entity has disconnected a network interface.
- Traffic flow was started to a MN.
- Traffic flow to a MN was terminated.
- Parameters regarding a traffic flow have changed.

When these changes occur, the DM entity will inform the platform. Figure 4.7 shows the operation method of the DM entity. The platform will analyse the information it receives and determine the type of change that happened based on the header of the message (Table 4.1). Then, it will identify the network entities that were involved, the parameters that changed and visualize them.

The DM entity is not included only in the visualization process, but also in the process of triggering network related actions. This can be seen in Figure 4.7. In order to trigger network related actions, the platform sends the request to the DM indicating the action that needs to be performed and the entities it involves. When the DM receives this request,

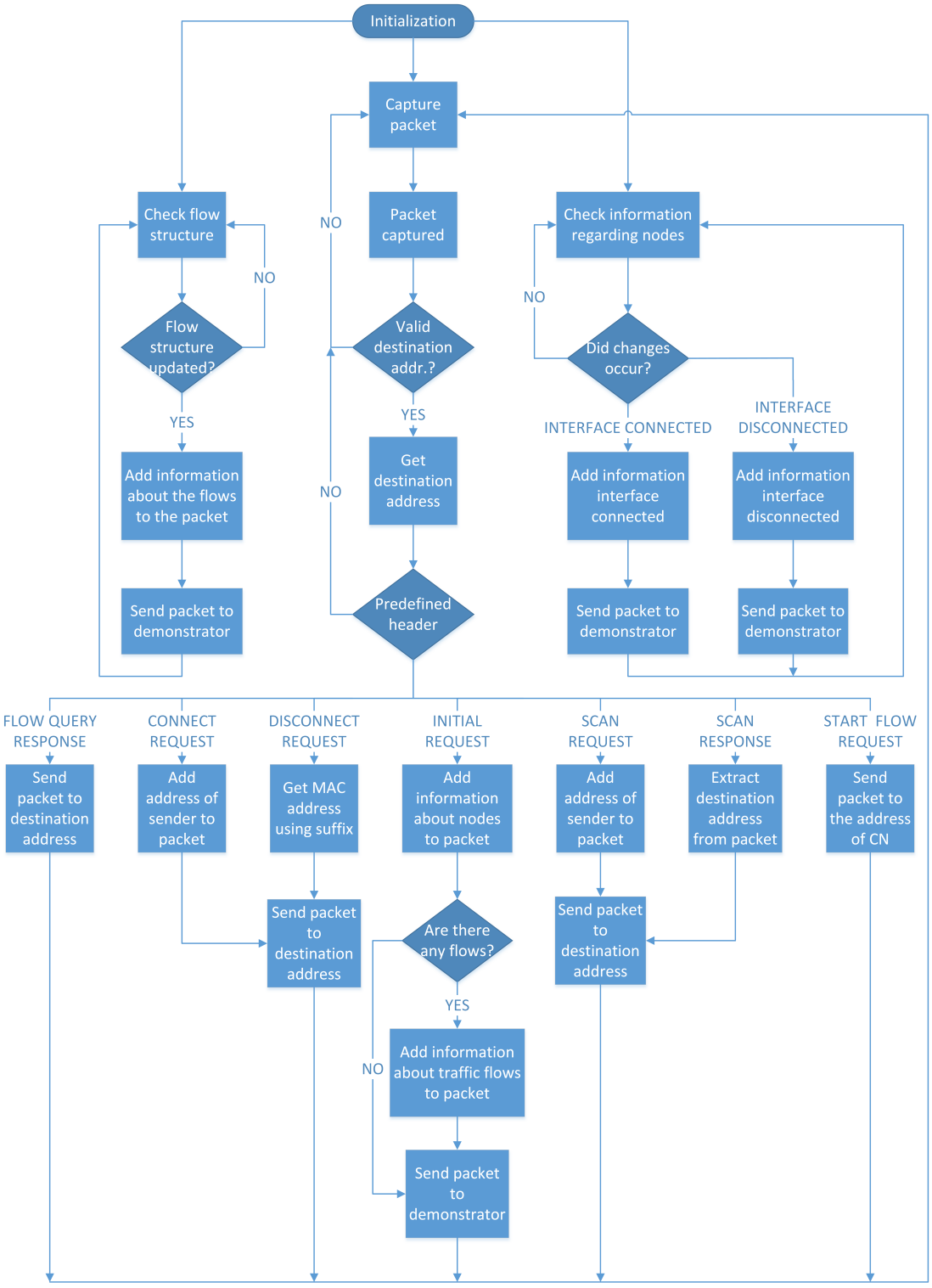


Figure 4.7: DM operation flow diagram

it will determine the action it needs to perform based on the header of that request (Table 4.2), and send it to the entities that are involved. Since the platform does not know the position of the other nodes in the network, the request to trigger a network related action is first sent to the DM, which then forwards it to the corresponding network entity. This request is then received by the UTS entity of the corresponding network entity. Figure 4.8 shows the operation diagram of the UTS entity.

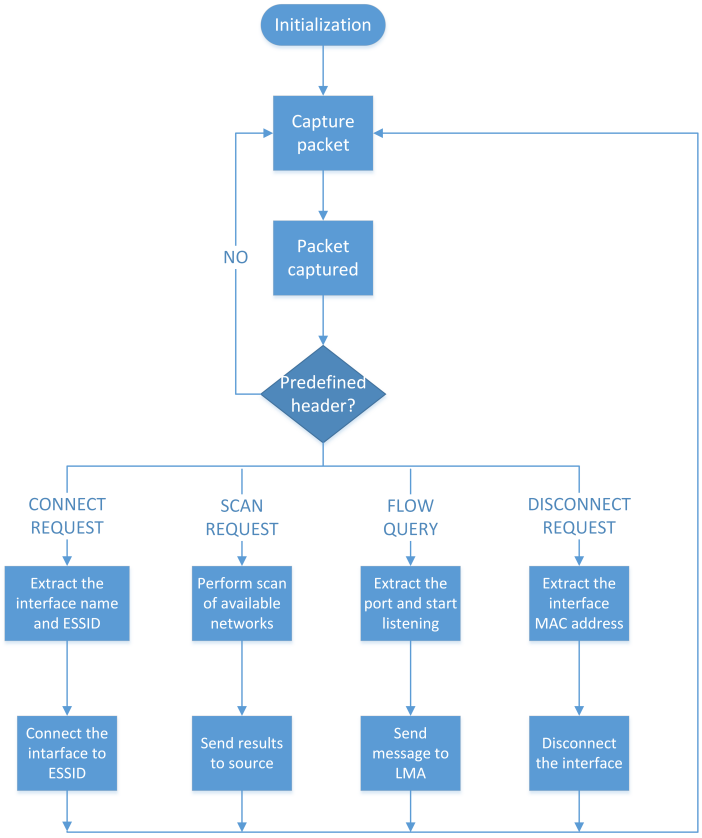


Figure 4.8: UTS operation flow diagram

After the UTS entity receives the request to trigger a network related action, it determines the type of the request based on the specific header, and executes the action for the corresponding network event. If there are multiple traffic flows for the same MN, there is a need to uniquely differentiate information that is sent about each flow. This is possible using the MN prefix, flow source address, flow destination address, flow source port, flow destination port, flow transport protocol. These parameters form an identifier used to uniquely differentiate each traffic flow that is present in the network. Two remaining parameters that are sent, flow rate and flow mean packet size, are related with the

multihoming traffic and used to visualize multihoming. In the case when no traffic flows are present at the moment when the DM receives the initial request from the platform, a message will be sent to the platform stating that there are no traffic flows present. The information sent to the platform is sent in a UDP packet. The reason to use UDP instead of TCP is related to the fact that the information is short and can fit inside one packet. Should the message needs to be sent not fit inside one packet, which would correspond to a situation when there would be a large number of entities in the network, the message would be divided into several packets and sent to the platform.

After the initial data is sent to the platform, the DM will only send the information when there is a change in the network either related to a change in the number of entities, connections between them or information regarding traffic flows and multihoming. When sending this data, the DM will only send information about the change that occurred, and not the information about the entire network. This reduces the information being sent through the network.

4.6 Interaction Process

This section describes the network related actions that can be triggered on the end-user terminal using the platform developed in this Dissertation. These actions are the following:

- Performing a scan of available networks.
- Connecting a network interface to an available network.
- Disconnecting a network interface from the network.
- Starting a traffic flow to the end-user terminal.

These events correspond to the scenarios described in Section 3.2.

4.6.1 Scanning Available Networks

The first method developed is the process of performing a scan of available networks at the end-user terminal. Figure 4.10 shows the message exchange of this event. By using the platform and selecting a certain user, the option *Scan available networks* is available. Figure 4.9 shows the scan option in the platform. This option performs the scan of networks that are in the range of that user terminal and to which it can connect. After the *Scan*

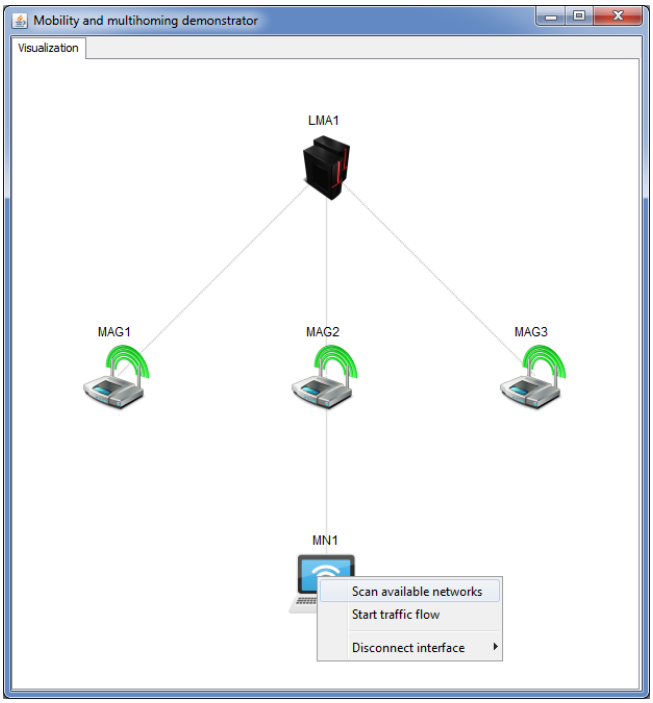


Figure 4.9: Scanning of available networks in the platform

available networks option is selected, the request to perform a scan of available networks for the specified end-user terminal is sent to the DM entity that is located on the LMA. This request contains the specific message header *SCANREQ* that indicates the type of the request, and the identifier of the MN specifying the end-user terminal for which the scan should be performed. After receiving the packet, the DM determines the type of the message located in the packet. It extracts the message header and determines that the message received is a scan request for a specific MN. Then, it extracts the identifier of the corresponding MN and sends the scan request to the specified MN after adding the IP address of the platform that originated the request, to the scan request in the packet. The packet is received by the corresponding MAG or mMAG, and forwarded to the MN which address is specified in the destination header. The MN receives the packet, analyses the header of the message and determines that the message is a scan request. Then, the MN performs the scanning of the available networks and sends the results to the platform, with the IP address the DM entity added to the scan request message. This address is extracted from the scan request message and inserted in the destination header of the packet containing the scan results.

The result of the scan process is a list that contains the list of network interfaces that

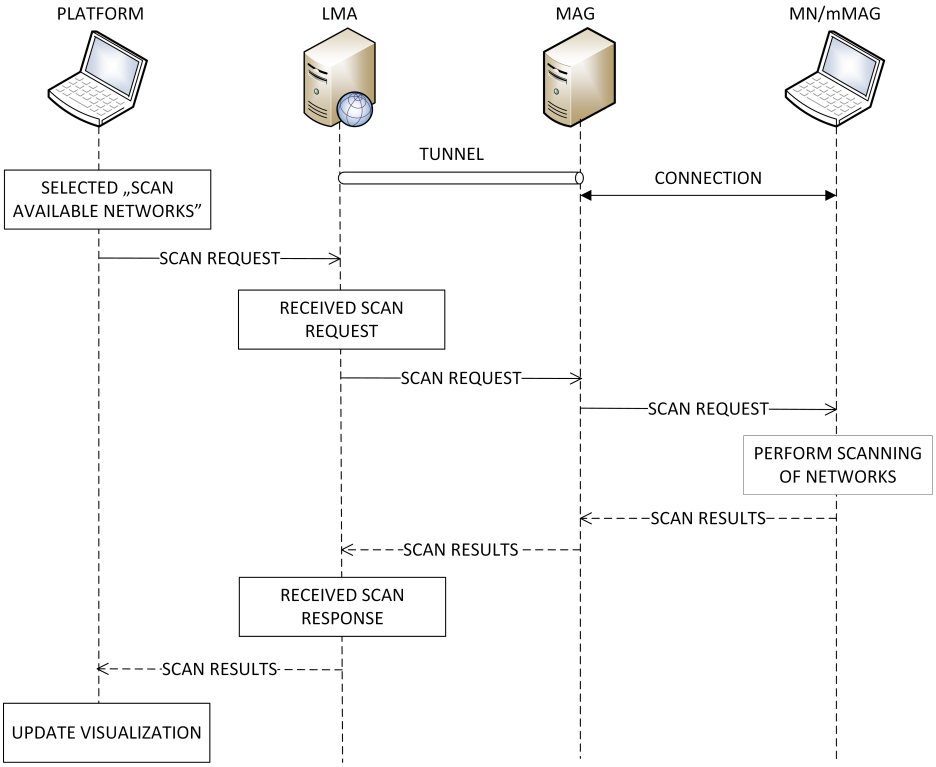


Figure 4.10: Message exchange in the scanning of available networks

are not connected to any network, and the list of networks that are available for each of those interfaces. The networks to which other network interfaces of that MN are connected are omitted from the scan result. This is done to prevent multiple network interfaces from being connected to the same network. The results of the scan event are then added to the message together with the header *SCANRSP*, to indicate that the message contains the scan results. This message is then sent to the platform in a packet. The packet is first received by the MAG or mMAG, and then forwarded to the DM entity. The DM entity, upon receiving the packet, analyses the header of the message and determines that the message contains the results of a previous scan event. It then forwards this packet to the platform which originated the request. The platform receives the results of the scan event, determines the message type by analysing the header of the message and updates the visualization. After the visualization is updated, the results of the scan event are presented for the MN, for which the scanning of available networks was performed. In the platform, the results of the scan are presented as available network interfaces, together with the list of networks for each available interface.

4.6.2 Connecting Network Interfaces

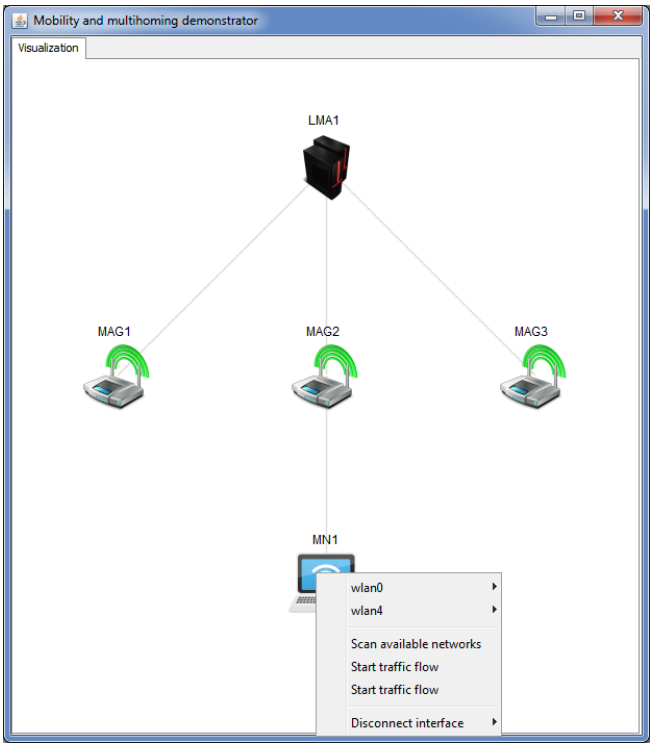


Figure 4.11: Connecting a network interface in the platform

An important feature of the platform is the ability to connect a network interface of the MN, to an available network we select in the platform. In this process we only take into account the network interfaces that are not already connected. This option of connecting a network interface, requires that the process of scanning the available networks explained in the previous section, is performed previously. After performing the scan of available networks for a selected MN, a list of available network interfaces is shown together with the available networks for that interface. Figure 4.11 shows this option in the platform. As previously explained, the networks to which other interfaces of that MN are connected are not presented to avoid double connections. After selecting the desired interface and network to which we would like to connect the interface, a request to connect the interface is sent. This request contains the parameters selected in the platform: the identifier of the MN, the identifier of the network interface and the identifier of the network.

After selecting the network interface and the network to which we would like to connect that interface, a request is constructed and sent to the DM entity. Figure 4.12 shows the messages that are exchanged in the process of connecting the network interface. Upon

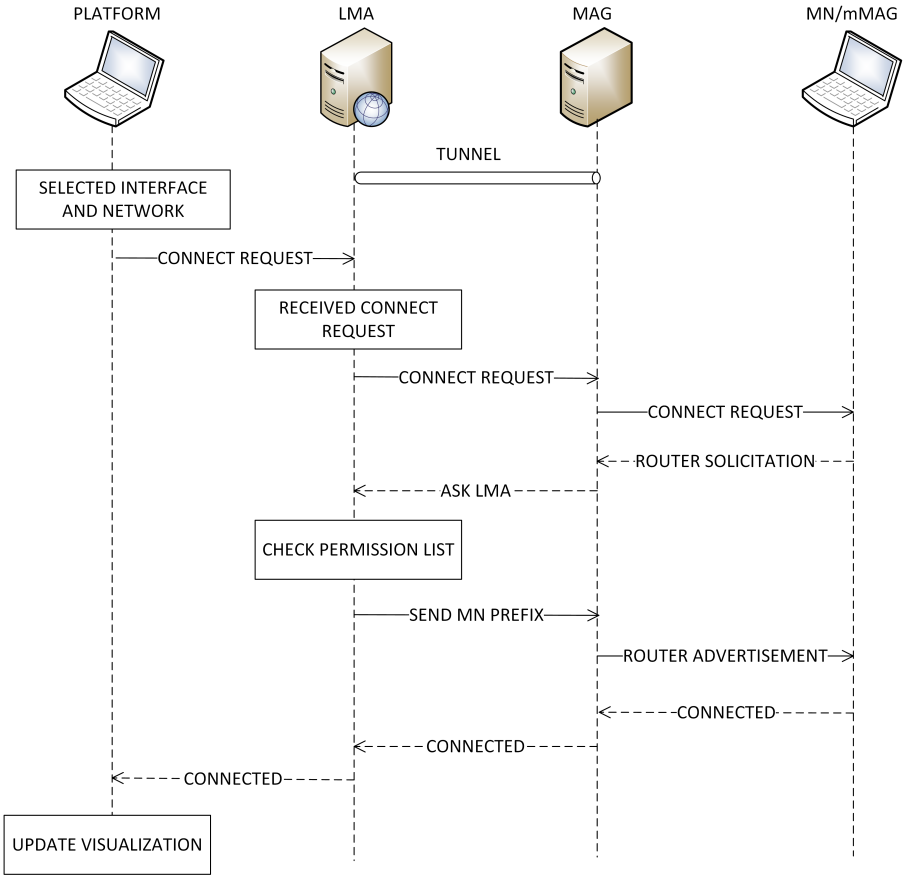


Figure 4.12: Message exchange in connecting a network interface

receiving the packet with the specific header *CONNECT* that corresponds to the request to connect an interface, the DM extracts the destination address from the packet and sends the packet to this address. This packet is then forwarded to the IP address of the interface we would like to connect. The packet is sent from the DM to the corresponding MAG or the mMAG, which then forwards it to the corresponding MN. When the MN receives the packet, it reads the header and, after determining that the packet corresponds to the request to connect the interface, the MN tries to connect to the PoA. For this purpose, the MN sends a RS to connect to the corresponding PoA. The MAG receives the RS and informs the LMA that the MN would like to connect its interface to the network. When the LMA is informed, it checks in the permission list if the MN has the permission to connect its interface to the network. If the MN has permission, the LMA will assign a network prefix to the MN and send it to the MAG which originated the request. Upon receiving the network prefix for the MN, the MAG sends an RA to the MN and the connection is

established. After the connection is established, the LMA sends a message to the platform, with the header *ADD_MN* indicating that a new network interface is connected to the network. The platform receives this information and updates the visualization, which in turn makes the change visible to the user and allows further interaction. In the case when the MN does not have permission to connect its interface to the network, the LMA will not send the prefix of the MN to the MAG. Therefore, the network interface of the MN does not connect to the network and the visualization service is not updated.

4.6.3 Disconnecting Network Interfaces

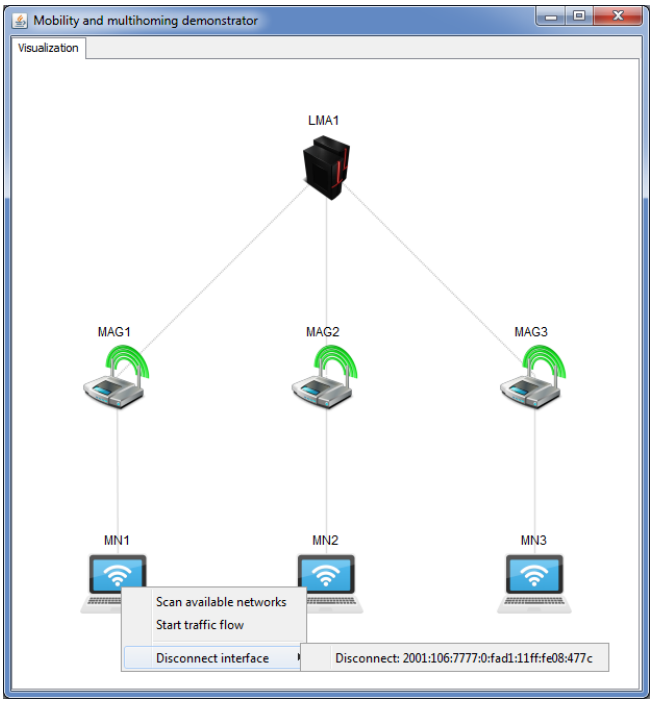


Figure 4.13: Disconnecting a network interface in the platform

The next interaction process explained is related to disconnecting a network interface of the MN. This implies that the network interface we would like to disconnect is connected to the network. Through the platforms GUI it is possible to select an arbitrary MN and choose the option to disconnect any interface, that is currently connected. Figure 4.13 shows the disconnect option in the platform.

Figure 4.14 shows the messages exchanged in this event. By selecting the disconnect option, a message is built based on the information of the selected interface. This message contains the specific header *DISCONN*, and corresponds to a request that can uniquely

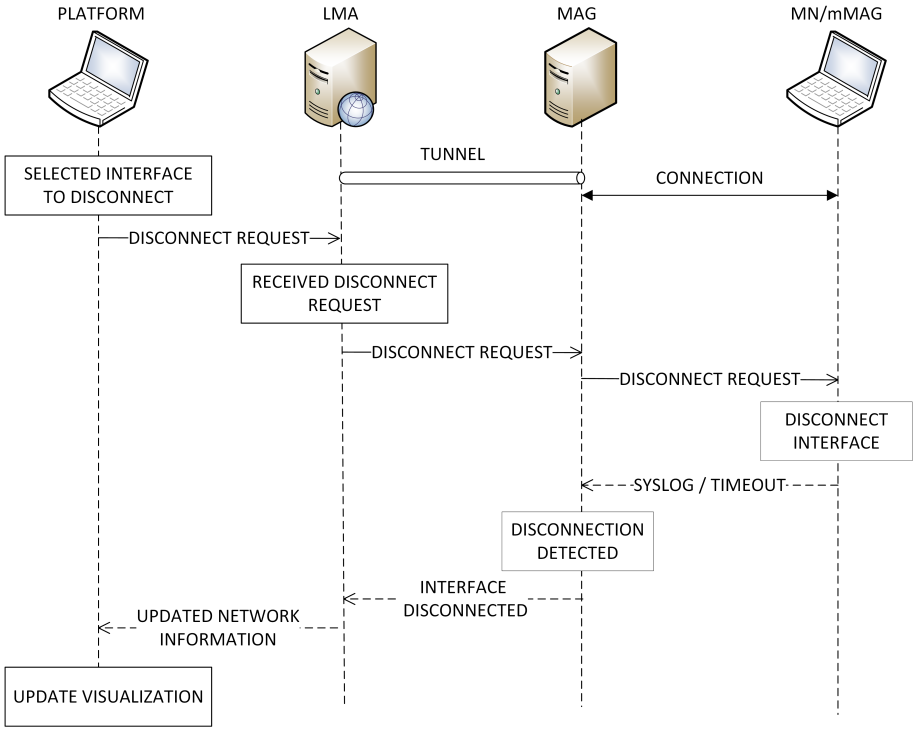


Figure 4.14: Message exchange in disconnecting a network interface

determine the interface that needs to be disconnected, and the position of the node with that interface in the network. This message is forwarded in a packet to the DM entity. The DM receives the packet and extracts the message from the packet. It determines that the message corresponds to the request to disconnect a network interface. The DM extracts from the request the identifier of the interface that needs to be disconnected, and based on it determines the MAC address. The DM then adds this MAC address to the packet together with the IP address of the platform which originated the packet. This packet is then sent to the MAG or mMAG, which then forwards it to the MN. The packet is received by the UTS entity of the MN which interface needs to be disconnected. After receiving the packet, the UTS extracts the message that was modified by the DM entity. Based on the specific header of the message, the UTS entity determines that the message corresponds to a request to disconnect a network interface. Then, it extracts the interface identifier and disconnects that interface from the network. When the interface is disconnected, the MAG detects that the interface disconnected, by either using *syslog* messages or after the connection timeout. Then, it informs the LMA, which sends a message to the platform with the specific header *DEL_MN*. This header indicates that a network interface has been

disconnected. The platform receives the message, determines its type using the AMOD module and updates the visualization using the VMOD module.

4.6.4 Starting Traffic Flows

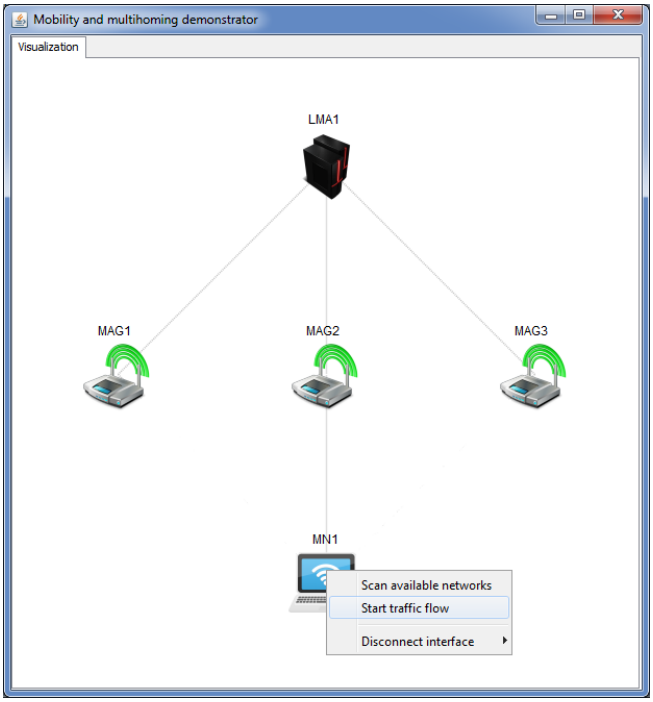


Figure 4.15: Starting a traffic flow in the platform

In order to demonstrate the application of multihoming, data needs to be sent to the end-user terminal. Therefore, as a part of this Dissertation, a process is developed to start a traffic flow from the CN to a certain end-user terminal. Figure 4.15 shows the option to start a traffic flow in the platform. After selecting the option *Start traffic flows* on an arbitrary MN in the platform, parameters required to start the traffic flow need to be inserted. Figure 4.16 shows these parameters in the platform.

They include the destination port, transport protocol, packet rate, packet size and time duration of the traffic flow. After these parameters are inserted, a message is constructed containing these parameters and the specific header *STARTFL*. This message is then sent to the DM. Figure 4.17 shows the messages that are exchanged in the start of the traffic flow. After receiving the message, the DM checks the type of the message and, based on the specific header, it determines that the received message corresponds to a request to start a traffic flow. Then, the DM entity first sends a query message to the end-user

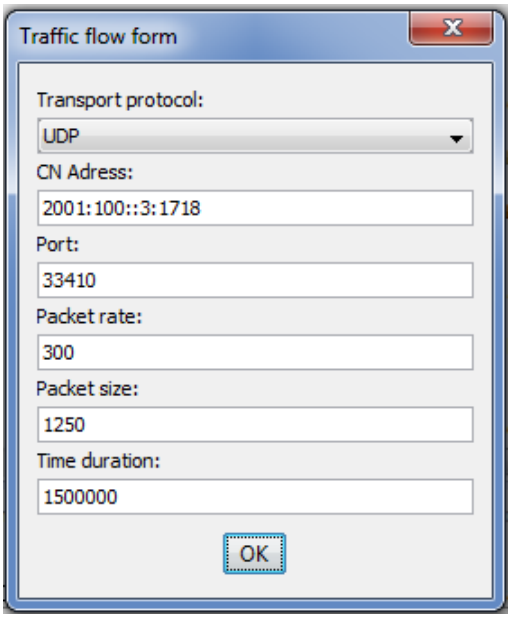


Figure 4.16: Parameters required to start the traffic flow

terminal to which the traffic data should be sent, to check if the end-user terminal is ready to receive the data. The query message is received by the UTS entity. After receiving the query, the UTS sends a confirmation message to the DM, in order to confirm that the end-user terminal is able to receive the data traffic. After the confirmation is received by the DM, it informs the CN to start the traffic flow with the specified parameters and indicates the designated end-user terminal. The CN starts the traffic flow to the end-user terminal, which is detected by the FM located on the LMA. The FM detects that a traffic flow is started and determines the multihoming rule after which information about the traffic flow and multihoming rule is sent to the platform in a message with the specific header *ADD_FR*. Upon receiving this information, the platform updates the visualization and presents the new traffic flow together with the corresponding information related to multihoming.

4.7 Chapter Considerations

This chapter presented the visualization and interaction platform developed in this Dissertation.

The architecture of the platform consists of multiple modules, which are able to communicate with each other and exchange information. The visualization is exactly based

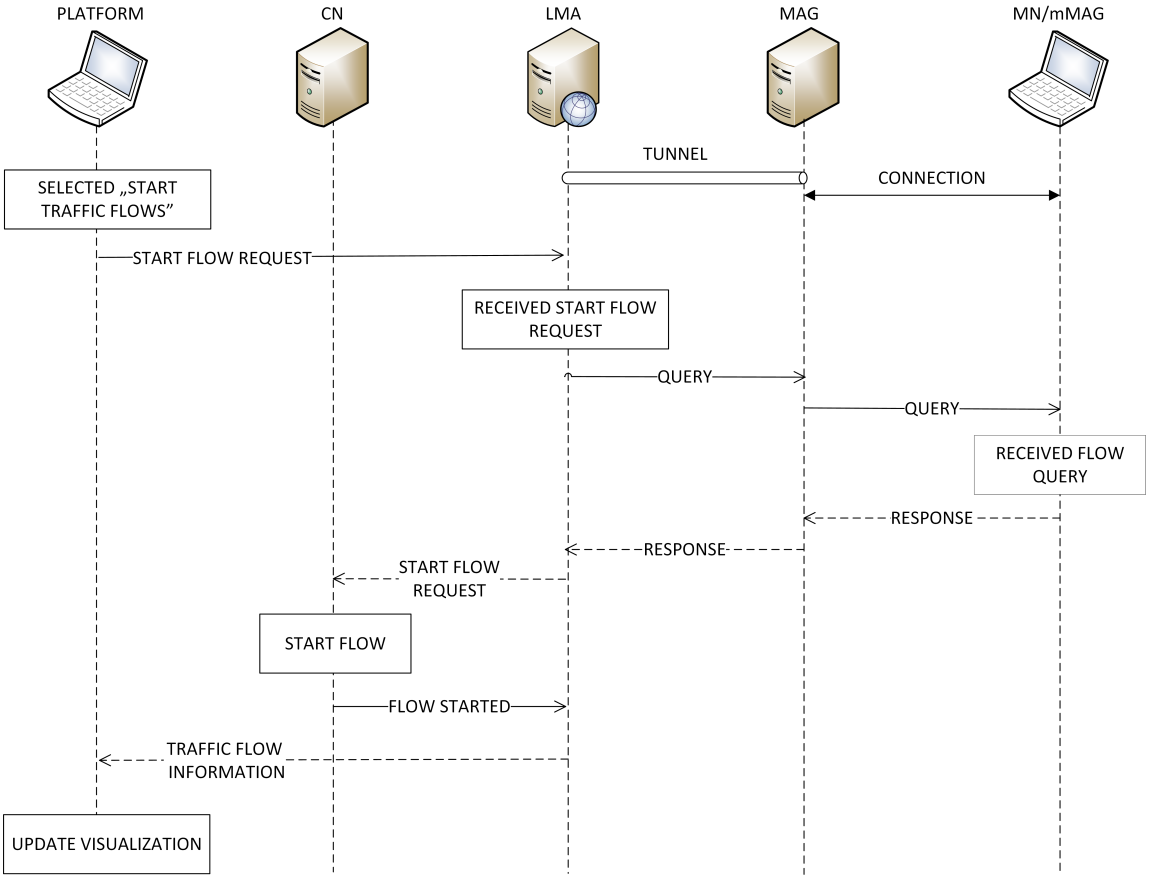


Figure 4.17: Message exchange in starting the traffic flows

on the information that is exchanged during this communication. The operation method of the platform is based on the communication between the platform and the framework explained in the previous chapter.

The presented interaction methods are mainly focused on triggering actions on the MN side. Should the network related actions be triggered on a MAG or mMAG, multiple MNs could be connected to it, and in a real network, we could interfere with the connections of other end-user terminals which could present a problem.

Sending the information in a format corresponding to a certain programming language would imply that every network entity that needs to send or receive that information should support the chosen programming language format. In the situation when the platform would be applied in a large network where the entities would not support the format in which the platform sends or receives information, this would require to make modifications on all the network entities which do not support the information format. In a network

where we would need to make modifications to a large number of network entities, this would present a problem. The possibility would also be sending this information in the form of bytes, but then when receiving this information we would loose the ability to quickly and with certainty determine to which network entity the received information corresponds. Sending the information as a sequence of characters makes it also possible for that information to be exchanged, which store the data in big-endian or little-endian. This presents a big advantage, since no prior knowledge about the machines in the network is required. Since the platform is developed in the Java programming language, it is required to have the Java Runtime Environment installed and properly configured.

The platform was developed with the aim to be used as a visualization platform that can be run on a single terminal. Since JUNG is based on Java programming language, with alterations it can be deployed and used as a web application. Changes needed to be made are regarding the way information is being sent to CMOD module of the platform. In the case the platform would be available as a web service, the entity integrated in the MN would need to have access to the Internet or send the information to a certain entity in the network that would be connected to the Internet. The requirement is that the end-user terminal running the platform has the *Java Runtime Environment* installed and properly configured.

After the platform is developed and implemented, it is important to demonstrate its features. Therefore, in the next chapter an evaluation of the platform will be performed, which will verify the platforms ability to operate in a real network.

Chapter 5

Evaluation

5.1 Introduction

In order to verify the implemented functionalities and examine the performance of the platform, it is important to test it in a real network. This chapter presents the evaluation of the platform developed in this Dissertation. The platform has both interaction and visualization abilities, which need to be taken into account when performing the evaluation. The evaluation of the platform is performed in a laboratory environment in both a real wireless network, and a VANET. For this purpose, testbeds are deployed and used to evaluate the platform and its performance. It is expected that the platform is capable of successfully visualizing both mobility and multihoming, as well as triggering network related actions. The visualization and the actions that are triggered are evaluated based on the network representation displayed in the platform. However, in order to evaluate the performance of the platform, metrics related to the visualization and interaction process are obtained and analysed in this chapter.

This chapter is organized in the following way. Section 4.2 presents the testbeds that are used to evaluate the platform. This section describes the entities of which the testbeds are composed, together with an overview of their configuration, and provides a detailed description of the machine characteristics.

Section 4.3 defines the metrics that are obtained and describes the methodology used to obtain these metrics. Based on the defined metrics, it is possible to perform an objective evaluation of the platform and of its performance.

Section 4.4 presents the visualization and results obtained from the testbed in a wireless network.

Table 5.1: Testbed1 characteristics

	CN	LMA	MAG1	MAG2	MAG3	MN1	MN2	MN3
CPU [GHz]	3.10	3.10	1.5	1.7	3.10	2.26	1.4	1.66
Memory [GiB]	0.496	1.0	0.496	0.496	3.4	2.9	3.4	0.992
OS	Ubuntu (v11.04)	Ubuntu (v11.04)	Ubuntu (v11.04)	Ubuntu (v11.04)	Ubuntu (v11.04)	Ubuntu (v11.04)	Ubuntu (v14.04)	Ubuntu (v12.10)
Linux Kernel	2.6.35	2.6.35	2.6.35	2.6.35	3.0.0	3.2.0	3.16.0	3.5.0

Section 4.5 presents the visualization and results obtained from the testbed in a VANET.

Finally section 4.6 provides chapter considerations and summarizes the evaluation performed in this chapter.

5.2 Testbed

This section describes the testbeds used to perform the evaluation of the visualization and interaction platform. In order to provide an objective evaluation of the platform, two testbeds are deployed in the laboratory environment. These testbeds are used to examine both the visualization and interaction abilities of the developed platform. The implementation of each testbed is presented in this section, together with an overview of the equipment used to deploy the testbeds.

5.2.1 Equipment Used

The first testbed contains the following entities: the CN, the LMA, the MAG and the MN. Table 5.1 shows the machine specifications of these entities. In order to create different Wi-Fi networks, three Cambria Network Platforms [37] are used as PoAs. The specifications of these PoAs are shown in Table 5.2. As we can see from the specification, the hardware characteristics of the PoAs used in the evaluation process are identical. Three personal computers are used as the MNs. They are the following:

- MN1: Asus laptop with Intel Core 2 Duo 32-bit processor and Ubuntu v11.04 operating system.
- MN2: HP laptop with AMD E1 64-bit processor and Ubuntu v14.04 operating system.

Table 5.2: Wireless PoA characteristics

Parameter	PoA1	PoA2	PoA3
Development kit	GW11016	GW11016	GW11016
CPU [MHz]	667	667	667
OS	OpenWRT (2.6.32.27)	OpenWRT (2.6.32.27)	OpenWRT (2.6.32.27)
Wireless card	Atheros (AR5213A-001)	Atheros (AR5213A-001)	Atheros (AR5213A-001)

Table 5.3: Testbed2 characteristics

	CN	LMA	MAG1	MAG2	mMAG1	mMAG2
CPU [GHz]	2.18	3.10	0.5	0.5	0.5	0.5
Memory [GiB]	0.476	7.529	0.059	0.059	0.059	0.059
OS	Ubuntu (v10.10)	Ubuntu (v14.04)	Veniam (v19.2)	Veniam (v19.2)	Veniam (v19.2)	Veniam (v19.2)
Linux Kernel	2.6.35	3.13.1	3.7.4	3.7.4	3.7.4	3.7.4

- MN3: Asus laptop with Intel Atom 32-bit processor and Ubuntu v12.10 operating system.

The MNs are equipped with three network interfaces and can connect to a PoA in their range, using Wi-Fi technology.

The second testbed contains the following entities: the CN, the LMA, the MAG and the mMAG. Table 5.3 shows the machine specification of these entities. The MAGs are fixed stations in the road, road side units, and the mMAGs are on board units in the vehicles. The on board units are equipped with both Wi-Fi and WAVE interfaces. The MAGs and mMAGs are running Veniam v19.2 operating system, which is based on the OpenWRT operating system.

5.2.2 Testbeds Implemented

In order, to evaluate the visualization and interaction abilities of the platform two testbeds are deployed.

The first testbed is depicted in Figure 5.1. It shows a real wireless network deployed in the laboratory environment. This testbed is aimed to evaluate the platform in a network

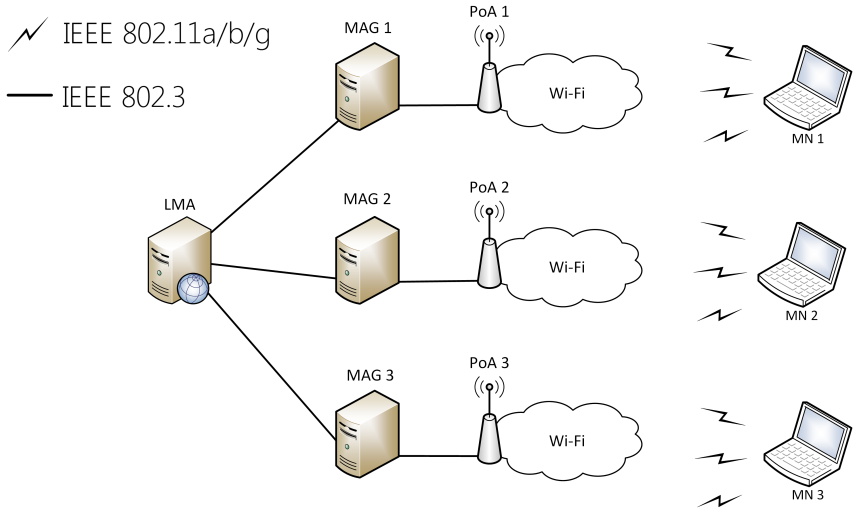


Figure 5.1: Testbed 1

that uses Wi-Fi technology. The configuration of the testbed aims to test the platforms ability to present accurate visualization, with a various number of network entities connected to the network. These entities can be connected to the network using multiple network interfaces. Different numbers of MNs are used in order to evaluate how the platform reacts to the increasing number of users and connected interfaces. In this evaluation, a user is considered a MN that is connected to the network with at least one network interface. When a MN disconnects from the network, it stops being considered a user. Should the same MN connect again to the network, it will once again be considered a user. Since by definition every user is connected to the network by at least one network interface, every time when the MN connects an additional network interface this will increase the amount of information that needs to be presented in the platform. The testbed described aims to evaluate the visualization and interaction abilities. For this purpose, the platform needs to visualize the actual network state, and the events triggered by the platform need to be performed in a transparent way.

The LMA and CN used in the first testbed are both virtual machines. The LMA is connected to three MAGs, using the Ethernet network. Each MAG is connected by Ethernet to the LMA and to a PoA. In order for the visualization process to function properly, the DM entity needs to be deployed on the LMA. Also, for the interaction process, the UTS entity needs to be deployed on the MN. The platform needs to be connected to the network and assigned an IPv6 address corresponding to the LMA network.

The second testbed is depicted in Figure 5.2. It aims to evaluate how the platform

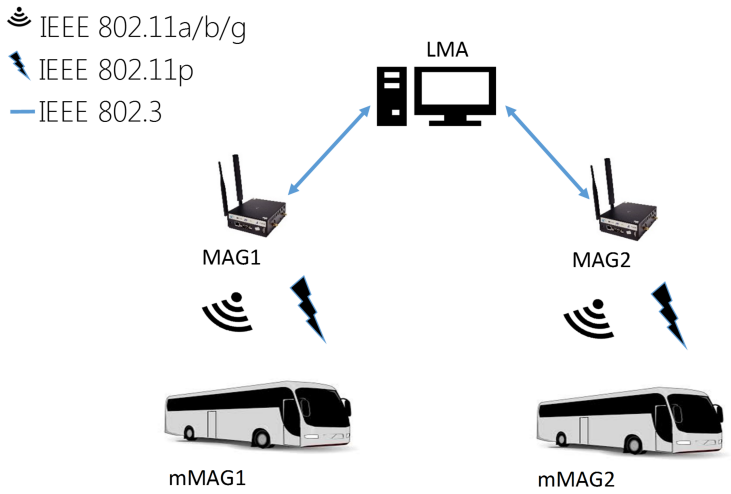


Figure 5.2: Testbed 2

performs in a vehicular network that also contains different access technologies, Wi-Fi and WAVE. The WAVE technology is an access technology developed especially for VANETs and is based on the IEEE 1609.X and IEEE 802.11p standards. If there is multi-hop in the network, the testbed can be used to evaluate how the platform visualizes and interacts with the network nodes in a multi-hop network. The on board units are equipped with a *Connection Manager* that is developed in a parallel MSc Dissertation. This *Connection Manager* chooses the best available access network in its range (based on certain parameters, such as RSS), and automatically connects to that network.

The CN used in the second testbed are virtual machines. The LMA is connected to the MAGs using the Ethernet network. The MAGs can be connected to the mMAGs using both Wi-Fi and WAVE technology. Both the MAGs and the mMAGs are deployed with the *Connection Manager*. Similar to the first testbed, the DM entity needs to be deployed on the LMA, in order for the visualization and interaction process to function properly. Also, the platform needs to be connected to the network and assigned an IPv6 address corresponding to the LMA network.

5.3 Methodologies and Metrics

This section defines the methodologies and metrics that are obtained in the evaluation process. The metrics are the following:

- Mean time required for the visualization.

- Mean time required to trigger network actions.

- Network overhead introduced by the platform.

These metrics provide an objective assessment of the visualization and interaction abilities of the platform. Although the objective is to develop a platform with visualization and interaction capabilities, it is also important to examine the performance of the platform. The aim is to visualize the network and network changes in the shortest time possible. Also, the interaction with the network nodes and triggering of network related actions should not impact the network or decrease its performance. Based on the defined metrics, an objective evaluation of the platform is carried out. Before presenting the obtained results, the metrics are clarified in order for the results to be seen in proper context.

The mean time required for the visualization corresponds to the time required for the platform to visualize, either the entire network, or the result of the triggered network action. The time required to visualize the entire network is the time from the moment when the initial request with the *GETNODES* header is sent, to the moment when the received response is analysed and visualized. The platform also provides interaction capabilities, for which the time required to trigger a network action should be determined. The mean time needed to trigger network related action corresponds to the time from the moment when a network action is triggered in the platform, to the moment when information about this action is received and visualized. The platform communicates with the visualization and interaction platform described in Chapter 3. This communication is done by exchanging messages between the platform and the framework needed to visualize the network or trigger network related actions. These messages introduce overhead in the network which can impact the performance of the network. The size of the overhead introduced in the network is evaluated, since it depends on the specific actions that need to be performed.

The triggering of traffic flows from the CN to the selected MN uses the Distributed Internet Traffic Generator (D-ITG) [44] in order to generate traffic data to the MN. The D-ITG is a traffic generator which can generate UDP and TCP data traffic. In order to analyse the packets of the traffic flows and determine the network overhead, the packets are captured using Wireshark [45], an open-source network analysis tool.

The time metrics are obtained using *shell* scripts on the MNs and a program in Java, developed especially for this purpose. The program is integrated in the platform and registers the time when an action is selected in the platform, or when information is received by the platform. The obtained metrics are analysed using a script in MATLAB [46].

The tests are performed 10 times, and the results obtained in the evaluation process present a 95% confidence interval.

5.4 Experimental Results of the Wireless Fixed Testbed

This section presents the results obtained in a real wireless network in the laboratory environment. The tests are performed using one MN, two MNs and three MNs. For each number of MNs, the evaluation is performed with a different number of network interfaces connected to the network. The MNs evaluated in the laboratory are connected to the network initially with only one network interface, then with two network interfaces, and finally with three network interfaces. The results presented in this section consider the visualization displayed in the platform and the graphs of the obtained metrics.

5.4.1 Scenarios and Visualization

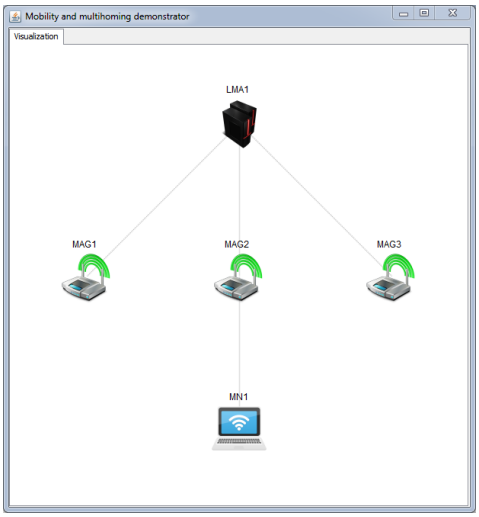


Figure 5.3: Visualization of the testbed in the platform

The primary objective of the platform is to visualize both mobility and multihoming. The first results show the visualization and interaction processes. For this purpose, the platform presentation of these processes will be showed. After the initialization of the platform, an illustration of the current network state is visualized. Figure 5.3 shows the initial testbed visualization. This visualization corresponds to the actual situation in the network, since initially only one MN is connected with only one network interface. Initially,

no traffic flows are present for the connected MN and the percentage of multihoming traffic is not visualized. As shown, the platform is able to correctly display the network entities and the actual network state.

Scanning Available Networks

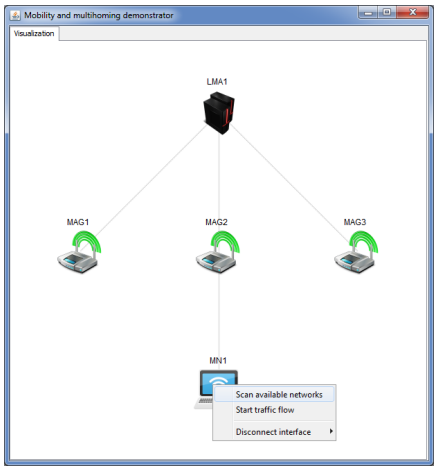


Figure 5.4: Visualization before the scan

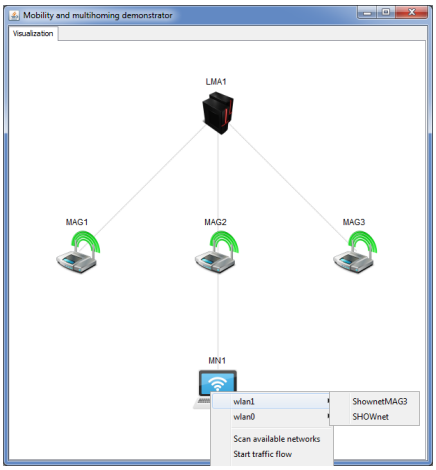


Figure 5.5: Visualization after the scan

The next platform functionality that is evaluated is triggering the scan of available networks. After the option *Scan available networks* is selected in the platform, a request is sent to the corresponding MN to perform the scan. Figure 5.4 and Figure 5.5 present the visualization related to the scan of available networks. The result of the scan process is shown in Figure 5.5. After the scan is performed by the corresponding MN, the available network interfaces are shown together with the Extended Service Set Identifications (ESSIDs) of the networks, that are in the range of that interface. Figure 5.5 shows that the available network interfaces are *wlan0* and *wlan1*. Both network interfaces are in the range of the same networks: *SHOWnet* and *ShownetMAG3*. These results correspond to the actual state of the network. Figure 5.4 and Figure 5.5 show that, for every network interface that is not connected, a scan is performed and ESSIDs of the networks that are in the range of that interface are presented.

Connecting a Network Interface

Triggering the *Scan of available networks* is a prerequisite in order to trigger the *Connect network interface* event. After the results of the scan process are visualized, the option

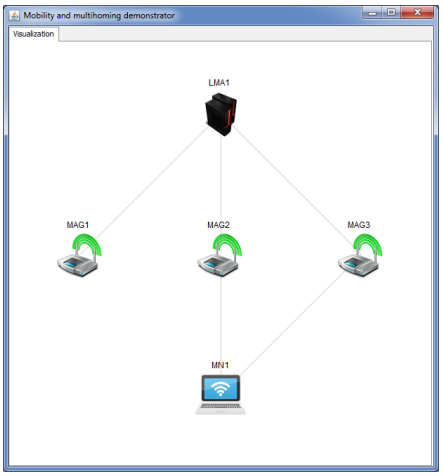


Figure 5.6: Visualization after triggering the connect action

to connect a network interface can be selected. Using the platform, the interface *wlan1* is selected to connect to the network *ShownetMAG3*. Figure 5.6 shows the result of this process. As shown, the network interface *wlan1* connected to the *ShownetMAG3* network. This demonstrates the platforms ability to trigger the action of connecting a network interface to a desired network.

Disconnecting a Network Interface

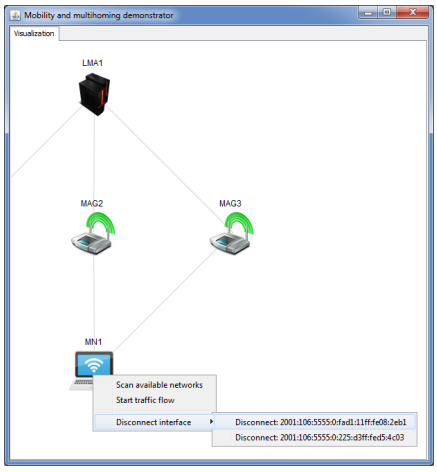


Figure 5.7: Visualization before disconnect

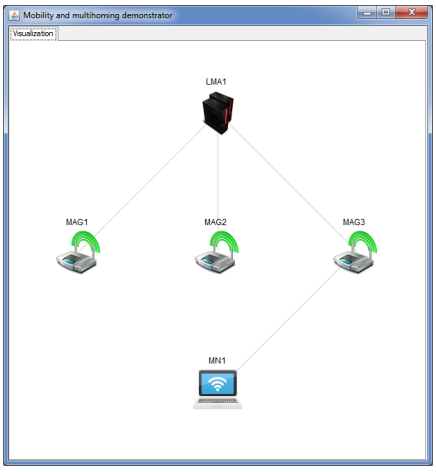


Figure 5.8: Visualization after disconnect

In order to trigger *Disconnect of a network interface* in the platform, a MN needs to be selected together with the network interface we wish to disconnect. Figure 5.7

shows the triggering of the disconnect action, and Figure 5.8 shows the result of the disconnect action. Figure 5.7 shows that a network interface with the IPv6 address *2001:106:5555:0:fad1:11ff:fe08:2eb1* is selected to be disconnected. The visualization shown in Figure 5.8 demonstrates that the interface is successfully disconnected. This test proves that the platform is capable of both triggering the action of disconnecting a network interface and visualizing the result of this action.

Starting Traffic Flows

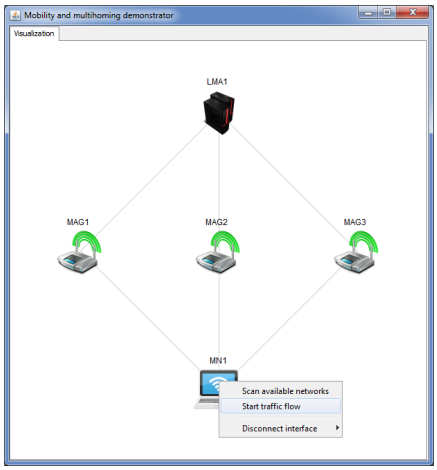


Figure 5.9: Selecting start traffic flow

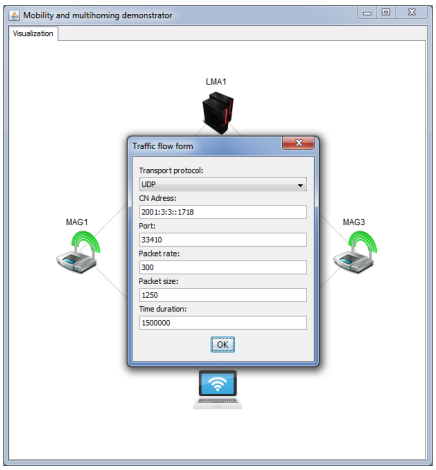


Figure 5.10: Input of the parameters

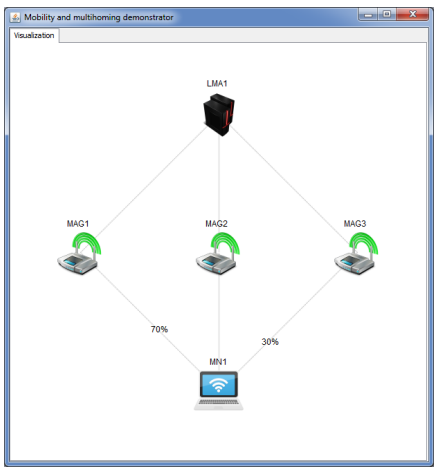


Figure 5.11: Visualizing the traffic flow

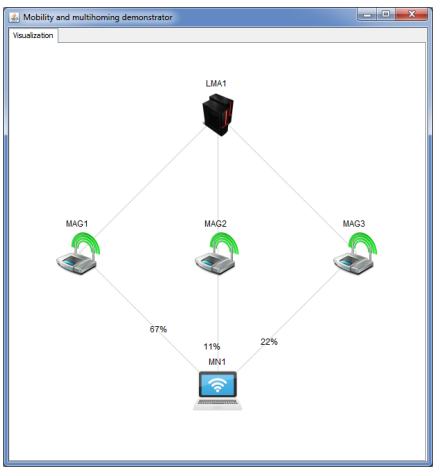


Figure 5.12: Visualization of multihoming

Triggering the start of traffic flows requires several steps to be performed. First, the

option *Start traffic flows* needs to be selected in the platform. Then, the parameters regarding the flow need to be entered. Figure 5.9 and Figure 5.10 show how this process is visualized in the platform. After the traffic flow is started from the CN to the MN, the platform will visualize the percentage of traffic located on each network link of the MN. Figure 5.11 shows that, when the flow is started, most of the traffic is routed through MAG1. Figure 5.12 shows the second visualization, based on which we conclude that the multihoming rule has been applied and the optimal percentage of traffic data is routed through each MAG. The percentage of traffic routed through each MAG is based on several characteristics. Figure 5.12 shows that the largest amount of data can be sent through MAG1.

5.4.2 Mean Time Required for Visualization

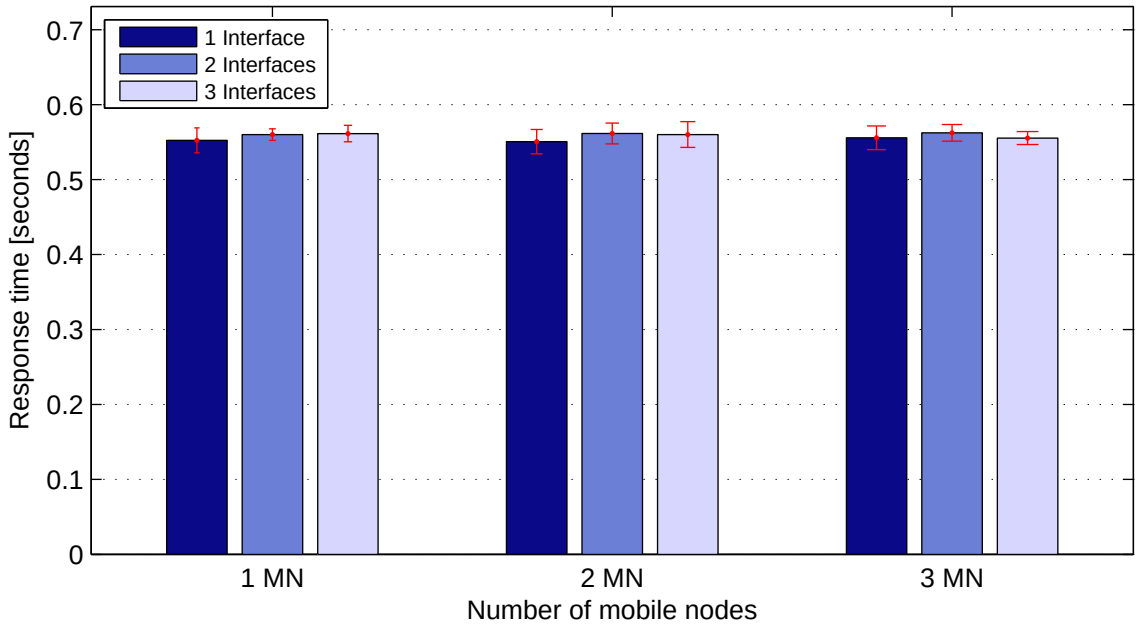


Figure 5.13: Time required for the initial visualization

The visualization abilities of the platform require a certain amount of time. Figure 5.13 shows the mean time required for the initial visualization. As shown, the time required for the visualization varies between 0.531 and 0.577 seconds. The time increases with the increase of the network interfaces that are connected, since with the increase of network interfaces, more information is received by the platform and needs to be processed and

visualized. Figure 5.13 shows that the time does not increase when more MN are connected to the network. This shows that the platform is able to visualize the network with increasing number of nodes without significantly increasing the time required for the visualization.

5.4.3 Mean Time Required to Trigger Network Related Actions

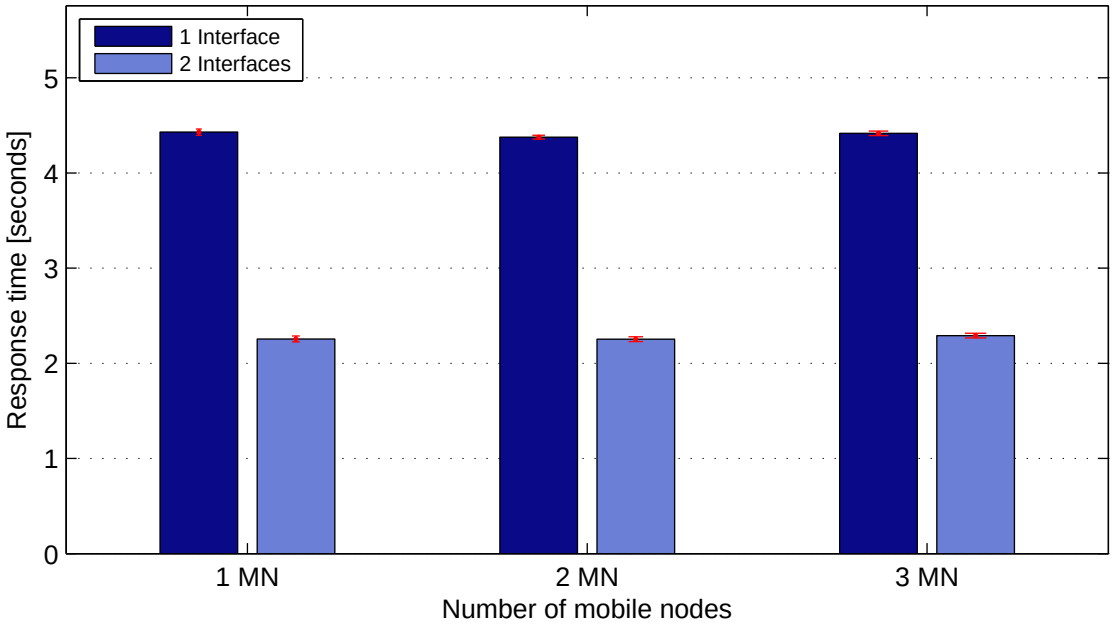


Figure 5.14: Time required to trigger the scan of available networks

The time required to trigger network related actions needs to be evaluated regarding the type of the actions that are triggered. Figure 5.14 shows the time required for the scan of available networks. The time is shown for the situation when one, two and three MNs are connected to the network. The evaluation is not performed for the case when the MN is connected to the network with three network interfaces. This corresponds to the situation when all network interfaces of the MN are connected, and the scan of available networks is only performed when the MN has at least one network interface that is not connected. The greatest time of 4.504 seconds corresponds to the situation when the MN is connected with one network interface and performs the scan of available networks. In this situation, the scan is performed for two network interfaces and the results contain the largest amount of information. However, since only one network interface is connected, the MN is not using multihoming. This time decreases when the MN has two interfaces

connected. In this situation, when two network interfaces are connected, the time varies between 2.256 and 2.442 seconds. Figure 5.14 shows that the time required for the scan of available networks does not increase significantly with the increase of the number of MNs connected to the network.

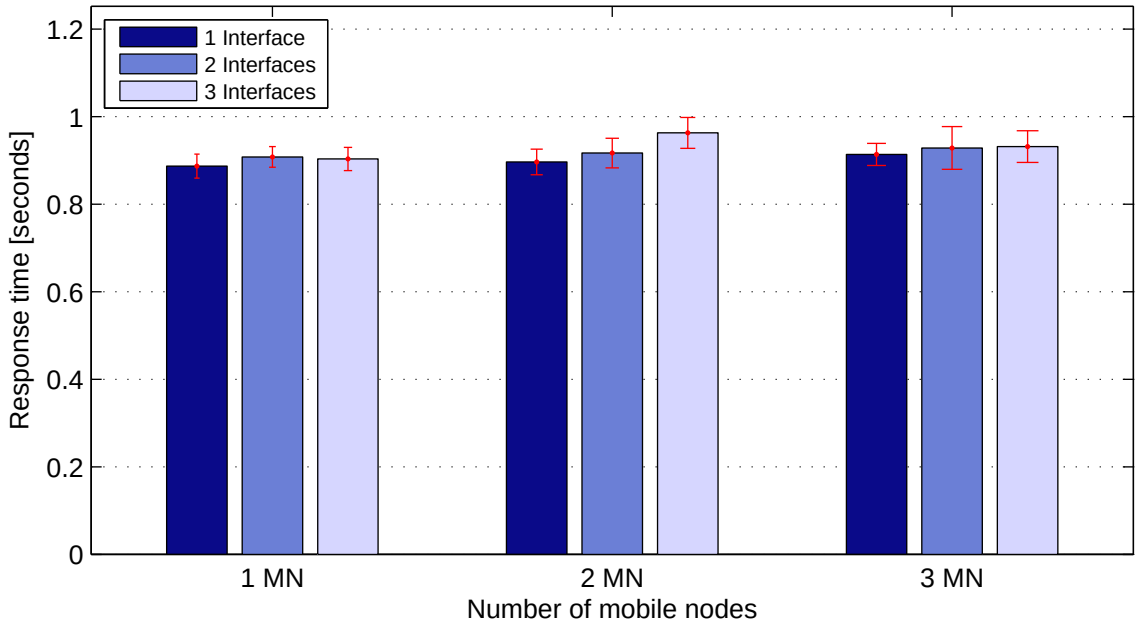


Figure 5.15: Time required to trigger the disconnect action

Figure 5.16 and Figure 5.15 show the time required to connect and disconnect a network interface. The connect action is not evaluated for the situation when the MN is connected to the network with three network interfaces. In this situation, all network interfaces of the MN are already connected, and no additional interfaces can be connected. As we can see, the time varies for both actions depending on the number of MNs and network interfaces connected. An observation is made that connecting a network interface requires a larger amount of time than disconnecting a network interface. The time required to disconnect a network interface varies between 0.879 and 1.047 seconds, and the time required to connect a network interface varies between 1.609 and 1.811 seconds. The reason why disconnecting a network interface takes less time than connecting a network interface is the following. The requests sent to trigger the disconnect and connect action are approximately of fixed length; however, the request to connect a network interface is of larger length than the request to disconnect a network interface. The disconnect request contains only the address of the interface which needs to disconnect, opposite to the connect request, which contains the

address of the interface that needs to connect and the ESSID of the network to which the interface should connect. Since the connect request is of larger length, more time will be required to extract information from that request and execute the corresponding action.

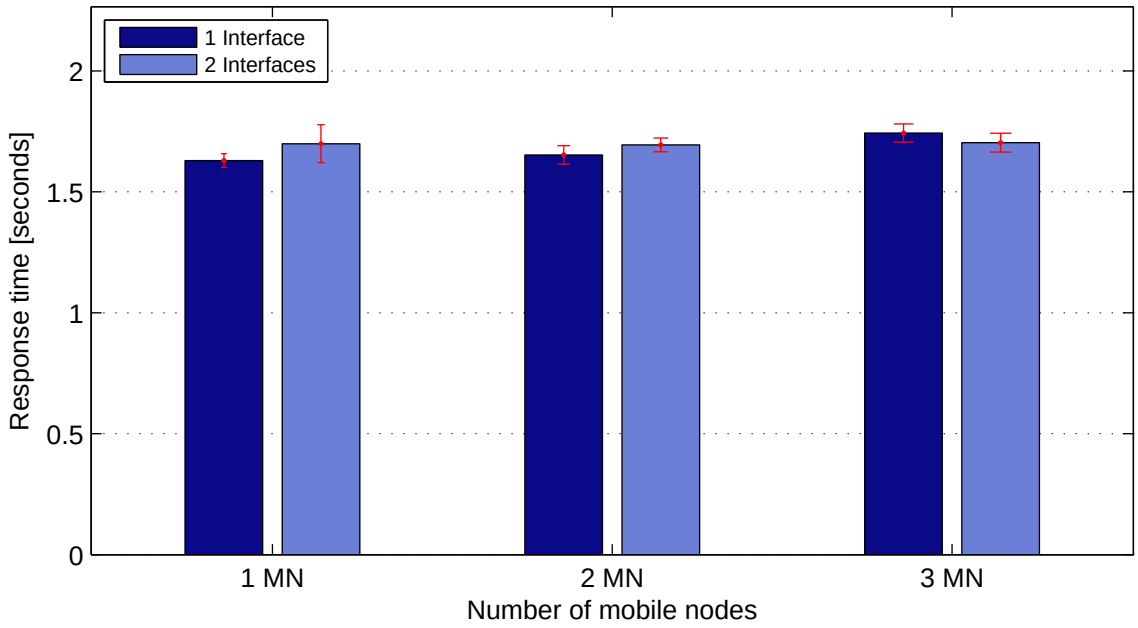


Figure 5.16: Time required to trigger the connect action

5.4.4 Network Overhead

The communication between the platform and the visualization and interaction framework introduces overhead in the network. The most significant overhead corresponds to the largest message sent to the platform. That message is the response to the initial request and contains information about the entire network. Figure 5.17 shows the network overhead of this message depending on the number of MNs present in the network and the number of network interfaces connected. The largest network overhead is 1054 bytes/event and corresponds to the situation when three MNs are present in the network and each MN is connected using three network interfaces. The smallest network overhead of 282/event bytes corresponds to the situation when one MN is connected with only one network interface. Figure 5.17 shows that the overhead increases with the number of MNs in the network and the number of interfaces connected. This is due to the fact that, when more MNs or interfaces are connected, more information is required to be sent and the size of the

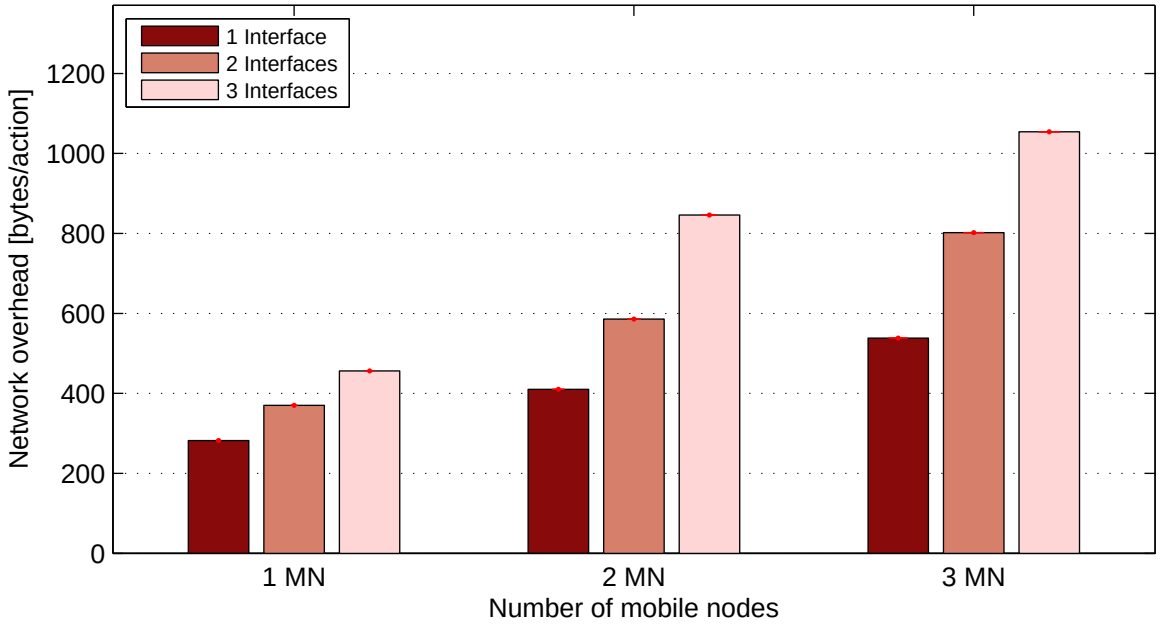


Figure 5.17: Network overhead of information about the entire network

overhead increases. Based on the results shown in Figure 5.17, the number of connected interfaces has a larger impact on the length of the network overhead than the number of connected MNs. This is because information about the network interfaces is larger than the information about the MNs. The information about the network interface includes both the identifier of that interface and the identifier of the MN to which that interface is connected.

Figure 5.18 shows the network overhead of the message that contains the results of the scan of available networks. The largest overhead of 324 bytes/action corresponds to the situation when one MN is connected with one network interface, and the smallest overhead of 206 bytes/action to the situation when one MN is connected with two network interfaces. As seen in Figure 5.18, the size of the network overhead decreases with the increase of the number of connected interfaces. This corresponds to the actual situation, since with every connected interface less information is required to be sent.

Figure 5.19 shows the network overhead of the information related to the start of a traffic flow. The overhead increases with the number of network interfaces connected, since with every new interface that connects more information is required to be sent. This overhead does not increase with the increase of MNs connected, since this does not imply that traffic flows are present for those users.

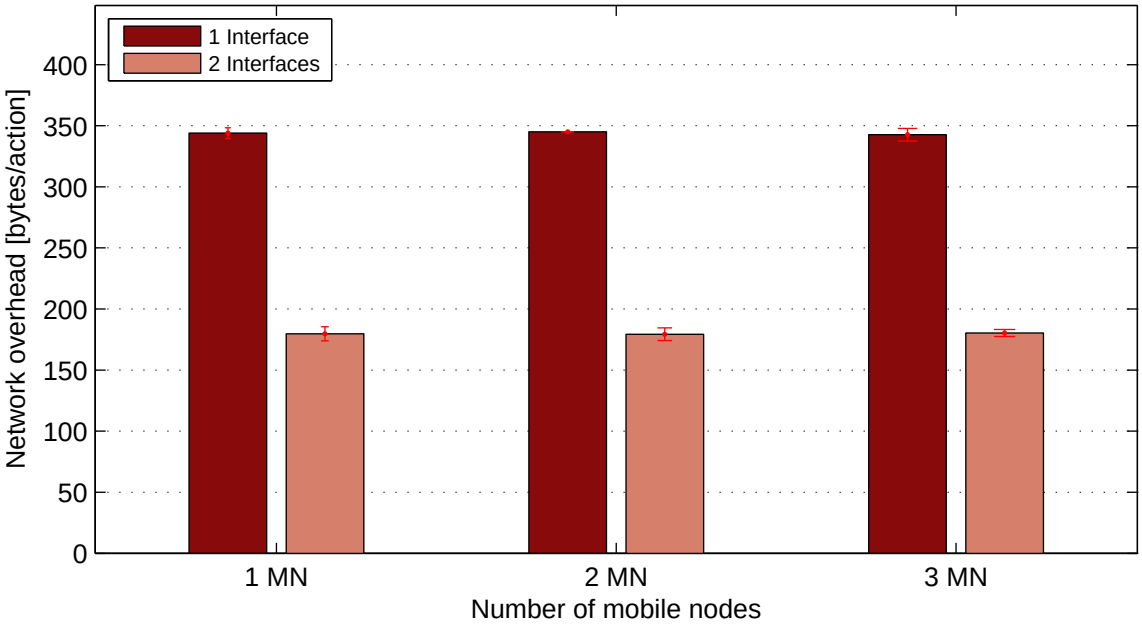


Figure 5.18: Network overhead for the scan response

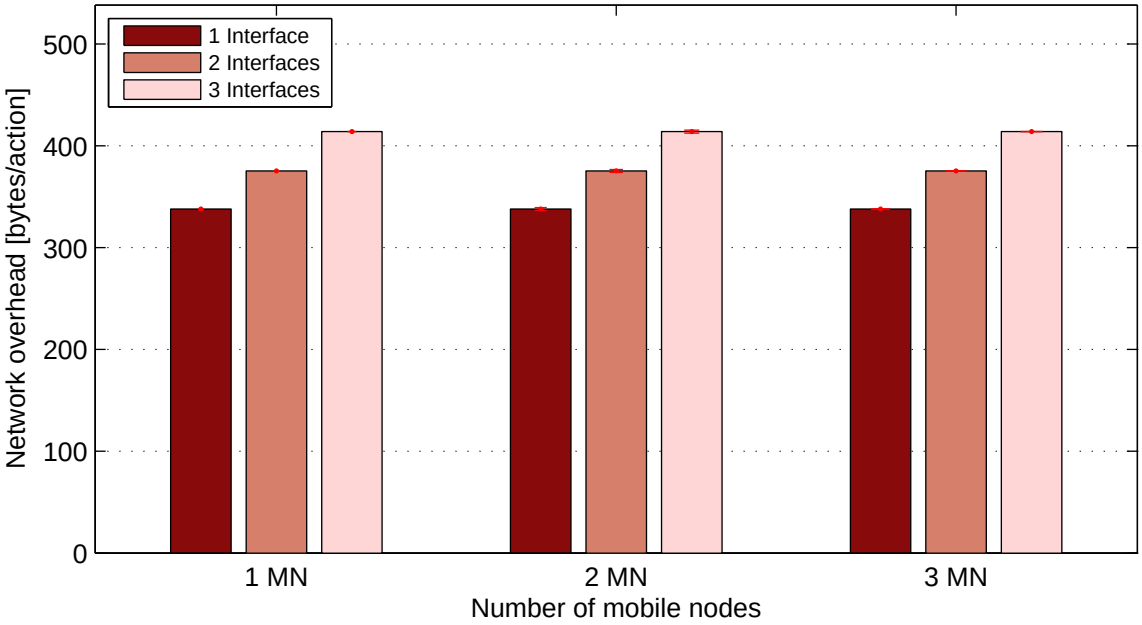


Figure 5.19: Network overhead of information that a flow has started

---

The process of triggering other network related actions and receiving information about changes that occur in network also introduces network overhead. The overhead introduced

in the network because of these actions does not change significantly with the increase or decrease of the number of MNs, or network interfaces. This is because the information sent or received by the platform is mostly of fixed length. The overhead introduced when a request is sent to disconnect a network interface varies between 128 and 130 bytes/action. When a request is sent to connect a network interface, it introduces a overhead in the network from 120 to 124 bytes/action. When a network interface disconnects, the information sent to the platform introduces an overhead of 127 bytes/action, and 145 bytes/action for the case when a network interface connects. The request to perform the scan introduces 144 bytes/action of overhead in the network. When a traffic flow is started, the information about it introduces an overhead of 191 bytes/action, and 262 bytes/action of overhead when the traffic flow stops. Based on these results, we can conclude that triggering these network actions does not introduce significant overhead in the network.

5.5 Experimental Results of the VANET Testbed

This section presents the results obtained in a real VANET in the laboratory environment. The tests are performed using one and two mMAGs. For each number of mMAGs, the evaluation is performed using both Wi-Fi and WAVE technology. Since, both the mMAGs are deployed with the *Connection Manager* developed in a parallel MSc Dissertation, the results presented in this section are regarding the visualization displayed in the platform and the graphs of the overhead, which is introduced in the network due to the visualization. The interaction process is not evaluated, since it would require the removal of the *Connection Manager*. The *Connection Manager* will choose the best available network for each network interface, and then automatically connect to that network. For this reason, the mMAGs that are evaluated in the laboratory are initially connected to the network with both Wi-Fi and WAVE interfaces.

5.5.1 Scenarios and Visualization

Figure 5.20 shows the initial visualization of the VANET testbed. The visualization corresponds to the initial situation in the network, since initially only one mMAG is connected to the network with both Wi-Fi and WAVE interfaces. No traffic flows are present for the connected mMAG and the percentage of multihoming traffic is not visualized. The visualization shown in Figure 5.20 shows the platforms ability to correctly display the network entities and the actual network state, when using different access technologies.

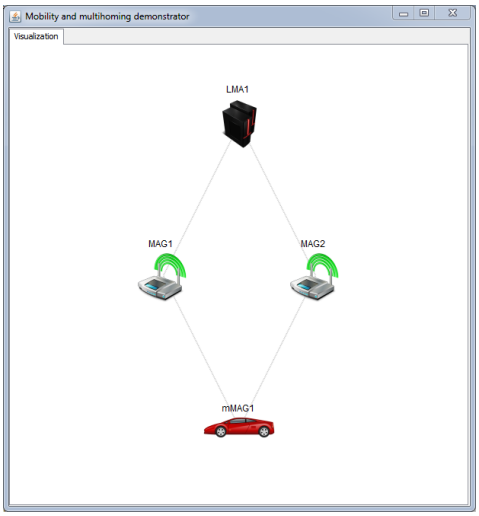


Figure 5.20: Visualization of the VANET testbed in the platform

Connecting a Network Interface

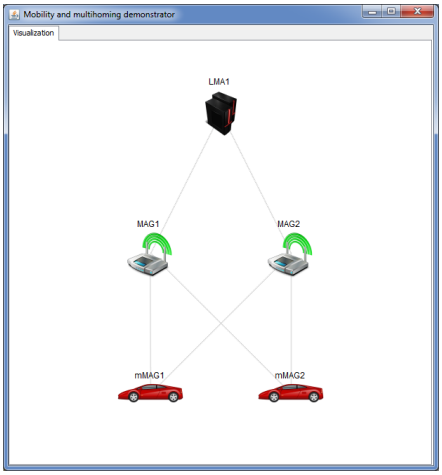


Figure 5.21: Visualization after connecting a network interface

After the initial visualization, the platform needs to be evaluated when a change occurs in the network. The first network action for which the platform is evaluated is the connection of a network interface. For this purpose, a new mMAG is connected to the network. Figure 5.21 shows the visualization after the mMAG has connected to the network using the WAVE interface. The visualization shows the correct number of network entities that are present in the network. However, the visualization shows that the new mMAG is connected to the network using two interfaces, both WAVE and Wi-Fi interfaces.

As explained previously, the mMAGs are deployed with the *Connection Manager*, which automatically connects a network interface to the best available network. Therefore, when the mMAG connects to the network with the WAVE interface, the *Connection Manager* detects that the Wi-Fi interface is not connected, and automatically connects it to the best available network in its range.

Disconnecting a Network Interface

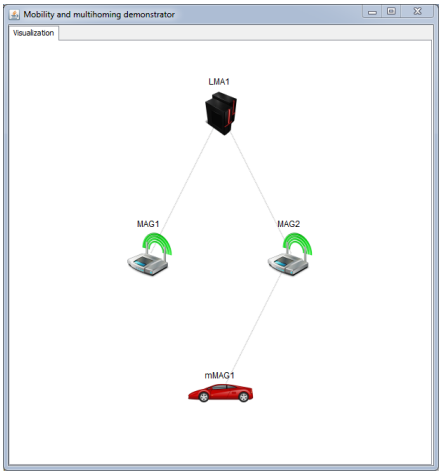


Figure 5.22: Visualization after disconnecting a network interface

The next network action for which the platform is evaluated is the disconnection of a network interface. Figure 5.22 shows the visualization displayed in the platform after the Wi-Fi interface of the mMAG is disconnected from the network. The visualization shows the platforms ability to successfully visualize the disconnection of a network interface, regardless of the type of access technology that is used.

Visualization of Traffic Flows

Since the primary function of the platform is to visualize both mobility and multihoming, it is important to evaluate the platform when a traffic flow is present in the network. Figure 5.23 shows the visualization when a traffic flow is started in the mMAG. As shown, the platform successfully visualizes the percentages of multihoming traffic that correspond to each network interface. Figure 5.23 shows that 80% of total traffic is routed through MAG1 and 20% of traffic is routed through MAG2. Based on the platforms ability to visualize multihoming when both Wi-Fi and WAVE technology are used, we can

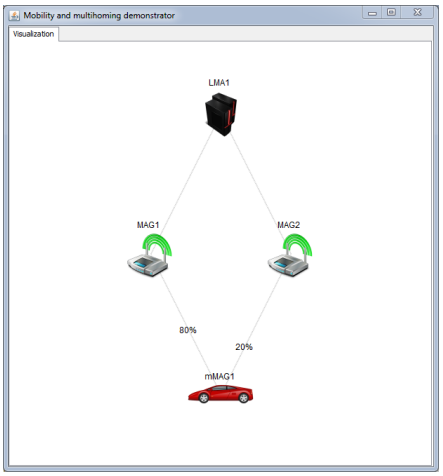


Figure 5.23: Visualization after starting a traffic flow

conclude that the platform is able of visualizing multihoming when the end-user terminal is connected to the network using different access technologies.

5.5.2 Mean Time Required for Visualization

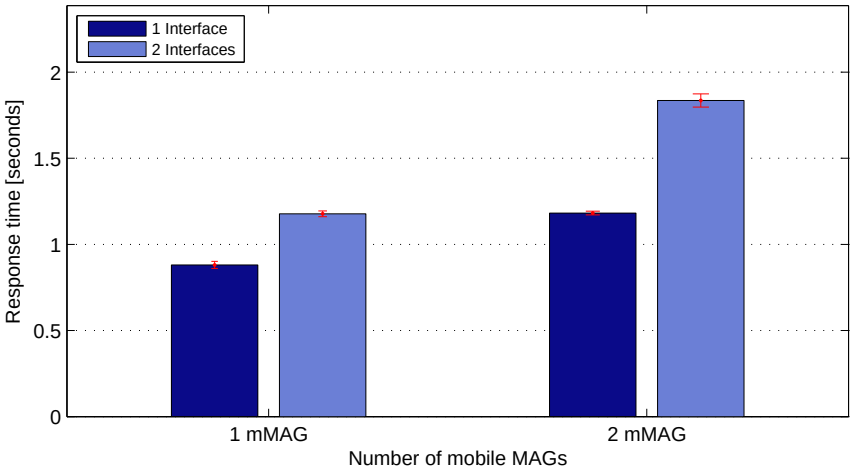


Figure 5.24: Time required for the initial visualization

Figure 5.24 shows the mean time required for the initial visualization obtained in the VANET testbed. As shown, the time required for the initial visualization varies between 0.891 and 1.842 seconds. The time increases with the increase of the number of mMAGs and network interfaces that are connected to the network. This is due to the fact that, with every new network interface or mMAG that connects to the network, more information

is sent to the platform, which needs to be analysed and visualized. This shows that the platform is able to visualize the increasing number of vehicles in the VANET in a reasonable amount of time.

5.5.3 Network Overhead

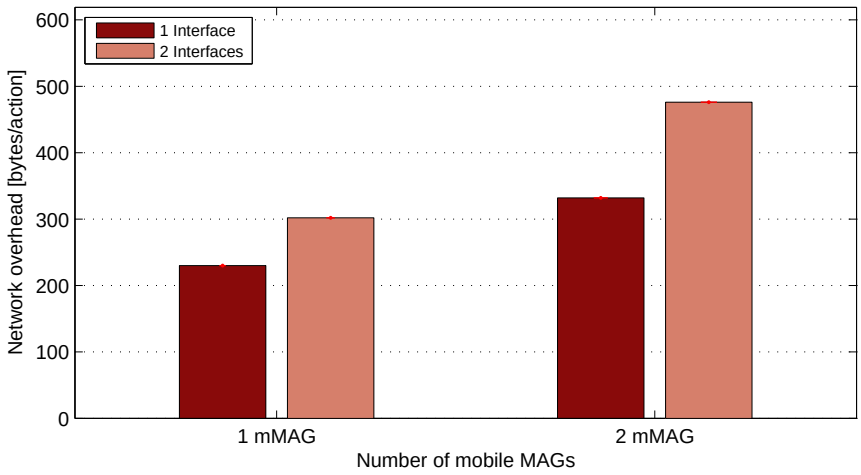


Figure 5.25: Network overhead of information about the entire network

The most significant overhead introduced in the network corresponds to the message that contains information about the entire network. Figure 5.25 shows the network overhead of this message, depending on the number of mMAGs present in the network and the number of network interfaces connected. The largest network overhead is 476 bytes/action and corresponds to the situation when two mMAGs are present in the network, and each mMAG is connected to the network with both WAVE and Wi-Fi interfaces. Figure 5.25 shows that the overhead increases with the number of mMAGs in the network and the number of network interfaces connected. The reason for this is that, when more mMAGs or interfaces are connected to the network, more information is required to be sent to the platform, and the size of the overhead increases. Based on the results shown in Figure 5.25, the number of connected mMAGs has a slightly larger impact on the overhead size, than the number of connected interfaces. The size of the information that is sent about the network interfaces and mMAGs varies depending on the number of interfaces that a mMAG has connected to the network.

5.6 Chapter Considerations

This chapter presented the evaluation of the developed platform. The evaluation was performed taking into account both the visualization and interaction abilities of the platform. The visualization abilities were evaluated using the display in the platform, and the performance of the platform was evaluated based on metrics that are obtained and analysed in this chapter. They indicate the amount of time that is required for the platform to visualize and trigger network related actions, and the overhead that the visualization and interaction introduces in the network.

The evaluation presented in this chapter was performed by deploying two testbeds in the laboratory environment. The first testbed aimed to evaluate the platforms ability to operate in a real wireless network. The second testbed aimed to evaluate how the platforms operates in a real VANET, using equipment developed specially for vehicular networks.

The results obtained from the first testbed demonstrate the platforms ability to properly function in a wireless network. The platform correctly visualizes the network state and is able to interact with the network nodes and trigger network related actions. However, while triggering the action of connecting and disconnecting a network interfaces, it was noticed that the version of the Linux operating system should be 12.10 or greater in order to for the disconnect action to function properly. An improvement could be made by checking the Linux version and performing the disconnection specific for that version. Also, the scan process demonstrated by the platform was evaluated with available Wi-Fi networks which do not require a password. Should a password be needed, then the action of connecting to an available network should also include the option to provide the password required to connect to the desired Wi-Fi network.

The second testbed was evaluated taking into account the visualization and the overhead that the visualization introduces in the network. The interaction abilities of the platform were not evaluated in the VANET, since the On Board Units are deployed with a *Connection Manager* that has automatic interaction capabilities and automatically connects the network interface to the best available network. The evaluation shows that the platform is able to successfully visualize mobility and multihoming in VANETs while the end-user terminal is connected to the network with multiple access technologies.

The results obtained from both testbeds show that the platform is able of visualizing mobility and multihoming in both a wireless fixed network and a VANET. The results indicate that the platform is able to perform visualization of the network and network events in a reasonable amount of time. The overhead introduced in the network is only

significant for a large number of network entities. The results show that the platform introduces a smaller amount of overhead in the VANET than in the wireless fixed network, for the same number of network entities and network interfaces connected. Also, based on the results, the time required for the initial visualization of the wireless fixed network is lower, compared to the time required for the initial visualization of the VANET.

Based on the evaluation performed in this chapter, final conclusions are made and presented in the following chapter.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The objective of this Dissertation was to visualize both mobility and multihoming, and to trigger network related actions. In this work, a platform was developed capable of visualizing mobility and multihoming, as well as interacting with network nodes in order to trigger network related actions. In addition to developing the platform, it was also necessary to modify an existing mobility protocol with multihoming support, and extend it in order to provide visualization and interaction capabilities. For this purpose, a visualization and interaction framework was developed and integrated in the protocol.

While developing the visualization and interaction platform, several requirements needed to be taken into account. The platform was developed to operate in both a small network where changes occur seldom, as well as in a large network that can be highly dynamic. Also, the framework that was integrated in the mobility protocol provides the necessary visualization and interaction capabilities without affecting any other functionality of the protocol that was used. For this purpose, two separate entities were developed that provide interaction and visualization support. The visualization of mobility, multihoming and the interaction with the network nodes is achieved through the communication of the developed platform with the framework that was extended. The information that is exchanged is accurate and introduces as little overhead in the network as possible. Also, the platform to visualize the information and network events in real-time processed the received information in a reasonable amount of time.

In order to provide an objective evaluation of the developed platform, tests were performed in the laboratory environment in both a wireless fixed network and a VANET. In

these tests, metrics were obtained in order to provide an objective evaluation. Based on these tests and the obtained metrics, it was possible to evaluate the developed platform and its performance. Based on the results the following conclusions are made:

- Visualization allows for network data to be presented in a intuitive and interesting way, that is superior to simple textual representation of information.

- The developed platform has the ability to visualize both mobility and multihoming.

- The platform is able to interact with the network nodes and trigger network related actions that were implemented.

- The communication between the developed platform and the framework, that is extended to provide visualization and interaction abilities, introduces overhead in the network. The size of the network overhead is dependent on the type of the action that is performed, but is small and is not expected to include scalability issues.

- The platform requires a certain amount of time for the visualization. The time required is minimal and does not increase significantly with the increase of network entities in the network or connections between those entities. This makes the platform suitable for deployment in large network scenarios with a large number of network entities and connections between those entities.

- The time required for the platform to trigger a network related action is significantly higher than the time needed for the visualization. This time depends on the time that is required by the end-user terminal to execute the commands that correspond to the actions that need to be triggered.

The deployment scenario of the platform can include static networks with only a few network nodes, or highly-dynamic networks with a large number of network nodes. The interaction and visualization capabilities of the developed platform make it a powerful network tool, which can be used to trigger network related actions from the end-user terminal on which it is running. We conclude that the developed platform can be used to visualize mobility and multihoming in a real network in real-time.

6.2 Future Work

This section proposes and discusses possible improvements of the work done in this Dissertation. These are the following:

- **Evaluation of the platform with a large number of users:** evaluate how the platform reacts and visualizes a large number of users. In order to visualize the network entities, information about all users is required by the platform, which in turn introduces overhead in the network.

- **Evaluation of the platform in highly dynamic environments:** evaluate how the platform performs when the disconnect and connect process occur rapidly for multiple users. This scenario should also include the situation when a user connects a interface and disconnects it as soon as the connection is established. The response time of the platform should be evaluated in these situations as well as the visualization of changes that happened.

- **Improving the graphical user interface:** although the existing GUI is very intuitive and provides good user experience, it would be beneficial to add more advanced features. Adding features such as graphs when presenting data, or statistics about the network, would allow better visualization of the relevant content.

- **Distributed gathering of information:** the LMA is involved in both the interaction and the visualization process. To prevent a single point of failure, information about the network should be obtained in a distributed way. In that case, the interaction with the network nodes should also be done without a central entity involved in the triggering process.

- **Implementing a data storage system:** storing the received information into a storage system or database would allow access to that information even after the platform is no longer running. By implementing a database, it would be possible to securely store the received information which could be used by other applications or programs regardless of the platform. The stored information could be used for analysis or even to reconstruct previous network events.

Bibliography

[1] Julie Steele and Noah Iliinsky. *Beutiful Visualization - Looking at Data through the Eyes of Experts*. O'Reilly Media, April 2010.

[2] Adam Perer and Ben Shneiderman. Balancing Systematic and Flexible Exploration of Social Networks. In *Visualization and Computer Graphics, IEEE Transactions on*, volume 12, pages 693–700. IEEE, November 2006.

[3] Kristin Koch et al. How *Much* the Eye Tells the Brain. In *Current Biology*, volume 16, pages 1428–1434. Elsevier, July 2006.

[4] Vikram Dayal. Anscombe's Quartet: Graphs Can Reveal. In *An Introduction to R for Quantitative Economics*, chapter 9, pages 59–63. Springer, March 2015.

[5] Stephen Few. Data Visualization for Human Perception. In *The Encyclopedia of Human-Computer Interaction*. The Interaction Design Foundation, 2nd edition, 2014.

[6] Lei Shi at al. 1.5D Egocentric Dynamic Network Visualization. In *Visualization and Computer Graphics, IEEE Transactions on*, volume 21, pages 624–637. IEEE, May 2015.

[7] Tatiana von Landesberger et al. Visual analysis of large graphs. In *Proceedings of Annual Conference of the European Association for Computer Graphics*, 2010.

[8] Alain Barrat José Ignacio Alvarez-Hamelin and Alessandro Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Advances in Neural Information Processing Systems 18*, pages 41–50. MIT Press, 2006.

[9] Stephen P. Borgatti. NetDraw: Graph visualization software. In *Analytic Technologies*. Lexington, 2002.

[10] Aric Hagberg, Daniel Schult, and Pieter Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science conference (SciPy 2008)*.

[11] Vladimir Batagelj and Andrej Mrvar. Pajek — analysis and visualization of large networks. In *Graph drawing software*. Springer Berlin Heidelberg, 2004.

[12] Vladimir Batagelj and Andrej Mrvar. Pajek-program for large network analysis. In *Connections*, volume 21, pages 47–57, February 1998.

[13] The Gephi Consortium. Gephi. Available: `https://github.com/gephi/gephi.github.io/tree/master/images/screenshots`, 2015.

[14] Mathieu Bastian, Sebastian Heymann, and Mathieu Jacomy. Gephi: an open source software for exploring and manipulating networks. *Proceedings of the Third International ICWSM Conference*, 2009.

[15] Joshua O'Madadhain et al. Analysis and Visualization of Network Data using JUNG. *Journal of Statistical Software*, 10:1–35, 2005.

[16] Yasuhiro Watashiba et al. OpenFlow Network Visualization Software with Flow Control Interface. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 475–477. IEEE, July 2013.

[17] Jerry Clarke, John Vines, and Ken Renard. The Network Interdisciplinary Computing Environment. In *High Performance Computing Modernization Program Users Group Conference (HPCMP-UGC), 2010 DoD*, pages 421–423. IEEE, June 2010.

[18] Yu Yang et al. SNAMP: A Multi-sniffer and Multi-view Visualization Platform for Wireless Sensor Networks. In *Industrial Electronics and Applications, 2006 1ST IEEE Conference on*, pages 1–4. IEEE, May 2006.

[19] Zhuming Bi. An integrated environment for visualization of distributed wireless sensor networks. In *Control and Automation (ICCA), 2013 10th IEEE International Conference on*, pages 312–317. IEEE, June 2013.

[20] Emmanouil Spanakis et al. Real-time graph visualization tool for vehicular ad-hoc networks: (VIVAGr: VIsualization tool of Vanet graphs in real-time). In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 729–734. IEEE, July 2011.

[21] Barberis C. et al. A customizable visualization framework for VANET application design and development. In *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, pages 569–570. IEEE, January 2011.

[22] Wei Zeng et al. Visualizing Mobility of Public Transportation System. In *Visualization and Computer Graphics, IEEE Transactions on*, volume 20, pages 1833–1842. IEEE, December 2014.

[23] Akihiro Inoue, Yuma Takahashi, and Akio Koyama. MANET Viewer III: 3D Visualization System for Mobile Ad Hoc Networks. In *Network-Based Information Systems (NBiS), 2013 16th International Conference on*, pages 178–185. IEEE, September 2013.

[24] Shohei Sato, Akio Koyama, and Leonard Barolli. MANET-Viewer II: A Visualization System for Visualizing Packet Flow in Mobile Ad-hoc Networks. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 549–554. IEEE, MArch 2011.

[25] Fares Benayoune and Luigi Lancieri. MoViTo: a Generic Visualization Tool for Mobility Analysis. In *Wireless Communication Systems, 2005. 2nd International Symposium on*, pages 356–360. IEEE, September 2005.

[26] Lei Shi et al. 1.5D Egocentric Dynamic Network Visualization . In *Visualization and Computer Graphics, IEEE Transactions on*, volume 21, pages 624–637. IEEE, March 2015.

[27] Jianli Pan et al. MILSA: A New Evolutionary Architecture for Scalability, Mobility, and Multihoming in the Future Internet. *Selected Areas in Communications, IEEE Journal on*, 28(8):1344–1362, October 2010.

[28] Diogo Miguel Augusto Lopes. Acesso à internet com handover de veículos através de gateways móveis. Master's thesis, University of Aveiro, 2013.

[29] C. Perkins, D. Johnson, and J. Arkko. Mobility Support in IPv6. RFC 6275, IETF, July 2011.

[30] C. Perkins. Mobility Support in IPv4, Revised. RFC 5944, IETF, November 2010.

[31] C. Perkins. IP Mobility Support for IPv4, Revised. RFC 5944, IETF, November 2010.

[32] Fábio Cristiano Outeiro de Aleluia Martins. Separação de identificação e localização para mobilidade de veículos. Master's thesis, University of Aveiro, 2014.

[33] Tetsuya Arita and Fumio Teraoka. PNEMO: A network-based localized mobility management protocol for mobile networks. In *Ubiquitous and Future Networks (ICUFN), 2011 Third International Conference on*, pages 168–173. IEEE, June 2011.

[34] S. Gundavelli et al. Proxy Mobile IPv6. RFC 5213, IETF, August 2008.

[35] S. Jeon, B. Sarikaya, and R. L. Aguiar. Network Mobility Support using Mobile MAG in Proxy Mobile IPv6 Domain. draft sijeon netext mmag pmip 00, IETF, April 2013.

[36] Jun Bi, Ping Hu, and Lizhong Xie. Site Multihoming: Practices, Mechanisms and Perspective. In *Future Generation Communication and Networking (FGCN 2007)*, volume 1, pages 535–540. IEEE, December 2007.

[37] Nelson Capela and Susana Sargento. Multihoming and network coding: A new approach to optimize the network performance. In *Computer Networks*. Elsevier, 2014.

[38] Allen L. Ramaboli, Olabisi E. Falowo, and Anthony H. Chan. Bandwidth aggregation in heterogeneous wireless networks: A survey of current approaches and issues. In *Journal of Network and Computer Applications*, volume 35, pages 1674–1690. Elsevier, November 2012.

[39] T. Daniel Wallace and Abdallah Shami. A Review of Multihoming Issues Using the Stream Control Transmission Protocol. In *Communications Surveys and Tutorials, IEEE*, volume 14, pages 565–578, May 2012.

[40] Alberto García-Martínez and Marcelo Bagnulo. The Shim6 Architecture for IPv6 Multihoming. *Communications Magazine, IEEE*, 48(9):152–157, September 2010.

[41] Open Air Interface Proxy Mobile IPv6. Mobile Communications Department at EURECOM in Sophia-Antipolis, France. Available: `http://www.openairinterface.org/`.

[42] Li Mingzhe, Mark Claypool, and Robert Kinicki. WBest: A bandwidth estimation tool for IEEE 802.11 wireless networks. *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*, pages 374–381, October 2008.

[43] Joshua O'Madadhain et al. JUNG - Java Universal Network/Graph version 2.0.1. Available: `http://jung.sourceforge.net/`, 2005.

[44] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks (Elsevier)*, 56(15):3531–3547, 2012.

[45] Angela Orebaughet et al. *Wireshark & Ethereal Network Protocol Analyzer Toolkit (Jay Beale's Open Source Security)*. Syngress Publishing, 2006.

[46] MATLAB. *version 8.2 (R2013b)*. The MathWorks Inc., Natick, Massachusetts, 2010.