Cognitive Radio Experimentation World

Project Deliverable D7.5.4

Showcase of experiment ready (Demonstrator)

Contractual date of delivery:	31-03-14
Actual date of delivery:	18-04-14
Beneficiaries:	WINGS
Lead beneficiary:	WINGS
Authors:	Dimitrios Karvounas, Andreas Georgakopoulos, Evangelia Tzifa, Katerina Demesticha (WINGS)
Reviewers:	Kostas Tsagkaris (WINGS)
Workpackage:	WP7 – External Test Cases
Estimated person months:	2
Nature:	R
Dissemination level:	PU
Version	1.1

Abstract: The deliverable analyses the showcase for experiment-based validation of control channels for cognitive radio systems. Elaboration on experiment setup and procedures for the realization of the showcase is being conducted.

Keywords: Control channels, cognitive radio systems, coverage expansion, capacity expansion, signalling evaluation, performance evaluation, D2D constructs, mesh networks

REVISION HISTORY

Version	Date	Author	Description
1.0	13/04/2014	WINGS	First version of document
1.1	18/04/2014	IMEC	Minor revision, final version

Contents

List of Figures

Executive Summary

This deliverable considers the concept of device-to-device (D2D) communications for the resolutions of persistent issues of mobile networks. Moreover, the impact of mobility is addressed by using a mobile robot of the w.iLab-t testbed. Certain procedures for the utilization of various elements of the testbed and information on the developed software provided by WINGS are included in this deliverable.

Table of contents

1 Experimental Environment

In this experiment, a Mobile Node (MN) that is moving around an area covered by many wireless Access Points (APs) is considered. The APs are registered (i.e. they send messages with their capabilities) to the Network Management Entity (NME) that has knowledge of the network topology. The MN is served by a specific AP and thus, it can have access to various applications (ping, video streaming, file transfer). Due to the fact that other APs are also transmitting and interfere to the received signal of the MN, the MN may experience low Quality of Service (QoS) or even go out of the coverage of the AP. Therefore, the proposed solution is to maintain the connectivity (or to improve the QoS) of the MN by exploiting the opportunities available because of the presence of neighboring (fixed) nodes. Specifically, a path is identified that leads to the AP through the exchange of control messages among the nodes that contain information regarding their status and capabilities and the most suitable nodes (according to a fitness value) create a wireless mesh network. This scenario is implemented in the w.iLab-t testbed, where a mobile robot acts as the MN, while the other nodes of the testbed act as the neighboring nodes and the APs. Each node is enhanced with Java-based agents that add functionalities according to the role that each node plays (i.e. MN, fixed node, AP, NME) and also implements the control channel through which the control messages are exchanged. In the MN, the received signal quality from the AP is monitored and if it drops below a certain threshold or if the MN loses its connectivity, the procedure of the identification of a path to the AP through the neighboring nodes is initiated in order to create the mesh network. When the mesh network is established, the applications mentioned above are executed in the MN in order to measure their performance (in terms of throughput, delay, jitter, packet drop probability) in relation with the number of hops and the distance from the node that executes the traffic generation (i.e. the AP). Therefore in the AP a media and a file server is installed, as well as a Java-based traffic generator that various parameters can be configured (distribution of the generated packets, transport protocol (TCP/UDP), packet size, transmission interval, number of sessions, etc.). The installed Java software enables also the real-time presentation of these statistics.

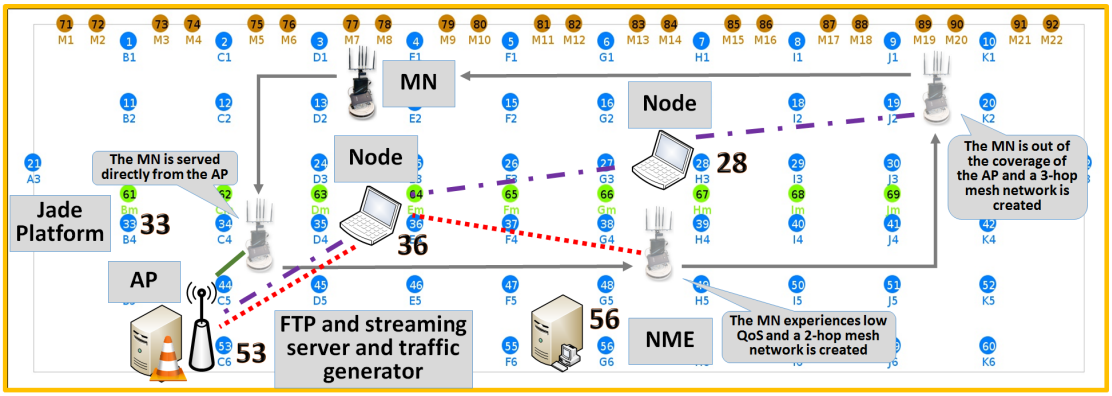

Figure 1 – Experimental Environment (in iMinds)

To wrap-up, the main objective of the experiment is to evaluate the impact of mobility in the context of device-to-device (D2D) communications. In this sense, we use one of the mobile robots of the testbed in order to move them around the playground and thus evaluate the impact to communication quality (e.g., achieved throughput, packet drop probability, delay etc.). The Chapter 2 describes the working demonstration.

2 Working Demonstration

In order to test the 802.11s (mesh) network's performance under various traffic conditions a custom configurable client – server traffic generator was developed. Since the mesh network maintains the IP address even after the network topology shifts (i.e. when the robot node is moving) the upper layers (TCP, UDP) maintain their connection as long as the Data link layer supports it. The application uses java TCP and UDP connection for the network traffic and has a rich graphical interface that displays many measureable aspects of the test bed context. These are the results of the wireless scan process that is being performed every a given interval, the calculation of the interference in each channel and data plots of the interface Throughput and QoS parameters. This framework acts as a host for the initial experiment scenario which is to monitor the quality of the WiFi (Infrastructure mode) network and, if it drops its quality below a given threshold, begin with the establishment of the mesh network.

This application combines information acquired from the wireless network drivers of the Linux OS, multithreaded network services and Enterprise java graphics for the best illustration of the output data. The acquisition of the linux kernel information is performed by invoking the "iwlist" binary via the Apache commons library. The result is then being stored in custom Wireless Network objects that are being passed to the application to update the data structures that are being translated to graphics (tables, plots etc). Another approach is also provided that uses a direct communication between Java runtime and the linux driver with the assistance of the JNI API. The communication protocol between the client and the server is a custom protocol that is based on signaling messages that control the session over a constant TCP connection (through the constant Access Point). This helps to coordinate the UDP transmission between the two peers with the increased robustness of the dedicated link. The UDP transmission's parameters (packet rate, packet size) are passed on the UDP demon at the initialization procedure and remain constant until the finalization message is received by the Server (Client-initiated termination).

The Server is a TCP multithreaded server that generates one thread per data session. The initiation of the data transmission is performed by the client after he has reserved an UDP Socket resource and communicates the UDP port to the server through the TCP connection. The Server given the knowledge of the destination IP and Port of the transmission begins initialization of the session thread. Additional information is also included in the "START" message so as to the packet interval (packet rate) , the packet size (in bytes) the packet resizing policy (Constant packet size, random Gaussian packet size) and also the number of packets that will be transmit. The server checks its internal capacity (which is hard-limited to 100 Sessions/Runtime) and replies with either a success or a failed message including also the unique identifier of the session (Integer [1-100]). Then the transmission of UDP packets is being performed until specified otherwise by the Client. The Client, at the other end of the transmission, begins receiving the UDP packets at the specified UDP socket resource and delivers them to the session layer when information (delay, throughput) is being calculated and stored in local variables. When the client realizes that the preset number of packets has been completed, sends a session termination message through the TCP channel to the server. The server then stops the UDP transmission, cleans up the resources and replies with an acknowledgement message that includes the number of sent packets for this session. With this data at hand, the session application calculates the packet drop probability of the session and completes the higher layer link report by invoking the session completed callback method.

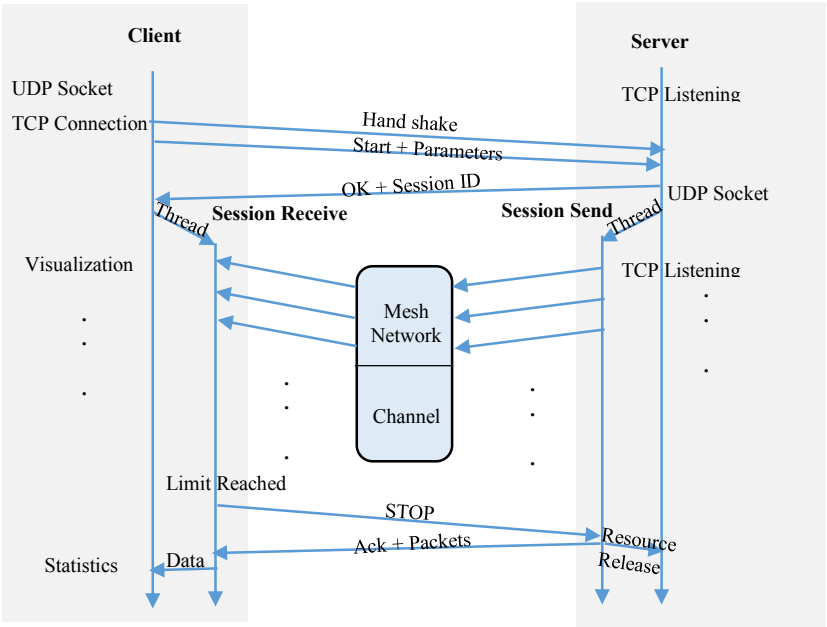

Figure 2 – Protocol Sequence Diagram

2.1 Application User Manual

In this chapter we will enumerate the functionalities of each application and give detailed information about its execution and prerequisites. As this application has two ends, the client side and the server side, we will explain the installation and execution procedure in both nodes.

2.1.1 Prerequisites

Server: The server application is a JavaFX graphical application with network usage requirements. The subnet in which it is executed must be a static IP network with no NAT or Firewall at any intermediate node. The required java runtime environment for its execution is JRE7u51 and later in order to utilize the JavaFX graphical features.

Client: The client application is a JavaFX graphical application with network usage and also script execution capabilities. For this reason root user capabilities are required for its execution and also static IP network, no NAT in any intermediate node and JRE7u51 virtual machine. The Operating system must be Linux based with kernel 2.6.x or later and preferably Ubuntu 8+. Also this application requires at least one functional wireless card (optimally two) in order to simultaneously monitor the WiFi channel and also transmit the UDP packets via the other wireless card.

2.2 Server User Manual

The server application has a graphical user interface with limited input parameters and a detailed session overview environment. After running the application (from its runnable .jar file) a user might be prompt to grant network / firewall permissions. For its optimal operation, the network rights must be granted in order to act as the TCP listening entity for the network. After that a graphical window will be revealed with 3 input fields of which only the last is editable (the other two are marked as deprecated). The editable field is the TCP listening port field at which the user must put the port number that will be listening for incoming connections. After specifying the port in that field the user then presses the “START” button and the scene is being transformed to a session overview panel. In this window an observer can notice all the running UDP sessions, their respective packet counts, client IP’s and session’s id along with a pie diagram of the server utilization. Closing the application will result in immediate resource cleanup via the java runtime environment garbage collection and the end of all sessions. The server application is designed to run in the one end of the mesh network

chain in order to monitor the effects of the data link layer transmission to the packet delivery rate at higher layers.

2.3 Client User Manual

The client application is a little more elaborate than the Server application because it is the center of output data gathering and visualizing. Running this application under Linux environment requires root privileges in order to execute the bash scripts as super user. After running the application (from its runnable .jar file) you will be prompt to enter the interface name that you wish to use as the scanning interface. It is important to put a valid interface name (i.e. wifi1, mesh0) in order to acquire its local ip which will be required later. Also, it must be noted that using the same interface for mesh network transmission and scanning will disable its transmission functionalities and cause a 100% packet drop performance so it is crucial that the "continuous refresh" option is disabled if such a scenario is in effect. After that the main window of the application will open with three visible areas : The data table, that continuously displays the latest wireless network information as it derives from the continuous wireless scanning from the selected wireless interface, the status bar (bottom) that will report for any problems in any program functions (error in session initiation , runtime exceptions etc) and finally the options bar that uses JMenu to navigate through the program's capabilities. There are four menu's each containing different commands:

- The File menu where the program gives the user the ability to begin capturing the wireless scan data in the form of serializable objects and write in a save file. Closing the program while that option is selected will write all the captured objects in that file and it can be accessed by other software for meta-analytics.
- The Action menu, where all the basic actions that can be performed with the program are stored. The available action are:
    - Monitor throughput (downlink, uplink) from a given interface.
    - Monitor Channel Quality [0-1] of any selected SSID's from the table.
    - Monitor SINR [varies] of any selected SSID's from the table.
    - Enabling the Monitor Agent, this agent changes the program to an environment where the user can set a quality threshold value, and the agent automatically monitors the selected (Connected) SSID in order to perform the mesh network initialization function and move to the mesh network.
    - Enabling /disabling the continuous refresh function that is executing the bash script (iwlist scan).
    - The Start Experiment command which prompts the UDP Session wizard in order to begin a number of UDP Sessions, bundled in the Experiment Wrapper. We will analyze this functionality later on.
- The View menu, where toggle-buttons are being placed in order to specify what context will be displayed in the main window. This context may be:
    - The Wireless Scan results table (as mentioned before).
    - The Interference table which displays the interference (dBm) in each WiFi channel.
    - The Experimental results view which has a number of graphical plots for each different QoS parameter measured during and after the completion of UDP Experiment.
- Finally the "about" menu where credits for the Software developers is being displayed.

The basic operation of the software is the Experiment operation, where the user begins the traffic generation with the remote Server and then measures the results of the experiment. This occurs upon selection of the "Start Experiment" action in the Actions menu. The experiment function reveals two new graphical elements of the program, the right panel which acts as an experiment description input form with all the parameters that are required to begin the experiment and the left panel which displays detailed information about each active sessions such as its completion percentage (in the form of a progress bar), the session name and also some runtime control buttons such as session

pause and session stop button. It must be noted that the left panel will not be displayed when no sessions are active. The input parameters of the session are basically all the information needed for the successful execution of a UDP Session with the server. They must be correctly completed by the user in order to perform various experiments that will display the QoS profile of the link. These fields are analyzed below:

1. Server destination IP: Here the user must specify the IP which will be used for the TCP signaling channel establishment. If the Client software is executed in the robot node, it must be noted that the Ethernet interface will be displayed during the mobility session, therefore the best link for sustaining the signaling channel would be the Wi-Fi access point (which is common for both the server and the client).
2. Local UDP IP: Here the user must specify the IP that is designated to the client node during the mesh network generation. This will solve the interface routing issue and will guide all the traffic via the mesh network (although alternate path exists via the Ethernet or the Wi-Fi AP).
3. Number of packets for each session: Here the user must specify the number of packets that will lead, upon completion, to the termination of the UDP Session. It must be noted that selecting zero (0) as the number of packets is preset to act as a wildcard for infinite packet transmission that will be interrupted only by the means of the STOP button in the left panel of the graphical user interface.
4. Number of sessions for this experiment: Here the user must specify the number of sessions that he wishes to establish with the remote server. Note that the session generation rate is 1 second + any session establishment delay from 0 to 2000 ms (2 seconds TCP timeout).
5. Packet size type: This enumeration gives the user a choice between CBR (Constant bit rate) and VBR (Variant Bit Rate) which is emulated by the means of packet payload size randomization. This software uses the built-in Java Gauss distribution (through the java.util.Random class) in order to generate random traffic with statistical mean and variation proportional to the selected packet size.
6. Packet interval (milliseconds): This is required to program the repetitive task of the server transmission to send a packet every <selected> milliseconds. Reversing the packet interval leads to the calculation of the packet rate.
7. Packet size (bytes): This field is required in either the CBR or the VBR mode. Reverse calculation of the UDP, IP and DLL header size has been performed in order to shape the LAYER 2 throughput to the selected value (as it results from Packet Size * Packet Rate).

Finally, after the user has completed the experiment form he can chose to either close the window, cancelling the experiment or to start the experiment with the selected options. The first action will be to check if the user input is of correct system values and if so the experiment will begin generating sessions and visualizing them in the (now visible) left panel. At the completion of the experiment, the experiment callback function will be invoked showing the results of each Session in the form of various QoS charts. These charts are:

- Packet drop probability: Statistical packet drop probability calculated as the number of packets send minus the packets received divided by the same number of packets send.
- Mean delay value: The statistical average delay between each consecutive packet received in the session.
- Jitter: The statistical variation (Jitter) of delay between each consecutive successful packet received by the session.
- Average Throughput. The application layer (payload bytes only) average throughput as it derives from diving the total number of received bytes by the total duration of the session.

These operations, output data and functionalities summarize the usage scenarios of the Client software we have developed for the experimentation of the Mesh networks and it provides us with multiple parameters to test in conjunction with the robot mobility aspects and the various mesh network topologies.

3 Conclusion

This deliverable considered the utilization of D2D communications for the resolutions of persistent issues of mobile networks. Moreover, the impact of mobility was taken into account through the utilization of a mobile robot which is available in the testbed. Certain procedures regarding the utilization of elements in the testbed were also analysed in the scope of this deliverable.