I. INTRODUCTION

Network simulator is a computer program which allows to model a communication network by specifying the structure and operation of network nodes and behavior of communication channels. Network simulators are mainly used in research but they are also very useful in education.

In this paper we focus on wireless local area network (WLAN) simulation. We are interested mainly in simulation of network layer and data link layer protocols. The simulation is to be performed by students within WLAN course projects. The first problem that needs to be resolved is the selection of appropriate platform on which students would develop their WLAN projects. One option is to use a custom network simulator developed from scratch by students. Unfortunately, to investigate a selected network protocol, e.g. MAC protocol, students would have to create a model of the entire network which would be too complex and time-consuming task, taking into account the time scheduled to complete the project. Another option (chosen by us) is to use a well-known simulation platform, which provides software environment for simulation of WLANs. We took into account only the publicly available, open source simulators, mainly ns-2, ns-3, and OMNeT++. Eventually, we chose network simulator 3 (ns-3).

The following arguments determined our choice: well-organized, free source code, already existing library of basic WLAN modules, both a library and simulation scripts written in C++ language, realistic environment (real IP/MAC addresses, real packets, BSD-like sockets, multiple interfaces per node, etc.).

The purpose of this paper is to show the usefulness of ns-3 as a simulation platform for students doing WLAN projects.

The rest of the paper is organized as follows. Section II contains a short description of ns-3. Section III presents the possible areas of applications of ns-3 in WLAN simulation. Section IV presents the example of IEEE 802.11s network simulation in ns-3 environment. Finally, Section V concludes the paper.

II. NS-3

Ns-3 is a discrete-event network simulator. It is publicly available, free software, which can be used for research and education. Ns-3 is developed within the ns-3 project [1], started in 2006. The aim of the project is to maintain an open environment for researchers to contribute and share their software. Ns-3 is primarily developed for Linux platforms, but it can also run in Windows, using Linux emulation environment called Cygwin, or virtual machine [2].

The ns-3 software is built on C++, i.e. it is a library containing a set of network simulation modules implemented as C++ objects [3]. Simulation scripts, which interact with this library, can be written in C++ or Python. The existing library of modules allows to simulate popular wireless networks (e.g. WiFi, WiMax) in a simple way, by writing a simulation script. Thanks to the availability of the source code of existing modules, it is possible to modify the operation of any module from the library. It is also possible to create new modules, implementing algorithms or protocols not present in the existing library.

Key objects in the simulator are Nodes and Channels [2], [5]. A Node represents a network element. It is a device (e.g. computer) to which Applications, Protocol Stack, and NetDevices can be added (Fig. 1). Applications are like processes in a normal system. They are packet generators or sinks which run on a Node. The interface between an Application and a Protocol Stack is called a Socket. A NetDevice is a network card which can be plugged into a Node in order to enable the Node to communicate with other Nodes in the network. Nodes communicate via Channels. Similarly to a real network device, which may contain separate interface for Ethernet, WiFi, WiMax, etc., an ns-3 Node may contain many NetDevices. Currently, in the ns-3 library, there are available both wireless interfaces (e.g.

---

**Fig. 1.** The basic model of a network in ns-3.
studied: way, among other things, the following WLAN issues can be simulated, which requires C++ programming skills only. This way, among other things, the following WLAN issues can be studied:

- network layer: various routing protocols (AODV, OLSR, DSDV),
- data link layer: MAC protocols (802.11-DCF, 802.11e-EDCA), layer-2 routing protocol (802.11s-HWMP), various rate control algorithms,
- physical layer: 802.11a/b/g physical layer models, various propagation loss models (e.g. Jakes, TwoRayGround, Nakagami, Friis, LogDistance, ThreeLogDistance, FixedRss, Random, etc.).

A more advanced use of ns-3 is testing custom algorithms or protocols. It requires the creation of new modules and their integration with already available modules. To do this, in addition to the knowledge of C++, students must also demonstrate some knowledge of Unix (e.g. sockets) and some familiarity with object-oriented simulation of discrete-event systems. Because the ns-3 code is well organized, students are usually able to quickly understand the code and make their own modifications of existing modules or write new ones which replace one protocol with another at a selected layer in the network stack.

Multi-radio multi-channel MAC protocol is the example of a new protocol for WLAN, which can by studied using ns-3.

IV. SIMULATION OF THE IEEE 802.11S NETWORK

The IEEE 802.11s is an amendment adding wireless mesh capabilities to the existing IEEE 802.11 WLAN standard. An important feature of IEEE 802.11s is the fact that the mesh network is implemented in the data link layer. Hybrid Wireless Mesh Protocol (HWMP) is a mandatory path discovery mechanism in IEEE 802.11s draft [6]. HWMP can operate in two modes: on-demand reactive mode and tree-based proactive mode.

The ns-3 library contains some modules which allow to simulate the 802.11s mesh WLAN. In this section, it is shown how students, using ns-3, can investigate the operation and performance of IEEE 802.11s network. For this purpose, two exemplary ns-3 simulations of 802.11s network are described. The simulations were performed with ns-3 ver. 3.9, using solely modules from its library.

A. The throughput of mesh WLAN

The first experiment consisted in studying the throughput of a mesh network. The throughput was defined here as the number of correctly transmitted bits per unit of time. The experiment was carried out in a network whose topology is shown in Fig. 2. Five mesh stations (STAs) were located in a line, at a distance of 120 m from each other. The transmission range of each station was 140 m and it covered the immediately neighboring stations.

For each station, the required transmission range was set by configuring the following parameters of the physical layer: Energy Detection Threshold, Transmission gain, and Reception gain. The model of a physical layer, proper to the mesh station, is implemented in YansWifiPhy class. The locations of the mesh stations were set using the GridPositionAllocator class.

III. USING NS-3 IN EDUCATION

Ns-3 is a useful educational tool for a simulation of communication networks, especially wireless ones. The easiest way to use ns-3 by students is to simulate a network consisting of modules stored in the simulator library. Modules currently available in ns-3 allow for the simulation of such wireless networks as WiFi, WiMax, and LTE. In order to simulate the network, a student has to write a simulation script, which requires C++ programming skills only. This way, among other things, the following WLAN issues can be studied:

- A wireless node’s specification may comprise a mobility model. Various mobility models are available in ns-3, e.g. constant position, constant velocity, constant acceleration, Gauss Markov, or random direction 2D.

- To create a network of nodes which exchange higher layers’ data, the protocol stack have to be installed in each Node. Ns-3 is a simulator intended primarily for Internet systems, therefore the Internet Stack is used. Ns-3 provides implementation of such TCP/IP related components like IPv4, ARP, UDP, TCP, IPv6, etc.

- A Node may be connected to more than one Channel through multiple NetDevices. A Channel is a physical connector between a set of NetDevices. A Node may be connected to more than one Channel through multiple NetDevices.

- The ns-3 tracing system is built on the concept of independent tracing sources and tracing sinks [4]. The tracing mechanism gives a possibility to reduce the amount of data outputted by the simulator, by tracing only required events. Moreover, output data can be formatted directly into required form, acceptable by programs used to analyze these data. Ns-3 generates two formats of trace files: pcap and ascci. The pcap (packet capture) format is used by many traffic analyzers, e.g. Wireshark. The simulation results generated by ns-3 can be easily formatted according to requirements of Gnuplot.

- Network simulations with ns-3 often require the use of scripts. A script parsing the content of the output file to extract required information from data gathered during simulation.

- The ns-3 library contains some modules which allow to simulate the 802.11s mesh WLAN. In this section, it is shown how students, using ns-3, can investigate the operation and performance of IEEE 802.11s network. For this purpose, two exemplary ns-3 simulations of 802.11s network are described. The simulations were performed with ns-3 ver. 3.9, using solely modules from its library.

- The throughput of mesh WLAN

The first experiment consisted in studying the throughput of a mesh network. The throughput was defined here as the number of correctly transmitted bits per unit of time. The experiment was carried out in a network whose topology is shown in Fig. 2. Five mesh stations (STAs) were located in a line, at a distance of 120 m from each other. The transmission range of each station was 140 m and it covered the immediately neighboring stations.

For each station, the required transmission range was set by configuring the following parameters of the physical layer: Energy Detection Threshold, Transmission gain, and Reception gain. The model of a physical layer, proper to the mesh station, is implemented in YansWifiPhy class. The locations of the mesh stations were set using the GridPositionAllocator class.
The mesh STA1 was configured to receive data. To this end, the PacketSink application was installed on the station with the task of receiving UDP traffic. All the remaining stations were transmitting stations, each running the OnOff application. This application is a traffic generator, which follows an On/Off pattern, i.e. On and Off states alternates. In the On state, data packets of a predetermined fixed size are generated, while in the Off state, the generation of the packets is suspended. In the case under study, OnOff application generated UDP traffic directed at mesh STA1 with a fixed intensity of 54 Mbps.

During the experiment, HWMP was operating in the reactive mode. The measurements were carried out for four datagram lengths – 100B, 800B, 1500B and 2300B – with a different number of hops between the end stations. The mesh STA1 was the recipient station while the transmitting station changed with each increase in the number of hops, i.e. for one hop, the mesh STA2 was the source; for two hops, the mesh STA3 was the source; for three hops, the mesh STA4 was the source; and for four hops, the mesh STA5 was the source.

To determine the throughput, an ns-3 tracing system was used. First, information on the number of bytes received correctly by mesh STA1 was gathered. To achieve this, the RxOk trace source was used. This source reports that the PHY layer has correctly received a packet, and makes the contents of the packet available. A trace sink attached to the RxOk trace source processed the received packets. This processing consisted in checking for packet type. If the received packet was a data packet, its size was recorded. The sizes of all correctly received packets were summed throughout the duration of the measurement.

The simulation script implemented a function that determined the throughput on the basis of the sum of the bytes received. The function saved the throughput result to an output file. The simulation script was run multiple times for varying datagram sizes and varying hop numbers. As a result, it could be determined how the throughput depends on the number of hops and datagram length, as shown in Fig. 3.

The aim of the second experiment was to analyze network traffic, study the operation of HWMP, and check whether the protocol implemented in ns-3 operates in compliance with 802.11s draft.

The experiment was run for the network whose architecture is shown in Fig. 4. The arrows show which stations in the simulation connected to each other. The stations were located statically, and the distances were deliberately selected. The locations of the mesh stations were determined using the ListPositionAllocator class. Like in the previous scenario, the OnOff application was used to generate UDP traffic. This time, it was run on mesh STA1. Mesh STA5 was the receiving station; the PacketSink application was run on this station.

Ns-3 makes it possible to save all the packets generated and received by the stations to a pcap file. This allows one to analyze the network traffic, and thereby to study the process of mesh network creation and path discovery. From the PREQ and PREP packets exchanged in mesh network during the path discovery process, the metric values, corresponding to individual connections, can be read. The Wireshark program was used to read the pcap file containing the frames transmitted in the mesh network.

Packets transmitted in an ns-3 simulated network are just like packets transmitted in a real network. The analysis performed using Wireshark showed that, for path discovery, the frames exchanged both in the proactive and reactive mode of HWMP match the frame format specified in the 802.11s recommendation. The subsequent stages of mesh network creation and path discovery using the HWMP protocol also follow the solutions recommended in the 802.11s draft. Missing fields were only found in packets exchanged during the process of the establishing of connections between the mesh stations (the Peer Link Open and Peer Link Confirm frames), which should have carried information on the connection parameters.

In mesh network simulation, it is possible to generate an xml report for each mesh station. This report contains statistics for the HWMP routing protocol and the Mesh Peer Link Management (MPLM) protocol. The report contains information on the number and size of the generated, sent and received packets of each type, information on whether a given station functions as a root, the parameters of the connections that the stations have established with neighbouring stations, and the metric values for each connection.

Based on the information contained in the report, it can be verified whether the stations established connections as predicted by the user, whether and how often the connections were broken, and what metric values were set. The reports
obtained for the network under study show that the stations established connections as shown in Fig. 4.

In the scenario under study, mesh STA1 and mesh STA5 could communicate using four different paths, plotted using different line styles in Fig. 5. Above the lines, the metric values for each connection are shown, which were read from the reports for each station after the simulation was completed. The example in Fig. 5 refers to the situation in which none of the stations functioned as a root, and the paths were determined using the reactive mode of HWMP. As the figure shows, in some cases the stations set different metric values for the same connection. For example, the metric value for the connection between mesh STA2 and mesh STA1 set by mesh STA2 was 24, while that set by mesh STA1 was 29.

The continuous line arrows in Fig. 5 denote the path with the smallest number of hops. The traffic analysis demonstrated that the data travelled between the end stations using a path with the lowest cost according to the metric values contained in the reports. One can thus conclude that the model of HWMP implemented in ns-3 operates in compliance with the 802.11s recommendation.

C. Modifications of routing protocol in 802.11s

The 802.11s draft defines the Airtime Link metric, which may be used by a path selection protocol to identify the best path [6]. The draft does not prevent other metrics from being used, i.e. the recommended metric may be replaced with any path selection metric. There are proposed many metric calculation algorithms [7]. Using ns-3, one can test one’s own algorithms of determining a metric. To do so, one needs to write a class whose functionality will determine the value of the metric.

REFERENCES