Improving ns-3 Emulation Support in Real-World Networking Scenarios

Helder Fontes
INESC TEC and
Faculty of Engineering
University of Porto
Portugal
hfontes@inesctec.pt

Rui Campos
INESC TEC and
Faculty of Engineering
University of Porto
Portugal
rcampos@inesctec.pt

Manuel Ricardo
INESC TEC and
Faculty of Engineering
University of Porto
Portugal
mricardo@inesctec.pt

ABSTRACT

A common problem in networking research and development is the duplicate effort of writing simulation and implementation code. This duplication can be avoided through the use of fast-prototyping methodologies, which enable reusing simulation code in real prototyping and in production environments. Although this functionality is already available by using ns-3 emulation, there are still limitations regarding the support of real network interfaces and easy configuration of the network settings, such as IP and MAC addresses.

In this paper we propose an improved version of the ns-3 emulation component by introducing new functionalities that address these limitations. The new functionalities include the support of new types of real network interfaces and the easier integration of emulation nodes with existing networks by means of a new auto-configuration mechanism for ns-3 nodes. Experimental results obtained in a laboratorial testbed and in a real vehicular network testbed demonstrate the new functionalities proper operation, and their backwards compatibility with previously coded ns-3 scenarios.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software—*Reuse Models*; I.6.M [**Simulation and Modeling**]: Miscellaneous

General Terms

Experimentation, Verification

Keywords

ns-3, Network Simulation, Network Emulation, EmuFdNetDevice, Vehicular Networks, Fast-Prototyping

1. INTRODUCTION

A common problem faced in networking research and development is the duplicate effort of writing simulation and implementation code. When using Network Simulator 3 (ns-3) [1], there are two main approaches to accomplish a shared protocol model implementation and avoid code duplication: Direct Code Execution (DCE) [9] – code developed outside ns-3 and reused in ns-3 simulations – and fast-prototyping [5] – code developed in ns-3 and reused in real prototypes. DCE may be the right choice if we are reusing compatible applications or protocols already developed in Linux. However, when developing new protocols, ns-3 provides abstractions that result in faster protocol development when compared to Linux [5]. ns-3 also provides detailed logging by using the ns-3 tracing facilities, which is an advantage to produce detailed simulation results when compared to protocols developed outside ns-3. This work is focused on the fast-prototyping approach and results from our past experience on developing a routing protocol [7] that needed to be thoroughly simulated and then implemented in a real prototype.

The fast-prototyping methodology proposed in [5] takes advantage of the built-in network emulation features of ns-3. ns-3 emulation allows simulations to be executed in real time and ns-3 nodes to use emulated network interfaces. Emulated network interfaces provide direct access to the real network interfaces of the Linux node hosting the ns-3 process execution. From the real networks' perspective, the ns-3 nodes are real nodes running a real network protocol.

When the fast-prototyping methodology is employed in a controlled, static testbed, the experimental scenario is usually characterized by Ethernet or Wi-Fi real networks that are administered by the experimenters themselves. The experimenters can then deploy emulated ns-3 nodes accessing those real networks, with emulated network devices, using configurations (e.g., MAC and IP addresses) that are predefined for the experiment and remain constant and controllable during the whole experiment timeframe. Yet, when the fast-prototyping methodology is used in a more complex and dynamic scenario – e.g., a vehicular network – different network access and usage characteristics take place.

Figure 1 depicts the use of emulation in a vehicular network scenario, where two interfaces are available to access real networks. The ppp0 interface represents the 3G connectivity, and enables IP over the 3G Network. This interface is only present in the real Linux node when there is an active 3G connection. In practice, it is common to have this

SIMUTOOLS 2015, August 24-26, Athens, Greece

DOI 10.4108/eai.24-8-2015.2261074

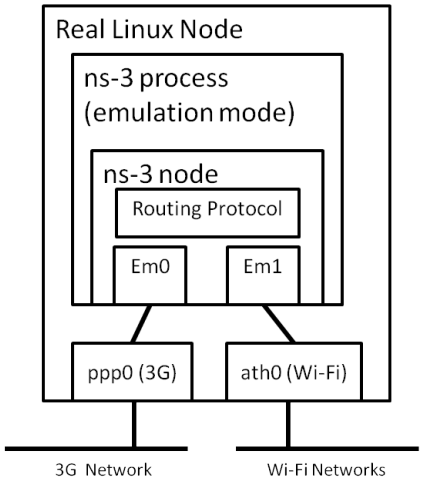

Figure 1: Emulation in a vehicular network scenario.

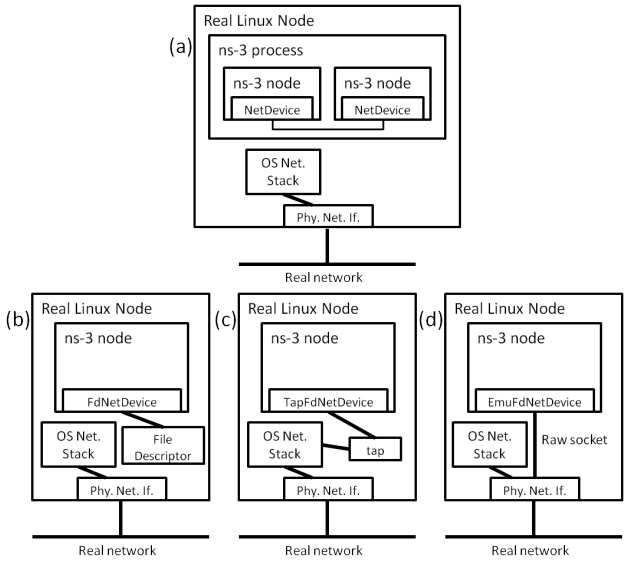

Figure 2: Overview of the communications provided by the ns-3 (a)NetDevice, (b)FdNetDevice, (c)TapFdNetDevice and (d)EmuFdNetDevice.

interface intermittently available, due to possible intermittent 3G connectivity and the dynamic IP renewal policy that may be imposed by the operator. The ath0 interface represents the Wi-Fi connectivity to multiple Wi-Fi networks available along the vehicle path, to which the real node connects opportunistically. Each of these networks may have its own administrator and assign specific dynamic IP level settings. Also, as an access control policy, network administrators may use MAC addresses to identify the users and provide IP level configurations.

The ns-3 emulation mode and the related emulated network devices, represented by Em0 and Em1 in Figure 1, were tested in the SITMe's [2] multi-technology vehicular network scenario, where the real network interfaces had characteristics similar to those previously mentioned. These characteristics precluded the use of fast-prototyping in the vehicular network scenario due to existing ns-3 limitations.

In order to support such characteristics and overcome the current ns-3 limitations, we propose an improved ns-3 emulated interface (EmuFdNetDevice), improving its compatibility and self-configuration to interact with real network interfaces. The result is an enhanced ns-3 emulation module that enables the use of fast-prototyping in complex and dynamic real network scenarios, such as vehicular network. The improved EmuFdNetDevice is backwards compatible with previously coded scenarios. As a result of this contribution, there is a work in progress to integrate the improved EmuFdNetDevice in the official release of ns-3.

The remainder of the paper is organized as follows. Section 2 presents an overview of different ns-3 communication types, focusing on the communication between an ns-3 node and a real network. Section 3 introduces the problem, with a detailed description of the characteristics of the vehicular network scenario challenging the current ns-3 EmuFdNetDevice. Section 4 describes the improved EmuFdNetDevice. Section 5 presents its functional evaluation using both a laboratory testbed and a vehicular testbed. Finally, Section 6 draws the major conclusions and points out future research directions.

2. OVERVIEW OF NS-3 COMMUNICATION TYPES

ns-3 is an event-driven packet level network simulator and is largely adopted by the scientific community to evaluate networking solutions in simulation environment. Being a packet level simulator, ns-3 allows to produce fully detailed simulation environments that accurately represent the real network behaviour; for instance, each network packet exchanged in the simulator uses exactly the same structure of a real packet. This realism enabled the development of the ns-3's emulation functionality that, in its essence, is the ability to run a simulation scenario in real time with the capability of exchanging network traffic between real and ns-3 nodes. From the real nodes' perspective, the ns-3 emulated network appears as an extension of the real network, with transparent exchange of network traffic between them.

In the real world, network nodes have network interface cards, allowing them to connect to a network and exchange network traffic. Likewise, in ns-3, simulated nodes are connected to a network using NetDevices. Figure 2 presents an overview of the two communication types possible when using ns-3; internal – between ns-3 nodes (Figure 2a) – and external – between an ns-3 node and the outside world, either the real Linux node hosting the ns-3 node or a real network (Figures 2b–d).

In Figure 2a two ns-3 nodes exchange network traffic over a virtual channel using standard NetDevices, hence, the nodes can not communicate with the outside of the ns-3 process. In Figure 2b an ns-3 node uses a specific NetDevice – the FdNetDevice – which allows exchanging network traffic with the real Linux node, using a file descriptor managed by the Operating System (OS) of the real node. In Figure 2c an ns-3 node is using a specialization of the FdNetDevice – the TapFdNetDevice – designed to exchange network traffic with the real Linux node using a tap interface. Every write from the ns-3 node via the TapFdNetDevice appears

to the real Linux node as incoming network traffic via the tap interface, and vice versa. Finally, in Figure 2d an ns-3 node is using another specialization of the FdNetDevice – the EmuFdNetDevice – designed to allow direct communication to outside of the real Linux node using a raw socket bound to a real network interface (e.g., eth0). This allows exchanging the ns-3 node's traffic with that specific real network interface, thus enabling the emulation functionality.

The fast-prototyping methodology [5] uses the EmuFdNetDevice module. Figure 1 illustrates an example of a vehicular mobile router prototype developed using the fast-prototyping methodology, where a routing protocol implemented in ns-3 is reused. The utilization of the EmuFdNetDevices – represented by Em0 and Em1 interfaces in Figure 1 – is important for the vehicular mobile router prototype, as they provide direct communications to the real networks as if it was a real node.

3. PROBLEM

In this section we introduce the problem, with a detailed description of the characteristics of the vehicular network scenario that cause incompatibilities with the current EmuFdNetDevice.

3.1 3G PPP interfaces provide IP level communication

Point-to-Point Protocol (PPP) [8] interfaces supporting IP over cellular networks are unsupported by ns-3. The EmuFdNetDevice module is designed to read and write Ethernet frames from/to real interfaces, not IP packets as it is the case for these 3G PPP interfaces. As such, the EmuFdNetDevice must be modified and capable of detecting whether the real interface is operating at L2 or L3 and adapt itself accordingly. Also, when working at IP level, the EmuFdNetDevice does not need Ethernet Address Resolution Protocol (ARP) [6] support. This aspect needs to be addressed as well; otherwise the installation of the Internet Stack in the node associated to that EmuFdNetDevice will fail due to the asserts that are made when installing the ARP protocol.

3.2 3G PPP interfaces are intermittent

3G related PPP interfaces are only available in Linux when there is an active 3G connection established. Throughout the experiment duration, the 3G connection can be lost due to (1) the lack of network coverage in some geographic areas, (2) forced reconnection by the operator to allow dynamic IP renewal, and (3) any sort of other communications problems preventing communications between the PPP Client and Server. This leads to the PPP session shutdown, and the interface ppp0 disappears. The EmuFdNetDevice module does not support this intermittence and has two possible undesirable behaviors: 1) if the real interface is not available when the emulation starts, the ns-3 aborts the execution; 2) if the real interface is available when the emulation starts but it disappears, the ns-3 process detects that the raw socket was closed, stops the EmuFdNetDevice, but does nothing to restart it.

3.3 Real MAC addresses have to be used

In a simulation scenario there is full control of the elements interacting in the simulation. Typically ns-3 self-generated MAC addresses are used and assigned sequentially to every simulation node to avoid MAC address collisions. However, when fast-prototyping is used, we may not fully control the scenario and the interactions with other nodes. So, in order to avoid MAC address collisions and network access control problems, the MAC address of the real interface has to be used by the emulated node. ns-3 does not include any MAC cloning functionality, which results in the need for error-prone, additional manual configurations.

3.4 IP configuration settings are dynamic

In IP networks, the interface auto-configuration provided by Dynamic Host Configuration Protocol (DHCP) [4] is frequently used. This auto-configuration mechanism allows a node to establish IP connectivity with other nodes using the given network settings, such as the IP address, network mask, and default gateway, leased by a DHCP server. In a vehicular network environment it is usual for a node to connect to different networks with the same interface, or use a mobile network that imposes IP address renewal from time to time. In the current version of the ns-3 EmuFdNetDevice, there is no mechanism to keep the network settings updated in the emulated node whenever the real network interface settings change. This will make the emulated node use wrong network settings and lose communication with other nodes.

4. PROPOSED EmuFdNetDevice

Motivated by the characteristics presented in Section 3, our work aims to expand the functionality of ns-3 by proposing an improved backwards compatible EmuFdNetDevice module, which supports new features to further improve the capability of running emulated and hybrid environments – emulated and real nodes interacting – over different real network scenarios.

4.1 Detection of the operating layer of real network interfaces

The current EmuFdNetDevice is hardcoded to read and write Ethernet frames; as such, it only supports real network interfaces operating at MAC level. Conversely, the improved EmuFdNetDevice inspects the underlying real interface, checks whether it has a MAC address assigned, and classifies the real interface as an IP or MAC level real interface accordingly. This information is saved in the new flag named *m_isL2NetDevice*, associated with the FdNetDevice, which is setup during the Helper execution. When the real network interface is operating at IP level, the improved EmuFdNetDevice disables the "NeedArp" setting, in order to avoid assertion errors when installing the Internet Stack in the node. The automatic detection of the network stack, associated to the underlying real interface, effectively avoids the need for added configuration and enables backwards compatibility with previously coded scenarios.

4.2 Support for intermittent real interfaces

The EmuFdNetDevice was designed assuming that the real interface to which it is bound has an identifier always present in the Linux's interfaces list. As explained in Section 3, this is not always true – e.g., the PPP interfaces representing 3G connections. The current EmuFdNetDevice has two possible undesirable behaviors, which were referred in Section 3. The improved EmuFdNetDevice expects and handles gracefully this behavior, according to the state machine presented in Figure 3. The improved EmuFdNetDevice allows

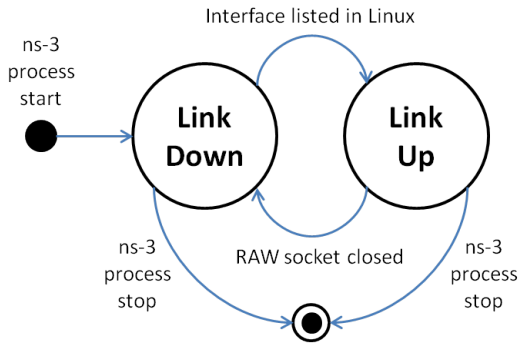

Figure 3: State machine representing how the improved EmuFdNetDevice handles communication using intermittent real interfaces.

the user to configure the support of intermittent behavior. When the related flag – named *m_isIntermitentInterface* – is set, the improved EmuFdNetDevice assumes the link is down, allowing the emulation process to be executed even if the real interface is not listed. In the "Link Down" state, the simulator checks periodically whether the real interface ID becomes listed again in the real Linux node. As soon as the interface ID becomes listed, a new raw socket is created and the communication is restarted, assuming the "Link Up" state. When the raw socket is unexpectedly closed, the ns-3 process sees the socket returning "-1". Instead of just stopping the device, the improved EmuFdNetDevice assumes again the "Link Down" state and actively tries to re-establish communication by creating a new raw socket and binding it to the new real interface. The ns-3 process can be stopped from either state.

The improved EmuFdNetDevice enables the support for intermittent real interfaces while keeping full compatibility with previously coded scenarios. The only drawback is that the user creating the emulation scenario must be sure about the identifier of the real network interface. In the current EmuFdNetDevice, the process is terminated and the user is informed about the non-existing/erroneous interface identifier. With the improved EmuFdNetDevice, if the identifier supplied by the user is wrong and the user configures the interface as being intermittent, the emulation process will run without ever binding to a real interface nor informing about the error.

4.3 MAC Address Cloning

Simulated nodes, running inside ns-3, use self-generated MAC addresses by default – e.g., 00:00:00:00:00:01. This is not a problem when running simulations, but can be a problem when using the emulation capability in a real network scenario – e.g., the case of fast-prototyping network protocols. In such case there will be a number of ns-3 instances running independently from each other, with emulated ns-3 nodes acting as real nodes. Being different ns-3 instances, means that the ns-3 mechanism for self-generating MAC addresses will assign, by default, coincident MAC addresses (starting in 00:00:00:00:00:01) to the ns-3 nodes running in the different instances. Manually managing the MAC addresses of each emulated node accessing a real network to avoid MAC collisions can be difficult and error-prone. Also, often the MAC address used by the EmuFdNetDevice has to match the MAC address of the real interface of the real node, in order to allow communication in the real network. Using the real interfaces' MAC addresses could then solve the MAC addresses collision problem and ensure compatibility with real networks requiring the use of the MAC address of the real interface. Because ns-3 lacks a MAC cloning feature, a configuration option was added to the EmuFdNetDevice instances, in order to allow automatic cloning in run time of the MAC address of the real interface to which the specific EmuFdNetDevice is bound. This feature is disabled by default, so that the EmuFdNetDevice operation is unchanged when running previously coded scenarios.

4.4 IP Address Cloning

In a real network scenario, IP level network configuration is often carried out using the DHCP protocol. This is common when connecting to real networks in a vehicular scenario, where multiple networks can be used and each network has a different IP configuration. Also, usually Internet Service Providers (ISPs) do not provide fixed public IP addresses; as an example, every time a PPP connection is established over a public 3G operator link, a new dynamic IP address is leased. Because the IP address changes every time a new connection is established, even during the same emulation process execution, the emulation scenario rapidly becomes outdated, with an IP address associated to the EmuFdNetDevice that is no longer valid nor corresponds to the real PPP interface anymore. When the emulated node tries to write packets with an outdated IP, they can be discarded or lead to an IP address collision. In order to avoid this problem, we added an IP address cloning feature to the EmuFdNetDevice, which is enabled by configuration in every EmuFdNetDevice instance. When enabled, ns-3 periodically verifies the IP address, network mask and default gateway of the real interface and applies those settings to the node running the emulated interface, successfully simplifying the deployment and auto-configuration of emulated nodes in a real environment. IP address cloning is disabled by default, assuring the EmuFdNetDevice's behavior expected by previously coded scenarios.

5. SOLUTION VALIDATION

This section describes the tests performed to validate the proper operation of the improved EmuFdNetDevice. Two scenarios were used to perform the tests: 1) a laboratory testbed, used to test the new features and assist the debug process in a controlled environment; 2) a real world testbed using the SITMe's project vehicular network.

5.1 Laboratory testbed

The first approach to validate the improved EmuFdNetDevice was to conduct the experiments in a small laboratory testbed, in order to easily test the proper operation in a fully controlled scenario.

Figure 4 presents the components that characterize the laboratory testbed, which is composed by two Real Linux x86 nodes:

Real Node #1 is a multi-interface real network node hosting an ns-3 emulated node. The real node has two physical Ethernet interfaces: 1) eth0, auto-configured at IP level using the DHCP client; 2) eth1, to establish a PPPoE connection from the PPPoE Client to

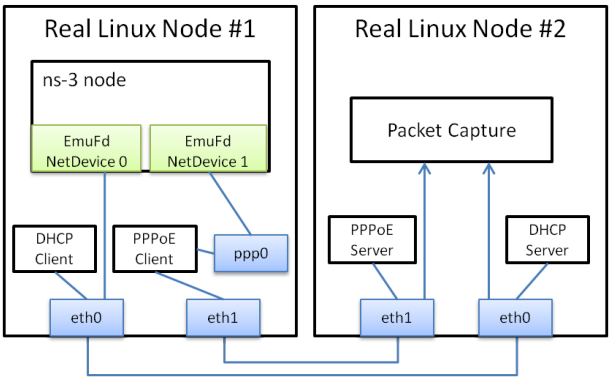

Figure 4: Laboratory testbed scenario.

the PPPoE Server. When the PPPoE connection is established, the ppp0 interface is listed in Linux. eth0 represents the use of real network interfaces operating at MAC level and ppp0 represents a real network interface working at IP level. The ns-3 node has two emulated network devices, one bound to the real node's eth0 and the other to the intermittent ppp0;

Real Node #2 is a communication peer used to interact with the emulated node, but it also implements the access control and auto-configuration mechanisms present in most real networks, such as DHCP server and PPPoE server. Real Node #2 has two Ethernet interfaces connected directly to the equivalent interfaces in Real Node #1. Real Node #2 also captures the network traffic from the two Ethernet interfaces, in order to assess the correct operation of the emulated node when communications between Real Node #2 and ns-3 emulated node are attempted.

In order to reproduce the conditions needed to test each functionality of the improved EmuFdNetDevice, the following procedures were considered:

PPP intermittent behavior. The PPPoE Server was periodically stopped and restarted in order to close and reestablish the PPPoE connection and make the ppp0 interface disappear and reappear in Real Node #1;

Dynamic IP configuration on eth0. Short duration IP leases were used, in order to generate frequent lease renewals. By using manual MAC-IP address associations in the DHCP Server that were periodically changed to different IPs, led to periodic IP address changes in the eth0 interface of Real Node #1;

Dynamic IP on ppp0. Every time the PPPoE Server was stopped, the configuration file was changed to assign a different IP addresses range to the PPPoE client, so that the ppp0 IP address was changed every time the PPPoE connection was re-established.

In order to generate network traffic between Real Node #2 and the ns-3 node running in Real Node #1, two mechanisms were used: 1) ICMP echo requests/replies between the two nodes; 2) an UDP echo server running in Real Node #2, which replied to UDP packets sent by the ns-3 node with exactly the same UDP payload. The traffic generated was captured using Wireshark [3], which was running in Real Node #2.

To validate each new feature of the improved EmuFdNetDevice, the following observations were made:

Detection of the operating layer of real interfaces. The use of two EmuFdNetDevices bound to eth0 and ppp0 interfaces confirmed the correct operation of this feature. In the case of the eth0 interface, the lower level transfer unit used by EmuFdNetDevice0 was the Ethernet frame, while with the ppp0 interface, EmuFdNetDevice1 exchanged IP packets;

Support for intermittent real interfaces. The use of the PPPoE protocol led to the creation of the ppp0 interface in Real Node #1. This interface was only listed intermittently according to the test conditions referred above, which allowed the test of the EmuFdNetDevice ability to recover from the following conditions: 1) ns-3 process started without the ppp0 interface available; 2) ppp0 interface unexpectedly becomes unavailable. In both situations, the ns-3 process detected the anomalous conditions and successfully kept the emulation running;

MAC Address Cloning. The EmuFdNetDevice0 successfully cloned the real eth0 MAC address, and every communication made to Real Node #2 appeared in the Wireshark logs as originating from the MAC address of the real eth0 interface. Also, the ns-3 emulated node replied successfully to every communication directed to it;

IP address cloning. The usage of different IP addresses during the tests, in both ppp0 and eth0 interfaces, allowed to successfully test whether the IP addresses were updated in the ns-3 node. This mechanism worked with success. The time interval between checks of the real interface's IP address is configurable. If the IP address changes very frequently, it is recommended to use low interval checks to keep the ns-3 node settings updated; yet, too small time interval uses more CPU resources.

After validating each new feature, the correct operation of the improved EmuFdNetDevice was confirmed.

5.2 Vehicular Network Testbed

With the improved EmuFdNetDevice tested in the laboratory testbed, the next step was to test it in a real world testbed. The selected testbed was the one used in the real pilot of the SITMe's project, composed by 11 buses with an on-board Linux router, supporting multiple access technologies to provide Internet access to bus passengers.

Figure 5 depicts the elements composing each of the 11 buses used in the SITMe's real pilot. This testbed used the Wireless Metropolitan Routing Protocol (WMRP) [7], a proactive multi-technology routing protocol based on OLSR, entirely developed in ns-3. Each bus had a Linux computer installed with five network interfaces, one to give network access to the passengers and other bus equipment, and the other four to connect to the outside world. Using the fast-prototyping methodology, the routing protocol ran in an ns-3 emulated node, as illustrated in Figure 5. This emulated node had as many Emulated NetDevices (EmuFdNetDevice)

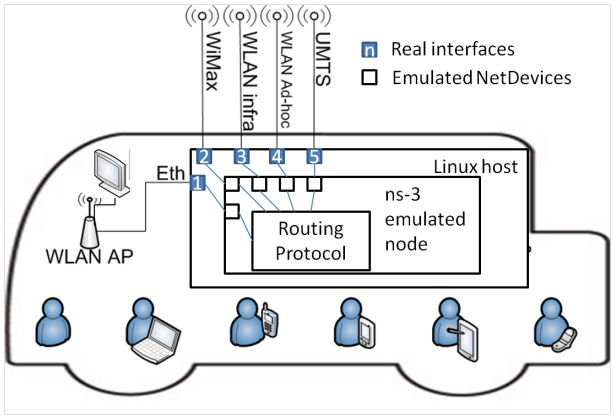

Figure 5: Elements present in each bus of the Vehicular network testbed scenario.

as the number of real interfaces connected to the real node. Through this mechanism, the ns-3 router had direct access to the real networks and acted as a real router from the real networks' perspective.

All features introduced in the improved EmuFdNetDevice were tested in this scenario: multiple interfaces were configured using dynamic IPs obtained via DHCP; the 3G operator provided IP level PPP interfaces, which were intermittent; MAC address cloning was used in all the interfaces with MAC address, to avoid MAC address collisions and obey to the ISP policy. WiMAX traffic was only allowed by the operator for specific real interfaces' MAC addresses.

This experiment ran successfully for more than one year with good results and very good feedback from the bus passengers. This real world usage of the improved EmuFdNetDevice proved its correct operation and usefulness, especially in a heterogeneous real world scenario such as the SITMe's real pilot, where multiple emulated instances needed to be deployed and auto-configured to allow communication in a demanding real network environment.

6. CONCLUSIONS AND FUTURE WORK

The fast-prototyping methodology has several possible applications and was already proven useful, although it fully depends on the ns-3's emulation capabilities. Currently, ns-3 emulation capabilities are enabled by running simulation code in real time and establishing network communications with real networks using the EmuFdNetDevice, which is bound to real network interfaces of the real node running the ns-3 process.

However, the use of the EmuFdNetDevice in complex and dynamic real network scenarios – e.g., a vehicular network – revealed the limitations of the current EmuFdNetDevice, thus invalidating the use of the fast-prototyping methodology in scenarios with such conditions. This was due to the incompatibilities with some real interfaces operation and the overhead and error-prone methods needed to configure each emulated instance. Motivated by the need to overcome these problems, we proposed an improved, backwards compatible EmuFdNetDevice including a set of features that addresses interface compatibility problems and introduces self configuration mechanisms to enable the fast-prototyping methodology in demanding real network conditions, such as the existing in vehicular network scenarios.

Using a laboratory testbed and a real world vehicular network testbed, it was possible to confirm the proper operation of the improved EmuFdNetDevice. Also, the usefulness of such features was demonstrated in complex real network scenarios, such as the SITMe's real pilot.

As future work, we plan to add the improved EmuFdNetDevice to the ns-3 project official code repository, and make these improvements available to the community. We are also working on other topics related to the fast-prototyping methodology, which are closely related to the work presented herein and will bring further optimizations to the ns-3 emulation mode. This includes the reduction of the processing overhead related to the emulation mode and the ability to perpetuate real experiments by using the traces obtained from the ns-3 emulated nodes to replicate the same exact conditions in simulation.

7. ACKNOWLEDGMENTS

The authors express their gratitude to the BEST CASE project ("NORTE-07-0124-FEDER-000056", "NORTE-07-0124-FEDER-000058" and "NORTE-07-0124-FEDER-000060") financed by the North Portugal Regional Operational Programme (ON.2 – O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF).

This work is financed by the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project UID/EEA/50014/2013, and fellowship SFRH/BD/69051/2010.

8. REFERENCES

[1] ns-3 website. http://www.nsnam.org/. Accessed: 2015-02-13.

[2] SITMe's project website. http://www.sitme.org/. Accessed: 2015-02-13.

[3] Wireshark website. http://www.wireshark.org/. Accessed: 2015-02-13.

[4] S. Alexander and R. Droms. DHCP Options and BOOTP Vendor Extensions, March 1997. RFC 2132.

[5] G. Carneiro, H. Fontes, and M. Ricardo. Fast prototyping of network protocols through ns-3 simulation model reuse. *Simulation Modelling Practice and Theory*, 19(9):2063–2075, 2011.

[6] D. Plummer. Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware, November 1982. RFC 826.

[7] M. Ricardo, G. Carneiro, P. Fortuna, F. Abrantes, and J. Dias. Wimetronet a scalable wireless network for metropolitan transports. In *Proceedings of the 2010 Sixth Advanced International Conference on Telecommunications (AICT)*, pages 520–525, May 2010.

[8] W. Simpson. The Point-to-Point Protocol (PPP), July 1994. RFC 1661.

[9] H. Tazaki, F. Uarbani, E. Mancini, M. Lacage, D. Camara, T. Turletti, and W. Dabbous. Direct code execution: Revisiting library os architecture for reproducible network experiments. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, pages 217–228, New York, NY, USA, 2013. ACM.