UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

DOTTORATO DI RICERCA IN INFORMATION AND
COMMUNICATION TECHNOLOGIES (ICT)
nell'ambito della Scuola di Dottorato in
INFORMATION AND COMMUNICATION TECHNOLOGIES (ICT)
Ciclo XXV

VIRTUALIZATION TECHNIQUES FOR THE STUDY OF EMERGENCY NETWORKS

Relatore (Tutor):
Prof. Ing.
MAURIZIO CASONI

Candidato:
GIORGIO CALARCO

Direttore della Scuola di Dottorato:
Chiar.mo Prof. Ing.
GIORGIO MATTEO VITETTA

Anno Accademico 2012 – 2013

Questo lavoro è dedicato a Irene e Catia, sempre con me.

Abstract

Communications for crisis support are today based on rate-limited Professional Mobile Radio infrastructures. However, protection agencies show a rising need for a broadband approach, to increase the reliability and the quality of communications, and the effectiveness of operations. Multi-part Emergency Networks, where different technologies are used at the Personal, Incident, Jurisdictional and Extended Area Networks might be employed to support first responders (FRs), mobile Emergency Operation Centers (MEOCs) and the remote Emergency Operation Center (EOC). However, the design of a heterogeneous, multi-technological telecommunication system offers several challenges. A significant target is to understand since the beginning if a network architecture matches the expected performance. In particular, it would be helpful to have a design tool with smart configuration capabilities, for examining the performance of alternative architectures, to choose the optimal infrastructure. A broad variety of methods can be adopted, ranging from pure mathematical models to real-world testbeds. If it were possible, the perfect tool should summarize the best of all the possible practices, that is, be as flexible and cost-effective as a software simulator; modular as a hardware emulator; and, realistic as a physical testbed. For these reasons, software network emulation has recently gained a lot of interest in the research community. The key idea behind it is to follow a hybrid approach: a software simulator is mixed with real components and applications, to increase the fidelity of results and to perform the validation of an infrastructure against real traffic. Similarly to pure simulators, software emulators are fast to configure and low-cost, and their realism depends on the network modeling accuracy. However, an emulator cannot run in a virtual simulated time, like simulators do, to coexist with real network entities, and it needs enough computing resources to respect real-time constraints when processing incoming data. If true applications exchange traffic through an emulated topology, the emulator should not introduce any extrinsic delay. Certainly, the plain execution of several types of networks within a single, real-time emulator requires a huge amount of computing resources. Therefore, the investigation of a heterogeneous infrastructure could be performed by the execution of many emulators in parallel (one for each part of the overall network). This task can be accomplished by running the different emulators onto several PCs. However, today, virtualization techniques have become very efficient and allow different applications to

be hosted onto a single hardware, as well as multi-core CPUs are widely available, even within the cheapest netbooks. Therefore, in this work, we investigated the architecture and the implementation issues of a modular and scalable testbed (NetBoxIT) which aims at taking advantage of these available techniques (software emulation, virtualization, and multi-CPU platforms) to simulate complex networks, possibly organized with different topologies and technologies. NetBoxIT supports the creation, and interconnection of several, coexisting virtual networks ("netboxes") onto a single, multi-core platform. In summary, using LXC containers and the NS-3 simulator, a number of distinct netboxes can run concurrently and mimic the separate portions of a heterogeneous network. NetBoxIT supports real-time data handling, with negligible timing overheads against the represented true network in many cases, and can be interfaced with external real nodes. We examined the testbed performance with several emulation trials, most of which are related to the reference Emergency Network under study within the EU 7th FP project "A holistic approach towards the development of the first responder of the future", to verify its realism in assessing multi-part networks evaluations.

Keywords - emulazione, virtualizzazione, emergenza, wireless, NetBoxIT

Contents

List of Figures

Chapter 1

Introduction

Heterogeneous networks design is frequently a demanding task. When different technologies and protocols are needed to inter-operate, it is difficult to foretell if a proposed infrastructure will really attain the supposed operating performance. Things can be even more difficult when such systems are to be adapted and optimized for the particular requisites of some special traffic profile (e.g., to respect a certain quality of service constraint in multimedia communications). The a priori availability of a flexible design tool could offer an important benefit, as it would help to define the best network architecture from the very beginning. In particular, it would be appealing to have an instrument that allows the fast examination and comparison of various architectures. Such a tool should offer some fundamental attributes: realism and repeatability, for the reliable evaluation of distinct, possible choices; flexibility and scalability, to cope with more and more complex scenarios; and, finally, be low-cost.

In this work, we discuss the design principles, the implementation issues, and the performance of a modular and scalable software testbed (NetBoxIT), that can be fruitfully employed to simulate heterogeneous communication systems, organized with various topologies and technologies. NetBoxIT is conceived to assist the creation and the combination of several virtual networks ("netboxes") on a single, multi-CPU hardware platform. In synthesis, using virtualization techniques and a network simulator, a number of independent netboxes (each using a different configuration) can run simultaneously, to imitate the different portions of a complex network. By means of software bridges or external equipment, these virtual networks can be joined to synthesize the behavior of the whole infrastructure under investigation. Figure 1.1 depicts the abstract schema of a netbox.

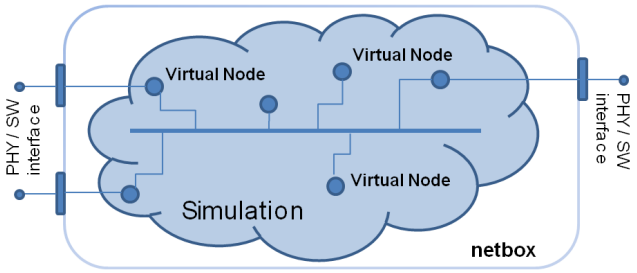

Figure 1.1: The conceptual schema of a netbox: a whole network is simulated within an insulation container and communicating with external entities or other netboxes by physical or software interfaces ©2012 Elsevier B.V.

The testbed is designed by following a new approach and its novel architecture aims at offering several advantages. First, it is modular, to let the designer define complex scenarios by combining simple building blocks. Container-based virtualization is employed to create a first level of protection among netboxes: the computing resources of the platform (mainly, the available CPU-cores and the memory address space) can be parceled and decoupled among them, so that their processing co-interference is avoided. To further reinforce this encapsulation, the network emulation logic (i.e., the operational parameters of each simulated network) is completely boxed-in within each corresponding netbox, and no network emulation processing is executed out of their boundaries. In this way, the obtained virtual networks are fully autonomous and self-contained, and can be created and employed as if they were hardware emulation devices. Furthermore, thanks to the kind of emulator that we selected, NetBoxIT can support the realistic modeling of several network standards, and offers a real-time management of incoming packets, with a controlled timing drift against the represented real network, so that data traverse the virtual networks with the same latencies that would occur in reality. Also, NetBoxIT is an open platform, since it can be transparently attached to external equipment, nodes, and true-world applications, so that the realism of simulations can be further increased. Finally, it is low-priced, being based on PC-class hardware and open-source, "off-the-shelf" software only.

This work is organized as follows: in Chapter 2 we give an overview about Emergency Networks in general, starting from their current deployments worldwide, and then discussing the trends and the emerging communication technologies that could be employed in the future; moreover, we describe the telecommunication infrastructure which has been defined within the EU 7th FP E-SPONDER project "A holistic approach towards the de-

velopment of the first responder of the future". In Chapter 3 we illustrate the traditional approaches to the general problem of network design and emulation, together with a brief survey of the existing simulation testbeds employed in this research field. In Chapter 4 we discuss the virtualization methodologies currently available, and we report the comparison assessments that we carried on for the selection of the most appropriate techniques to employ for NetBoxIT. In Chapter 5 we detail the architecture and the (hardware and software) components we used for our testbed, and the motivations that led to their choice. In Chapter 6, using the E-SPONDER infrastructure as an exercise, we focus on the most representative experimental trials we performed, with the aim to prove that the testbed is feasible for the heterogeneous networks modeling. Moreover, we bring to light the performance constraints of NetBoxIT, the improvements that are applicable today, and the advances that could be introduced in the near future.

Chapter 2

Emergency Networks: state of the art and trends

Under the circumstances of a disaster, public telecommunication infrastructures are usually not trustworthy enough for rescue services. These networks can easily become overloaded, or can be damaged and turn out to be unusable. For this reason, the preferred choice to support the emergency personnel communications nowadays is to utilize reserved Professional Mobile Radio (PMR) infrastructures like TETRA (Terrestrial Trunked Radio) or APCO P25 (Association of Public-Safety Communications Officials - Project 25). The European TETRA, an ETSI standard adopted in more than a hundred nations worldwide, is mainly destined to speech and messaging among aid operators (fire fighters, security forces, etc.), and offers a basic, low-rate data transmission (3-14 Kb/s). Its main advantages are the high spectral efficiency, the support for point-to-point and point-to-multipoint digital communications, the fast call set-up, the autonomous relaying among users, and the adoption of encryption schemes to ensure the confidentiality of information. P25 is a TIA APCO standard widely adopted in North America by federal and local public safety agencies. It is an analog/digital system with a preponderant attention for interoperability, which is even possible with analog legacy radios. However, P25 and TETRA are not compatible with each other. The major limitation of P25 is represented by its cost, since a radio equipment can be easily priced some thousands of dollars. Moreover, in recent years, some security flaws have been reported, mainly related to the DES-OFB and ADP encryption schemes, to jamming vulnerability, and to possible traffic analysis.

However, many recent experiences have clearly demonstrated that the safety and cor-

rectness of emergency operations are strictly related to the accuracy and timeliness of information that is made accessible to rescuers. For instance, the rapid availability of video streams, images, terrain maps, etc. can largely improve the situation awareness within the command and control chain: this facilitates the identification of the most appropriate actions, and reduces the possibility of errors and misunderstandings. Under this perspective, present-day PMRs are starting to show their limits, and their low data rates seem the main constraint to improve the effectiveness of first responders. Further, PMR networks have a fundamental problem: since they rely on fixed, ground relay stations, they could be absent in a particular geographic area, as well as they are exposed to be damaged by the calamity itself. On the contrary, first responders (FRs) and their vehicles (MEOCs, or Mobile Emergency Operations Centers) should operate with no concern for the self-sufficiency and stability of their internal communications.

2.1 Emergency Networks, current requirements and future opportunities

As just stated above, existing PMRs offer rate-limited, simple communications capabilities. On the other hand, public protection agencies show a rising interest for exchanging enhanced information (video, maps, etc.) and for a better reliability and quality of communications with the aim to improve the effectiveness and the safety of operations. Under an organizational point of view, the following general requirements arise and should be fulfilled by the system architecture of an Emergency Network:

1. interoperability;
2. reliability;
3. flexibility;
4. scalability.

A possible approach is to design such an infrastructure by the functional and/or physical decomposition in several subsystems, each offering different features, and following a hierarchical and layered scheme. Figure 2.1 presents a high-level overview of an

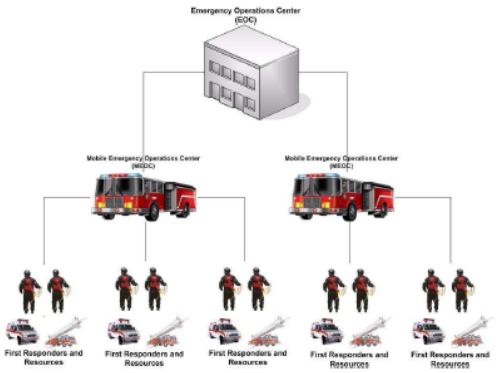

Figure 2.1: An abstract overview of an Emergency Network, where the EOC, the MEOCs, and First Responders are represented ©2010 IEEE

Emergency Network infrastructure, where all the possible key elements are employed: the Emergency Operations Control Center (EOC), the Mobile Emergency Operations Centers (MEOCs) and the First Responders. The overall system architecture is therefore based on the design and integration of its three main parts, i.e., the FRU (First Responder Units), the MEOC and the EOC.

Communication is critical for First Responders' (FRs) operations, since for them it represents the means to connect with the command and control centers at any time, and exchange the information that is necessary for the decision making process. Interoperability is an important property, since it permits and draws to the so-called "all connected" context, where there is a continuous contact among the first responders and the operation centers. FRUs equipment should conform to this major requirement, and be based on flexible radio interfaces that can comply with the most varying user needs and constraints. It seems also important to follow a modular approach for the design of both the software and the hardware components of the FRUs, to disjoint the development of the users applications from the other parts of the terminal device (e.g., the indoor/outdoor navigation module, the environmental sensors monitor, the telecommunication module that will be in charge of the inter-operation with the MEOC, and so on). The MEOC will act as a virtual relay among the first responders acting at the incident field and the EOC personnel located at the remote command headquarters. Hence, the MEOC has to be designed to deal with several, different issues, and many different requirements must be taken into account: communications equipment and IT infrastructure, interoperability of systems and services, security of communication and data exchanges, the proper selection of a vehicle able to host the nec-

essary equipment, and the ability to locally perform some of the EOC functions in a mobile condition. The EOC somehow represents the spine for the operations of the E-SPONDER system. In fact, the EOC is presumably located at the headquarters of the public protection agency, where all the strategic decision are taken, and therefore is involved in the collection, selection, analysis, and dissemination of all the information generated at the disaster site and forwarded through the MEOC. Several fundamental components are usually positioned at the EOC, of course the communication and IT infrastructures, as well as the Geographical Information System (GIS), the web portal, the Decision Support System, etc.

Commonly, after the design of the discrete components, it is inevitable to integrate them and assemble the overall system, and therefore a particular care should be devoted to the interactions among the FRUs, the MEOCs and the EOC. In particular, inter-communications issues must be analyzed and solved, so that the different system entities interact and cooperate smoothly. More in detail, communications interoperability in an Emergency Network is a fundamental asset that enables to coordinate the emergency response more effectively, not only within the single protection agency, but also across agencies from different jurisdictions. For instance, flexible and re-programmable radios can be useful to operate in a wide range of spectrum bands and using different protocols, since the personnel at the local level or in rural areas could employ different equipment than state agencies (or even neighboring jurisdictions). It is important to observe that an Emergency Network, besides being capable of inter-operating with heterogeneous network entities, is frequently structured as a Wireless Mesh Network (WMN, [1, 2]): in fact, the transmitting nodes are characterized by mobility, they can (directly or indirectly) communicate with each other, and the whole infrastructure might be connected to the rest of the Internet as well.

To facilitate the propagation of the local information from the crisis site towards the remote EOC (and vice versa), satellite communications can play a crucial role. The ability of the satellite signal to reach even the most isolated regions (especially those where the traditional network infrastructures are missing or disrupted) is fundamental in ensuring the continuous monitoring of operations and in supporting the work of first response crews. If we refer again to Figure 2.1, data from the FRs can be transmitted in real-time to the local MEOC, where it can be filtered, aggregated and relayed to the remote EOC. Therefore, the connection among FRs and the EOC can be guaranteed through the use of satellite communications in any moment. Modern satellite equipment operating in the Ku or Ka bands

are highly portable and can be carried in almost any crisis site via land or sea transport vehicle. In this manner, voice and data transmissions can be deployed to any (outdoor) location, including areas close to the incident site.

Together with the satellite coverage, other wireless communication technologies are good candidates for the Emergency Network infrastructures of the future, not only because they allow for mobility, but also for their low cost and flexibility. Flexibility is meant as "communications interoperability", which is is frequently seen as a fundamental issue for coordinating emergency response teams, particularly across agencies from different jurisdictions. Nowadays, the innovations in radio interoperability provide for individual devices that can communicate on a wide range of frequencies and using many different standards.

An other point to mention is that, even if modern technologies are continuously improving the reliability of wireless systems, transmission outages (e.g., due to obstacles, fading, or electronic failures) are always possible and could affect any of the links depicted in Figure 2.1, making part of the system inoperable. Therefore, the Emergency Network architecture should be designed and developed holding this issue in mind: the redundancy and replication of information paths seems the most appropriate solution to mitigate the effect of possible interruptions.

Another large set of constraints for the design of an Emergency Network comes from the analysis of the users' requirements. For instance, the wireless coverage should be arranged quickly at the disaster scenario, and provided by transportable, low-maintenance equipment. In addition, it should be able of organizing autonomously, with no human intervention, and it should offer enough capacity to support the real-time services cited above. In brief, inter-operable, self-organizing, broadband communication infrastructures are envisioned.

Finally, security is crucial: encryption, proactive management, and robustness in respect of possible links and nodes failures should also be considered during the definition of the network architecture.

2.2 The E-SPONDER Project proposals

To satisfy all these necessities, broadband, ad-hoc wireless mesh networks, created among all the possible players located on the crisis scenario, seem to be the best candidate for the

replacement of PMRs in the future. In WMN systems, FRs can configure their connections "on the fly" and communicate autonomously, with no need of pre-existing infrastructures. Once established, a whole FRs mesh can then get linked to the remote EOC (and to the Internet) through the nearby vehicular MEOCs, or (as a backup option) directly by means of any infrastructured public network possibly survived. Therefore, FRs terminal units should employ different radio interfaces, besides being wearable, energy efficient and with a long-life power supply. The MEOCs might employ different transmission channels (e.g., the satellite or, again, a public infrastructure) to create the back-haul link towards the EOC, as well as they might be conceived to form an intermediate layer of meshed links (by means of broadband inter-MEOC connections) that can further increase the robustness of the whole network (since the number of available satellite paths toward the EOC can be increased accordingly). Vehicular radios on board the MEOCs have less weight and power restrictions, but they also must be equipped with different radio interfaces to allow the creation of the various wireless communications just described.

2.2.1 The envisioned architecture

The E-SPONDER project perspective ([3, 4]) is to develop a real-time, data centric platform that will adopt several standards and technologies to enable the interoperability and reliability of communications, and that will be based on a hierarchical and modular infrastructure, to fulfill the emergency operational flexibility and scalability needs. In general, a large number of different network topologies and telecommunication technologies could be employed to reach these targets.

Figure 2.2 sketches a possible layered scheme of a typical crisis scenario (PAN, IAN, JAN, and EAN denote the Personal, Incident, Jurisdictional, and Extended Area Networks, respectively). This architecture is composed by:

- a number of personal area networks (PAN). A PAN comprises wireless sensors devices, used to gather information about the health of FRs (breathing, body temperature, blood oxygenation, etc.), their positioning, or environmental conditions (temperature, radiations, chemical compounds, etc.);

- a number of incident area networks (IAN). The first element within a IAN is represented by the FRUs, which act both as concentrators (for the PAN's data traffic)

and as user terminals (in order to provide the first responder with support informations). Moreover, the IAN includes one or more MEOCs. Figure 2.2 depicts some possible topologies: for instance, FRs can be connected to their MEOC using a star-based network; or, they can establish a mesh, to increase the coverage and the reliability. More interestingly, a IAN can also be structured with a hybrid topology, e.g., with a mesh for intra-FRs communications, but with the team commanders equipped with long-range radio device to connect the whole FRs mesh to its reference MEOC. To resume, we can describe the whole resulting network topology as an inter-technological wireless mesh network (WMN);

- a jurisdiction area network (JAN); this is the back-haul network for the traffic generated by the IANs. A JAN consists of fixed infrastructures (which can be used as a backup), together with the geostationary satellite, which is employed to provide the main interconnection path with the EOC. It is worth noting that the future next-generation satellites will allow the direct meshing of land users, with no involvement of relaying paths through the provider ground stations: this will offer an additional degree of reliability for the Emergency Network, as well as a reduced level of latency for the traffic among the MEOCs and the EOC;

- an extended area network (EAN); this is any possible geographical network which provides the Internet connectivity for the EOC. In the E-SPONDER infrastructure this will fundamentally act as a backbone for different JANs.

A IAN, in general, can include one or several MEOCs and FRs teams. At the EAN/JAN level, the MEOCs provide for the main back-haul link to the EOC using a geostationary satellite, or take advantage of any reachable infrastructured network for backup links. The satellite can also be used to enhance the resiliency of the MEOCs mesh: for instance, if the satellite connection is locally faded out by heavy rain, snow or smoke, it is possible to route all the data traffic through a dedicated intra-IAN wireless link to a different MEOC, which, instead, could still be reached by the satellite (typically, heavy weather conditions are statistically limited in space and time).

It is worth noting that such an outage can be likely in many crisis scenario, so it is relevant to reduce the need of using the satellite link as much as possible. Therefore, caching

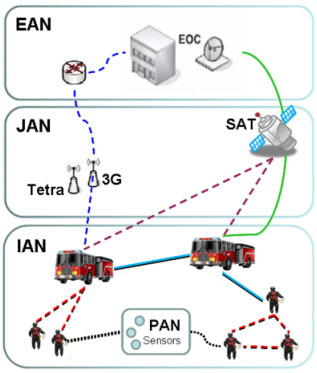

Figure 2.2: The multi-layer scheme of the E-SPONDER network infrastructure ©2011 IEEE

solutions to store all (or most) of the significant data available at the EOC within the MEOC can be extremely helpful to make the local operations more independent. Moreover, data sharing among different MEOCs can further reduce the use of the satellite link.

The modularity of the E-SPONDER architecture makes easy to adopt different technologies to implement the various portions of the network and helps the realization of a flexible and scalable infrastructure, where a proper number of MEOCs and FRs can be positioned at the crisis zone depending on the actual need.

2.2.2 The role of radio technologies for First Responders communications

As stated above, the E-SPONDER project aims at conforming to an "all-connected" approach, which is focused on the reliable interconnection among the FRs, the MEOCs, and the EOC. Nowadays, there are various standard wireless technologies that can be integrated to ensure the best possible connectivity, and to satisfy the requirements previously discussed. In particular, a possible choice is:

- a IEEE 802.11 network (star- or mesh-based) for the first level FRs network (i.e., among the team and their commander);
- the IEEE 802.16 standard to link the team commander and his/her whole team to the MEOC, and for an optional meshed backbone among MEOCs. The standard proposes several different profiles: the Time Division Duplexing profile (where the up-link and down-link capacities can be dynamically adapted to the traffic variations)

is not the most efficient, but certainly seems the most flexible and appropriate for an Emergency Network; moreover, the TDD profile is explicitly conceived to support the creation of WiMAX mesh topologies;

- the ETSI DVB-RCS and the LTE/3G/TETRA standards to support, respectively, the main (broadband) satellite and the backup (infrastructured) back-haul links among the MEOCs and the EOC and/or public Internet;
- the IEEE 802.15 standard for the data retrieval from the FRs sensors and positioning device (so that commanders can keep the environment and the staff health constantly monitored).

Another option for FRs connectivity could be given by LTE femtocells (in place of IEEE 802.11): even if initially conceived for small indoor environments, this technology could be easily adapted to this particular field of applications, reducing the number of radio interfaces of the FRU. Finally, the ETSI DVB-RCS Next Generation ([5]) could be adopted in the future, offering IP-based transport, refined QoS management, VoIP optimizations and mesh support (i.e., the option of direct links among MEOCs via satellite relaying), introducing an additional degree of resilience to the system.

To achieve the E-SPONDER targets, the system should be designed to be easily deployable and to require only limited external configuration. This constraint is imposed by the unique characteristics of a crisis event, which is abnormal by definition. Certainly, it would be significant if the first responders were able to form an autonomous communication infrastructure (in an ad-hoc manner) from the very first moment, to be immediately operational: this is possible if the devices carried by first responders are able to self-discover and self-configure. In a few words, we can say that the paradigm of autonomic networking is foreseen to play an indisputable role within the E-SPONDER ecosystem. Once the network is organized, topology changes should be monitored continuously, to select the optimal routing paths, preferably in a proactive manner. Additional robustness can be introduced within the system by the means of cooperative transmission techniques, based on MIMO diversity mechanisms, dynamic beam-forming, or relaying.

Chapter 3

Methodologies and trends in Network Design

A large spectrum of methods can be utilized for networks design, ranging from mathematical models to real-world testbeds. Mathematical queuing theory is the most abstract technique, normally proficient for qualitative assessments, but it can result into unmanageable levels of complexity in many circumstances (e.g., for the modeling of mobile nodes within mesh networks). Physical testbeds, that use real equipment, are the most trustworthy, but can be impracticable for the study of multi-technological networks, since the time and costs due to their implementation rises quickly, as well as they are not always compatible with a laboratory environment (in particular when the study is focused on wide-ranging radio networks, for instance, a cellular infrastructure or a satellite link). Hardware emulators are represented by special-purpose, standalone equipment: they are usually reliable to mimic the behavior of a certain network and can be possibly cascaded to re-create a heterogeneous environment. However, these devices are usually expensive, and frequently they cannot be customized by the researcher. Simulation tools are probably the most popular choice: they are flexible and adaptable, cheap to deploy, and can offer repeatable evaluations, as well as they can (even if, not always) provide for trustworthy networks modeling. However, the simulation of intricate and compound infrastructures commonly demands a large quantity of computing resources; furthermore, software simulators cannot be integrated with real-world entities, since by definition they lack the entry points to exchange information with external applications and systems.

In the recent past software network emulation has started to obtain a lot of interest

in the networking community. The strategy behind it is to adopt a hybrid approach then, where a software simulation (running on a general-purpose PC system) is combined and exchanges information with true network entities, with the target of increasing the accuracy of results and to accomplish the examination of a certain system against real traffic. As native simulators, software emulators are cheap and swift to configure; similarly, their realism is strictly related to the accuracy of the internal network models. Nevertheless, a central difference is that an emulator does not simulate the data exchanges in virtual time units, since it is compelled to coexist with real-world external players. To accomplish this task, the emulator should be provided with enough computational resources to observe the real-time execution constraints when processing the ingress data: if real applications inject or receive some traffic to/from an emulated topology, the emulator must guarantee not to introduce any extra latency during its processing (or, if so, that this is limited and measurable). There are consequently some crucial conditions to respect, primarily for the choice of the hardware and software components of the emulation platform.

In theory, it would be perfect to create an evaluation framework that captures the best of all these methodologies: the flexibility of simulators; the modularity and scalability of hardware emulators; and, finally, the realism of physical testbeds. In addition, it should be configurable, programmable, and manageable through open interfaces. Finally, it would be amazing if it was technologically close to the target system, so that little effort is required in order to transform it in a working prototype. We have tried to develop NetBoxIT as the synthesis among all these targets. Consequently, a framework that aims at being:

- a general-purpose, flexible instrument, capable of supporting the study of heterogeneous networks;
- modular: the “netbox” (i.e., the “network in a box”) in Figure 1.1 should synthesize a whole, independent network segment (as if it was a hardware network emulator), and should be entirely insulated, with its own exclusive computational resources, so that each can be fully separated from the others during their concurrent execution;
- open, that can interfaced with external devices, networks, nodes and applications by standard physical interfaces;
- based on cost-effective, commodity hardware, with no dedicated, special-purpose devices (e.g., FPGA or DSP cards), or involving large-scale distributed platforms;

- based on "off-the-shelf" and reusable software elements only (and avoiding complex customizations);
- with a high-quality level of realism, possibly validated against real-world systems, in particular within the PHY/MAC layers;
- real-time, with no insertion of temporal overheads during the emulation of a network; moreover, "time warping" techniques [6] should be avoided: even if they can scale up the dimension of the simulation limitlessly, they become clearly inappropriate if the testbed must be connected with true-world applications;
- efficient, hence employing the minimum quantity of computational resources, to enhance the scalability.

3.1 Background

Traditional assessment tools are founded on the usage of physical devices or simulation environments, but, considered separately, these are not always adequate to fulfill the above-mentioned needs. For instance, a wireless physical testbed is obviously purely realistic, but it can also be expensive to be deployed, especially if novel technologies or protocols must be experimented on a large scale or for nomadic scenarios. Moreover, the obtained results can be sometimes impossible to be reliably reproduced, because of the uncertain nature of the radio channel. The ORBIT Testbed [7] is probably representing the state-of-the-art in the context of this kind of approach: basically, it employs a huge, re-configurable grid of IEEE 802.11a/b/g nodes placed in a large reserved hall under controlled boundary conditions.

As stated above, simulators are largely practiced, since they offer an adaptable, and repeatable modeling of a certain network scenario, but they are also far from being useful at the implementation stage. Nowadays, there are several simulators available, for instance NS-2, OMNet++, QualNet, Opnet, NS-3, just to name a few. But, not all are open or can provide emulation capabilities (e.g., be able to exchange true packets with a true network). Among these, NS-3 ([8],[9]) is currently providing the best simulation performance and minimal memory footprint[10], even if it is still missing the significant number of models of NS-2 [11].

To find a possible way to join the benefits of the physical and simulation techniques, many hybrid testbeds have been proposed in the past years. The WHYNET/TWINE [12], MiNT [13], MeshTest [14] and EWANT [15] projects followed this strategy, and combined the simulative, emulative, and physical approaches altogether for the design and evaluation of wireless networks. Other testbeds (e.g. Dummynet [16], NetPath [17]) are focused on the use of "virtual links" for the modeling of a network and as a means of interconnection among physical or virtual nodes. This strategy can be followed both on single-machine testbeds, and on distributed ones. In any case, special-purpose software elements (commonly, shapers, customized tunnels, or bridges) are to be specifically designed and employed to mimic the nuances of the (real) network (propagation delay, bandwidth, packet latency, packet loss probability, bit error rate, etc.). ModelNet [18] is also a good example of this vision: it is based on a modified version of Dummynet, and it probably represents the earliest attempt to emulate entire networks on a single host. A set of user applications (running on a number of physical nodes) can exchange information through a set of "core" nodes that are used to simulate the bandwidth, latency, and loss profile of a target network topology. The network emulation is performed within the core using the ipfw firewall and the ModelNet specific kernel module: the firewall selects the ingress packets, whilst the kernel object moves the datagrams into a sequence of "pipes" that mimic the emulated topology characteristics (it can be noticed how this technique represents a field of application of the queuing theory). The ModelNet scheduler runs within kernel land at high priority and is based on a 10Khz clock tick (i.e., with 100 µs granularity) to move packets from pipe to pipe. As observed in [16], this approach usually loses the precise representation of the MAC layer behavior, for instance, the effects of framing (e.g., check-sums insertion), channel access method, or frames retransmissions. This coarseness of the MAC layer representation can be overcome by introducing a probabilistic model to adjust the frames latency (however this additional element must be re-engineered every time a different MAC protocol is employed). Netkit [19] is a lightweight, flexible emulator (based on User-Mode Linux [20]) that provides a set of tools for the setup of complex network scenarios. Virtual nodes are created within a customized Linux virtual machine (that runs as a user-space process) and are inter-connected by means of virtual hubs (the "uml_switch"), which substitute the behavior of a switch or hub device. Moreover, these virtual nodes can exchange traffic with external networks, because they are provided with TAP devices connected to the physical NICs. Nevertheless, an uml_switch is basically implemented by

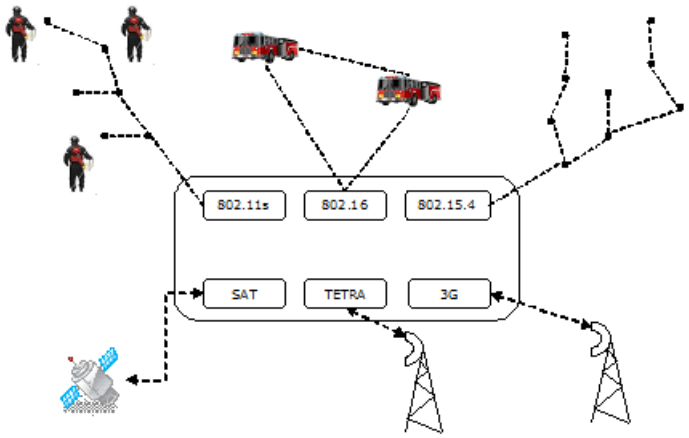

Figure 3.1: The MEOC connectivity scheme, with the envisioned radio interfaces ©2010 IEEE

a user-space (daemon) process, that receives packets from the virtual nodes via UNIX domain sockets, a design choice that seems not strictly suitable for the real-time emulation. Therefore, the best context of application for Netkit appears didactics.

3.2 A hybrid approach to the study of a Mobile Emergency Operations Center

In the E-SPONDER project vision, the MEOC telecommunication subsystem is fundamentally required to act as an active relay among several, different types of networks. It should acquire, trans-code, and distribute all the information produced or to be received from different players (FRs, packet radio users, sensors, the EOC, and additional MEOCs operating on the field). A MEOC-centric perspective is depicted in Figure 3.1.

One of our first targets was to assemble an assessment tool for the studying of the depicted infrastructure, since using a proper evaluation tool from the very beginning could have been a strategic breakthrough towards an optimal E-SPONDER infrastructure. But, one that could be arranged quickly, with no long-term prototype integrations, and that could be immediately used for an a priori evaluation, so that it could help us to steer the succeeding design and development stages more consciously.

At the best of our knowledge, there is no existing hardware or software all-in-one equipment that is ready to inter-operate with all the networks depicted above. The MEOC

gateway should, in fact, include several interfaces, to deal with:

- the satellite link;
- the LTE, 3G, TETRA base stations;
- the Wi-Fi first responders network;
- the WiMAX network (for the interconnection between the MEOC and the FRs commander and/or the other MEOCs);
- a 802.15 sensor network (for the environmental monitoring).

Moreover, in our choice of a testing environment for the MEOC, the following aspects were also taken into account:

- the internal MEOC routing subsystem should offer proper performance in terms of forwarding rate among the different radio networks;
- we needed to estimate which is the adequate capacity of the access and back-haul wireless links, to sustain the aggregated traffic volumes exchanged among the command center and first responders;
- traffic differentiation and QoS policies needed to be studied, to guarantee the smooth coexistence of the different applications (e.g. voice calls, live video streaming, GIS maps retrieval, etc.);
- the testbed should be technologically as close as possible to the final equipment arranged on board; better yet, it was highly desirable to follow a step-by-step approach, so that the initial testing tool could evolve and progressively employ the final scenario depicted in Figure 2.2.
- and, of course, flexibility, scalability, and realism.

Because of all these purposes, we were particularly determined to adopt a hybrid approach, which seems the most feasible to our needs, and to follow this gradual development path:

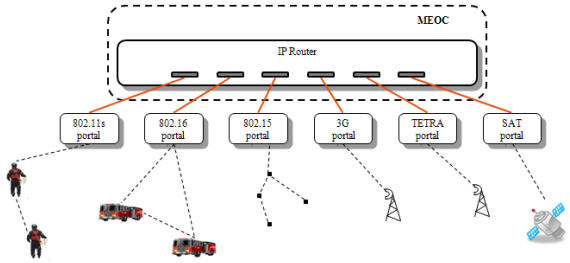

Figure 3.2: The hybrid approach to the MEOC design: several wireless networks modems are connected to the IP router by the means of Ethernet links ©2011 IEEE

1. as a first step, instead of considering the router and its various radio interfaces all together as a unique device (as in Figure 3.1), we split the functionalities of the MEOC equipment in two subsystems: a standard router, provided by conventional Ethernet interfaces only, and a set of radio devices (or, in general, whatever kind of network portals). In brief, we extract the wireless interfaces from the router and move their circuitry within a set of external radio modems (of course, Ethernet has to be considered just as an intermediate protocol among the modems and the router). It is worth noting that this approach is not just a theoretical expedient to simplify the problem, but it can also be a practical implementation method in many real-world situations. Figure 3.2 exemplifies this scheme;

2. the second step is to replace the physical modems (and their related networks) with a certain number of network emulators (one for each of the networks to be emulated). These are in charge of generating or collecting the traffic going/arriving to/from the router, and, of course, to mimic the behavior of the corresponding network. Figure 3.3 depicts this step;

3. finally, for a partial or complete transition toward a true prototype, the emulators can be progressively replaced by the real-world networks, by attaching the true modems or NICs to the router, bringing back all the MEOC functionalities within an all-in-one equipment.

It is worth stressing that in case of such a complex and heterogeneous infrastructure, the preliminary interconnection of the MEOC router with all the radio networks, in the attempt to pursue the immediate implementation of a physical testbed, would demand an intense (and expensive) effort to be carried on with no clear perspectives about the final

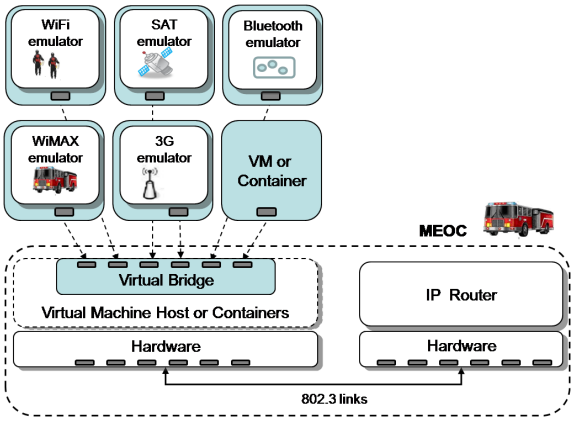

Figure 3.3: The hybrid approach to the MEOC design: physical radio modems (and their related networks) are replaced by software objects (network emulators) in charge of emulating the portions of the Emergency Network ©2011 IEEE

results. As an example, the straight integration of the MEOC router into a IEEE 802.11s mesh network implies that it should behave as the mesh portal, and therefore it should comply with all the 802.11s MAC layer mechanisms (e.g., it should manage the Hybrid Wireless Mesh routing Protocol, the MCCA QoS preservation, the congestion control, the Simultaneous Authentication of Equals security authentication, etc.). At the time of the E-SPONDER project kick-off, the IEEE 802.11s protocol was not implemented on any commercial device on the market (and the IEEE was still far from publishing the final specifications). In such a situation, a working MEOC prototype could have been possible only by a from-scratch activity of software development (mostly dedicated to modify the firmware of a common Wi-Fi card). However, some software 802.11s simulators were already existing and used by the research community for studying purposes, and could be employed to conduct some (rough, but useful) evaluations within an emulation context.

To resume, in many situations, a hybrid approach can be advantageous to shift in the future the time-consuming task of the prototype implementation, insulating the core functions of the equipment from the "network interoperability" puzzle, and allowing the exploratory study of the planned infrastructure from the very beginning.

Chapter 4

Virtualization and Emulation Techniques for the deployment of Virtual Networks

Hardware virtualization is fundamentally based on the abstraction of the functionalities of a physical component. The benefits of virtualization are particularly noticeable on recent, multi-core or multi-CPU PC hardware, where the constant addition of computational capacity permits and encourages the multiplexing of the physical resources among several virtual systems (Virtual Machines or GuestOSs). Therefore, in a virtualization platform a special software layer (called the Virtual Machine Host - VMH, or the hypervisor, or the Virtual Machine Monitor - VMM) is committed to split the low-level shared resources among the high-level operating systems (and the applications running within them). In short, virtualization recreates in software a number of emulated hardware platforms, on which the operating systems run in the illusion of being on a real machine. From the opposite viewpoint, virtual machines do not attain the hardware resources straightaway, but exclusively through the VMH. Virtualization can bring various advantages, e.g., improved security, central management, live backup and migration of GuestOSs, low-costs, energy efficiency, just to mention a few. However, some disadvantages can also arise: a virtual guest can suffer the performance decay related to the interference with the concurrent guests, as well as the VMH inevitably introduces an overhead that can slow down the access to CPUs, memory, hard disk, network cards, etc., by the GuestOSs.

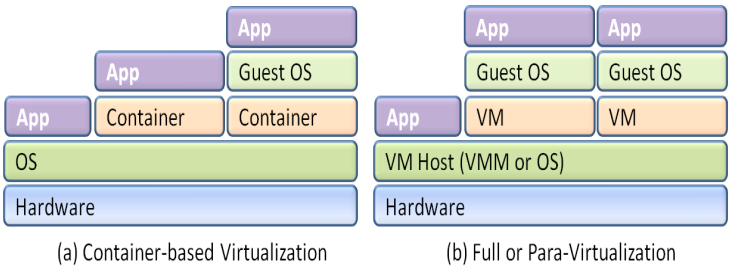

Figure 4.1: Containers vs. traditional virtualization techniques ©2012 Elsevier B.V.

4.1 Traditional vs. container-based virtualization

Nowadays, two similar methodologies are primarily employed for virtualization: the first choice is to run the VMH directly on the bare hardware (this is the kind of VMH named "native VMH"); or, a second option is to have the VMH running on top of an underlying operating system (in this case, the VMH is a "hosted VMH"). Native and hosted VMHs offer the so-called "full virtualization" if their GuestOSs are installed and running with no modifications: in these situations, the physical hardware is entirely hidden and exposed to the GuestOSs in a virtual form. In other words, the VMH is in charge with the complete emulation of any underlying physical device. This technique is the simplest and the most inter-operable, but not the most efficient, since the VMH intervenes in any access to the physical resources and its overhead inevitably affects the functioning of the GuestOSs. In the attempt to improve the performance, a viable option is to modify the GuestOSs, so that these can use a special API that provides the direct access to some hardware resources, with no VMH involvement. This methodology is usually called "para-virtualization" and it offers the reward that the overhead is much smaller. VMware, VirtualBox, and KVM are famous virtualization platforms that employ the full virtualization, whilst XEN is a well-know example of para-virtualization.

Containers are a more recent virtualization method (also named OS-level virtualization). It was developed as an enrichment of the UNIX "chroot" security mechanism in the beginning, but then it was enhanced to provide more isolation and the complete control and management of the resources. Differently from traditional virtualization techniques, where the installation of a whole Guest Operating System is compulsory, containers can be employed to host even a single, standalone application alone (see Figure 4.1). The key idea behind containers is that the kernel of the hosting operating system is revised to per-

mit the coexistence of multiple isolated namespaces, instead of just one. Moreover, this environment is enriched by several resource management features, which can be employed to avoid (or limit) the effect of one container's activity on the other ones. This virtualization technique usually brings a small overhead, since an application running within a container can use the native kernel API interface, and consume the physical resources directly, with no hardware emulation and without the additional involvement of the VMH. OpenVZ, Linux-VServer, Virtuozzo, FreeBSD Jails and LXC [21] are the most practiced container-based solutions.

4.2 Virtualization applied to network emulation

During the past decade, the research community has begun to combine the software emulation and the virtualization techniques for the deployment of the so-called "virtual networks". A well-established practice is funded on the employment of a number of traditional Virtual Machines (VMs), running within a single (or a small set of) PC, interconnected by a set of software links that mimic the networks among them. In a certain sense, this approach exploits the availability of multi-processor platforms to re-create the same environment of a distributed testbed within a single physical node. Several research testbeds have been designed starting from this schema, typically employing full- or para-virtualized virtual machines. However, the execution of an application together with an entire GuestOS is a task that consumes a lot of computational resources, and these architectures usually exhibit scalability problems. In some experiments, this problem has been faced by the introduction of "time-warping" techniques [6], that slow down the clock tick for the virtualized applications: this trick allows to scale up the testbed complexity limitless, but also makes impossible the direct exchange of data between a virtual machines and a true-world external entity.

In recent times, container-based virtualization has started to attract some interest. Mainly, this is related to its lightweight execution environment, which can be used to execute even a single application alone, without the mandatory installation of a whole operating system. An interesting example is represented by the Trellis testbed [22], which exploits containers to run a set of concurrent, virtualized applications. This platform is somewhat similar to ours, but only from a technological point of view, since there are some basic distinctions related to how they employ virtualization: first, each container is used to host exclusively a

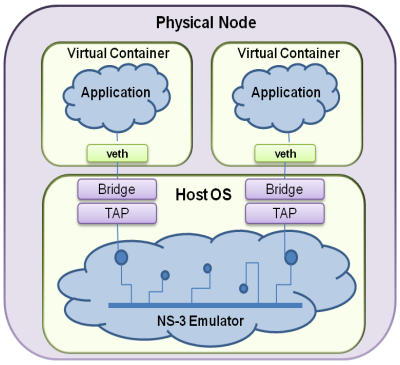

Figure 4.2: Virtual Networks built with a conventional employment of Linux containers ©2012 Elsevier B.V.

virtual router, and not an entire virtual network, as we will show; moreover, programmable virtual links (EGRE tunnels, created within the kernel space of the host OS, that can offer only plain shaping capabilities) are employed to mimic the networks among the routers. Another testbed comparable to ours is [23]. This is a widely adopted scheme, resumed in Figure 4.2 and published on the NS-3 Wiki web site. Here, containers are used to run a number of applications which are linked through a network emulator. It is worth noting however that in this platform the (single) emulator is not virtualized, but running within the raw Linux Operating System. A further little difference is represented by how the VMs are connected to the network emulator (i.e., using Tap interfaces).

There are some few examples of container-based testbeds. Mininet [24] is a framework for the quick prototyping of complex virtual networks on a single PC: the virtual nodes are represented by different virtual machines, whilst the network links are created with Virtual Ethernet peers [25]. To enforce bandwidth constraints and QoS policies to a network connection, the in-kernel Linux Traffic Control is utilized; however, only wired links can be simulated within this platform. CORE (Common Open Research Emulator, [26]) is an emulation framework that exploits Linux or Free BSD container-based VMs to create both wired and wireless virtual networks. Real-world applications can be executed without modifications, and possibly connected to true networks to extend the testbed with external equipment. However, CORE deals only with the strict emulation of the IP and higher protocol layers, and employs a plain simulation engine for the MAC and PHY layers. Virtual links are based on a tailored version of Linux kernel bridges, that are employed to set up the network bandwidth, latency, and probability errors. Moreover, on/off connectivity can

be utilized to re-create the impairments of a wireless network. To overcome its limitations in realism, CORE can be joined with other (underlying) emulation frameworks, for instance EMANE [27] and NS-3, following the same architectural schema drawn in Figure 4.2. Other similar integrations of containers and NS-3 are presented in [28] and [29], where LXC nodes communicate through a single NS-3 emulated network.

Differently from these examples, NetBoxIT introduces two architectural novelties. First, we reverse the conventional paradigm of Figure 4.2 and we employ containers to virtualize a number of network emulators (instead of applications), and run them concurrently (and not just a single one within the raw operating system) in a multi-namespace context. This idea aims at a modular framework, which seems more advantageous on modern multi-core systems, where we can split the overall CPU resources and assign them separately to each container (i.e., for the execution of several and concurrent virtual networks). Second, Ethernet is the only protocol utilized for the interconnections among (or, with) virtual networks. This choice is not only much more inter-operable, but is also useful for enhancing the realism of the testbed: first, NetBoxIT is ready to be coupled with a large collection of true equipment (switches, routers, wireless modems, etc.); moreover, it becomes much easier to attach real-world applications (running on external PCs) to the emulations (it is worth noting that this choice is also helpful to preserve some testbed computational power which can be more fruitfully reserved for the network emulations tasks only).

4.3 A performance comparison between LXC Containers, VMware, and VirtualBox

With the purpose of selecting the most appropriate software tool for the implementation of virtual networks, we carried on some experiments where a very simple testbed configuration was utilized: in particular, the VMH is configured to host a single VM instance, therefore creating a single, insulated execution context, to which we assign some predefined CPU resources and a single emulated Ethernet card. Moreover, inside this virtual machine, a NS-3 simulation is executed, wherein a single network node is created with the task of exploiting the VM virtual card to transmit a UDP-based packet flow. A kernel bridge is employed to bind the virtual card of the VM to the true physical Gigabit Ether-

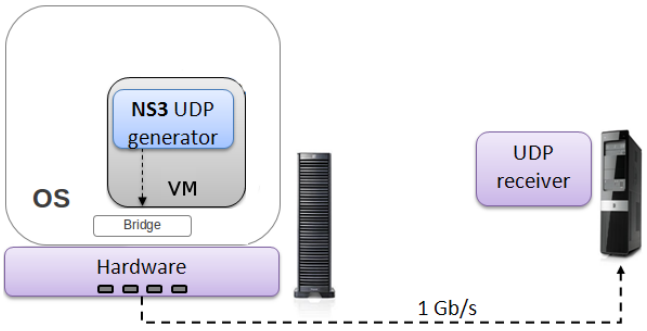

Figure 4.3: The testbed configuration employed to perform a comparison among different virtualization techniques

net card of the testbed. Finally, a second PC is simply utilized to measure the rate of the received packets. Figure 4.3 depicts the configuration employed for these experiments.

A first set of tests is performed using VirtualBox as the VMH, wherein an Ubuntu Linux operating system is hosted. Therefore, the NS-3 packet generator is executed within the Ubuntu environment. Then, we configure the generator to produce packets at an increasing rate, until the MLFGR (Maximum Loss-Free packet Generation Rate) is reached. "Loss-Free" means that all the generated packets are successfully delivered to the receiving PC. Figure 4.4 resumes these performance trials when an increasing number of CPU cores is reserved for the VM (the "debug" label is related to a configuration in which NS-3 was running in debug mode with 16 cores). The best MLFGR is 9800 packet/s, and it is obtained when only a single core is assigned to the VM. This is not entirely surprising: first, the NS-3 generator is not able exploit the multi-core architecture, since its source code is not multi-threaded; moreover, the use of several virtual processors tends to decrease the coherency of the physical CPU cache, since the VirtualBox scheduler is free to migrate the NS-3 generator across the different cores, so that each migration causes the cached data to be invalidated and to be re-fetched. Increasing the packet rate beyond the MLFGR threshold is scarcely helpful, and we verified that at the MLFGR the emulator is already consuming all the available (virtual) CPU provided by the VM. In a few words, the performance of the packet generator is clearly CPU-bounded.

A second sequence of trials is then performed using VMware Server as the VMH. Again, Ubuntu is used for the host operating system wherein NS-3 is running. However, differently from VirtualBox, this VMware release cannot assign an individual CPU core to

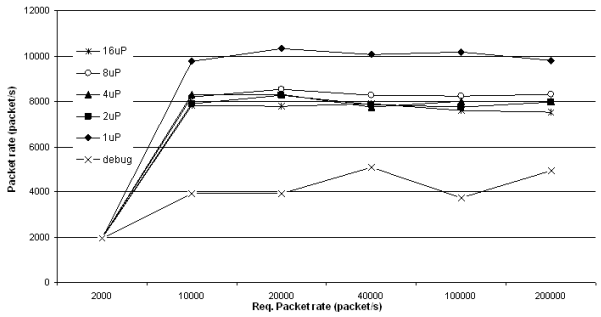

Figure 4.4: NS-3 over VirtualBox – Achieved vs. Required packet rate (64-byte UDP packets) ©2011 IEEE

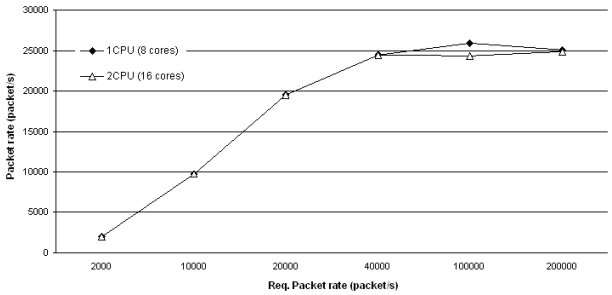

Figure 4.5: NS-3 over VMware – Achieved vs. Required packet rate (64-byte UDP packets) ©2011 IEEE

a VM, but only an entire CPU (i.e., a whole group of 4 physical cores). Figure 4.5 reports the outcomes of these tests: the obtained MLFGR is about 24600 packet/s, significantly better than VirtualBox. Once more, it can be noticed how the addition of further processing power (two CPUs vs. one CPU) does not help the performance.

Figure 4.6 illustrates what happens when LXC is employed. It is worth noting that now NS-3 is directly virtualized within its own container, with no hosting OS. Once again we increase the number of CPU cores per VM. To have an baseline reference, we also tried the NS-3 emulator directly onto the raw Linux operating system, avoiding virtualization entirely (the "no LXC" curve). Some significant outcomes emerge: first, LXC outperforms the other virtualization techniques, and reaches the highest MLFGR; second, independently from the number of CPU cores assigned to the VM, the curves are all coincident (below the MLFGR threshold); finally, the "no LXC" curve is also overlapped with the others, which suggests that the overhead introduced by LXC could be considered

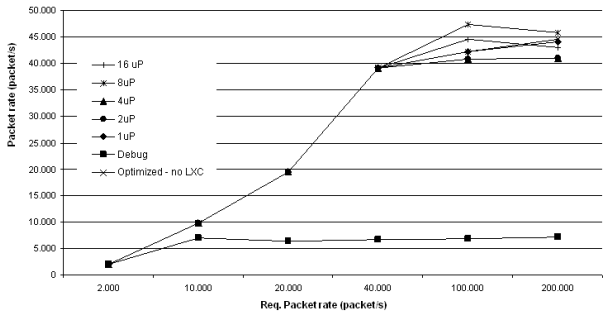

Figure 4.6: NS-3 over LXC – Achieved vs. Required packet rate (64-byte UDP packets) ©2011 IEEE

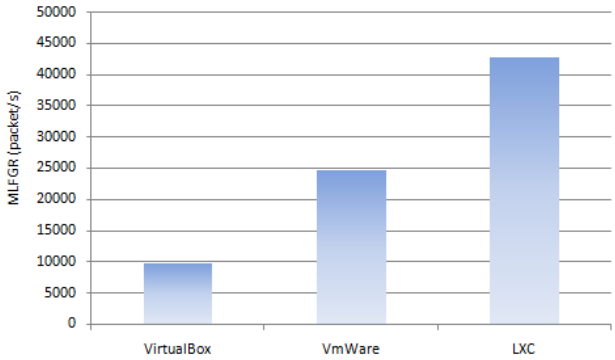

Figure 4.7: MLFGR comparison between VirtualBox, VMware, and LXC (64-byte packets)

negligible (or not existing at all).

In Figure 4.7 we sketch the general comparison of the best MLFGR values for all the examined virtualization techniques (see Table 4.1 - second column for the precise values), whilst in Figure 4.8 we summarize their performance figures altogether.

To complete our evaluations, we finally assessed each of the VMHs with a further run of trials (Table 4.1) where the packet generators are configured to send out 512-byte packets at the maximum loss-free rate registered with 64-byte packets. It is worth noting how all the three techniques scale up linearly when the frames size is increased, showing that the MLFGR is fairly independent from the packet size. Furthermore, LXC achieves the maximum throughput, about 175 Mb/s, which is an interesting result, since it shows that a single core per VM seems fairly adequate to deal with the traffic load possibly expected by any of the E-SPONDER network segments (e.g. 54 Mb/s from the Wi-Fi portal, or tens of Mb/s from the WiMAX/SAT links).

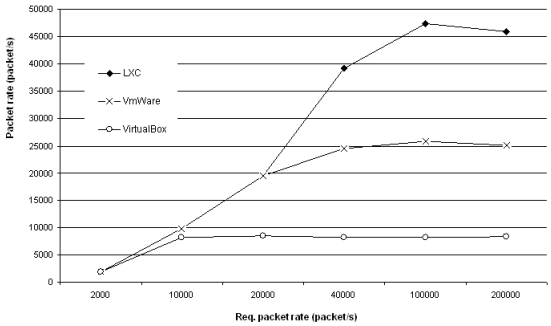

Figure 4.8: Achieved vs. Required packet rate - general comparison (VMHs are configured with the best possible tuning; 64-byte UDP packets)

Virtualization Technique	**MLFGR (packet/s) (64-byte packet)**	**Throughput (Mb/s) @ MLFGR (512-byte packet)**
LXC (1 core)	42870	175,6
VMware (1 CPU - 8 cores)	24600	100,7
VirtualBox (1 core)	9770	40,0

Table 4.1: Throughput of the virtualized NS-3 packet generator, at the MLFGR and using 512-byte packets ©2011 IEEE

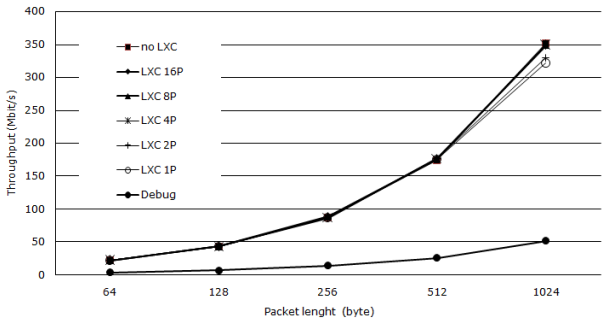

Figure 4.9: Maximum throughput of the NS-3 packet generator over LXC, at the maximum rate, and increasing the packet size (64-128-256-512-1024 byte)

To conclude, in Figure 4.9 we report the maximum throughputs obtained with LXC alone and with different frames sizes, to show that LXC scales up rather linearly.

Chapter 5

NetBoxIT hardware and software architecture

At this point, it seems useful to briefly recall the reasons that have initially motivated our activity and the goals intended to be attained. The Emergency Networks of the future need to be broadband and highly inter-operable infrastructures, where different technologies are employed at the different organizational levels of the security agency. This approach naturally brings to the design of heterogeneous networks, which are, however, complex infrastructures. And, their investigation can frequently be a striking challenge. A smart emulation tool can be helpful to support the designer's decisions, in particular if it is open and ready to be joined with real-world devices, so that the realism of the experiments can be enriched and true applications can be tested.

From this perspective, we thought to design and assemble a platform that is supported by open components, i.e., general-purpose hardware and "off-the-shelf" software; modular, where independent emulation entities (the "netboxes") with exclusive computational resources can be arranged similarly to hardware emulators, and run concurrently within a single PC; realistic, with a high level of trustiness, proved against real-world networks (for example, using software simulators already established to meet this target); real-time, that is, capable of providing real-time emulations; efficient, i.e., that utilizes the minimum amount of resources, so that the testbed scalability is maximized; and, finally, interoperable, to transparently exchange information (using standard interfaces and protocols) with external networks, devices, and applications. Moreover, to further enhance the testbed modularity and the re-usability of its components, NetBoxIT was designed to avoid the us-

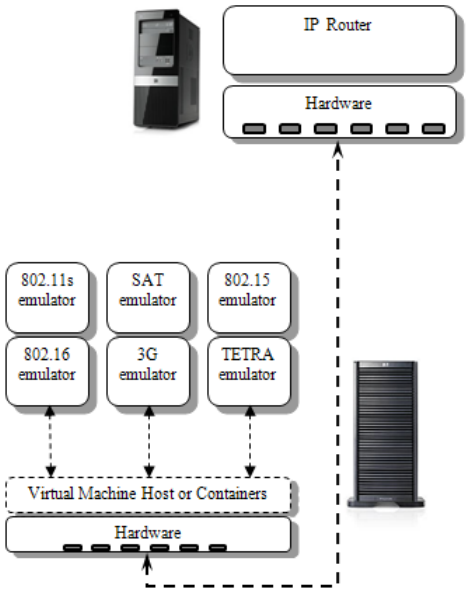

Figure 5.1: The functional scheme of the emulation testbed

age of special-purpose virtual links (that, in literature, are frequently employed to introduce delays, packet losses, etc. to mimic the real network behavior, but often with a questionable realism in comparison with several well-tested network simulators) to interconnect the virtual networks. In fact, such an approach requires the specific design and the introduction of ad-hoc software components. On the contrary, we aim at emulating the attributes of any network entirely within the boundaries of the related netbox, and at using only transparent and plain connections among them.

In the following of this chapter, we will present the hardware and software elements that were chosen to achieve the aforementioned targets, and the main reasons for which they were selected. It is worth remarking that we start from the conceptual schema depicted in Figure 1.1 as a reference for the design and implementation of a netbox. At the same time, we cannot forget that we want to use the netboxes to assemble the whole scheme of the E-SPONDER network depicted in Figure 2.2, and that the expected result is going to be a framework similar to the one sketched in Figure 5.1.

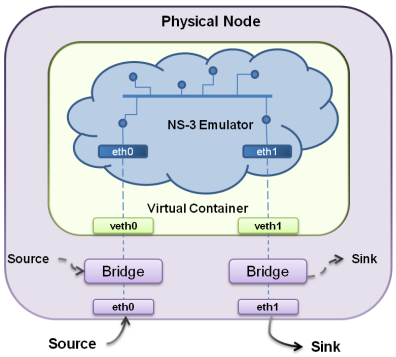

Figure 5.2: The architecture of a netbox: a whole network is emulated and encapsulated within an insulating virtual machine. A netbox can be interfaced to external entities by physical cards or directly to other netboxes by Ethernet software bridges ©2012 Elsevier B.V.

5.1 Netboxes internals

Figure 5.2 summarizes the architectural template and the I/O endpoints of a generic netbox. This schema tends towards the translation, in terms of concrete and existent software technologies, of the abstract entity presented in the first Chapter.

A netbox, or we should more properly say “a network-in-a-box”, is a programmable component, designed to behave similarly to a hardware network emulator. Its core is a whatever simulation engine that is capable to act as an emulator, i.e., where some nodes within the simulated network are provided with a virtual NIC and are able to produce and emit (or, to receive and inspect) true Ethernet frames. In our specimen, these virtual cards are bound to the virtual interfaces (veth0, veth1) of the VM wherein the emulator is running. It is straightforward to observe that, together, the emulator and the virtual machine constitute an ensemble which is equivalent to the abstract schema already represented in Figure 1.1. The input/output interfaces of the netbox can be either associated with the physical cards of the testbed (to exchange real information with external agents in real-time mode), or linked to other netboxes by the means of software bridges (instantiated within the Linux kernel). Hence, it becomes possible to put together the different portions of a heterogeneous network, starting from a group of different netboxes (each with an independent internal configuration) that are combined to form the overall infrastructure under investigation. In Figure 5.3 we show a cascade of three netboxes that synthesizes the emulation of a heterogeneous network within a single physical node.

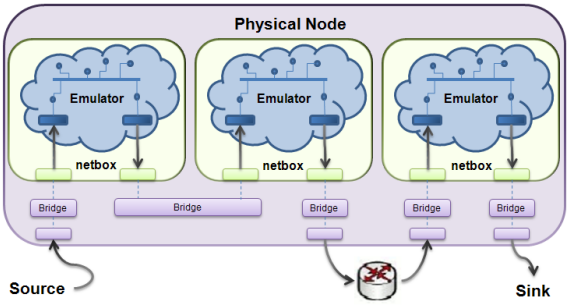

Figure 5.3: An example of interconnections between different netboxes (with external applications or through a routing equipment, and among themselves, using the physical NICs and software bridges respectively) ©2011 IEEE

To understand the architectural novelty introduced by NetBoxIT, it is useful to compare the schema in Figure 5.2 against the traditional approach depicted in Figure 4.2, where: 1) the emulator is not virtualized, but running as a single instance onto the raw Linux OS; 2) the virtual machines are employed to execute the users application, rather than the emulators.

5.2 Virtual machines and LXC containers

Once the internal schema of the netbox has been defined, it is then worth deciding which virtualization technology is (the most) feasible for the execution of the network simulator. Traditional full- or para-virtualization techniques are normally employed to run several operating systems (GuestOSs) concurrently. In our situation, one could certainly use a GuestOS to run the emulation environment within, and synthesize a netbox. Nevertheless, this aggregation does not seem the best solution, because the virtualization of a entire operating system for hosting just a single application consumes lots of computational resources. This assumption was preliminarily confirmed by our previous investigations (reported in Chapter 4 and in[30]), where different technologies (VMware, VirtualBox, and LXC Linux Containers) were assessed during the run of a mere traffic generator. We have seen that containers achieve the best loss-free throughput independently from the frames size, and, moreover, we noticed that they did not introduce any imperceivable computational overhead, since the performance did not differ from what obtained onto the raw Linux OS. Therefore, LXC containers have been elected as the key virtualization technology for our

testbed. It is worth noting that we are going to use them to split the overall CPU-cores among a number of distinct netboxes, and we want to do this in a scalable and modular fashion, so that an increasing number of concurrent emulators can be executed with no co-interference. This aspect is not to be forgotten when we will use containers in more complex scenarios, in particular when the overall E-SPONDER infrastructure will be emulated, since the imperfect (computational) encapsulation of the network emulators could bring to defective and unrealistic results, especially for the infringement of the real-time timings. LXC release 0.7.4 was employed during all of the following tests.

5.3 The NS-3 simulator

The next step is to select the emulation engine that demonstrates to be the most efficient and flexible for our needs. There are several simulators in the network design arena (in Chapter 3 we listed NS-2, NS-3, OMNeT++, QualNet, etc.), but very few are also exploitable for emulation. Among these, NS-3 immediately appears the most interesting. The first reason lies in the large number of available network models; what is even more amazing, the Wi-Fi and WiMAX modules provide a high-quality level of realism, validated from scratch against true-world systems, with the PHY/MAC layers specifically conceived to reproduce the behavior of the real equipment very strictly. Moreover, it exhibits two different emulated Ethernet interfaces, Tap and EmuNet. The latter is exploitable within our schema, since it permits a netbox to use the testbed physical NICs.

Furthermore, NS-3 is capable to run the simulations in real time, within a pre-defined temporal skew limit. In this manner, ingress data are handled by the emulator with the identical timing they would encounter when traversing the true network. Definitely, NS-3 does not represent a compulsory choice: analogous (or better) emulators could be plugged in the future; however, NS-3 seems nowadays the most efficient in terms of execution speed and memory footprint[10], besides being is open and free. Experiments have been carried on using NS-3.9.

5.4 The Click Modular Router

Commercial routers are commonly not extensible or open to modifications by the researcher, and frequently too expensive to be justified for a rough, preliminary evaluation at the prototyping stage. Instead, the network designer can benefit from the exercise of an open router, which can be modified and adapted with respect to a particular traffic pattern (e.g., multimedia communications), or wherein new functionalities can be experimented. The Click Modular Router [31] is a a Linux-based software framework initially developed at the MIT, that supports the arrangement of high-speed, modular, PC-based routers. Its main advantages are flexibility and extensibility, and various services can be created by simply connecting or modifying the basic libraries or creating new ones from scratch. Even if performance is not the main goal of the project, Click can be dynamically used as a kernel object, taking advantage of the kernel-space priority and substituting the standard in-kernel networking layers. As an example, Click was already exploited to examine [32] possible QoS and admission control schemes some years ago.

5.5 The hardware platform

NetBoxIT is based on a low-cost, entry-level server system especially suited for virtualization tasks, and employing a Dual Xeon 2.4GHz E5530 chipset. Therefore, it offers a total number of 8 physical, independent CPU-cores, since the two CPU sockets enclose 4 cores each. This server PC incorporates four single-port Gigabit Ethernet cards, all attached to the peripheral PCI-Express I/O bus. To perform experiments with external and true devices, this platform is wired (by Ethernet links) to an additional RFC1812-compliant software router (based on a Dual-core E2200 chipset), which can be possibly employed to forward the information between the distinct virtual networks. A Linux 2.6 kernel is installed on both these two PCs. Moreover, two auxiliary PCs can be exploited to run (real or synthetic) traffic sources, with the aim to create the traffic profiles required for the test of the virtual networks.

To resume, we planned to achieve our testbed design goals by the combination of general-purpose hardware and properly selected open-source, off-the-shelf software components. The hardware is based on a low-cost, server-class PC, where the multiple CPU-

cores can be split (and assigned in a rigorous manner) to allow many execution contexts (the netboxes) to run disjointedly. Besides, the platform employs a PCI-Express I/O bus, so that the overhead of internal data transfers is negligible. LXC is the technology that allows to distribute the available CPU-cores (as well as other resources, for instance, the NICs and the memory space) among the concurrent emulation tasks. NS-3 promises to be realistic, since many of its simulation models are validated against true networks from scratch: nonetheless, in the following we will be careful to examine if it offers trustworthy results even when used as a virtualized emulator. Additionally, NS-3 claims to sustain the real-time processing of packets, so that information can be managed with the exact timings they would meet in reality. What is more, both LXC and NS-3 seem very efficient: the former appears not to introduce any perceivable overhead, whilst the latter provides high simulation performance compared to the competitors [10]. Finally, with the aim to be fully inter-operable, NetBoxIT utilizes Ethernet as a means to support information exchange among netboxes and other external entities: this is particular helpful to attach the testbed to real devices and networks, to increase the realism of the investigations and to allow the evaluation of true applications.

Chapter 6

NetBoxIT applications and performance evaluation

In this chapter, we illustrate a selection of the experimental trials performed during the study of Emergency Networks. It is worth underlining that we do not intend here to show the overall outcomes of the E-SPONDER project, but we merely take its network infrastructure as an exercise, with the aim at proving that the testbed is suitable for the modeling of heterogeneous networks in general. Hence, we analyze the trials results under a particular viewpoint, focusing on the intrinsic properties of NetBoxIT itself, meant as an emulation machine. As a consequence, our discussion starts from the simulation of radio networks, but then moves on the testbed constraints, the improvements that were introduced immediately, and the emerging software technologies that could be applied in the near future to further improve its performance.

6.1 A case study: heterogeneous wireless Emergency Networks

In Chapter 2 we illustrated the general picture of the telecommunication infrastructure envisioned within the E-SPONDER project. Now, in Figure 6.1 we focus on the more particular and simpler network configuration that has been chosen for the pilot demonstration of the project, where only the main interconnecting paths are selected. On the crisis field, an IEEE 802.11 link is employed among the first responders and their commander,

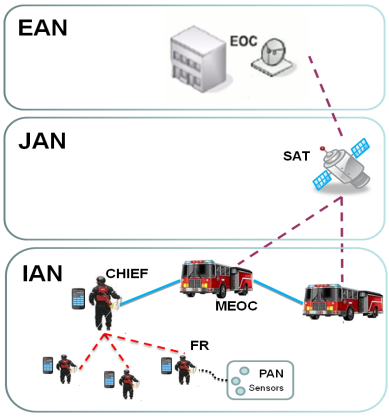

Figure 6.1: The Emergency Network under investigation, based on Wi-Fi, WiMAX and satellite networks (dashed-red, blue and dashed-purple lines, respectively) ©2012 Elsevier B.V.

the IEEE 802.16 standard provides the long-range communication between the commander (and his/her whole team) and the closest MEOC, and a geostationary satellite supports the coverage among the MEOCs and the remote EOC. Moreover, we envision the satellite system to implement the DVB-RCS Next-Generation standard [5], so that a direct MEOC-SAT-EOC (or even a MEOC-SAT-MEOC) path can be created, with a significant reduction of propagation delays (since the ground station hubs are not involved in the forwarding activity anymore). Finally, an IP router is located by any of the MEOCs, to select the information to be routed between the different networks and to circumscribe the propagation of (bandwidth-consuming) broadcasts within the IAN layer.

6.2 Experimental results in wireless networks emulation

At the best of our knowledge, containers have never been used to create virtual networks in the way we do, and, hence, it was crucial to analyze the testbed behavior in networks assessments with a certain care. Hence, we started by carrying out a sequence of wireless emulations based on very naive scenarios, easily comparable against the real measurements reported in literature; but, we gradually increased the complexity of the netboxes aggregation, to verify some properties that were determinant to confirm our architectural choices: scalability, computational load, realism, and timing overheads. Finally, we experimented some available techniques (i.e., direct LXC-to-NIC interconnections and VETH peers) to remove the kernel bridges and obtain some important performance improvements.

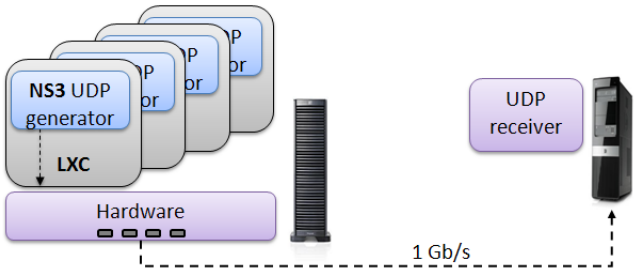

Figure 6.2: The testbed configuration utilized for the scalability evaluation

6.2.1 Scalability

With "scalability" we refer to the testbed ability to support, within the limits of the available hardware resources, an increasing number of concurrent netboxes. At the same time, these should not exhibit any sort of malfunctioning related to their computational interferences, but perform their tasks unperturbed. Under this perspective, we conducted a series of experiments where a pool of LXC containers (each with a single CPU-core assigned) is utilized to encapsulate an NS-3 packet generator each (Figure 6.2). These are in charge of producing a stream of packets at the maximum rate, so that their virtual CPU gets exhausted. Figure 6.3 reports the obtained aggregated throughput, measured at the receiving PC, when the number of generators is increased and the available CPU-cores are progressively employed. It is straightforward to notice that, independently from the frames size, the throughput grows up fairly linearly. This result discloses a marginal co-interference among netboxes, and seems to confirm that NetBoxIT is reasonably scalable in exploiting the computational resources of the hardware platform.

6.2.2 Computational load

The amount of CPU processing required by a netbox to run an emulation task represents its computational load, and is inversely related to the efficiency of the internal simulation engine (but, usually proportional to the complexity of the simulated scenario and/or to the accuracy of the network modeling). This quantity must be carefully monitored, since if a netbox gets overloaded, it can fail, or, worse, it can slow down and do not respect the necessary real time timings, bringing to misleading results. Since NS-3 is based an event-driven engine, it consumes the available CPU proportionally to the occurrence of ongoing

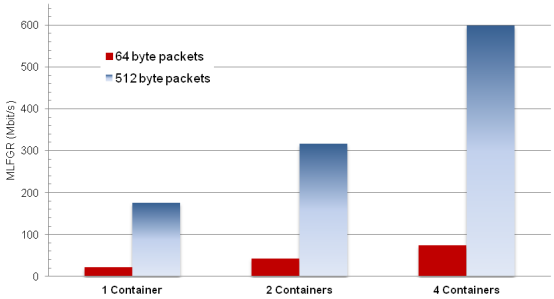

Figure 6.3: The aggregated throughput obtained by the concurrent execution of an increasing number of netboxes, each running ad UDP generator at the maximum rate (64- and 512-byte frames) ©2012 Elsevier B.V.

events. Consequently, an experimental method to measure the maximum CPU load of a netbox is to inject a packet stream at an increasing rate, so that it is gradually pushed toward saturation. For a rough estimate, we utilized two independent netboxes (with a single CPU-core assigned each), in charge of emulating a Wi-Fi and a WiMAX point-to-point link respectively. A 64-byte unidirectional flow traverses them, and its packet rate is raised up until the wireless channels is completely saturated. At the same time, we tracked the CPU load currently consumed by the netbox processing: this always kept below the 55% (for the Wi-Fi netbox), and the 65% (for the WiMAX netbox) of its maximum, well far from overloading conditions (and with additional room for increasing the complexity of the simulation further). More interesting, the CPU consumption grew up quite linearly, and a 10x increase of the packet rate produced just a 5x increase of the CPU utilization.

Figure 6.4 illustrates the measurements recorded during our evaluation of the Wi-Fi-based netbox. It is worth noting that the gross capacity of the Wi-Fi link (54 Mb/s) is saturated with a input flow of just 20 Mb/s (obtained with 64-byte packets at about 40,000 packet/s), a difference that is due to the high overhead of the 802.11 PHY and MAC layers (much higher than the Ethernet flow arriving at the netbox input card).

Certainly, a question arises: what if a single CPU-core becomes inadequate to face more complex tasks? A first solution is that a too intricate simulation could be split among several, simpler netboxes; an other method is trying to assign more than a single core to a certain netbox, since some NS-3 models support multi-threaded simulations; or else, since we exploit Ethernet as a means of communication, it is even possible to scale up the emulation to more than just a single PC and deploy a distributed testbed.

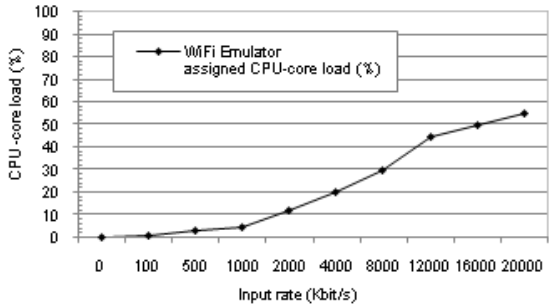

Figure 6.4: Wi-Fi emulator CPU load vs. offered input traffic (AP-STA distance 10m, 64-byte packets) ©2011 IEEE

6.2.3 Realism

As previously stated, the realism of a simulator is strictly related to the accuracy of its internal network modeling. Even if many of the NS-3 models are accurately designed and developed from scratch to be validated against true experimental data, we use them to practice emulation, and within a virtualized environment.

These novelties bring some additional constraints and possible risks: the virtual networks are compelled to handle information in real time; moreover, we add an additional software layer (the virtual machine), which can introduce some processing (and timing) overheads; and, the interconnections between the netboxes are built within the Linux kernel, which has its own limitations and requires its own resources. Prudentially, we started from the simplest configurations and introduced further new elements progressively, since the realism of the testbed is certainly its most crucial quality.

6.2.3.1 Preliminary experiments: a plain Wi-Fi link emulation

As a first step, we employed a simple Wi-Fi netbox, with a single CPU-core assigned to it (Figure 6.5). The NS-3 emulator runs in real-time mode, and hosts a plain point-to-point 802.11a Line-Of-Sight link between a stationary access point (AP) and a mobile station (STA). The transmitting power of the Wi-Fi devices is set to 16 dBm. As it can be noticed, the traffic source and sink are executed onto the same platform of the emulator (instead onto external PCs), hence their clock reference is shared and unique: this choice avoids the need for any clock synchronization between them and offers an absolute timing precision during latency measurements.

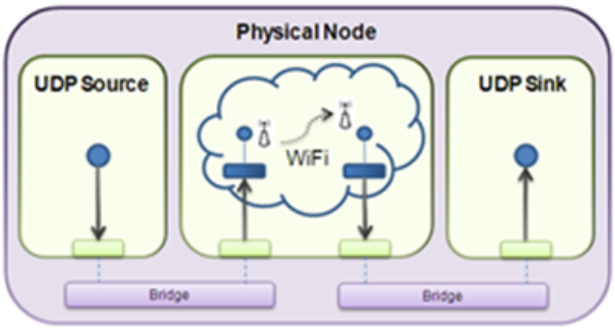

Figure 6.5: The Wi-Fi netbox under evaluation

It is also to mention that we do not use NS-3 to implement the traffic generator and the receiver here (we employed the RUDE & CRUDE applications [33]), since the timestamps appended to the datagrams by NS-3 are referred to the (local) simulation start time, and not to the (global) system clock. Therefore, since it is virtually impossible to start the generator and the receiver simulations simultaneously, the latency measurements would be biased, a tortuousness that we preferred to skip.

Figure 6.6 shows the Wi-Fi good-put, measured at the output interface of the netbox, when the STA-AP distance is varied. This result is in good accordance with the existing validations reported in literature [34, 35]. Figures 6.7 and 6.8 report the outcomes (throughput, delay, and jitter, respectively) of the Wi-Fi emulated network when the offered load is progressively increased and the STA-AP distance is set equal to 80 meters (the maximum good-put is about 7 Mb/s). The latency keeps below 1 ms, whilst jitter is limited to about 120 µs (except when the offered load is 8 Mb/s, and the Wi-Fi link capacity is exceeded).

Moreover, in other experiments not reported here, we observed that these latency values were very similar to those obtained within pure NS-3 simulations: hence, the extra latency introduced by the two kernel bridges seems minimal. To verify this impression and evaluate the timing overhead introduced by the bridge alone, we removed the Wi-Fi emulator and we measured the delay of a direct source-bridge-sink interconnection. The result is that, independently from packets size and rate, the bridge alone introduces an average delay of just 12µs, about two orders of magnitude smaller in respect of the typical (intrinsic) Wi-Fi latency.

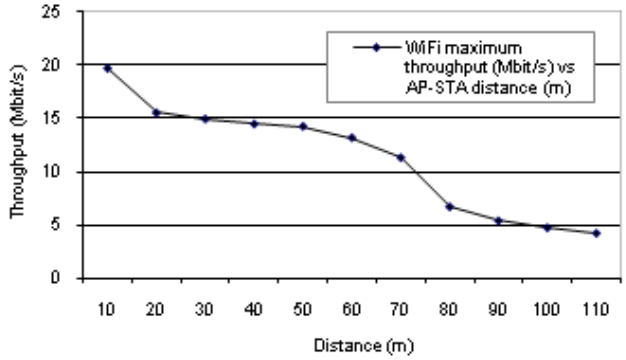

Figure 6.6: Wi-Fi good-put vs. AP-STA distance (500-byte packet) ©2011 IEEE

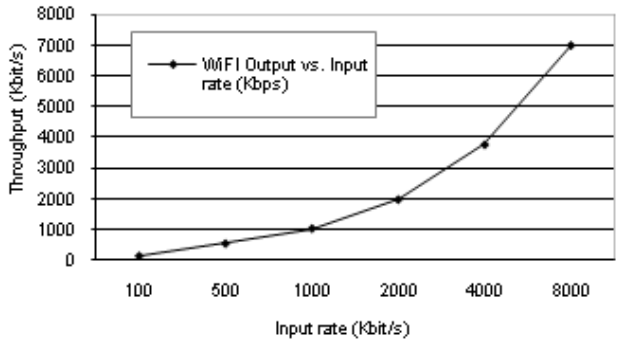

Figure 6.7: Wi-Fi IP good-put vs. offered load (Kb/s) (AP-STA distance 80m, 500-byte packets

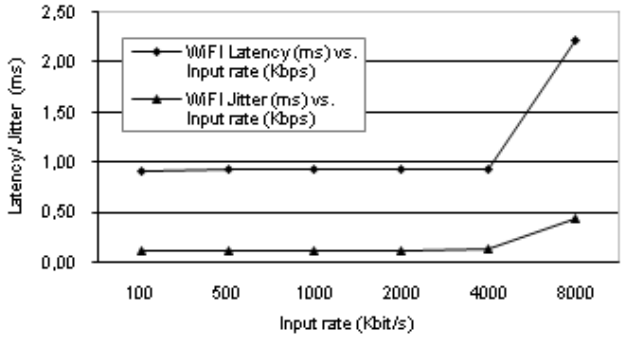

Figure 6.8: Source-to-Sink average latency and jitter vs. offered load (AP-STA distance 80m, 500-byte packets) ©2011 IEEE

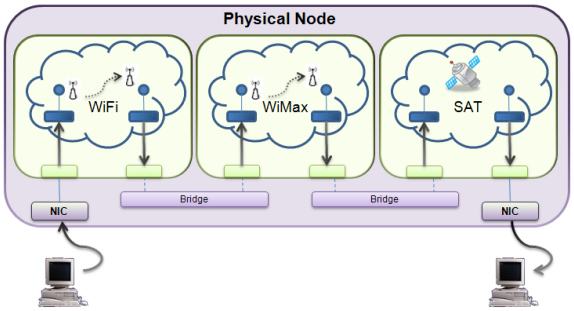

Figure 6.9: Trials configuration, with distinct NS-3 instances (the blue clouds) used to emulate the Wi-Fi, WiMAX, and satellite networks ©2012 IEEE

6.2.3.2 Performance measurements of a heterogeneous wireless network (using synthetic traffic sources)

As a second step, we now imagine to have a member of the FRs team positioned on the crisis field, and with the need to communicate with the remote EOC. Therefore, a VoIP flow traverses the entire infrastructure: at first, through the Wi-Fi access network toward the commander; then crossing the WiMAX link between the commander and the MEOC, and finally proceeding through the satellite return channel toward the EOC.

Figure 6.9 shows how this scenario is translated in terms of netboxes. The key components are (from left to right): an external PC, that creates the voice stream produced by the FR portable device; a Wi-Fi netbox, that emulates the IEEE 802.11a connection between the FR and his/her commander; a WiMax netbox, that synthesizes the long-range commander-to-MEOC link; a SAT netbox, that implements the return/forward channels of the satellite; and, last, a PC that represents the EOC personnel receiving device. It is worth saying again that each netbox is configured to handle the ingress traffic in real-time mode (i.e., with a strict upper-bound of the timing skew inside the emulator) and employs just a single CPU core. Table 6.1 resumes the main configuration parameters of the emulators.

These simulations are performed by injecting a one-way, end-to-end flow (500-byte UDP packets) into the network return channel (between the source and the receiving PCs), and increasing its rate progressively. Figure 6.10 reports the good-put for each traversed network segment, measured at the output (virtual) interfaces of the Wi-Fi, WiMAX and SAT netboxes respectively. This test was carried on to verify whether the emulators perform coherently in terms of served traffic in respect of the represented real networks.

Network segment	**Technology**	**TX Power (dBm)**	**Channel propagation model**	**Distance**	**Theoretical Gross Capacity**
FR-FR Commander	802.11a	16	FRIIS, LOS	80 m	54 Mb/s
FR Commander - MEOC	802.16TDD 10MHz @ 5 GHz	30	FRIIS, LOS	2.5 km	Return/Forward channel: 18/18 Mb/s
MEOC-EOC	DVB-RCS-NG	-	Latency=254ms, PER=10^{-6}	76000 km	Return/Forward channel: 2/10 Mb/s

Table 6.1: The NS-3 emulators operational parameters

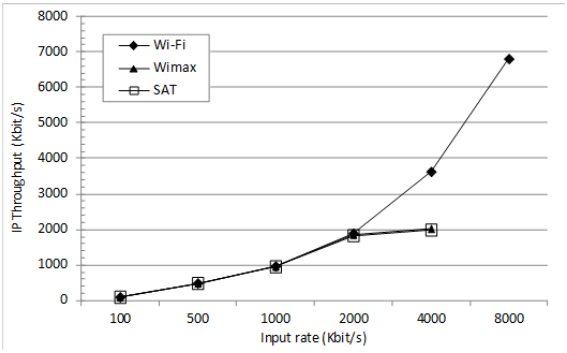

Figure 6.10: The maximum (return channel) IP good-put measured for each network segment (500-byte packets) ©2012 IEEE

The satellite return link behaves as expected, since its capacity is limited by construction (2 Mb/s); instead, the measured WiMAX throughput (slightly less than 2 Mb/s) was a little surprising. However, the scrupulous reading of the 802.16 (TDD profile) standard and the examination of existing validations ([36, 37]), helped us to clarify that the obtained results are honestly accurate. In fact, the 18 Mb/s capacity resumed in Table 6.1 is referred to the theoretical gross value at the physical layer. In practice, instead, some decreasing factors concur to determine the lower effective good-put. For example, only the ¾ of the 256 OFDM sub-carriers are effectively employed for data transmission, decreasing the capacity to 13.5 Mb/s. Moreover, at the distance of 2.5 km between the FRs commander and the MEOC, the modulation technique used cannot be the most efficient (64-QAM), but the more robust QPSK ¾ (this decreases the capacity of a factor 3, at just 4.5 Mb/s). The error correction coding (FEC) is an other overhead that contributes to reduce the good-put further (at 3.3 Mb/s). Moreover, the TDD subframe itself (the frame part that conveys the useful data symbols) enwraps some overhead at the physical layer, since the transmission of a sequence of consecutive OFDM symbols must be always preceded and followed by a predefined number of trailing and guard symbols. Finally, the TDD frames must also carry information related the scheduling, ranging, and control information that must be continuously exchanged between the Base Station and the Subscriber Station. Similar considerations could be done for the Wi-Fi network, as well; in fact, the measured IP good-put (7 Mb/s) is in agreement with the experiments reported in literature under similar conditions ([34, 35]).

In order to verify if the testbed realism even more accurately, we then introduced some further complexity within the scenario. A second sequence of experiments was carried out to evaluate the behavior of the emulated network when several user applications are transmitting information concurrently from the FR device to the EOC. Table 6.2 resumes the ingress traffic profiles used.

These tests are combined with the application of various QoS policies within the WiMAX segment, so that distinct traffic categories are handled with different priorities. In detail, the rtPS (real-time Polling Service) scheduling policy is applied to the VoIP-like flows, the VIP-like flows are treated with the UGS highest priority (Unsolicited Grant Service), and bulk data transfers are managed with the lowest BE (Best Effort) scheduling. We executed a series of latency and jitter measurements, where we gradually joined additional traffic within the return channel. The latency values are recorded between the input interface of

Input traffic	UDP payload	Source rate (Kb/s)	Ingress IP load (Kb/s)	WiMAX scheduling
1 VoIP-like CBR flow	74 byte	16	48	rtPS
1 Video-like CBR flow	256 byte	388	401	UGS
8 UDP data CBR flows	256 byte	8 x 34	305	BE, Best Effort

Table 6.2: The user traffic profiles employed to assess the network QoS capabilities

the Wi-Fi netbox and the output interface of the SAT netbox in Figure 6.9. Figure 6.11 compares the end-to-end latency and jitter suffered by a (single) VoIP flow and by a (single) VIP flow, individually crossing the network infrastructure. The performance of the VIP-like flow is slightly the worst, despite the higher priority, but this is due to its larger packet size, that requires a higher scheduling and transmission time within the WiMAX uplink subframes.

As a matter of fact, when the two flows are overlaid, and the WiMAX segment becomes busier, the VoIP flow starts experiencing some degradation, whilst the VIP quality is still fully preserved (Figure 6.12). Of course, this happens since the rtPS scheduling priority does not offer a so stable and reliable quality protection as UGS does. A possible discussion topic is that it could be a better choice to employ a UGS policy even for the VoIP flow, to better preserve its quality. Nevertheless, this would be a wasteful setting: in a real situation, VoIP agents are usually configured to exploit the VAD (voice activity detection) technology, since this is the most efficient way to utilize the available bandwidth (VAD is able to suppress the packets transmission when a user is silent, and it is well known that during a phone call only about 50% of the time the caller is truly speaking and vice versa for the callee). Under this viewpoint, rtPS (by which the user application flexibly updates its bandwidth requests when necessary) is usually preferable to UGS (that statically reserves the bandwidth once for all), because it allows a more efficient use of the available bandwidth.

Figure 6.13 shows the effect of the aggregation of eight, additional data flows. The VIP quality is still guaranteed, whilst the VoIP flow is affected by a larger latency; obviously, latency and jitter are much worse for the best effort data flows. However, the WiMAX channel is still well far from saturation, since the total load of the user applications sums up to just 754 Kb/s.

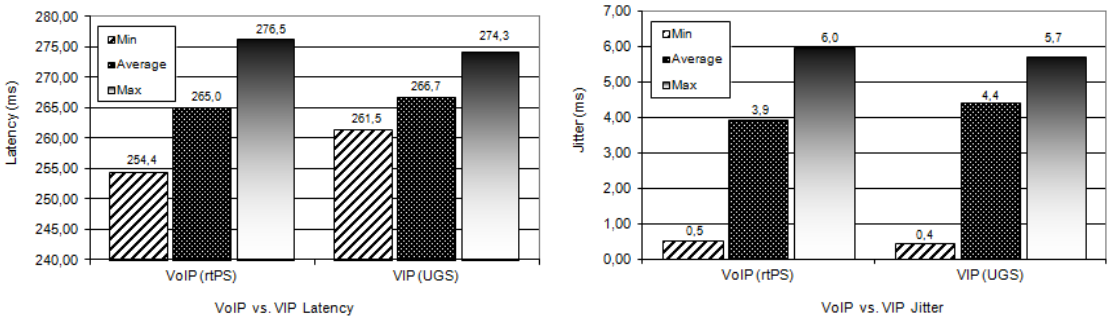

Figure 6.11: Latency and Jitter comparison between a single VIP flow (UGS scheduling) vs. a single VoIP flow (rtPS scheduling) ©2012 IEEE

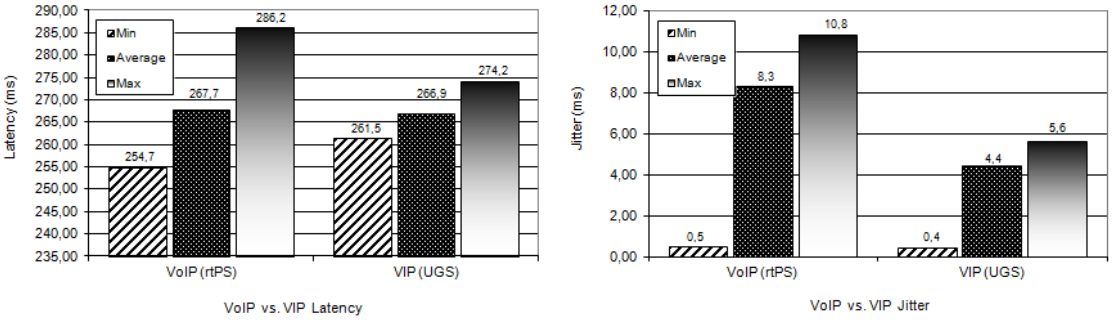

Figure 6.12: Latency and Jitter comparison between simultaneous VIP (UGS scheduling) and VoIP (rtPS scheduling) flows ©2012 IEEE

Finally, we deliberately decided to overload the WiMAX uplink channel, to evaluate if its QoS scheme was working well yet (and, accordingly, the realism offered by the emulation was still trustworthy). To obtain this result, we increased the FRs commander – MEOC distance up to 7.5 km; and, at the same time, we placed two VoIP flows side by side, instead of using just a single one. Therefore, the WiMAX uplink turns in a bottleneck, since its capacity at the IP layer lowers to 775 Kb/s, whilst the injected traffic load raises up to 802 Kb/s, enough to saturate the channel. Figure 6.14 depicts the performance characteristics obtained in this situation, showing that the VIP flow is still well preserved, the VoIP flow is further degraded, whilst the bulk data transfer performs poorly, as we actually expected to observe.

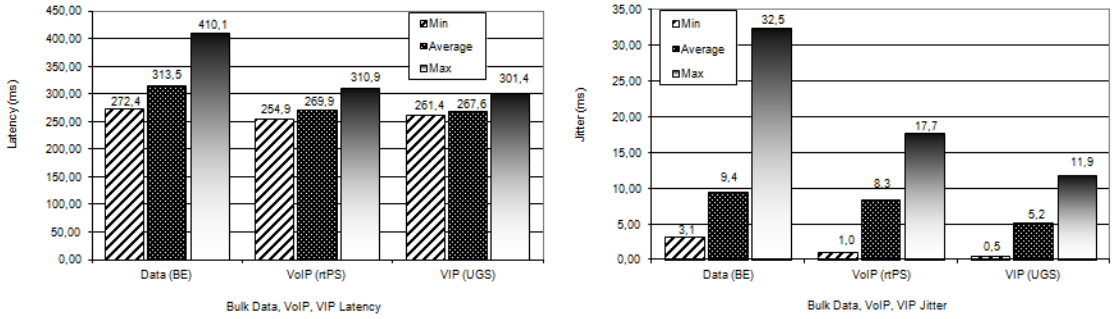

Figure 6.13: Latency and Jitter comparison between simultaneous VIP (UGS scheduling), VoIP (rtPS scheduling) and 8 data flows (BE scheduling) ©2012 IEEE

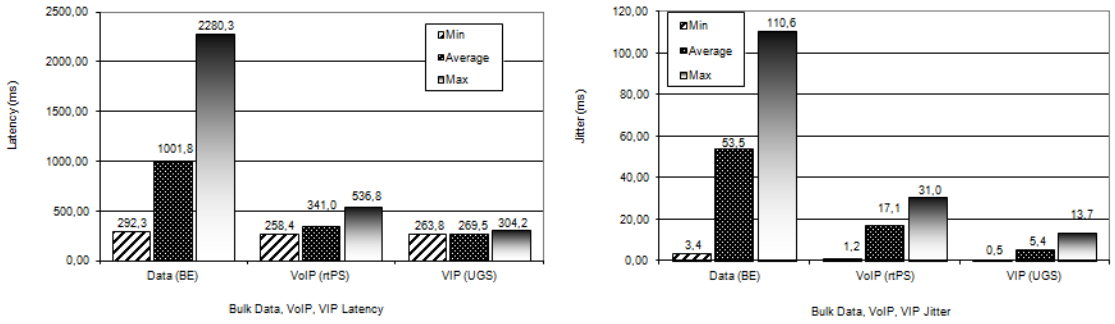

Figure 6.14: Latency and Jitter comparison between simultaneous VIP (UGS scheduling), 2 VoIP (rtPS scheduling) and 8 data flows (BE scheduling), in WiMAX overloading conditions ©2012 IEEE

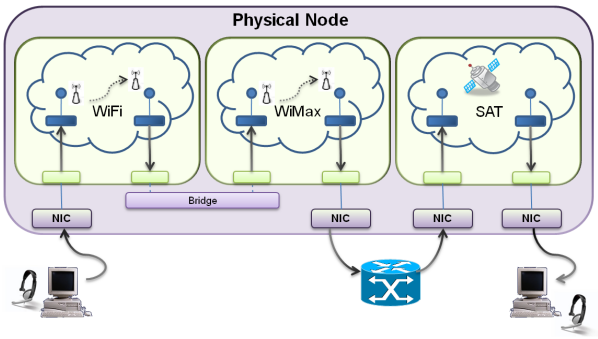

Figure 6.15: The E-SPONDER Emergency Network trials, exposing the NS-3 emulators (the blue clouds) used to emulate the Wi-Fi, WiMAX and satellite networks, the external Click Modular Router, and the real VoIP agents ©2012 Elsevier B.V.

6.2.3.3 Performance measurements of the Emergency Network (with real traffic sources and routing equipment)

As we stated several times, one of the key advantages of NetBoxIT is represented by its interoperability, and its easy connection with external, real-world equipment. In Figure 6.15 we sketch the testbed configuration that was employed for our last evaluations related to the testbed realism: three distinct netboxes (again using a single core each) are instantiated and interconnected to translate our reference scenario in terms of virtual networks. And, again, we suppose to have a first responder at the IAN with the need to communicate with the remote EOC personnel, generating a VoIP stream that traverses the whole Emergency Network.

However, we introduce now two novelties in comparison with the schema depicted in Figure 6.9: first, the WiMAX and satellite networks are separated by a true, external IP router (the Click Modular Router), attached to the physical NICs of the platform, as it would actually be within the MEOC vehicle; second, the ancillary PCs do no longer generate synthetic traffic, but, they are utilized to run a pair of real-world LinPhone [38] agents (with true headsets and microphones), as the source and the sink of the VoIP-based speech. The three wireless links are configured by using the same operational parameters previously resumed in Table 6.1, whilst Table 6.3 illustrates the configuration used for the LinPhone clients.

Moreover, it should be mentioned that, similarly to what could be done in a real deployment, the WiMAX simulator is configured to schedule the uplink (i.e., from the FR to

Agent	Audio Codec	Sampling rate (KHz)	Frame (ms)	Codec delay (ms)	Coding rate (Kb/s)	Ethernet Throughput (Kb/s)	VAD
LinPhone rel. 3.4	Speex WB	16	20	50	28 (CBR)	49.6	NO

Table 6.3: VoIP agents configuration ©2012 Elsevier B.V.

	Latency (ms)			Jitter (ms)		
	Mean	Max	Min	Mean	Max	Min
Return Channel	269.2	279.5	255.8	7.9	22.1	1.3
Forward Channel	303.0	350.0	256.4	42.9	55.6	25.5

Table 6.4: End-to-end latency and jitter experienced by a VoIP flow traversing the Emergency Network through the Return and Forward channels ©2012 Elsevier B.V.

the EOC) VoIP traffic using the rtPS (real-time Polling Service) QoS policy, which is the best suited to support voice communications in terms of efficiency and timing constraints; on the contrary, the downlink WiMAX scheduling is configured to employ a plain Best Effort policy, so that the EOC-to-FR communication is treated with no particular priority.

A SIP-based, two-way voice call is then opened between the two speakers. The Wireshark packet sniffer[39] is employed to capture and record the frames timing. In particular, it is executed within the raw Linux operating system of the emulation platform, so that it is able to receive all the frames at any of the testbed interfaces. More precisely, not only the physical NICs can be monitored, but also any other virtual interface instantiated within NetBoxIT (i.e., the bridges and the virtual cards of the netboxes). This is a key advantage: one can acquire the general picture of the packets timing along their journey through the testbed, in a single shot. Moreover, this can be done with an absolute timing reference (and no need for complex synchronization mechanisms), since both Wireshark and the netboxes are all hosted within the same platform, and their clock tick is shared. A track of 13 minutes of true conversation (40000 frames) was recorded for further off-line analysis (to keep the analysis plain we did not activate the VAD feature during these measurements).

Table 6.4, Figure 6.16, and Figure 6.17 show the statistics associated with the end-to-end latency and jitter experienced by the VoIP connection along the return (FR-to-EOC) and forward (EOC-to-FR) channels. It is worth mentioning that these values are measured between the couple of physical NICs where the two VoIP agents are attached: hence, we

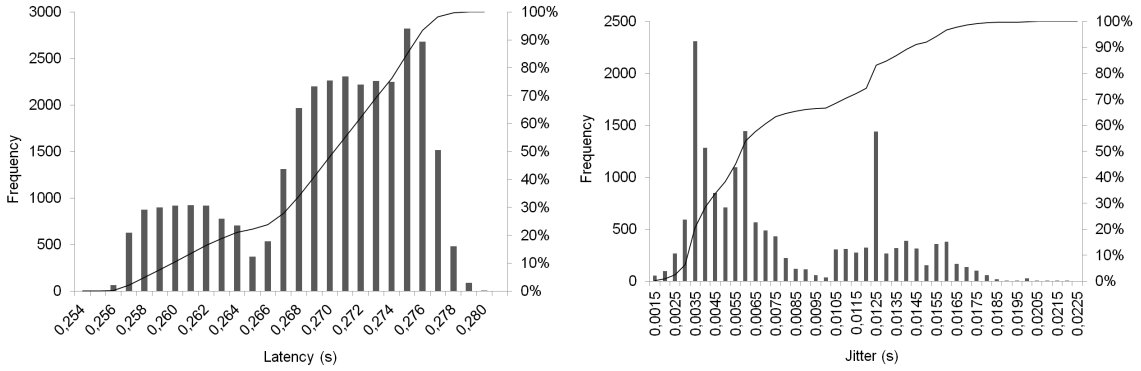

Figure 6.16: End-to-end latency and jitter (probability frequencies and cumulative function plots) experienced by a VoIP flow traversing the Emergency Network return channel ©2012 Elsevier B.V.

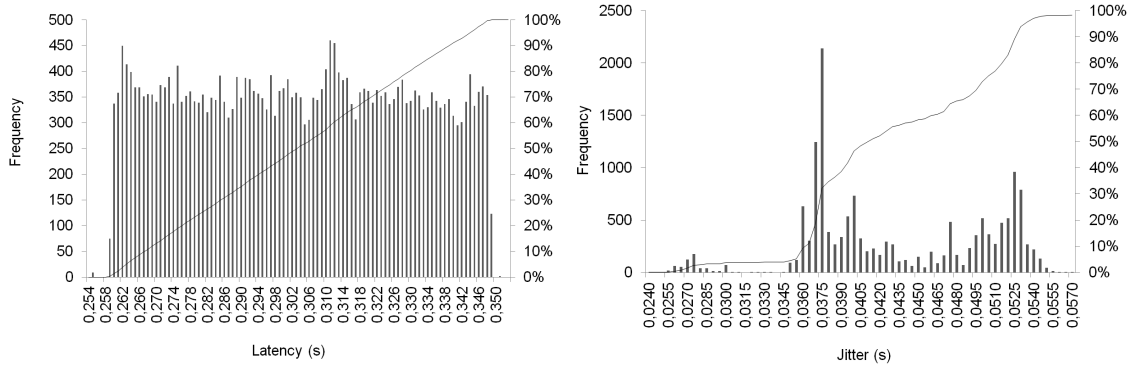

Figure 6.17: End-to-end latency and jitter (probability frequencies and cumulative function plots) experienced by a VoIP flow traversing the Emergency Network forward channel ©2012 Elsevier B.V.

are strictly evaluating the latency of the networks, and not the delays that occur within the two endpoint PCs. In other words, we do not consider the voice encoding and packetization latency, since these are not rigorously related to the network behavior, but dependent on the particular type of encoding adopted.

It is easy to notice that the distributions of the return channel are much stricter than those of the forward channel, highlighting the advantage of the rtPS scheduling. But, from our point of view, this experiment also demonstrates the very realistic functioning of the testbed. For instance, the uplink average latency (269.2 ms) seems in agreement with our theoretical expectations. In fact, in TDD WiMAX, the uplink request-grant procedure is roughly based on this procedure: the SS asks the BS the permission to transmit and indi-

cates the number of time slots (TXOPs, or transmission opportunities) that are needed; the BS verifies the SS request, and replies with the indication of the usable TXOPs; however, very frequently, these can be scheduled only within the next uplink frame (the immediate availability of TXOPs within the current frame is a more unusual event), consuming a time interval that is usually at least equal to the current uplink frame duration (10 ms). More simply, we can say that, most of the times, the SS is authorized to transmit only after 10 ms. Moreover, since the audio source and the WiMAX TDMA framing are (unavoidably) desynchronized, there is a continuous, gradual time shift (that cycles from 0 to 10 ms) between the arrival of a new packet from the source and the frames synchronism in WiMAX: therefore, the delay experienced by the packets within the WiMAX network should sum up to a value close to 15 ms in the average (which must to be added to the 254 ms delay introduced by the satellite link).

6.2.4 Timing overheads

With "timing overheads" we refer to both the hardware and the software latencies, extraneous to the strict real-time emulation running inside the netbox, that might affect the packets processing. Nevertheless, in the following we limited our study exclusively to the software-based latencies. Of course, this is just a first order approximation: the hardware architecture itself introduces its own skews (for instance, the PCI subsystem cannot enforce an absolute timing guarantee during the transfer of data from/to memory to/from NICs; or, an other example is that a NIC introduces some delay when queuing the incoming frames within its buffers). However, the physical data transfers within the PCs architecture are today so fast that the required times can be ignored in practice.

In principle, we wish that packets could traverse the sequence of virtual networks with the exact timings that would occur in the real heterogeneous network. However, we are utilizing a software-based platform, where this target is unachievable in absolute, but only with a certain degree of approximation, since some computing delays are unavoidable.

For our aims, it is certainly crucial to obtain a fair real-time data management at least within the netboxes (i.e., between the emulators input and output ports). This is simple to do, because NS-3 permits to compel an upper limit for the jitter of an emulation (i.e., the maximum displacement between the emulation clock and the system wall clock referenced by the NS-3 real-time scheduler): if this is not honored, the simulation engine aborts its

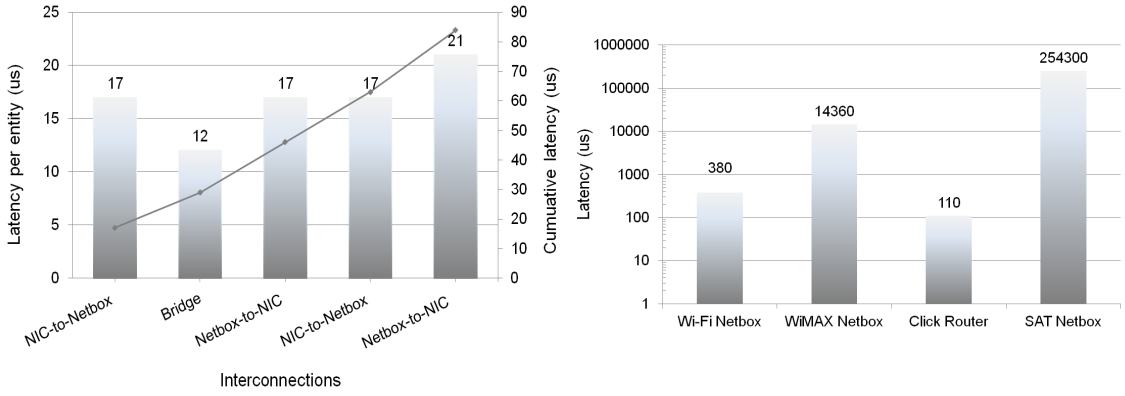

Figure 6.18: Overheads and emulation timings of the testbed ©2012 Elsevier B.V.

processing and reports the trouble. Hence, we mainly focalized our attention on determining the skews outside the netboxes, and analyzing if these had some concrete impact on our measurements or not. For instance, the virtual interface of a LXC container requires a bit of time to retrieve packets from a physical NIC and make them available for the hosted application, since they are retained for a while by the kernel bridge in the midst (since this is in charge of carrying on its lookup and copying operations).

For the evaluation of these "extrinsic" overheads, we accomplished a series of tests where a one-way UDP flow traverses the network configuration already depicted in Figure 6.15. The frames are tracked using Wireshark once again, and their timings are illustrated in Figure 6.18. The left bar graph accounts for the delays suffered by the frames while they traverse each of the interconnecting elements. For example, about 17 µs are required for the propagation of an ingress frame between the first physical NIC and the virtual NIC of the container attached to it (however, Wireshark records the frame when this has already been copied within the receiving ring buffer in kernel land: thus, the hardware latencies are not taken into account). The same latency occurs to copy a frame from the output port of a netbox toward the transmitting buffer of a physical NIC. Instead, traversing a bridge located between a pair of netboxes requires approximately 12 µs. The cumulative timing overhead sums up to 84 µs, not so relevant in many common situations.

The bar graph on the right of Figure 6.18 shows the delays recorded between the input and output interfaces of the Wi-Fi, WiMAX and SAT netboxes, or between the two physical NICs attached to the Click Modular Router ports, so that the single contribution of each network segment emerges. It is worth noting how the latency of the WiMAX netbox is

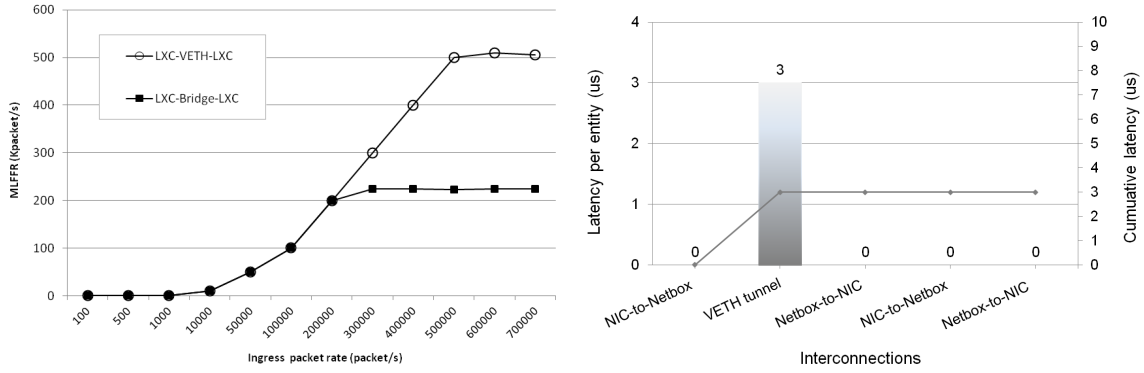

Figure 6.19: Virtual Ethernet peers vs. Bridge forwarding rate (left); timing overheads of the testbed (right) after the bridges removal ©2012 Elsevier B.V.

about 14.3 ms, not so far to the value (15 ms) speculated from the discussion above (it should be considered that now we are injecting a one-way flow, instead of using a two-way conversation as we did before, and thus the delay within the Wi-Fi network is certainly lower, since collisions and retransmissions between the AP and the STA cannot occur).

6.2.5 Virtual Networks interconnections

In the previous experiments, we extensively employed the software bridges as a means of interconnection among the physical and/or the software elements within the testbed. Each bridge is created as an in-kernel "object", that is able to store, copy, and deliver the incoming frames arriving either from a user-space application (the NS-3 emulator), or from a network card. Moreover, the outcomes discussed just above indicate that these interconnections could represent the main limitation for the realism of the testbed, since their latency could draw to misleading results if the number of interconnected elements continues to grow, or during the emulation of fast links (e.g., Fast-Ethernet or Gigabit connections), in which the network propagation time becomes comparable (or even smaller).

A new feature introduced by the Linux kernels after 2.6.35 is the support for the straight association between LXC containers and the physical NICs: this innovation makes it possible to remove most of the bridges from the testbed configuration, with a very positive impact, since their expensive processing is skipped and the related timing overheads completely disappear.

Moreover, even the remaining bridges (for instance, the link between the Wi-Fi and

the WiMAX emulators) can be replaced with a more efficient forwarding scheme. Virtual Ethernet peers (VETH, [25]) are a lighter in-kernel forwarding mechanism, which permits a faster delivery between two endpoints applications, than kernel bridges. In essence, the Ethernet frames created within a netbox are copied to kernel land and looped back to user land, for being received by the peer netbox. This technology employs the AF_NETLINK sockets to modify the destination address of a frame, so that this is not transferred to a physical NIC, as usual, but enqueued for the receiving application. Figure 6.19 (left) compares the maximum forwarding rate that can be reached by a Virtual Ethernet peer against a conventional bridge, proving the considerable advantage of the latter.

When both the two techniques are exploited, all the software bridges can be removed from within the testbed: Figure 6.19 (right) can be directly compared to the graph in Figure 6.18 (left), showing how the overall overhead is reduced from 84 µs to 3 µs.

6.3 NetBoxIT performance analysis

The assessment of the testbed capacity (i.e., the ability to manage the incoming frames within real time boundaries) is essential to realize what are the performance constraints of our architecture, at what point the system breaks down, and how internal timing overheads behave when the testbed is overloaded or the number of virtual networks is scaled up. In Figure 6.20 we illustrate the most significant results obtained during a series of investigations carried on with these targets in mind.

Our first trial was focalized on evaluating the NetBoxIT capacity, as its ability to manipulate and forward the offered load, measured in packets per second. Similarly to what happens with software routers, NetBoxIT performance proves to be largely independent from frames size, since both the hardware overheads (e.g., the time necessary to receive or transmit frames through the I/O bus), and the software overheads (related to the memcpy operations of the frames) vary minimally. Therefore, we describe here only the experiments performed with short 64-byte frames. We change from 1 to 8 the amount of netboxes to emulate a chain topology of 2-16 IP routers, connected through 1 Gb/s Ethernet links. It is worth remarking that any of the netboxes employs just a single CPU-core: therefore as the chain becomes longer, we progressively consume all the available computing resources of the platform. It should also be recalled that our hardware is based on a Dual Quad-core Xeon chipset, wherein we have two physical CPUs (socket #0 and socket #1), encasing 4

cores each. For clearness, we will apply the following conventions: the cores belonging to socket #0 will be enumerated 0,2,4,6, whilst the ones belonging to socket #1 will be called 1,3,5,7. We assembled five different configurations of the testbed, wherein the number of hops is gradually increased, with some variations in the selection of the employed cores:

setup A) 2 IP routers are simulated inside a single virtual network. Hence, we first create a container and we run a NS-3 simulation within; then, the NS-3 configuration defines two virtual routers linked by a 1 Gb/s Ethernet link; finally, each router is provided with an EmuNet device (i.e., a virtual NIC) and directly bound to the physical (Gigabit) NIC of the testbed (we do not employ software bridges anymore during these trials). In this way, the frames arriving from an external source are offered to the first router by the physical NIC, forwarded to the second router through the simulated link, and then forwarded by the second router and transmitted through the second physical NIC towards an external sink. This lone netbox runs on core #2 (the second core of socket #0).

setup B) 4 IP routers are separated in pairs and each couple runs within its own netbox (each constructed like in setup (A)). In this example, a lone Virtual Ethernet peer is instantiated to join the two virtual networks. During this trial, core #2 is selected for the first netbox, and core #4 for the second netbox.

setup C) 8 IP routers are split in 4 couplets, each emulated within its netbox. In this configuration, 3 Virtual Ethernet peers are utilized to connect the four netboxes. During this test cores #1, 3, 5, 7 are used (the entire socket #1).

setup D) This setup is entirely similar to setup (C), but cores #0, 2, 4, 6 (the whole socket #0) are employed.

setup E) 16 IP routers are split and run in pairs within 8 netboxes, with 7 Virtual Ethernet peers employed to interconnect them. During this test we utilize all the available CPU-cores.

In short, the described topologies are practiced to connect a transmitter with a sink through a variable number of routers. By changing the number of hops forming the path, we change the amount of work that NetBoxIT performs to simulate the end-to-end route. In any of the experiments, a single, external PC is utilized to create an UDP stream, as well as to receive

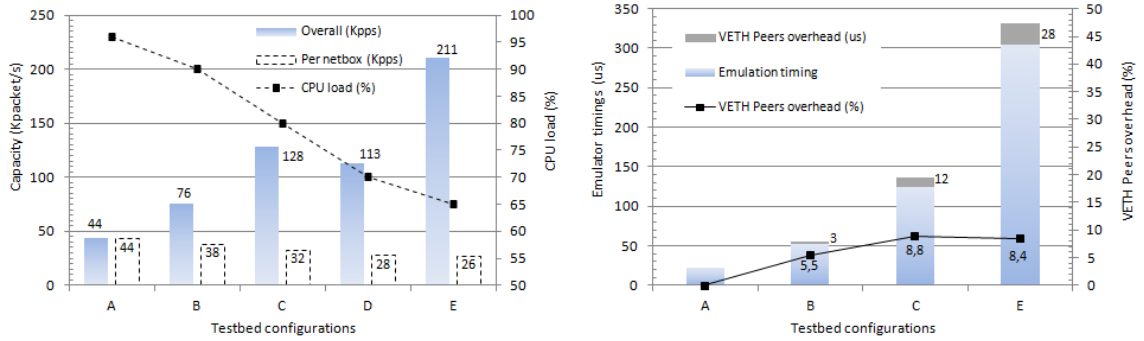

Figure 6.20: Cumulative (filled bars) and per netbox (dotted bars) capacity, and average CPU-core load (left); timings and interconnections overheads in NetBoxIT, during the emulation of a chain of Gigabit Ethernet routers (right) ©2012 Elsevier B.V.

the same frames and measure the testbed capacity. The left graph in Figure 6.20 shows the maximum value of the forwarding capacity, in packets per second, for any of the testbed configurations. In detail, the dotted bars represent the maximum value of the forwarding rate that can be obtained by each netbox forming the chain; besides, the filled bars show the amount of frames that can be instantaneously delivered by the platform as a whole. Within the same graph, we also plot the (average) load of the CPUs during the experiments. It is first to clarify that in a chain of netboxes, where various cores are employed, their load gets usually aligned, after a short transient. Indeed, the frames departing from a netbox become the input for its follower; and, the first netbox of the chain inevitably defines (i.e., limits) the performance of all its followers. An exception to this rule is when we execute both a netbox and a NIC driver onto the same core, for instance core #0 in setups D and E. In such conditions, that specific core sustains a higher load than its followers.

For the baseline situation of 2 hops, the measured testbed capacity is approximately 44,000 packet/s. It is worth clarifying that we are strictly referring here to (the capacity of the) emulator meant as a “forwarding black box”, (i.e., the software and hardware ensemble that receives and transmits packets from/to the physical NICs), and not to the capacity of the virtual segment interconnecting the two routers within the NS-3 simulation (whose nominal throughput of 1 Gb/s is chosen large enough not to be a bottleneck). In this situation, the CPU is utilized at its 95.6% and clearly constitutes the bottleneck for the emulator: the testbed enters a state in which all its committed resources are used to pull 44,000 packet per second from the input NIC, emulate the 2-hop virtual network (comprising the routers

IP stacks, their table lookup algorithms, the propagation of frames, the transmission and reception activities, etc.), and push the packets to the output network card. Once we try to emulate 4 hops or more, the forwarding rate of any single netbox decreases more and more. For 16 hops, any netbox forwards about 26,000 packet/s, whilst the platform as a whole handles 211,000 packet/s. In this worst case, even so, a netbox is still capable to perform the reliable emulation of a 10Mb/s Ethernet link with 64-byte packets and of a 100Mb/s link with frames larger than 500 byte. On the contrary, a 1 Gb/s connection can never be implemented, even with 1500-byte frames. For the sake of a comparison, we can observe that the capacity of a single netbox is certainly lower than in ModelNet [18], where an emulation node using a 1.4 GHz CPU forwards about 120,000 packet/s. However, the two systems are based on quite different kinds of architectures and network modeling. ModelNet "pipes" are certainly faster and lighter than NS-3 C++ objects, but they are also elementary. On the contrary, in NS-3 even a simple NIC (e.g., the CsmaNetDevice/EmuNetDevice used in these trials) offers a set of configuration parameters that make its functioning astonishingly similar to that of a true adapter (e.g., one can access the MAC address, MTU, DIX/LLC EncapsulationMode, queues lengths, CRC checksums, TX/RX statistical counters, error model, etc.). As it is easy to guess, a trade-off between performance and realism is unavoidable, and certainly the NS-3 superior accuracy comes with a price to pay. Certainly, we can speculate that ModelNet could probably forward about 200-300,000 packet/s on our 2.4 GHz CPU. Nevertheless, it is worth remarking that the ModelNet testbed is based on the principle that the emulation is entirely carried on within the kernel land of the host system (where "pipes" are instantiated), at the highest priority; whilst, our netboxes are executed within the low priority user land context. Hence, even if the performance of a single netbox appears to be inferior, we gain the additional flexibility that we can more easily execute several simulations in parallel onto different CPUs. The consequence is that the global amount of packets (the cumulative capacity in Figure 6.20) that is managed thanks to parallelization makes NetBoxIT comparable to ModelNet when the number of virtual networks grows, despite of the lower priority of the user space processing.

It seems now appropriate to investigate what are the bottlenecks of the testbed, to appraise whether the performance can be improved. In Figure 6.20 (left) we plot the curve of the CPU load, that gradually falls off from 96% to 65%: it seems clear that, as we step up the number of emulations, the utilization of the available cores becomes more and more

difficult. Certainly, we can say that the decrement of the forwarding capacity cannot be related to a lack of computing resources, since the CPU cores remain even partially idle. At the same time, however, this behavior is not completely surprising: a lot of recent research carried out about software routers demonstrates that it is not sufficient to expand the number of physical threads to obtain a better performance, even when the software threads are split with care among the physical threads. Some recent surveys also pointed out to the issues that should be specifically addressed. In [40] a careful analysis of Linux scalability over multi-core platforms brought to light different issues that could also apply to our experience. First, the protection of shared data structures against race conditions represents the main (but not the only) source of many scalability limits within multi-threaded platforms. In fact, when any concurrent tasks need to use the same data (and, hence, to read/write from/to a shared memory location), data coherency must be preserved. The first consequence of this constraint is that a thread may (or, must) lock a shared structure when using it: hence, a computational system where a growing number of tasks are cooperatively running and processing the same data structures is also a system where the tasks waiting times tend to become increasingly larger. A second problem, which is normally (but not necessarily) related to the first, is that when a thread writes to a shared memory segment, the cache coherency protocol has to quash any related cache entry: hence, the tasks using the same data must re-fetch them again, directly from the (slower) main memory, since their copy is no longer synchronized with the actual value. This problem is often related to locking (since locking is usually the technique to perform safe data touches), but can occur even with lock-free shared structures.

We can initially speculate to meet all these possible limitations within our architecture. For example, shared data and locks are both certainly utilized. In fact, Linux moves the frames from the NIC to the main memory (via DMA controller) using a pool of kernel buffers, that are referenced from a single, circular list of pointers (the well-known “ring buffer”). This list is a shared structure, since it allows several applications to send/receive data to/from the same NIC at the same time, and therefore it requires some form of protection. The default method is to employ a locking technique to regulate its updating (that occurs each time a frame buffer needs to be retrieved or is freed). This results in a contention for the lock, and can induce a bottleneck within the Linux networking subsystem. In NetBoxIT, the kernel networking layer is surely involved in packets copies to/from the netboxes, since these are just a special example of receiving/transmitting user land applica-

tions. Nevertheless, we did not find any clear evidence of a relation between the NetBoxIT performance limits and the ring buffer locking mechanism. First, our measurements in Figure 6.19 (left) demonstrated the forwarding capability of a VETH peer (which is running exactly within the Linux networking layer) to be at least 500,000 packet/s. And, even when we employ eight netboxes in setup (E), and a 26,000 packet/s stream flows through the kernel 16 times (bouncing up and down, from kernel to user land, and vice versa), the amount of packets handled by the kernel (about 420,000 packet/s) is still below this limit. An other reason comes from the knowledge of the consequences of a highly locked ring buffer: in such a situation, the DMA controller cannot acquire free memory pointers fast enough, and frames transfers between the NICs and the memory bank are stuck. As a result, the FIFO buffer within the NIC gets filled quickly, and the incoming frames start being dropped at the NIC transceiver (this events are usually reported by the NIC driver as "FIFO Errors"). Furthermore, the frames that are able to enter the system are queued within the NIC for a longer and longer time interval (typically, a sharp increase of their delay can be observed, from microseconds to milliseconds; for instance, see in [41]). As a matter of fact, we did not experience these anomalies during our trials: we did not register any frame drop within the NIC, as well as the VETH peers latencies remained low (3-4 µs).

Furthermore, it is important to notice that, by a comparison of (C) and (D) in Figure 6.20, it is a best practice not to utilize the CPU-core #0 to host a netbox, if it is possible, since this choice lowers the performance. Indeed, this core is rather preferred by the kernel to run the NIC driver, since the APIC (Advanced Programmable Interrupt Controller) delivers the IRQs (Interrupt Requests) generated by the cards to core #0 by default. Consequently, any task running on this core is often interrupted, and expensive context switches must be performed.

As a second step, we also speculated about the real impact of cache misses on system performance. This issue is certainly present: for instance, frames headers must be updated at the output of each hop, since their MAC destination address needs to be changed. As a practical example, in setup (B), any frame (i.e., its data structure in memory) is touched by the first netbox (which runs on core #2) before being passed to the second netbox (running on core #4): this is a typical condition in which a cache miss occurs, and core #4 is obliged to fetch data from the main memory. These considerations appear certainly true, however this problem alone does not explain the baseline performance that we obtained, since the

interconnecting bus between the CPU and the DDR3 bank provides a huge capacity (68 Gb/s) and the re-fetch of a 64-byte buffer can be accomplished in a few nanoseconds. Therefore, we can resolve that locking techniques and cache invalidations are not sufficient to justify the performance decrease that we registered.

There is however another phenomenon that has not been analyzed yet. Despite virtualization, the NS-3 emulator is a process like any other, and it is executed within the Linux user context: therefore, it is obliged to transmit or receive frames only through the kernel. One of the milestones within the Linux API is surely represented by a couplet of system calls, "copy_to_user" and "copy_from_user", that are invoked every time that data must be moved between the kernel and the user space. Indeed, for security reasons, kernel- and user-land tasks must operate in a completely separated memory address space; these two functions are in charge to execute all the necessary checks and preparations, before they perform the copy of data between the two memory spaces. Moreover, there is some other processing that involves a frame during its path between the NIC driver and the user application: after it is stored within a kernel buffer, a long sequence of activities are executed within the TCP/IP layers to manage the frames data and meta-data, making the extraction of the payload a quite expensive operation. This is the reason why several fast packet processors run inside the kernel itself, or as a special in-kernel module (like the Click Modular Router).

In order to have a baseline point of reference, and figure out how things work when packets routing is carried out in user space, we evaluated the user-land version of a Click IP router within our testbed. During these trials we measured a boundary forwarding rate of 120,000 packet/s, with a decrease of a factor 6 compared to the capacity of the in-kernel Click onto the same hardware (about 750,000 packet/s). This difference that can only be induced by the transit of packets through the kernel, and not to a lack of computing resources (the router employs just the 15% of the CPU). Hence, the NetBoxIT capacity recorded in setup (A) seems to have a sense, if one considers that NS-3 is not particularly optimized for routing activities, and that the routers within the simulation are even two.

Very recent studies are in agreement with this reasoning and demonstrate that skipping the kernel during the packets processing can lead up to tremendous improvements. For instance, in [42], the novel Netmap library is experimented to minimize the kernel participation into the shift of data between network cards and applications: multiple packets are moved by a single system call, dynamic packet buffering is substituted by static allo-

cations, and packet buffers are directly shared between the kernel and user space, so that their replication is reduced. The benefit is huge: a Netmap-aware, user-space bridge is 10 times faster than the in-kernel version; or, pktgen, the well-known packet generator, can easily saturate a 10 Gb/s interface with 64-byte packets when employing Netmap, whilst the usual kernel version can barely accomplish a rate of 4 Mpacket/s.

In Figure 6.20 (right) we also reported the end-to-end overheads (caused by VETH peers) accumulated by the frames along their journey, against the intrinsic emulation latency required by all the netboxes of the chain (singularly, the processing of each netbox varies between 23 µs in setup A, and 38 µs in setup E, depending on the related packet rate). As we previously mentioned, the VETH overheads always kept to a minimum of 3-4 µs, and did not increase as we were prolonging the topology (as a quick comparison, the ModelNet granularity - 100 µs - can bring to a cumulative overhead of 1 ms in a 10-hop configuration). Moreover, it is worth noting that when we were experimenting the emulation of the Emergency Network, the latencies of the respective networks were much larger than the interconnections overheads: therefore, these could be neglected. But now, using fast networks, they can be much more comparable. Hence, the main doubt is: how relevant is this overhead in respect of the kind of network that we emulate ? Regrettably, there is not a unique answer. For example, an overhead of 4 µs can be perfectly satisfactory, if we compare it against the cumulative delay of a chain of two 10-100 Mb/s segments transmitting a 500-byte frame (800-80 µs), but it is surely not acceptable if we emulate a couple of Gigabit links with 64-byte packets, since in the latter case the VETH overheads can be even greater than the network transmission delays. Nevertheless, further improvements are almost possible, and in the next Section 6.4 we will discuss about the techniques that could probably be utilized to overcome these limitations in the near future.

The last remaining question that must be answered is how far we can go with this testbed architecture, and which kind of experiments we can carry out reliably. Firstly, it is necessary to remark that we measure the capacity of the emulated networks (inside NS-3), and the performance of a whole netbox (as an emulation machine) by employing two different metrics: the former in bits per second, whereas the latter in packets per second. When we specify the capacity of a link within a NS-3 simulation, its value represents the key constraint to take into account: hence, the time required for transmitting a frame varies accordingly with the size of the frame itself (for instance, if we set up a 100 Mb/s link, then a 500-byte frame is transmitted in 40 µs). But, differently, the capacity constraint of

a netbox derives from its computing efficiency, and is measured in terms of packet rate (which is fairly independent from the frames size).

To establish if a netbox is sufficiently efficient to honor the timing of the hosted virtual network, one should always bear in mind that the capacity of the emulator overrides the capacity of the emulated network without exception. Let us clarify this with a brief sample. In setup (A), the boundary rate of a single netbox has been documented to be 44,000 packet/s. As the frames processing is not influenced by their size, that netbox can reliably accommodate the emulation of an link whose capacity can vary within 22.5 Mb/s and 528 Mb/s, depending on the ingoing frames size (from 64 up to 1500-byte). Or rather, since the emulator needs 23 µs to handle each frame, we cannot instantiate within the NS-3 emulation a network template that is faster than that. In other words, we cannot reasonably require the simulated network to be faster than the emulator itself. This sentence could sound obvious, but it should be clearly held in mind that, if a mixed traffic load is offered to a netbox, the strictest constraint (to respect to assure the reliability of the emulation) is determined by the size of the shortest frames (i.e., if we refer to setup (A) again, and we suppose to inject 64 and 500-byte frames concurrently, the emulator offers a capacity limited to 22.5 Mb/s, and it is a nonsense to instantiate a link with a higher capacity within the NS-3 emulation).

6.4 Future opportunities

As we previously mentioned, Netmap is a library for the high-speed frames exchange between a user application and a network device. In theory, it would be interesting to exploit it to create a fast software channel between a netbox and a physical NIC. Unfortunately, this cannot be done neither straightforwardly, nor easily, as the EmuNet device employed within the NS-3 simulation is not ready to employ the Netmap library: in fact, it relies on the conventional kernel mechanisms as a means to communicate with a network card. Instead, Netmap exposes the NIC buffers directly within the application address space, employing the memory-mapped I/O Unix system calls. Thanks to this mechanism, every time an application has to exchange information with a device, it should simply retrieve a file descriptor related to a shared memory area, before reading from or writing into it; after that, data can be asynchronously migrated toward the physical device with a reduced kernel involvement. Thus, a completely new approach that requires the modification of the

application source code.

Moreover, another interesting scenario could be opened by the utilization of shared buffers for high-speed communications between different NS-3 emulators directly, with no kernel participation. At present, we think that VALE [43] could be a tool somewhat similar to what we were aiming at. Preliminary results are indicating that it can be used to exchange 60-byte frames between user processes at 17.6 Mpacket/s, just 50-60 ns per packet, a latency about two orders of magnitude smaller than with VETH peers. Such an advance could support 10 Gb/s linkages among netboxes even with short packets, allowing NetBoxIT to be used in a much wider range of investigations.

Chapter 7

Conclusions

In this work we presented NetBoxIT, a modular, flexible, and scalable platform that employs LXC containers and the NS-3 simulator for the investigation of heterogeneous networks. We described its design and implementation issues, and exploited it for the assessment of the end-to-end performance of a multi-part Emergency Network.

Our research was mainly focused on examining, by means of plain and intuitive trials, if the proposed testbed architecture could lead to realistic and reliable results. Moreover, NetBoxIT has been conceived to be an open framework, where virtual networks can inter-operate with real devices and true applications, simply exploiting the testbed Ethernet ports. As a consequence, we could attach a true VoIP application and a real IP router to carry on some experiments, with the aim at increasing the realism of the emulations. Of course, this feature could be utilized for a subjective estimate of the quality of experience perceived by the end user and refine the network functioning parameters consequently.

Finally, we investigated the performance bottlenecks of our architecture, to show which are its current limitations and to identify how these could be overcome in the future.

Bibliography

[1] M. Portmann, A. Pirzada, Wireless mesh networks for public safety and crisis management applications, Internet Computing, IEEE 12 (1) (2008) 18 –25. doi:10.1109/MIC.2008.25.

[2] I. F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: a survey, Comput. Netw. ISDN Syst. 47 (4) (2005) 445–487. doi:10.1016/j.comnet.2004.12.001. URL http://dx.doi.org/10.1016/j.comnet.2004.12.001

[3] D. Vassiliadis, A. Garbi, G. Calarco, M. Casoni, A. Paganelli, R. Morera, C.-M. Chen, M. Wodczak, Wireless networks at the service of effective first response work: The e-sponder vision, in: Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on, 2010, pp. 210 –214. doi:10.1109/ISWPC.2010.5483797.

[4] G. Calarco, M. Casoni, A. Paganelli, D. Vassiliadis, M. Wódczak, A satellite based system for managing crisis scenarios: The e-sponder perspective, in: Advanced satellite multimedia systems conference (asma) and the 11th signal processing for space communications workshop (spsc), 2010 5th, 2010, pp. 278 –285. doi:10.1109/ASMS-SPSC.2010.5586905.

[5] H. Skinnemoen, Creating the next generation dvb-rcs satellite communication amp; applications: The largest standards initiative for satellite communication inspires new oppurtunities, in: Advanced satellite multimedia systems conference (asma) and the 11th signal processing for space communications workshop (spsc), 2010 5th, 2010, pp. 147 –154. doi:10.1109/ASMS-SPSC.2010.5586868.

[6] D. Gupta, K. Yocum, M. Mcnett, A. C. Snoeren, A. Vahdat, G. M. Voelker, To infinity and beyond: time warped network emulation, in: In ACM Symposium on Operating Systems Principles, 2005.

[7] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, M. Singh, Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols, in: Wireless Communications and Networking Conference, 2005 IEEE, Vol. 3, 2005, pp. 1664 – 1669 Vol. 3. doi:10.1109/WCNC.2005.1424763.

[8] NS-3 Project Homepage.
URL http://www.nsnam.org

[9] T. R. Henderson, S. Roy, S. Floyd, G. F. Riley, ns-3 project goals, in: Proceeding from the 2006 workshop on ns-2: the IP network simulator, WNS2 '06, ACM, New York, NY, USA, 2006. doi:10.1145/1190455.1190468.
URL http://doi.acm.org/10.1145/1190455.1190468

[10] E. Weingärtner, H. Vom Lehn, K. Wehrle, A performance comparison of recent network simulators, in: Proceedings of the 2009 IEEE international conference on Communications, ICC'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 1287–1291.
URL http://dl.acm.org/citation.cfm?id=1817271.1817510

[11] The network simulator ns-2.
URL http://www.isi.edu/nsnam/ns/

[12] J. Zhou, Z. Ji, R. Bagrodia, Twine: A hybrid emulation testbed for wireless networks and applications, in: In IEEE INFOCOM 2006, Society Press, 2006, pp. 23–29.

[13] P. De, A. Raniwala, S. Sharma, T. cker Chiueh, Mint: A miniaturized network testbed for mobile wireless research, in: In The 24 th Annual Joint Conference of the IEEE Computer and Communications Societies, 2005, pp. 2731–2742.

[14] T. Clancy, B. Walker, Meshtest: Laboratory-based wireless testbed for large topologies, in: Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on, 2007, pp. 1 –6. doi:10.1109/TRIDENTCOM.2007.4444659.

[15] S. Sanghani, T. X. Brown, S. Bhandare, S. Doshi, Ewant: The emulated wireless ad hoc network testbed, in: In IEEE WCNC, 2003, pp. 1844–1849.

[16] M. Carbone, L. Rizzo, Dummynet revisited, SIGCOMM Comput. Commun. Rev. 40 (2) (2010) 12–20. doi:10.1145/1764873.1764876.
URL http://doi.acm.org/10.1145/1764873.1764876

[17] S. Agarwal, J. Sommers, P. Barford, Scalable network path emulation, in: Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 219–228. doi:10.1109/MASCOT.2005.61.
URL http://dx.doi.org/10.1109/MASCOT.2005.61

[18] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, D. Becker, Scalability and accuracy in a large-scale network emulator, SIGOPS Oper. Syst. Rev. 36 (SI) (2002) 271–284. doi:10.1145/844128.844154.
URL http://doi.acm.org/10.1145/844128.844154

[19] M. Pizzonia, M. Rimondini, Netkit: easy emulation of complex networks on inexpensive hardware, in: Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities, TridentCom '08, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2008, pp. 7:1–7:10.
URL http://dl.acm.org/citation.cfm?id=1390576.1390585

[20] User-mode Linux Kernel.
URL http://user-mode-linux.sourceforge.net/

[21] The LXC Linux Containers.
URL http://lxc.sourceforge.net

[22] S. Bhatia, M. Motiwala, W. Muhlbauer, Y. Mundada, V. Valancius, A. Bavier, N. Feamster, L. Peterson, J. Rexford, Trellis: a platform for building flexible, fast virtual networks on commodity hardware, in: Proceedings of the 2008 ACM

CoNEXT Conference, CoNEXT '08, ACM, New York, NY, USA, 2008, pp. 72:1–72:6. doi:10.1145/1544012.1544084.
URL http://doi.acm.org/10.1145/1544012.1544084

[23] A. Alvarez, R. Orea, S. Cabrero, X. G. Pañeda, R. García, D. Melendi, Limitations of network emulation with single-machine and distributed ns-3, in: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, SIMUTools '10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2010, pp. 67:1–67:9. doi:10.4108/ICST.SIMUTOOLS2010.8630.
URL http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8630

[24] B. Lantz, B. Heller, N. McKeown, A network in a laptop: rapid prototyping for software-defined networks, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Hotnets-IX, ACM, New York, NY, USA, 2010, pp. 19:1–19:6. doi:10.1145/1868447.1868466.
URL http://doi.acm.org/10.1145/1868447.1868466

[25] VETH.
URL http://code.google.com/p/veth

[26] J. Ahrenholz, Comparison of core network emulation platforms, in: MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010, 2010, pp. 166 –171. doi:10.1109/MILCOM.2010.5680218.

[27] J. Ahrenholz, T. Goff, B. Adamson, Integration of the core and emane network emulators, in: MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011, 2011, pp. 1870 –1875. doi:10.1109/MILCOM.2011.6127585.

[28] J. Zhang, Z. Qin, Taprouter: an emulating framework to run real applications on simulated mobile ad hoc network, in: Proceedings of the 44th Annual Simulation Symposium, ANSS '11, Society for Computer Simulation International, San Diego, CA, USA, 2011, pp. 39–46.
URL http://dl.acm.org/citation.cfm?id=2048370.2048376

[29] M. Skjegstad, F. Johnsen, J. Nordmoen, An emulated test framework for service discovery and manet research based on ns-3, in: New Technologies, Mobility and

Security (NTMS), 2012 5th International Conference on, 2012, pp. 1 –5. doi:10.1109/NTMS.2012.6208683.

[30] G. Calarco, M. Casoni, Virtual networks and software router approach for wireless emergency networks design., in: VTC Spring, IEEE, 2011, pp. 1–5.
URL http://dblp.uni-trier.de/db/conf/vtc/vtc2011s.html

[31] E. Kohler, R. Morris, B. Chen, J. Jannotti, M. F. Kaashoek, The click modular router, ACM Trans. Comput. Syst. 18 (3) (2000) 263–297. doi:10.1145/354871.354874.
URL http://doi.acm.org/10.1145/354871.354874

[32] G. Calarco, C. Raffaelli, Implementation of implicit qos control in a modular software router context, in: Proceedings of the Third international conference on Quality of Service in Multiservice IP Networks, QoS-IP'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 390–399. doi:10.1007/978-3-540-30573-6_30.
URL http://dx.doi.org/10.1007/978-3-540-30573-6_30

[33] (C)RUDE.
URL http://rude.sourceforge.net//

[34] N. Baldo, M. Requena-Esteso, J. Núñez Martínez, M. Portolès-Comeras, J. Nin-Guerrero, P. Dini, J. Mangues-Bafalluy, Validation of the ieee 802.11 mac model in the ns3 simulator using the extreme testbed, in: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, SIMUTools '10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2010, pp. 64:1–64:9. doi:10.4108/ICST.SIMUTOOLS2010.8705.
URL http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8705

[35] J. A. R. P. de Carvalho, H. Veiga, P. A. J. Gomes, C. F. F. P. R. Pacheco, N. Marques, A. D. Reis, Laboratory performance of wi-fi point-to-point links: a case study, in: Proceedings of the 2009 conference on Wireless Telecommunications Symposium, WTS'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 143–147.
URL http://dl.acm.org/citation.cfm?id=1689059.1689088

[36] O. Grondalen, P. Gronsund, T. Breivik, P. Engelstad, Fixed wimax field trial measurements and analyses, in: Mobile and Wireless Communications Summit, 2007. 16th IST, 2007, pp. 1 –5. doi:10.1109/ISTMWC.2007.4299213.

[37] J. Martin, B. Li, W. Pressly, J. Westall, Wimax performance at 4.9 ghz, in: Aerospace Conference, 2010 IEEE, 2010, pp. 1 –8. doi:10.1109/AERO.2010.5446943.

[38] LinPhone, free SIP VoIP Client.
URL http://www.linphone.org

[39] Wireshark.
URL http://www.wireshark.org

[40] S. Boyd-Wickizer, A. T. Clements, Y. Mao, A. Pesterev, M. F. Kaashoek, R. Morris, N. Zeldovich, An analysis of linux scalability to many cores, in: Proceedings of the 9th USENIX conference on Operating systems design and implementation, OSDI'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 1–8.
URL http://dl.acm.org/citation.cfm?id=1924943.1924944

[41] R. Bolla, R. Bruschi, The ip lookup mechanism in a linux software router: Performance evaluation and optimizations, in: High Performance Switching and Routing, 2007. HPSR '07. Workshop on, 2007, pp. 1 –6. doi:10.1109/HPSR.2007.4281242.

[42] L. Rizzo, Netmap: a novel framework for fast packet i/o, in: Proceedings of the 2012 USENIX conference on Annual Technical Conference, USENIX ATC'12, USENIX Association, Berkeley, CA, USA, 2012, pp. 9–9.
URL http://dl.acm.org/citation.cfm?id=2342821.2342830

[43] VALE, a switched ethernet for virtual machines.
URL http://info.iet.unipi.it/~luigi/papers/20120608-vale.pdf

Publications

Journals

- G. Calarco, M. Casoni, On the effectiveness of Linux containers for network virtualization, Simulation Modelling Practice and Theory, Volume 31, February 2013, Pages 169-185, ISSN 1569-190X, 10.1016/j.simpat.2012.11.007.

Conferences

- Calarco, G.; Casoni, M.; Paganelli, A.; Saladino, D.; Schaap, M.; , “A Resilient ICT System for Managing Crisis Scenarios”, 7th International Conference on Critical Information Infrastructures Security, 2012

- Calarco, G.; Casoni, M.; , "NetBoxIT: An open testbed for heterogeneous networks study," Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International , vol., no., pp.956-961, 27-31 Aug. 2012 doi: 10.1109/IWCMC.2012.6314334

- Calarco, G.; Casoni, M.; , "NetBoxIT: Virtual emulation integrated testbed for the heterogeneous networks design," Local & Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on , vol., no., pp.1-2, 13-14 Oct. 2011 doi: 10.1109/LANMAN.2011.6076929

- Calarco, G.; Casoni, M.; , "Virtual Networks and Software Router Approach for Wireless Emergency Networks Design," Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd , vol., no., pp.1-5, 15-18 May 2011 doi: 10.1109/VETECS.2011.5956396

- Calarco, G.; Casoni, M.; Paganelli, A.; Vassiliadis, D.; Wódczak, M.; , "A satellite based system for managing crisis scenarios: The E-SPONDER perspective," Advanced satellite multimedia systems conference (asma) and the 11th signal processing for space communications workshop (spsc), 2010 5th , vol., no., pp.278-285, 13-15 Sept. 2010 doi: 10.1109/ASMS-SPSC.2010.5586905

- Vassiliadis, D.; Garbi, An.; Calarco, G.; Casoni, M.; Paganelli, A.; Morera, R.; Chen, C.-M.; Wodczak, M.; , "Wireless networks at the service of effective first response work: The E-SPONDER vision," Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on , vol., no., pp.210-214, 5-7 May 2010 doi: 10.1109/ISWPC.2010.5483797

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° [242411].