AiroLAB: Leveraging on Virtualization to Introduce Controlled Experimentation in Operational Multi-hop Wireless Networks

Roberto Doriguzzi Corin, Roberto Riggio, Daniele Miorandi, and Elio Salvadori

CREATE-NET, via alla Cascata 56/D, IT - 38100, Povo, Trento, Italy

Abstract. Network Virtualization represents one of the most promising approach to unlock innovation in current network technologies. This paper presents *AiroLAB*, a wireless network virtualization solution aimed at providing Wireless Internet Service Providers with an effective tool to support experimental testing of novel network–level solutions on *production* networks. In the paper, the design choices which lie at the basis of *AiroLAB* are presented and discussed together with a prototype implementation. The outcomes of measurement activities performed on a small–scale testbed demonstrate the strength of the proposed framework in preserving guaranteed performance for production traffic while allowing several experimental instances to run on a network in operation.

Keywords: network virtualization, multi–hop wireless networks, embedded devices, resource constrained environment.

1 Introduction

One of the most promising approaches to enable innovation in today's network is Network Virtualization (NV) [1,2]. In general terms, NV refers to the possibility of pooling together low–level hardware and software resources belonging to a networked system into a single administrative entity. In such a way network resources could be effectively shared in a transparent way among different logical network instances. NV differs from "conventional" virtualization techniques (aimed mostly at virtualizing computing and storage resources) in that it explicitly addresses low–level resources (e.g., bandwidth), well below the IP waist. From an academic standpoint, NV can be seen as a tool for evaluating novel Internet architectures ("clean–slate approaches") in large–scale realistic environments. Similarly, from a business perspective, NV can change the functional role of Internet Service Providers (ISPs) by decoupling the provisioning of the physical infrastructure from the provisioning of communication/computing resources. This could set the basis for the introduction of new business models and stakeholders in the Internet ecosystems (i.e. Infrastructure Providers, Virtual Network Providers and Service Providers). Finally, NV can enable a smooth and controlled introduction of novel services in an operational network by providing means to isolate them from already deployed applications, thereby unlocking innovation in telecommunication networks.

While several NV architectures and solutions have been proposed in recent years, most (if not all) of them were designed and developed for wired networks, characterized by virtually unlimited processing/storage power and link bandwidth (PlanetLab [3], VINI [4], etc). On the other hand, a particularly challenging, yet interesting, domain for NV is that of wireless multi–hop networks [5,6] with a particular emphasis on Wireless Mesh Networks (WMNs). WMNs are a cost–effective access networking paradigm, which represents an interesting solution in scenarios where a fixed wired infrastructure is either not feasible (e.g., in the case of mobile nodes such as vehicular ad hoc networks) or not economically attractive (e.g., access to Internet in developing countries). However, despite these expectations, very few studies have been performed, so far, on virtualization in resource–constrained environments in general, and multi–hop wireless networks in particular. Furthermore, the available literature on the theme focuses mainly on comparing how different wireless medium virtualization techniques affect the overall network slices performance in term of isolation and stability [7,8].

The aim of this paper is to introduce *AiroLAB*, a novel network virtualization framework specifically tailored to multi–hop wireless networks [9]. AiroLAB was designed to provide Wireless Internet service providers (WISPs) with an effective virtualization solution. In *AiroLAB*, an innovative mechanism to assess wireless link capacity and realize soft–performance isolation among virtual networks instances allows production traffic to share part of the available network resources with a variable number of network slices, where novel solutions, such as new routing protocols, services or network operation tools, can be experimentally tested in a severely controlled yet realistic environment with no impact on the operational traffic. Compared to [9], which is mainly focused on the description of the objectives and constraints that have driven the design of our Network Virtualization framework, in this paper we provide a more accurate analysis of the experimental results obtained in a small–scale wireless testbed.

The paper is organized as follows: Sec. 2 provides an overview of the main challenges behind Network Virtualization, emphasizing how *AiroLAB* differentiates from solutions proposed in literature. Section 3 describes the *AiroLAB* architecture and protocols. Section 4 presents the results of experimental tests carried out with a prototypical implementation of *AiroLAB* while Sec. 5 concludes the paper.

2 Network Virtualization: An Overview

Network Virtualization research main challenge refers to the definition of appropriate and efficient algorithms, architectures and protocols to effectively share a common physical network infrastructure, splitting it into several logical instances (generally referred to as "slices") composed of virtual links and virtual network nodes [10]. Network nodes should be fully programmable to allow the instantiation of several network instances, each one potentially based on a different architecture. Several projects worldwide are working on the various aspects underpinning NV: GENI in USA [11], 4WARD [12], FEDERICA [13] in Europe and AKARI in Japan [14].

An effective NV solution shall satisfy several requirements [15]: scalability (performance should not depend on the number of slices), isolation (among slices), flexibility (fully programmable network elements), efficiency (limited overhead due to virtualization), manageability (it should work in multi-domain scenarios) and heterogeneity (in term of underlying technologies, end-users,...). When network virtualization solutions are used for running concurrent research experiments in dedicated testbeds, two further requirements are lack of inter–experiments interference and experiments repeatability.

Most of the research groups proposing NV mechanisms in wireless networks have been focusing in large–scale testbed scenarios, i.e. ORBIT [16] or GENI [11], where several experiments could run concurrently on the same physical infrastructure. Due to their strong focus on creating a stable testing environment where experiments should be repeatable and should not affect each other when running on concurrent slices, most of the works have focused on evaluating wireless virtualization techniques to realize performance isolation. This has been obtained by exploring both dimensions of (i) virtualization of the wireless medium (through multiplexing techniques like SDM, FDM, CDM or TDM [17]) and (ii) virtualization of the network node. Studies regarding the feasibility of each of these approaches have been already provided in literature with an analysis of their pros and cons [7,8,18,19].

Compared to those works, the scenario addressed in this paper is a sort of intermediate one between such purely research approaches applied to *dedicated testbed* network infrastructures, and a conservative approach (pursued so far by most of the operators) where novel services or recent protocols are tested on a small-scale testbed separated from the main production network. In fact, *AiroLAB* investigates techniques and architectures to provide a NV framework specifically tailored for *production networks*, where operational traffic is fully guaranteed over a "privileged" slice, while all novel (experimental) services and protocols under test are run on "background" (lower–priority) slices. Our aim is somehow similar to the ones pursued by solutions such as Cabernet [20,21] and generalized in [22] on carrier-class networking equipment; however, given the specific technological domain under consideration, we are providing an emphasis toward the possibility for a Wireless ISP (WISP) to perform experimental activities in a controlled fashion directly on its production network.

3 The Architectural Framework of *AiroLAB*

Multi–hop wireless networks are usually built using commodity components and are characterized by rather limited computing capabilities. Such a scenario calls for a radically new approach to network virtualization, where the trade–off between flexibility and scalability shall, due to pragmatical consideration, drift towards the latter. Before analyzing the intricacies behind the proposed NV architecture, a simplified network scenario is described in this Section to emphasize objectives and constraints that have driven the *AiroLAB*'s design.

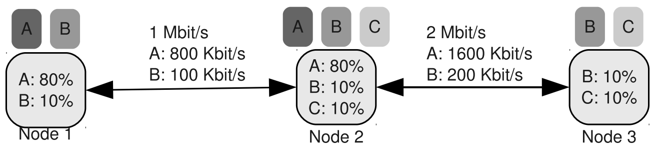


Fig. 1. Simplified deployment scenario

Fig. 1 sketches a simplified setup where a network, composed of three nodes organized in a string topology, is running three distinct *slices*: one production slice (A), and two experimental slices (B and C). In this scenario, links are symmetric and their capacity is assumed to be time–invariant. Moreover, mesh routers are equipped with a single radio interface.

In this simplified scenario, the production slice A is assigned 80% of the resources in the network, while the two experimental slices equally share the remaining 20% of resources. It is worth noticing that, with our architecture, we do not aim at supporting hundreds or even tens of concurrent slices, instead we foresee a scenario where 5 to 10 slices share the overall network resources. Such limitation is mandated by the computing and storage constraints that characterized currently used wireless multi–hop networking devices.

Traffic shaping is performed at each node in order to limit the amount of network resources used by each *sliver*. In this simplified setup the resources that each sliver can exploit are upper bounded by a fixed threshold derived from the relative performance goal given during the planning phase. As a result, slice A "sees" an 800 Kb/s bidirectional link between node 1 and node 2, while the available bandwidth between node 2 and node 3 is 1600 Kb/s. In this setup some bandwidth is voluntary left unused. However scenarios where a sliver can have full access to all the available bandwidth are also supported.

3.1 Assessing Wireless Link Capacity

Estimating the capacity of a wireless link is not trivial due to the use of a shared medium; in fact interference coming from external sources, changes in the propagation characteristics or interference from the same signal traveling along different paths make the link's total capacity fluctuate over time. Even if we limit our attention on communications realized using the IEEE 802.11 facility of standards, an ideal estimator of the link capacity from an Access Point toward a generic Station should take into account both the the data frame SNR (measured at the receiving station) and the ACK frame SNR (measured at the access point). Such a level of precision is difficult to achieve without introducing additional signaling and/or modifying the standard IEEE 802.11 MAC operations.

In this work we decided to use an indirect way of assessing a link's total capacity based on the rate adaption functionalities already available in current IEEE 802.11 devices. Rate adaptation algorithm aims at dynamically selecting the transmission rate in order to achieve optimal performance under varying operating conditions. Rate adaptation is left unspecified by the IEEE 802.11 standard, as a result of the years a considerable number of solutions have been proposed by both the academic and the industrial worlds.

Our work builds on top of currently available rate–control algorithms for IEEE 802.11–based wireless networks. In particular we exploit the Minstrel [23] algorithm. The minstrel rate–control algorithm aims at selecting the transmission rate that maximizes the throughput. In order to so, the algorithm collects statistics of all the packets that have been transmitted. This data is then exploited to compute the probability of a successful transmission P_{ab} between a pair of nodes, a and b, for each available data–rate. In order to cope with environmental changes, minstrel uses an Exponential Weighted Moving Average (EWMA) based approach. EWMA has a smoothing effect, so that new results have a larger influence on the selected rate. Finally, if D_{tx} is the time spent for a single transmission, and B is the packet length, the empirical throughput T_{ab} that is used by *AiroLAB* as an estimation of the wireless link capacity is computed as follows:

$$T_{ab} = \frac{P_{ab}B}{D_{tx}}, \qquad (1)$$

3.2 Providing Soft–Performance Isolation

AiroLAB provides soft–performance isolation between slivers by leveraging on the Hierarchical Token Bucket (HTB) traffic shaping facilities provided by the Linux kernels 2.6.x. HTB organizes traffic classes in a tree structure; each class is assigned an average rate (*rate*) and a maximum rate (*ceil*). Three class types exist: root, inner and leaf. A root class corresponds to a physical link; its bandwidth is the one currently available for transmission. Leaf classes, placed at the bottom of the hierarchy, correspond to a given type of traffic (e.g., TCP-controlled or VoIP etc.). Two internal token buckets are maintained for each class. Classes which have not exceeded their rate can unconditionally transmit; classes which have exceeded their allowed rate but not their upper limit (ceil) can transmit only borrowing unused bandwidth, if available, from other classes. In order to borrow bandwidth, a request is propagated upwards in the tree. A request that would exceed the ceil limit is terminated. A request that would satisfy the allowed rate is accepted. A request that would not satisfy the allowed

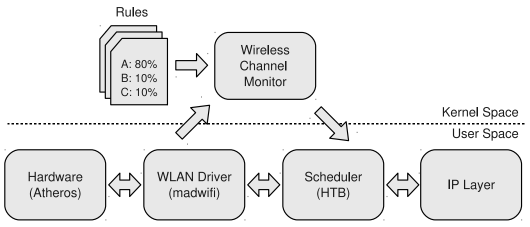


Fig. 2. *AiroLAB* wireless channel monitor architecture

rate constraint but the ceil one is propagated upwards until the procedure is completed.

Due to the stochastic nature of the wireless links capacity, an HTB scheduler alone is not able to deliver performance fairness among competing traffic flows in wireless networks. In order to address such an issues we devised and implemented a wireless channel monitor which exploits the channel statistics computed by the wireless driver in order to properly distribute the available bandwidth among the slivers running in a node. Figure 2, sketches the the architecture of the *AiroLAB* wireless channel monitor. The overall link capacity T_{ab} is assigned to the HTB's root class, while each sliver is associated to a leaf class in the HTB hierarchy. Available bandwidth is distributed among the slivers according to a set of input *policies*. The wireless channel monitor is implemented in the form of a software process running within each wireless router and periodically updates the HTB's configuration in order to reflect the actual channel capacity. HTB's configuration is also updated if either a new slice is deployed over the network or if the *policies* have changed.

3.3 A Description of the Node–Level Architecture

The *AiroLAB* framework design, whose architecture is sketched in Fig. 3, has been centered around two open source tools based on GNU/Linux operating

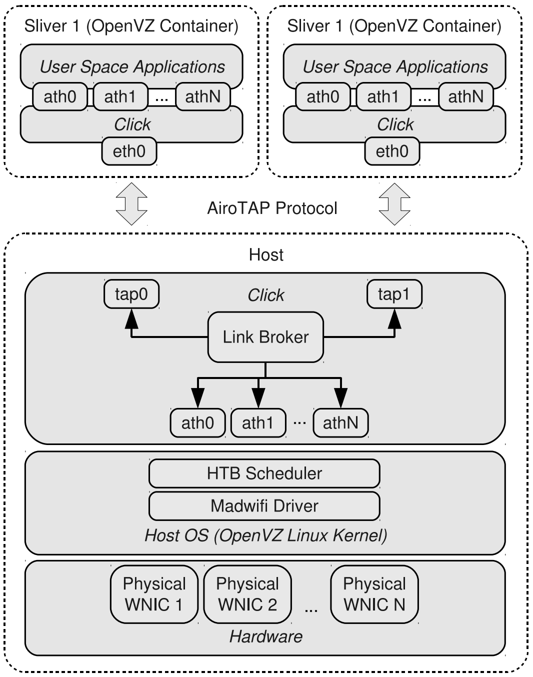


Fig. 3. *AiroLAB* node–level architecture

system: OpenVZ [24] and Click [25]. OpenVZ consists of a modified Linux kernel tree that supports virtualization, isolation and resource management and a set of user–level tools that allows the installation, configuration and maintenance of the virtual environments (also known as *containers)*. Container–based virtualization solutions are typically characterized by reduced overhead and thus better performance. They also provide good performance isolation (in terms of CPU cycles, memory consumption, and storage), because processes running within a container do not significantly differ from processes running in the hosting system. Thus, it is possible to apply existing resources sharing techniques, such as HTB for traffic scheduling. The major drawback of container-based virtualization solutions is that, since a single kernel is used for every sliver, kernel modifications are not allowed.

Within OpenVZ, each Virtual Environment (VE) performs and executes exactly like a stand–alone host; a container can be rebooted independently and can have root access, users, IP addresses, memory, processes, files, applications, system libraries and configuration files. Moreover, OpenVZ provides a resource management system that controls the amount of resources available for the environments. The controlled resources include parameters such as CPU power, disk space, and set of memory-related parameters. Furthermore, unlike alternative container-based solutions such as Linux-VServer [26], OpenVZ provides full virtualization of the networking subsystem allowing each virtual environment to create its own internal routing or firewall setups.

Due to the limitations imposed by the use of OpenVZ, namely the impossibility to run customized kernel images in different slivers, we decided to implement our virtualization stack in user–space using the Click modular router [25]. Our approach is not meant to replace OpenVZ, but rather to extend it in order to support flexible virtualization of the wireless resources. In fact, albeit characterized by an higher overhead in comparison to pure kernel–level implementation, Click–based solutions are highly customizable allowing us to circumvent the flexibility limitations of typical container based solutions [27]. Table 1 summarizes the trade–offs involved in the most relevant virtualization techniques currently available, namely containers, hypervisor, and hosted VMM. *AiroLAB* belongs to the second columns (Containers w/ Click) in that on the one hand container–based virtualization is used to achieve performances and scalability, and, on the

Table 1. Taxonomy of network virtualization techniques and relevant features [27]

	Containers	**Containers w/ Click**	**Hypervisors**	**Hosted VMM**
Scalability	*Good*	*Good*	*n.a.*	*n.a.*
Fault/Security Isolation	*n.a.*	*n.a.*	*Good*	*Good*
Performance Isolation	*Good*	*Good*	*Good*	*Good*
Flexibility	*Poor*	*Good*	*Good*	*Good*
Code Re–Usability	*n.a.*	*Poor*	*Poor*	*Good*
Efficiency	*Good*	*Good*	*Good*	*n.a.*

other hand, user–space wireless network virtualization delivers high flexibility in terms of packet processing capabilities.

Click is used both within each sliver (*guest click*) and at the host operating system level (*host click*). More specifically, the Click instance running within a sliver provides the guest environment with a set of virtual interfaces (*ath0, ath1, . . . , athN*) implemented as Linux TAP devices. A TAP device operates at layer 2 of the traditional ISO/OSI networking stack and simulates an Ethernet device. User-space process, running within a sliver, can exploit the virtual interfaces to implement their routing strategy. Communication over the virtual interfaces can be done using three different frame formats:

– 802.3 headers (Ethernet). Used to expose a standard Ethernet interface.
– 802.11 headers (WiFi). Used to expose a wireless interface complaint with the IEEE 802.11 protocol. In this case the user–space applications must properly encapsulate their traffic in 802.11 frames.
– Radiotap. Used to expose a raw wireless interface. In this case the user–space applications must properly encapsulate their traffic using the radiotap [28] header format. The radiotap header format is a mechanism to supply additional information about 802.11 frames, from the driver to user–space applications, and from a user–space application to the driver for transmission.

In either situation, outgoing traffic is encapsulated by the *guest click* process and sent to the *host click* process through the virtual interface *eth0* provided by the OpenVZ Container. Please note that, if the user-space application is already using the radiotap header, no additional encapsulation is performed by the *guest click* process and the frame is delivered unchanged to the host operating system. The *host click* process receives the incoming frame and dispatches it to the suitable device according to a set of policies maintained by the *Link Broker.*

The *Link Broker* is a software module that can expose different connectivity graphs to the various slivers without requiring that the nodes must be physically separated (i.e., out of radio range). Connectivity graphs are defined on a per-slice basis allowing us to define a different topology for each slice. This is particularly useful to test novel routing strategies on a subset of the nodes. Moreover, if wireless routers are equipped with multiple radio interfaces, it is possible to create multiple slices (whose cardinality equals the number of radio interfaces) operating on orthogonal frequency bands, implementing therefore an FDM wireless network virtualization solution. Hybrid solutions where only a subset of the slivers operates on orthogonal frequencies are also supported. Albeit network connectivity graphs are defined at deployment time, they can change during the network operations in order to create connectivity scenarios that simulates different operating conditions (i.e. link failures/outages).

3.4 An Example of Network–Level Configuration

A possible use case of *AiroLAB* is sketched in Figure 4, where a production slice exploiting a legacy version of a routing protocol is running in parallel with

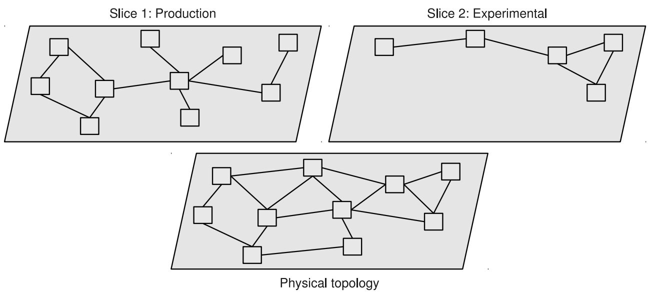


Fig. 4. Network–level configuration: an example with one production slice and one experimental slice sharing a common physical substrate

an experimental slice where novel routing strategies are being tested. In this scenario the *Link Broker* is used to expose two different connectivity graphs to the production and the experimental slices. On the other hand, the *Wireless Channel Monitor* is used to redistribute the available link bandwidth among the competing slices, 80% to the production slices and 20% to the experimental slices in this cases. Please note that, a minimum bandwidth, e.g. 1 Mb/s, can also be allocated to the production slice.

4 Results of the Experimental Activities

The main objective of the experimental measurements described in this Section is to prove the effectiveness of the *AiroLAB* framework in preventing traffic on a privileged slice being affected by traffic from other (lower–priority) slices, therefore guaranteeing a peaceful coexistence between operational and experimental traffic in a production network.

The wireless routers employed in the experimental set-up are built exploiting the PCEngines ALIX 2C2 (500MHz x86 CPU, 256MB of RAM) processor board. Operating system and application are stored on a 1 GB Compact Flash. Connectivity is provided by 2 Ethernet channels, 2 miniPCI slots and one serial port. PCEngines ALIX boards are equipped with two Mikrotik R52 WiFi IEEE 802.11a/b/g cards based on the Atheros AR2412 chipset. OpenWRT [29] has been selected as Operating system for our testbed, even though its original kernel has been replaced with a kernel provided by OpenVZ. The software configuration of the wireless routers is summarized in Table 2.

Table 2. AiroLAB wireless routers setup

Operating System	OpenWRT trunk (release 14748)
Linux kernel	OpenVZ 2.6.18-028stab056
Wireless drivers	MadWiFi trunk (release 2568)
Virtualization tools	vzctl-3.0.23, vzquota-3.0.12

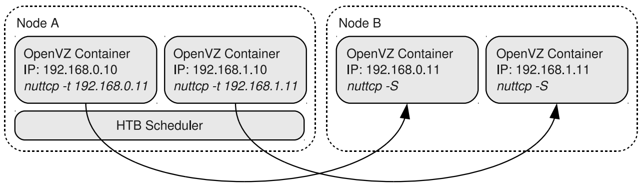


Fig. 5. Representation of the packet scheduling process for the case with two slivers

Several experimental scenarios have been set up to demonstrate *AiroLAB* performance isolation capabilities. The HTB configuration exploited during the measurements campaigns creates a traffic class for each sliver. For each traffic class, we specify the minimum (*rate*) and the maximum (*ceil*) throughput. Figure 5 shows the node setup for the case of two slivers. The performance metrics considered in each experimental scenario are throughput and delay. The results have been obtained by averaging the samples obtained as *nuttcp* benchmarks over 300 seconds with an averaging interval of 10 seconds.

In the first scenario, we use two wireless nodes, each one running two concurrent slivers sharing the same wireless interface. The privileged slice (#1) has higher transmission priority and a minimum guaranteed outbound bandwidth set to 10 Mb/s, and it provides an offered load of 10 Mb/s. The second slice (#2) has no minimum guaranteed outbound bandwidth, and it generates traffic off and on periodically with varying loads. The graph plotted in Fig. 6 shows the throughput and delay distribution per slice. As expected, when the wireless link is not saturated (from 0 to 140 secs), *AiroLAB* correctly limits the impact of slice (#2) on the privileged slice, by guaranteeing a stable throughput and averaged delays always below 10 msec. Of course, as soon as the offered load on Slice #2 leads to link saturation (from 140 to 160 secs), the throughput of Slice #1 is slightly affected as well as its averaged delay which increases up to 20 msec. However, compared to a similar test presented in [18], *AiroLAB* doesn't show any "drop to zero" effect when the second slice starts to carry some traffic, thus

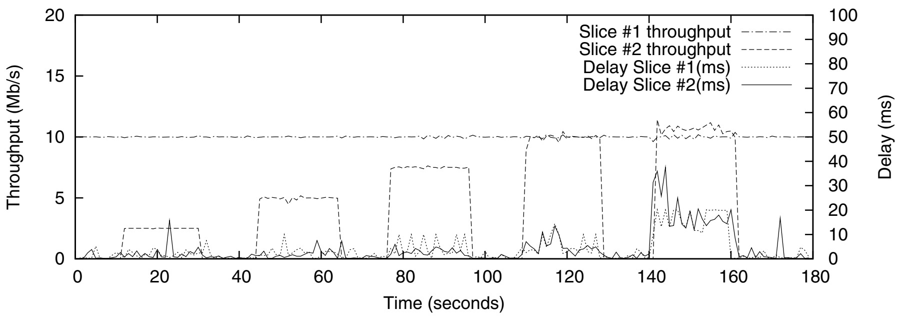


Fig. 6. Analysis of the cross–coupling effect among a privileged slice (#1) and an experimental one (#2) in term of throughput and delay

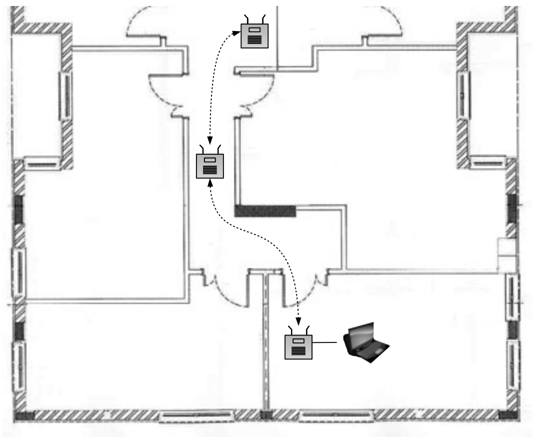
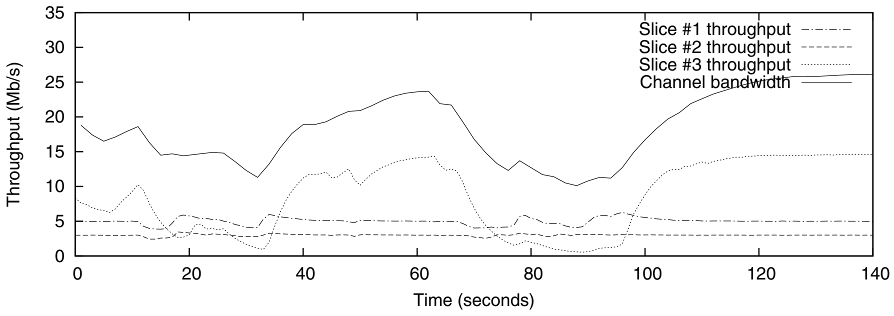


Fig. 7. The testing setup involved 2 nodes deployed in a typical office environment

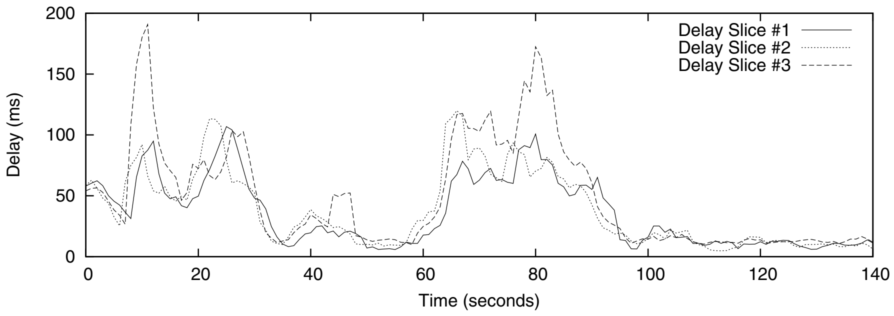
showing a more stable environment. It is worth noticing that no CPU reservation policies provided by OpenVZ have been used on the "priviliged" slice: such operation would have further limited the impact on the average delay for this slice, mainly due to some buffer processing delay on each physical node.

In the second scenario, we want to test the ability of the proposed architecture to effectively preserve production traffic in challenging conditions. To this purpose, the experimental setting sketched in Fig. 7 has been set up. We considered two wireless nodes, each one running three slivers sharing the same wireless interface. The experimental setup includes a wireless node connected to a PC lying on a desk in Office 1. Changes in link quality are emulated by moving the second node from Office 1 to another room. A continuous UDP flow is generated among the two nodes; its rate is such that the wireless link is *always* saturated.

In this scenario, there are two privileged slices (#1 and #2) with higher transmission priority and a minimum guaranteed outbound bandwidth set to 5 and 3 Mb/s respectively, while the remaining slice (#3) has no guaranteed bandwidth (one can suppose a WISP having slice #1 for production traffic and the remaining slices #2 and #3 for, respectively, testing a novel video–streaming service and for network management and monitoring).Moreover, Slice #1 and #2 provide an offered load of 5 and 3 Mb/s, while Slice #3 has no upper bounds on the maximum throughput it can inject in the wireless link. The results plotted in Fig. 8 show the throughput and delay distribution per–slice in different conditions of available wireless link capacity. As expected, *AiroLAB* guarantees that the throughputs of Slice #1 and #2 are only slightly affected by wireless link conditions to detriment of Slice #3. Results related to throughput measurements are summarized in Tab. 3. The impact on the average delay per slice is higher mainly due to the saturation conditions of the experimental scenario. However, it is worth noticing that an average delay lower than 150 msec is tolerable for video–streaming–based services; while network-management traffic can tolerate even higher delays [30]. As per the previous test scenario, the impact



(a) Throughput



(b) Delay

Fig. 8. Relative performance for the three slices in the second scenario

Table 3. Throughput statistics

	Slice #1	**Slice #2**	**Slice #3**
Average	5	3	8,85
Minimum	3,85	2,42	0,54
Maximum	6,27	3,46	14,6
Std Deviation	0,47	0,14	5,1
Confidence interval (95%)	±0,08	±0,02	±0,82

on the average delay for the slice running the production traffic could have been further minimized through an appropriate configuration of the CPU reservation via OpenVZ.

5 Conclusions

In this paper, we have presented *AiroLAB*, a novel virtualization framework specifically tailored to multi–hop wireless networks. *AiroLAB* has been designed with the explicit goal to empower WISP with an effective tool to allow production traffic to safely share part of the available network resources with a variable number of network slices where novel solutions, such as new routing protocols, services or network operation tools, can be experimentally tested in a severely controlled yet realistic environment. The architecture and protocols at the hearth of *AiroLAB* have been presented, discussed and compared with existing solutions. A first prototypical implementation of *AiroLAB* capable of supporting performance isolation between concurrent slices is described, together with experimental measurements obtained in a small–scale wireless testbed.

Despite the encouraging results presented in the paper, the proposed framework requires further development before reaching the stability level needed to enable its wide adoption. Among the possible research directions to enhance the current architecture, we believe that the ability of leveraging, by means of appropriate FDM approaches, the presence of multiple wireless interfaces (such that, for example, different slices could associate to dedicated radio channels), could definitely improve the efficiency and scalability of *AiroLAB* in realistic scenarios. Moreover, integration with currently available frameworks for automatic NV resources allocations, such as OMF [31], shall be cosidered for future evolution of the framework.

References

1. Feldmann, A., Kind, M., Maennel, O., Schaffrath, G., Wehrle, C.: Network Virtualization An Enabler for Overcoming Ossification. ERCIM News 77, 21–22 (2009)
2. Papadimitriou, P., Maennel, O., Greenhalgh, A., Feldmann, A., Mathy, L.: Implementing Network Virtualization for a Future Internet. In: Proc. of 20th ITC Specialist Seminar on Network Virtualization, Hoi An, Vietnam (2009)
3. Planet Lab project, `http://www.planet-lab.org`
4. VINI project, `http://www.vini-veritas.net`
5. Bruno, R., Conti, M., Gregori, E.: Mesh Networks: Commodity Multihop Ad Hoc Networks. IEEE Communications Magazine 43(3), 123–131 (2005)
6. Akyildiz, I., Wang, X., Wang, W.: Wireless mesh networks: a survey. Elsevier Computer Networks 47(4), 445–487 (2005)
7. Smith, G., Chaturvedi, A., Mishra, A., Banerjee, S.: 'Wireless Virtualization on Commodity 802.11 Hardware. In: Proc. of ACM WinTECH, Montreal, Quebec, Canada (2007)
8. Mahindra, R., Bhanage, G., Hadjichristo, G., Seskar, I., Raychaudhuri, D., Zhang, Y.: Space Versus Time Separation for wireless virtualization On an Indoor Grid. In: Proc. of EURO NGI, Krakow, Poland (2008)
9. Doriguzzi Corin, R., Riggio, R., Miorandi, D., Salvadori, E.: AiroLAB: A Framework Toward Effective Virtualization of Multi-hop Wireless Networks. International Journal of Communication Networks and Distributed Systems (2010) (accepted for publication)
10. Evans, J., Raychaudhuri, D., Paul, S.: Technical Document on Overview Wireless, Mobile and Sensor Networks. The GENI Project Office, Tech. Rep. GDD- 06-14 (2006)
11. GENI project, `http://www.geni.net`
12. 4WARD project, `http://www.4ward-project.eu`
13. FEDERICA project, `http://www.fp7-federica.eu`
14. AKARI project, `http://akari-project.nict.go.jp`

15. Chowdhury, N.M.M.K., Boutaba, R.: Network Virtualization: State of the Art and Research Challenges. IEEE Communications Magazine (July 2009)
16. Orbit Lab, `http://www.orbit-lab.org/`
17. Paul, S., Seshan, S.: Technical Document on Wireless Virtualization. The GENI Project Office, Tech. Rep. GDD-06-17 (2006)
18. Singhal, S., Hadjichristo, G., Seskar, I., Raychaudhuri, D.: Evaluation of UML based wireless network virtualization. In: Proc. of EURO NGI, Krakow, Poland (2008)
19. Bhanage, G., Seskar, I., Zhang, Y., Raychaudhuri, D.: Evaluation of OpenVZ for wireless testbed virtualization. WINLAB Rutgers University, Tech. Rep. 331 (2008)
20. Feamster, N., Gao, L., Rexford, J.: How to lease the Internet in your spare time. ACM SIGCOMM Computer Communications Review, 61–64 (January 2007)
21. Zhu, Y., Zhang-Shen, R., Rangarajan, S., Rexford, J.: Cabernet: Connectivity architecture for better network services. In: Proc. of Workshop on Rearchitecting the Internet (2008)
22. Schaffrath, G., Werle, C., Papadimitriou, P., Feldmann, A., Bless, R., Greenhalgh, A., Wundsam, A., Kind, M., Maennel, O., Mathy, L.: Network Virtualization Architecture: Proposal and Initial Prototype. In: Proc. of ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures, Madrid, Spain (2009)
23. Minstrel, `http://linuxwireless.org/`
24. OpenVZ, `http://openvz.org/`
25. The click modular router project, `http://read.cs.ucla.edu/click/`
26. Linux-VServer, `http://Linux-VServer.org/`
27. Nakao, A., Ozaki, R., Nishida, Y.: Corelab: An emerging network testbed employing hosted virtual machine monitor. In: Proc. of ACM ROADS, Madrid, Spain (2008)
28. Linux Radiotap, `http://www.radiotap.org/`
29. OpenWRT Linux Distribution, `http://openwrt.org/`
30. Szigeti, T., Hattingh, C.: End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs. Cisco Press (2004)
31. OMF, Control and Management Framework, `http://omf.mytestbed.net`