AiroLAB: A Framework Toward Effective Virtualization of Multi-hop Wireless Networks

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

CREATE-NET, via alla Cascata 56/D, IT - 38123, Povo, Trento, Italy

Abstract. In this work we introduce *AiroLAB*, a novel network virtualization framework specifically tailored to multi-hop wireless networks. *AiroLAB* departs from conventional network virtualization approaches by focusing on embedded, resource–constrained devices and by aiming at providing Wireless Internet Service Providers with an effective virtualization mechanism where network resources are shared between production traffic and a variable number of experimental slices allowing novel solutions and services to be tested in a controlled yet realistic environment. In the paper, the design choices at the hearth of *AiroLAB* are presented, together with an early–stage prototype implementation and experimental results obtained in a small–scale wireless network testbed.

Key words: network virtualization, multi–hop wireless networks, communications networks, embedded devices, resource constrained environments

1 Introduction

Network Virtualization (NV) is currently regarded as one of the most promising approaches to unlock and enable innovation in today's network [1, 2]. Generally speaking, NV can be seen as a tool for:

– Evaluating new, not necessarily backward compliant Internet architectures ("clean–slate approaches") in large–scale realistic environments, thereby helping to overcome the current Internet ossification;
– Changing the functional role and business model of Internet Service Providers (ISPs) by decoupling the provisioning of physical infrastructure from the provisioning of communication/computing resources. In such a way, the role currently covered, in most countries, by ISPs would be taken over by different entities: Infrastructure Providers, Virtual Network Providers and Service Providers. This could provide the basis for the introduction of new stakeholders in the Internet ecosystems improving the competition in this sector;
– Enabling the smooth and controlled introduction of novel services in an operational network by providing means to isolate them from already deployed applications, thereby promoting innovation in telecommunication networks;
– Moving logical instances of nodes and services across an infrastructure in order to optimize network performance and minimize operational expenditures. As

an example, moving services close to the users may lead to a decrease in the power consumption of a physical network, therefore contributing to a limitation of the network's carbon footprint.

Challenges in NV research include 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 [3, 4]. 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 [5], 4WARD [6], FEDERICA [7] in Europe and AKARI in Japan [8].

NV solutions depend heavily on the network substrate on which they need to operate. In particular, one challenging environment for NV solutions is that of wireless multi–hop networks [9, 10]. Wireless multi–hop networks are emerging as a cost–effective paradigm for enabling communications in those scenarios where the deployment of a fixed wireline infrastructure is either non feasible (e.g., in the case of mobile nodes such as vehicular ad hoc networks) or non economically attractive (e.g., access to Internet in developing countries). The complexity of introducing NV capabilities in such networks is exacerbated not only by the fragile stability of the wireless links, but also by the limited processing power and storage capabilities typically available on the devices employed in such systems.

While several NV architectures and solutions have been proposed in recent years, most (if not all) of them were developed for wired networks, characterized by virtually unlimited processing/storage power and link bandwidth (PlanetLab [11], VINI [12], G-Lab [13], etc). On the other hand, very few studies have been performed on resource–constrained environments in general, and multi–hop wireless networks in particular. Further, they have mainly focused on how different wireless medium virtualization techniques affect the overall network slices performance in term of isolation and stability [14, 15].

In this paper we introduce *AiroLAB*, a novel virtualization framework specifically tailored to multi–hop wireless networks. *AiroLAB* aims at providing Wireless Internet Service Providers (WISP) with an effective virtualization solution, allowing 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. In the rest of the paper, the building blocks that are at the base of the *AiroLAB*'s architecture and the protocols devised in order to support its operations are presented, discussed and compared with existing solutions. A first prototypical implementation of *AiroLAB* including link channel capacity estimation is described, together with experimental measurements obtained in a small–scale wireless network testbed.

The remainder of the paper is organized as follows. Sec. 2 surveys the main challenges in the design and development of network virtualization solutions for multi–hop wireless networks. Section 3 describes the *AiroLAB* architecture and protocols. Section 4 reports and discusses the outcomes of experimental tests carried out with a first prototype implementation of the proposed framework. Comparison with existing relevant solutions is carried out in Sec. 5. Finally, Sec. 6 concludes the paper with a discussion of open issues and relevant research directions for improving the *AiroLAB* framework.

2 Challenges

In this section we first provide a list of requirements any network virtualization environment shall satisfy regardless the technological domain it is applied to. Then, we analyze how such requirements need to be tailored to account for the peculiar features of wireless multi–hop networks.

2.1 Requirements for a Virtualization Environment

An effective network virtualization shall satisfy the following general requirements [3]:

– *Scalability*: depending on the network segment and technological domain under consideration, the performance offered shall not depend on the number of slices active in the system.
– *Isolation*: a full isolation among slices should be guaranteed in order to improve tolerance to faults, security and privacy in the virtualized infrastructure. In particular, potential misconfiguration in one network slice should not affect or destabilize the other ones.
– *Legacy support*: backward compatibility should be guaranteed in network virtualization environments by –at least– hosting a legacy implementation of a network instance (e.g. production network) in a slice.
– *Flexibility*: full programmability of the network elements must be ensured especially in environments where clean–slate approaches are to be investigated.
– *Manageability*: network management tasks on a single network slice should be performed without the need for coordination across administrative boundaries.
– *Heterogeneity*: heterogeneity should be supported at different levels: underlying networking technologies; end–to–end virtual networks spanning heterogeneous combinations of (sub)networks; end-user devices.
– *Efficiency*: the additional overhead introduced by the virtualization framework shall be minimal.

Two further requirements should be added to this list in the case network virtualization techniques are used for running concurrent research experiments [11]:

– *Inter–experiments interference*: network virtualization introduces some form of compromised performance for co–existing experiments. Any performance degradation associated with experiments should be carefully quantified when mapping virtualization to scientific experiments.

– *Experiments repeatability*: repeatability of results must be assured through proper resource sharing techniques to avoid unpredictable performance across multiple experiment runs.

Compared to existing approaches, these last two requirements are less relevant for our framework. Our aim is in fact to provide an environment for production networks, where operational traffic is fully guaranteed, while experimental services and protocols run on lower–priority slices. This scenario is a sort of intermediate one between (i) a purely research approach, i.e. ORBIT [16] or GENI [5], where novel services or protocols are executed on shared but dedicated testbed infrastructures, and (ii) 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 this scenario, AiroLAB is a NV framework whose objectives are somehow similar to the ones pursued by solutions such as Cabernet [17, 18] and generalized in [19]; however, given the specific technological domain under consideration, we are providing an emphasis toward the possibility for a WISP to perform experimental activities in a controlled fashion directly on its production network, fostering the deployment of novel solutions and services. In such a scenario, experimental traffic is expected to be given a best effort treatment, thus delivering the guaranteed service to the WISP customers. As a result, the share of bandwidth made available for experimental services might change dynamically and thus negatively affect the repeatability of experiments. It is the authors' standpoint that, in a production environment, repeatability shall be traded for performance isolation and scalability in order to give paying services preemptive access to the network resources.

2.2 Problem Definition and Challenges

Network virtualization in wireless networks needs to address two additional major issues: (i) how to isolate wireless resources belonging to network slices coexisting at the same time to ensure minimal interference among them, and (ii) how to control wireless resource utilization to ensure that a slice does not infringe the resources of another slice. Several techniques have been proposed to guarantee the isolation of wireless resources among concurrent slices[14]:

– *SDM (Space Division Multiplexing)*, where physical wireless nodes are partitioned in space, forming separate sub–networks, thereby minimizing the interference among different slices.
– *FDM (Frequency Division Multiplexing)*, where different slices are partitioned in the frequency domain by leveraging on the availability of multiple wireless interfaces on each network node.
– *CDM (Code Division Multiplexing)*, similar to FDM, but assigning different codes to each slice.
– *TDM (Time Division Multiplexing)*, whereby slices are partitioned in the time domain by assigning them a specific timeslot for their communication needs.

While studies regarding the feasibility of each of these approaches (or combinations thereof), with their pros and cons have been already provided in literature [14, 20, 15], we are interested here in investigating techniques and architectures to allow an effective isolation between concurrent slices on a multi–hop wireless network through a finer control of wireless and node resources usage in the network.

In particular, *AiroLAB* aims at achieving a level of flexibility which none of the aforementioned techniques, used in a stand–alone fashion, could provide. Further, we target the provisioning of methods for ensuring that a privileged slice (typically the one carrying the production traffic) can have guaranteed resources while the ones devoted to experimental activities may share the remaining (possibly time–varying) network resources.

3 *AiroLAB* Architecture and Design

In this section, we introduce the design of *AiroLAB*, a novel virtualization framework specifically tailored to multi–hop wireless networks. Such networks are usually built using commodity components and are characterized by rather limited computing capabilities, in comparison to the traditional carrier–class networking equipment exploited in projects such as FEDERICA [7], AKARI [8] or GENI [5]. As a matter of fact, we argue that a virtualization environment suitable for production–level wireless networks must satisfy the following requirements (in decreasing order of importance): efficiency, flexibility, and isolation, while constraints on heterogeneity and scalability can be relaxed.

In this section we will first illustrate the objectives and constraints that have driven the *AiroLAB*'s design and then we will describe in details its architecture. Integration with currently available frameworks for automatic NV resources allocations, such as OMF [21], will be considered for future evolution of the AiroLAB framework.

3.1 Baseline scenario

Most of the network virtualization architectures devised so far [7, 8, 5] aim at providing multiple isolated environments where experiments can be run in parallel over real–world networks. *AiroLAB*, on the other hand, aims at providing wireless networks operators with a comprehensive virtualization solution where production traffic (i.e. the traffic generated by the end–users), shares part of the available network resources with a variable number of experimental slices where novel solutions, e.g. routing protocols, are being tested.

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 the next sections we will

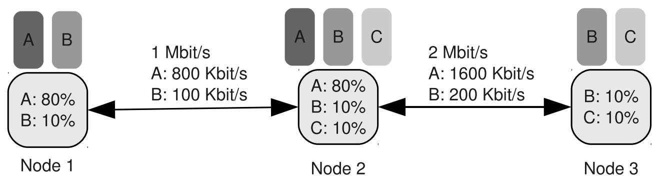

Fig. 1: Simplified deployment scenario.

generalize *AiroLAB*'s design to asymmetric links with fluctuating capacity in multi–radio/multi–channel setups.

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. As a matter of fact, to the best of the authors knowledge, the most powerful embedded wireless router processing board currently available on the market, the Gateworks Cambria GW2358-4, is equipped a 667MHz ARM CPU and 128MB of RAM.

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.

3.2 Link Capacity Estimation

Due to the use of a shared medium, estimating the capacity of a wireless link is not trivial. 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 Stations 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 *mac80211* [22] an implementation of the IEEE 802.11 stack for the GNU/Linux operating system. More specifically, mac80211 is a framework that GNU/Linux developers can use to write drivers for IEEE 802.11 wireless devices. At the time of this writing, the *mac80211* framework supports two different rate–control algorithms: PID and minstrel. In this work we focus on the latter 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, the empirical throughput T_{ab}, is computed as follows:

$$T_{ab} = \frac{P_{ab}B}{D_{tx}}, \tag{1}$$

where D_{tx} is the time spent for a single transmission, and B is the packet length. *AiroLAB* uses T_{ab} as an estimation of the wireless link capacity.

3.3 Providing Soft–Performance Isolation

Soft–performance isolation between slivers is provided using the Hierarchical Token Bucket (HTB) techniques supported by the Linux kernels 2.6.x. HTB allows network administrators to implement precise traffic shaping policies. Before describing the *AiroLAB* approach for providing performance isolation, this section briefly summarizes the main features of HTB. 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 rate constraint but the ceil one is propagated upwards until the procedure is completed.

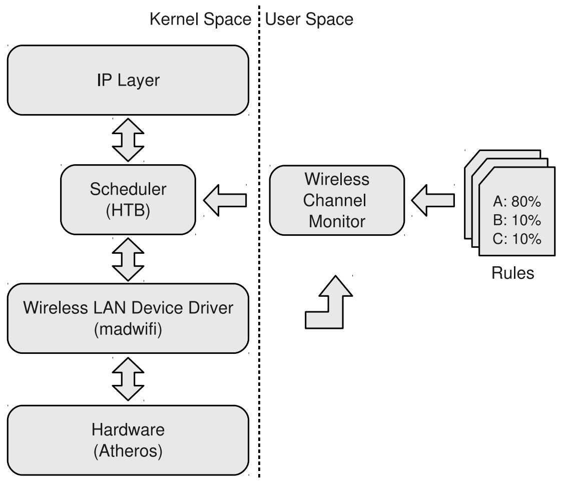

Fig. 2: *AiroLAB* wireless channel estimator architecture.

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 issue 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 some input *policies*. Through such *policies* it is possible to assign each sliver with a minimum, maximum and/or an average bandwidth. An additional relative priority can be specified for slivers that share the same traffic class, if no relative priority is specified the bandwidth available for that traffic class is equally shared between the competing slivers. 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.

3.4 Node–Level Architecture

In this section we shall describe the *AiroLAB* node's architecture (see Fig. 3) in details. *AiroLAB* is built around OpenVZ [23], an open source virtualization solution for the GNU/Linux operating system, and Click, an open source modular router currently available on GNU/Linux, all the children of BSD, and MAC OSX.

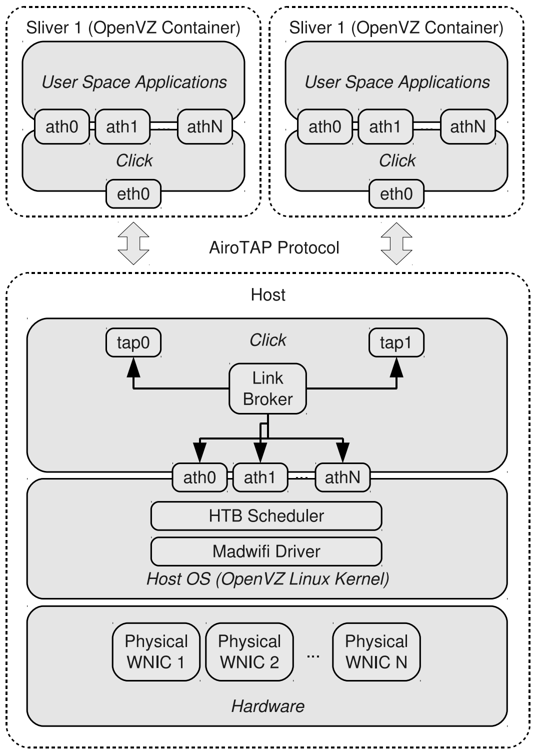

Fig. 3: *AiroLAB* node–level architecture.

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 alterna-

Table 1: Taxonomy of network virtualization techniques and relevant features [26].

	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.*

tive container-based solutions such as Linux–VServer [24], 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 own wireless network virtualization stack in user–space using the Click modular router [25]. It is worth noting that, our approach is not meant to replace OpenVZ, but rather to extend it in order to support flexible virtualization of the wireless resources. A Click router is built by assembling several packet processing modules, called *Elements*, forming a directed graph. Each element is in charge of a specific function such as packet classification, queuing, and interfacing with networking devices. Click comes with an extensive library of elements supporting various types of packet manipulations. Such a library enables easy router configuration by simply choosing the elements used and the connections among them. Finally, a router configuration can be easily extended by writing new elements. Albeit characterized by a 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 [26].

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 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 [27] 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 simulate different operating conditions (i.e. link failures/outages).

3.5 Network–Level Configuration: an example

Figure 4 sketches a possible use case, where a production slice exploiting a stable version of a routing protocol is running in parallel with 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 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 Experimental set up and results

This section provides (i) an overview of the hardware and software setup used in the experimental activities, (ii) a description of the experimental scenarios

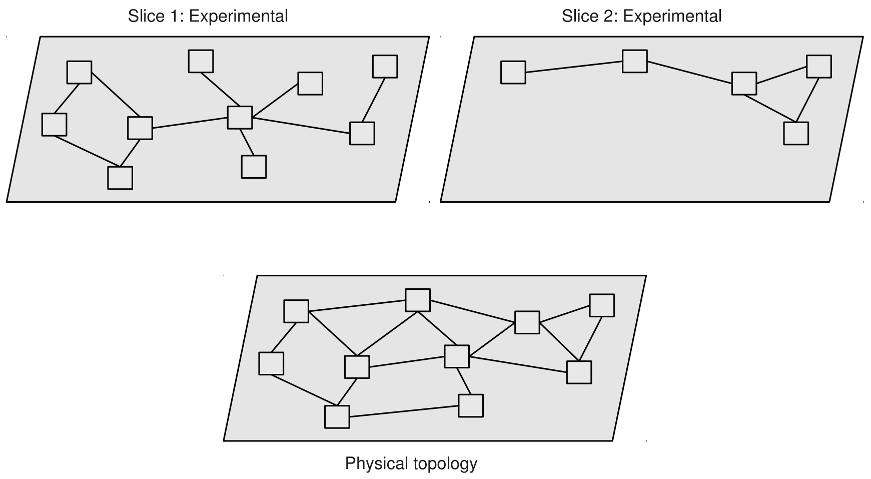

Fig. 4: Network–level configuration: an example with one production slice and one experimental slice sharing a common physical substrate.

where the tests were conducted, and (iii) a report and discussion of the tests outcomes.

4.1 Hardware/Software Platform Setup

The hardware used in our experimental activity consists of ALIX nodes provided by PC Engines [28]. Wireless routers 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 two Ethernet channels, two 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. OpenWRT is a minimalist BusyBox/Linux distribution released under a GPL license. It provides an automated system for downloading the source code for both the kernel and the userspace tools, and compiling it to work on any supported platform. Moreover, it is characterized by a small memory and disk footprint making it suitable for a wide rage of networking devices. Finally, it provides hardware configuration and maintenance abstraction through a custom system and package configuration facility called UCI (Universal Configuration Interface) and exploiting MIB-like structure in order to streamline device management using SNMP [30]. Customizations to the standard OpenWRT distribution include several scripts and tools necessary to manage the virtual environments. Moreover, the original OpenWRT kernel has been with a kernel provided by OpenVZ which was configured and recompiled to be installed on our nodes. The software configuration of the wireless routers is summarized in Table 2.

Table 2: AiroLAB Software 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

4.2 Experimental Settings

In order to assess the effectiveness of the *AiroLAB* framework in preventing traffic on a privileged slice being affected by traffic from other (lower–priority) slices, the scenario described in 3.1 has been set up. Evaluation of multi–radio/multi–channel setting with asymmetric link is left as future work. Moreover, comparison with a purely Space Division Multiplexing (SDM) approach is out of the scope of this work in that (i), such a study has already been carried out in [15, 20], and (ii) the particular deployment scenario envisioned for *AiroLAB*, namely virtualization of production–quality networks, would require a experimental staging network which would undermine the proposed system's main goal of sharing a single infrastructure for both paying services and experimentation activities.

The network deployment is sketched in Fig. 5 and consists of two wireless nodes, each one running two or more slivers (or virtual nodes) 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. Through such a test, we want to test the ability of the proposed architecture to effectively preserve production traffic in challenging conditions.

As described in Sec. 3, we used the HTB packet queuing discipline to control the outbound bandwidth of a given link. Our HTB configuration defines a traffic class for each sliver currently active within a given node. For each class, the minimum/guaranteed (*rate*) and the maximum (*ceil*) throughput are specified. Afterward, the HTB configuration defines which packets belong to which class depending on the source IP address. In this way, it is possible to limit the outbound bandwidth on a per–slice basis. Figure 6 shows the node setup for the case of two slivers: on node A, HTB ensures that the outbound bandwidth to carry the UDP stream is divided between the two slivers as configured.

4.3 Experimental Outcomes and Analysis

As *AiroLAB* is concerned with the provisioning of isolation and guaranteed performance to a privileged slice, tests were focused on the ability to ensure isolation in terms of guaranteed throughput over the shared wireless medium. Results have been obtained by averaging the samples obtained as *nuttcp* benchmarks over 300 seconds with an averaging interval of 10 seconds. It is worth stressing that, *AiroLAB* aims at providing a virtualization architecture where experimental services and protocols can be tested and evaluated over a production network

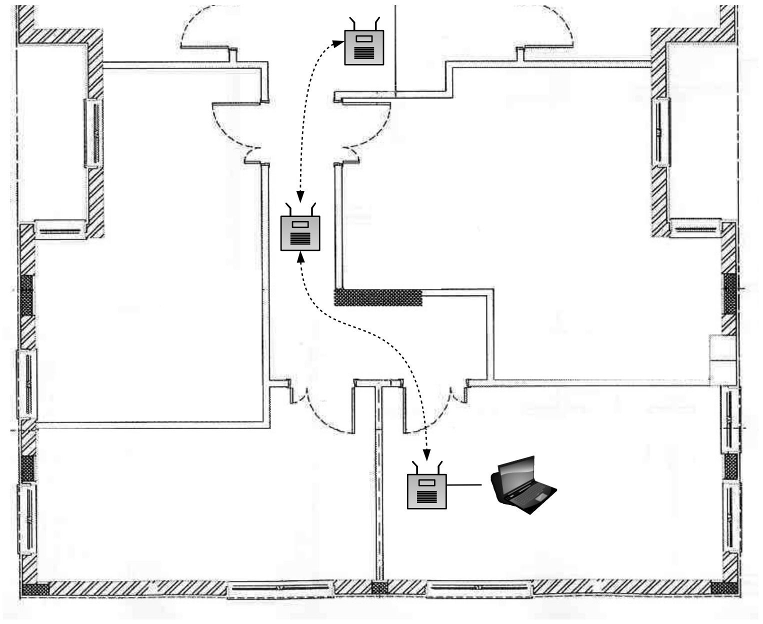

Fig. 5: The testing setup involved 2 nodes deployed in a typical office environment. One of the nodes is powered by means of a a rechargeable battery and is moved around in order to assess the capability of the wireless channel monitor to adapt to changing operating conditions.

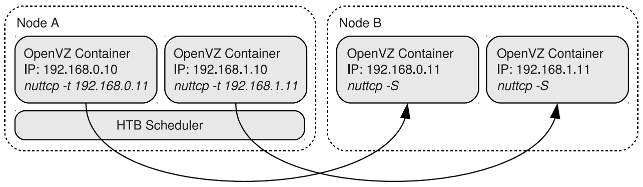

Fig. 6: Representation of the packet scheduling process for the case with two slivers.

with minimal impact on the operational traffic. In such a scenario experimental repeatability is traded for performance isolation between a single privileged production slice and several concurrent experimental and/or monitoring slices.

In the first scenario, three slices are running simultaneously. The privileged slice (#1) has higher transmission priority and a minimum guaranteed outbound bandwidth set to 5 Mb/s, while the remaining two slices (#2 and #3) have a minimum guaranteed outbound bandwidth of 128 Kb/s. Moreover, slice #1 and #2 have no upper bounds on the maximum throughput they can inject in the wireless link, while slice #3 has a 5 Mb/s limit. The graph plotted in Fig. 7 shows the throughput distribution per slice in different conditions of available wireless link capacity. As expected, *AiroLAB* guarantees that, even when link conditions are worsening, at least 5 Mb/s are allocated to slice #1 while the remaining link

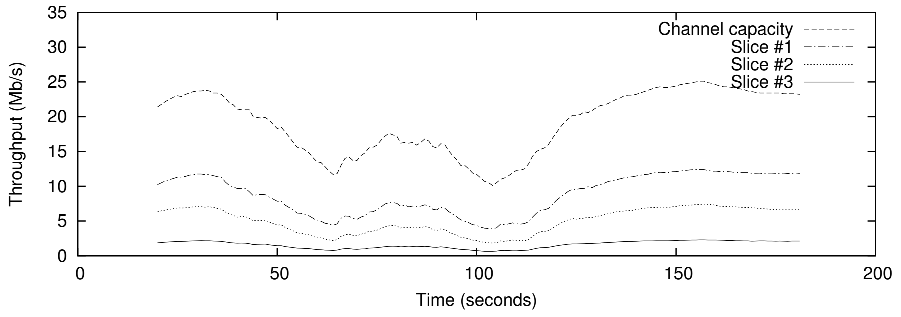

Fig. 7: Average throughput for the three slices in the first scenario. The privileged slice #1 has a minimum guaranteed bandwidth set to 5 Mb/s. Slices #2 and #3 have a minimum guaranteed bandwidth of 128 Kb/s. Slice #3 has an upper bound to the bandwidth set to 5 Mb/s.

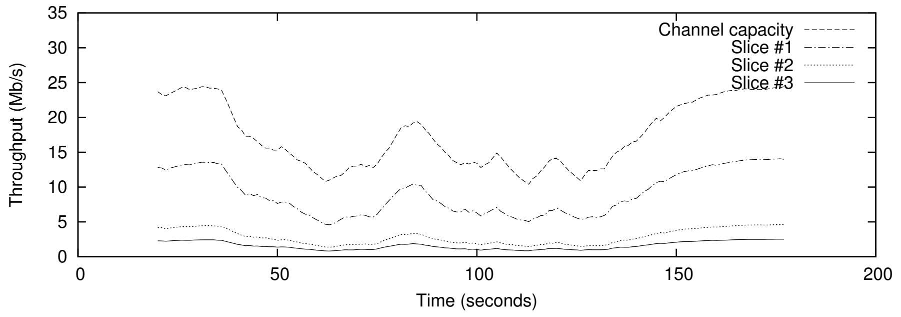

Fig. 8: Average throughput for the three slices in the second scenario. The privileged slice #1 has a minimum guaranteed bandwidth set to 5 Mb/s. Slices #2 has an upper bound to the bandwidth set to 5 Mb/s.

capacity is left available for use in a best effort fashion. It is worth noting that if the link capacity is lower than 5 Mb/s only slice #1 will be allowed to transmit.

In the second scenario, the privileged slice has 5 Mb/s of guaranteed outbound bandwidth, while Slice #2 and #3's outbound bandwidth is limited to 5 Mb/s. The results, reported in Fig. 8, show that the upper bound limit imposed to both slice #2 and #3 lets the maximum throughput experienced by slice #1 be higher than in the previous scenario. It is worth noting that, the throughput of slice #1 can go below 5 Mb/s if the radio conditions become bad, however this is an expected behavior given the volatile nature of the communication medium.

The third scenario presents two slices only, where one slice requires a limited but stable available bandwidth. The privileged slice (#1) has higher priority and a minimum guaranteed outbound bandwidth set to 2 Mb/s and an upper bound of 3 Mb/s, while slice #2 has a minimum guaranteed outbound bandwidth set to 128 Kb/s. The results, reported in Fig. 9, show that also in the worse

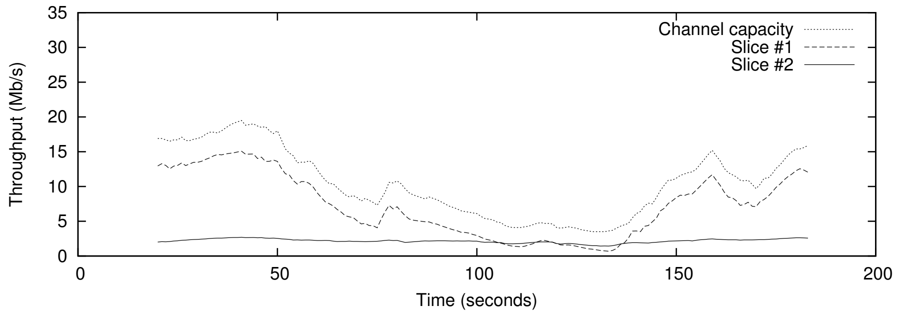

Fig. 9: Average throughput for the two slices in the third scenario. The privileged slice (#1) has a minimum guaranteed outbound bandwidth set to 2 Mb/s and an upper bound of 3 Mb/s. #2 has a minimum guaranteed outbound bandwidth set to 128 Kb/s.

Table 3: Overhead introduced by the AiroLAB wireless channel monitor.

	Scenario #1	**Scenario #2**	**Scenario #3**
Average	0,22	0,24	0,12
Minimum	0,11	0,12	0
Maximum	0,40	0,38	0,41
Std Deviation	0,09	0,07	0,08
Confidence interval (95%)	±0,01	±0,01	±0,01

working condition, *AiroLAB* guarantees the outbound throughput of slice #1 by adaptively tuning the bandwidth provided to slice #2.

We further evaluated the efficiency of the proposed framework, in terms of overhead introduced by *AiroLAB*. We considered the following performance metric, which represents the share of the channel capacity that remains unused:

$$\eta = \frac{C - \sum\limits_{i} \theta_i}{C}, \tag{2}$$

where C is the total capacity available and θ_i is the throughput of slice #i. The results, averaged over each single test, are reported in Table 3.

5 Related Work

Other groups have proposed mechanisms to obtain virtualization in wireless networks. All of them have been focusing mainly on using network virtualization to run large–scale testbeds where several experiments could run concurrently on the same physical infrastructure. Very few have been considering how to leverage these techniques in real production networks. 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 strategies. This has been obtained by exploring both dimensions of (i) virtualization of the wireless medium and (ii) virtualization of the network node.

In [14], the authors have presented their experiences in the design and implementation of a TDM-based wireless virtualization on a large-scale indoor 802.11 wireless testbed facility (ORBIT). Two major challenges have been identified:

– node synchronization, where NTP mechanisms do not provide an adequate level of precision to allow all network nodes to perform tasks at specific times;
– device state, due to the limited access to the state stored on the wireless devices.

Even though more expensive mechanisms to guarantee node synchronization could be potentially used (based, e.g., on the use of GPS), the main outcome of this paper is that TDM presents limits in the virtualization of the wireless medium.

The work presented in [15] compares two wireless virtualization strategies: SDM and Virtual Access Points. VAP is a logical abstraction running on a physical AP while emulating the behavior of a conventional AP to all network stations. Compared to a TDM mechanism, VAP does not require strict synchronization among different nodes. However, this scheme is limited to fixed star topologies and it needs traffic shaping mechanism to ensure fair access to wireless resources among experiments. The paper reveals benefits and weaknesses for both space (SDM) and time separation (VAP) schemes. Furthermore, it provides a policy manager for controlling inter-experiment interference by assigning and enforcing bandwidth assigned to each slice in VAP-based testbeds. However, the main limitation of the proposed approach is that incorporating arbitrary topologies in a slice or across VAPs allocated to the experiment is not feasible.

A detailed analysis of the performance of an FDM based mechanism for virtualizing wireless networks is presented in [20]. Virtualization is obtained here by using multiple wireless interfaces per node and by assigning different channel frequencies to each interface. Then, each interface is assigned to a virtual device running inside the wireless node. A performance comparison between a virtualized radio node and a non–virtualized one is presented in terms of throughput, delay, and jitter. Cross–coupling effects between two experiments have been investigated by studying the transient behavior associated with sudden changes in traffic on one of the slices. While showing the advantage of FDM as robust wireless virtualization strategy (and feasible, due to the widespread availability of multi-radio devices), this paper highlights the limitations of a User-Mode Linux (UML) approach to node virtualization, especially in specific experimental conditions (e.g. UDP experiments using small packets or when working in saturation conditions).

In [31] the authors compared the performance of two different node virtualization approach: UML and OpenVZ, while considering FDM as the option for the virtualization of the wireless medium. OpenVZ shows more stable behavior than UML with UDP experiments using small packets as well as when working in link saturation; it also provides excellent isolation in terms of the observed transient response of the experiments.

While OpenFlow has some similarities with *AiroLAB*'s objectives, especially in term of sharing a production network with experimental traffic [32], this initiative does not focus on a virtualization mechanism, but instead leverages a novel architecture to control the traffic passing through switches and APs by identifying flows and policies for handling them.

6 Conclusions and Discussion

In this paper, we have presented *AiroLAB*, a wireless network virtualization solution aimed at providing suitable means to support experimental testing of novel networking solutions on production networks, while preserving guarantees performance for production traffic. The design choices at the basis of *AiroLAB* have been presented and discussed. A prototype implementation has been presented, and outcomes of experimental activities, performed on a small-scale testbed, reported and discussed.

While the results presented appear encouraging, the framework needs to be further developed before reaching the stability level needed to support its wide deployment. One research direction that appear of interest for enhancing the current architecture (mainly in terms of efficiency and scalability) is to use an FDM approach, whereby different slices are associated with a dedicated wireless interface in a multi–radio deployment. Finally, we intend to integrate *AiroLAB*'s wireless networks virtualization capabilities within currently available frameworks for automatic NV resources allocations, such as OMF [21].

References

1. A. Feldmann, M. Kind, O. Maennel, G. Schaffrath, and C. Werle, "Network Virtualization: An Enabler for Overcoming Ossification," *ERCIM News*, vol. 77, pp. 21–22, April 2009.
2. P. Papadimitriou, O. Maennel, A. Greenhalgh, A. Feldmann, and L. Mathy, "Implementing Network Virtualization for a Future Internet," in *Proc. of 20th ITC Specialist Seminar on Network Virtualization*, Hoi An, Vietnam, 2009.
3. N. M. K. Chowdhury and R. Boutaba, "Network Virtualization: State of the Art and Research Challenges," *IEEE Communications Magazine*, July 2009.
4. "Technical Document on Overview Wireless, Mobile and Sensor Networks," The GENI Project Office, Tech. Rep. GDD-06-14, 2006.
5. GENI project, http://www.geni.net.
6. 4WARD project, http://www.4ward-project.eu.
7. FEDERICA project, http://www.fp7-federica.eu.
8. AKARI project, http://akari-project.nict.go.jp.
9. R. Bruno, M. Conti, and E. Gregori, "Mesh Networks: Commodity Multihop Ad Hoc Networks," *IEEE Communications Magazine*, vol. 43, no. 3, pp. 123 – 131, Mar. 2005.

10. I. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Elsevier Computer Networks*, vol. 47, no. 4, pp. 445 – 487, Mar. 2005.
11. Planet Lab project, http://www.planet-lab.org.
12. VINI project, http://www.vini-veritas.net.
13. German-Lab project, http://www.german-lab.de/.
14. G. Smith, A. Chaturvedi, A. Mishra, and S. Banerjee, "Wireless Virtualization on Commodity 802.11 Hardware," in *Proc. of ACM WinTECH*, Montreal, Quebec, Canada, 2007.
15. R. Mahindra, G. Bhanage, G. Hadjichristo, I. Seskar, D. Raychaudhuri, and Y. Zhang, "Space Versus Time Separation for wireless virtualization On an Indoor Grid," in *Proc. of EURO NGI*, Krakow, Poland, 2008.
16. Orbit Lab, http://www.orbit-lab.org/.
17. N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," *ACM SIGCOMM Computer Communications Review*, pp. 61–64, January 2007.
18. Y. Zhu, R. Zhang-Shen, S. Rangarajan, and J. Rexford, "Cabernet: Connectivity architecture for better network services," in *Proc. of Workshop on Rearchitecting the Internet*, 2008.
19. G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy, "Network Virtualization Architecture: Proposal and Initial Prototype," in *Proc. of ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures*, Madrid, Spain, 2009.
20. S. Singhal, G. Hadjichristo, I. Seskar, and D. Raychaudhuri, "Evaluation of UML based wireless network virtualization," in *Proc. of EURO NGI*, Krakow, Poland, 2008.
21. OMF, cOntrol and Management Framework, http://omf.mytestbed.net.
22. Linux Wireless, http://linuxwireless.org/.
23. OpenVZ, http://openvz.org/.
24. Linux-VServer, http://Linux-VServer.org/.
25. E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Transaction on Computer System*, vol. 18, no. 3, pp. 263 – 297, Aug. 2000.
26. A. Nakao, R. Ozaki, and Y. Nishida, "Corelab: An emerging network testbed employing hosted virtual machine monitor," in *Proc. of ACM ROADS*, Madrid, Spain, 2008.
27. Linux Radiotap, http://www.radiotap.org/.
28. "PC Engines." [Online]. Available: http://www.pcengines.ch/
29. "OpenWRT Linux Distribution." [Online]. Available: http://openwrt.org/
30. J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol," IETF RFC 1157, May 1990, http://www.ietf.org/rfc/rfc1157.txt.
31. G. Bhanage, I. Seskar, Y. Zhang, and D. Raychaudhuri, "Evaluation of OpenVZ for wireless testbed virtualization," WINLAB Rutgers University, Tech. Rep. 331, 2008.
32. OpenFlow project, http://www.openflowswitch.org.