computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

VIRTUAL WLAN: EXTENSION OF WIRELESS NETWORKING INTO VIRTUALIZED ENVIRONMENTS

Ghannam Aljabari [1], Evren Eren [2]

[1] Palestine Polytechnic University, Hebron, Palestine, galjabari@ppu.edu
[2] University of Applied Sciences Dortmund, Emil-Figge-Strasse 42,
D-44227 Dortmund, Germany, eren@fh-dortmund.de

Abstract: *In wired Ethernet networks (IEEE 802.3), a physical network interface can be connected to different network segments or shared among multiple virtual machines. In wireless LAN (IEEE 802.11) sharing a wireless network interface is recognized to be a difficult task. However, virtualization can solve this problem. In this paper we will introduce a software platform for hosting multiple virtual wireless networks over a shared physical infrastructure by means of open source virtualization techniques. We present the design, implementation, and performance testing of this platform. Results have shown that the hosting platform can extend wireless networking into virtualized environments without compromising the performance, isolation, or wireless LAN security mechanisms.*

Keywords: *Virtualization; Wireless LAN; Virtual network; Hypervisor; KVM.*

1. INTRODUCTION

Virtualization technology has been widely adopted in data centers to optimize resource sharing and utilization. This technology has helped to consolidate and standardize hardware and software platforms in data centers, i.e. servers and storage. The main benefit of virtualization technologies is savings in power and infrastructure costs in addition to improving availability, scalability, and security.

In recent years, virtualization has been pushed forward to also virtualize physical network infrastructures. By allowing multiple logical networks to co-exist on a shared physical infrastructure, network virtualization provides flexibility and manageability. Network virtualization often combines hardware and software resources to deploy virtual networks for different architectures. The term virtual network has been used to describe different types of network virtualization such as VLAN (Virtual Local Area Network) and VPN (Virtual Private Network). But recently, network virtualization is moving toward *virtualized environments*.

Virtualization of wireless LANs (WLANs) has become one of the important issues in network virtualization and also for cloud computing by now. It is useful in many scenarios such as hosting multiple wireless service providers on a single shared physical infrastructure, providing wireless services with different authentication mechanisms, and for virtual test bed environments. Hence, there are some research activities in this field [1-4].

There are several approaches to system virtualization and several software implementations, both open source and commercial. However, most of the virtualization approaches are mainly developed for wired Ethernet networks, and are not suitable for virtualizing wireless LAN interface due to the nature of wireless LAN devices. More specifically, the limitations of current virtualization approaches are due the difficulties in emulating wireless LAN management functions [3]. Therefore, existing virtualization approaches require a separate physical wireless LAN device for each virtual machine (VM) to have its own wireless network.

A viable solution to address the above issue is by giving all VMs access to the same wireless network and rely on network virtualization techniques such as VLAN or VPN to provide isolation for VM network traffic. However, this solution will add additional cost and overhead for configuring and maintaining a secured connection to all VMs. As a result, a new approach is needed to enable a single wireless network interface to be shared among several VMs without compromising the performance, isolation, or wireless LAN security mechanisms.

By means of open source virtualization techniques, it is possible to create multiple virtual wireless networks through one physical wireless LAN interface, so that each virtual machine has its

own wireless network. Available open source solutions such as KVM, hostapd, and VDE provide the software infrastructure to deploy and implement such an approach on Linux operating system (OS). This paper aims at demonstrating this approach.

2. BACKGROUND

Virtualization approaches enable running multiple OSs and applications concurrently on the same physical machine, eliminating the need for multiple physical machines. Each VM has its own operating system and applications such as the physical machine [5-7]. Thus making the applications unaware of the underlying hardware, yet viewing computing resources as shared resource pools available via virtualization.

The primary benefits offered by virtualization are *resource sharing* and *isolation*. Unlike real environments where physical resources are dedicated to a single machine, virtual environments share physical resources such as CPU, memory, disk space, and I/O devices of the host machine with several VMs. With isolation, applications running on one VM cannot see, access, and use resources on other VMs [5].

Virtualization provides a software abstraction layer on top of hardware. This layer is called *Virtual Machine Monitor* (VMM), also known as a *hypervisor*. The main task of the VMM is to manage the hardware resource allocation for VMs and to provide interfaces for additional administration and monitoring tools [5]. However, the functionality of the VMM varies greatly based on architecture and implementation.

Today, two alternative approaches exist to virtualization on x86 hardware architecture. In the so called *full virtualization* approach, VMs and guest OSs run on top of virtual hardware provided by the VMM. However, the VMM has to provide the VM with an image of an entire system, including virtual BIOS, virtual CPU, virtual memory, and virtual devices to allow the guest OS to run without modification. As a result, the guest OS or application is not aware of the virtual environment. The main advantage of full virtualization approach is that it supports any platform and provides complete isolation of different applications, which helps make this approach highly secure. However, this approach has poor performance in trying to emulate a complete set of hardware in software [5,7].

KVM, which stands for *Kernel-based Virtual Machine*, is a full virtualization solution that takes advantage of *hardware-assist* features such as Intel VT and AMD-V to improve the performance of guest OSs [8]. The first generation of hardware assist features was added to processors in 2006, so that KVM hypervisor supports only newer x86 hardware systems. Using KVM, several fully VMs can be created and operated in Linux environments, since KVM adds VMM capabilities to the Linux kernel. KVM hypervisor consists of two main components: a set of kernel modules providing the core virtualization infrastructure such as CPU and memory management, and a user space program that provides emulation for I/O hardware devices, currently through QEMU [9].

OS assisted virtualization or *paravirtualization* presents each VM with an abstraction of the hardware that is similar but not identical to the underlying physical hardware. This approach requires modifications to the guest OSs that are running in the VMs. As a result, guest OSs are aware that they are executing on a VM, allowing for near-native performance [5].

Xen is an open source virtualization software based on the paravirtualization approach. The Xen hypervisor runs directly on hardware, allowing the host machine to run multiple modified guest OSs concurrently [6]. Modifying the guest OS is not feasible for non-open source platforms such as Microsoft Windows. As a result, such OSs are not supported in a paravirtualization environment. Recently, unmodified guest OSs are also supported by Xen. In this mode, Xen provides a fully abstracted VM with hardware support (Intel VT and AMD-V) referred to as *hardware virtual machine* (HVM) [10].

With the adoption of virtualization in data centers, a new layer of network virtualization is emerging that provides inter- and intra- VM connectivity and has many of the same functions provided by the physical networking hardware. Today, this layer is providing connectivity to tens of VMs for a physical machine [11].

The main network components provided by virtual networking, as shown in Fig. 1, are *virtual Ethernet interfaces*, used by individual VMs, and *virtual switches*, which connect the VMs to each other [12]. VMs can also be configured with one or more virtual Ethernet interface to offer different virtual network appliances for virtual environments such as *virtual routers* (VR) and *virtual firewalls.* VRs are essential components in the virtual networking infrastructure because they operate in much the same way as physical routers, forwarding and routing packets based on standard routing protocols such as RIP and OSPF. Virtual firewalls provide the usual packet filtering and monitoring role provided via a physical network firewall. Thus, virtual networking components manage communication between co-located VMs, and connectivity to physical machines.

Modern OSs provides the ability to create virtual network interfaces that are supported entirely in software. From the OS's point of view, these interfaces behave similar to physical network interfaces. However, the virtual interface does not send the packets into the wire, but makes them available to userspace programs running on the system. Virtual network interfaces are commonly referred to as TAP and TUN interfaces under Linux. TAP interfaces operate with Layer 2 packets, while TUN interfaces can handle Layer 3 packets. VMs use the TAP interface to create a network bridge with the physical network interface [2].

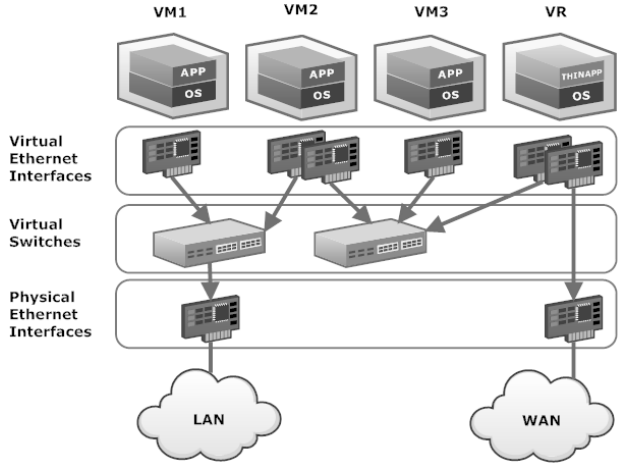

Fig. 1 – Virtual networking components

Most of the virtualization approaches also provide some form of virtual networking. For example, VMware virtualization software has a distributed switch for virtual machine networking [13]. Linux-based virtualization platforms, including Xen and KVM, generally use network bridging or *Virtual Distributed Ethernet* (VDE) switch [14]. A network bridge acts like an Ethernet hub; passing all traffic. While, VDE provides Layer 2 switching, including spanning-tree protocol and VLAN support.

Open vSwitch is an open source software switch that provides connectivity between the VMs and the physical interfaces. It implements standard Layer 2 and Layer 3 switching with advanced features such as traffic monitoring (e.g. NetFlow), port mirroring (e.g. SPAN), basic ACL (Access Control List) and QoS (Quality of Service) policies. The Open vSwitch consists of two components: a fast kernel module and lightweight userspace program. The kernel module implements the forwarding engine, while the userspace program implements forwarding logic and configuration interfaces. Open vSwitch supports multiple Linux-based virtualization software, including Xen and KVM [11,15].

Quagga is an open source routing software that provides implementations of TCP/IP based routing protocols such as OSPF, RIP, and BGP. In addition to traditional IPv4 routing protocols, Quagga also supports IPv6 routing protocols [16]. Vyatta software [17] incorporates open source routing and security projects such as Quagga, IPtables, OpenVPN and many others into a network OS for x86 hardware platforms. Vyatta also can be delivered as VMs, providing routing, firewalling, VPN, and more for virtual and cloud computing environments. Thus, Vyatta network OS complements virtual networking components by delivering the virtual router, virtual firewall, and virtual VPN in the hypervisor.

3. VIRTUALIZATION OF WLAN INTERFACE

A network interface can be shared and hence virtualized using either a *software* or *hardware based approach*, as shown in Fig. 2. In software-based approach, network interface virtualization is completely implemented as software to provide virtual network interfaces (VIF) for multiple VMs [3,18,19]. In this approach, *bridging* functionality is often enabled on the physical network interface to grant all VMs access to the same physical network.

Full virtualization techniques provide virtual network interfaces by emulating legacy Ethernet devices for simplicity. The virtual network interfaces appear to the VM as virtualized hardware devices within the hypervisor. With this technique, no modification is required for the guest OS. However, there is a significant performance overhead due to the context switching between VM and hypervisor. In the paravirtualization technique, the paravirtualized driver is used in the guest OS to achieve high I/O performance. However, this method requires modifying the guest OS and having a special driver to expose some details of the hardware [3].

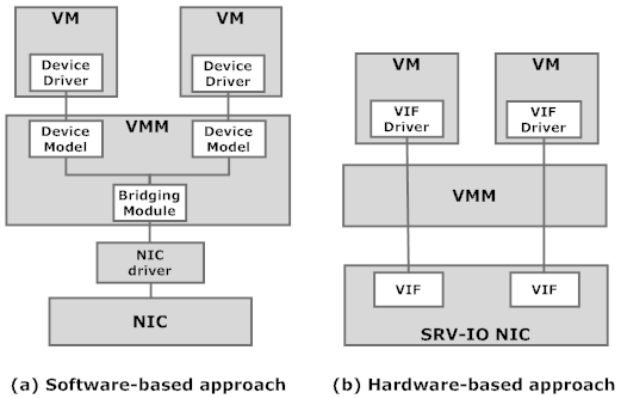

Fig. 2 – Network interface virtualization approaches

The second approach depends on hardware virtualization support to partition a physical network device to multiple virtual network interfaces. Then, each virtual interface can be assigned directly to a specific VM. While this approach reduces the

performance overhead of software-based network interface virtualization, it increases the complexity, maintainability and cost of network devices [3,18,19]. An example of hardware-based approach is *Single Root I/O Virtualization* (SR-IOV) where a single PCI device can be divided into multiple Virtual Functions (VFs). Each VF can then be used by a VM, allowing one physical device to be shared among multiple VMs. As a result, close to native I/O performance can be achieved, in addition to fair sharing of the bandwidth [20].

Virtualization of a wireless LAN interface is more complicated than for wired network interface because the capacity of the wireless LAN channel varies with radio signal strength and interference from other wireless LAN devices. This requires including complex management functions into wireless devices to achieve efficient and reliable communication. Examples of such management functions include data rate adaption, power management, and power control. The device driver, which is part of the OS, is also involved in such management functions for control and configuration. In contrast, wired LAN devices are data centric and have very little management functions [3].

A typical WLAN device consists of: RF transceiver, Baseband, and MAC layer. The RF transceiver performs radio signal transmitting and receiving, while the Baseband mainly responsible for digital signal processing. RF transceiver and Baseband are generally referred to as PHY layer. The MAC layer often consists of a hardware controller on the WLAN device and a software driver on the host computer. Most of the wireless LAN functions such as authentication and authorization are performed at MAC layer [3].

In the beginning, the MAC layer was entirely managed by the firmware on the wireless LAN device. This approach is called FullMAC, where full MAC layer functionality is executed by the hardware controller on the wireless device. New implementation of wireless LAN devices is based on SoftMAC approach, where most of the MAC layer functionality is moved to device driver on the host computer, with the firmware providing a set of functional primitives [2]. This approach provides a high degree of software control over the MAC layer functions, while still allowing the PHY layer to define the radio waveform.

MultiNet [21], which was later named VirtualWiFi, proposes a software based approach to virtualize a single wireless interface. Virtualization of wireless LAN interface is implemented with intermediate driver, called MultiNet Protocol Driver, which continuously switches the radio resources across multiple wireless networks. This approach has been adopted in Microsoft Windows 7 to give a user the ability to simultaneously connect to multiple IEEE 802.11 networks with one WiFi card. However, MultiNet approach was not designed to support the VM environment [3].

Recently, a novel virtualization approach on 802.11 MAC layer has emerged in the wireless industry. Multiple virtual wireless LAN interfaces are separated at MAC layer sharing the same PHY layer [3]. As shown in Fig. 3, multiple virtual MAC entities can be active and share a common PHY layer via *Time Division Multiplexing* (TDM) on the same channel. This approach reduces costs, eliminating co-channel interference, and offering smooth roaming as clients move through the WLAN's coverage area. WLAN products that provide support for such an approach include Atheros, Intel, and Marvell.

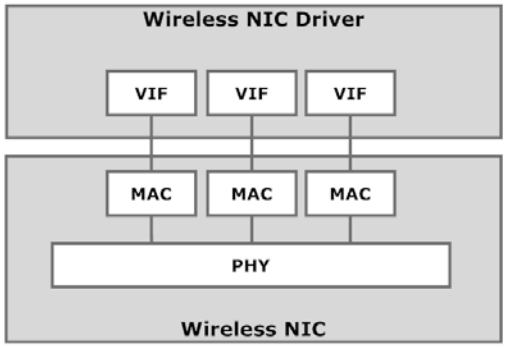

Fig. 3 – Wireless network interface virtualization

In the case that different virtual MACs need to operate on different RF channels, a *time-critical scheduling* is required for multi-channel MAC functions. Implementing such solution will allow the PHY layer to switch between different RF channels and keep virtual MACs in synchronization with the associated networks. Several research efforts have been made in implementing multi-channel virtualization approach for WLAN devices such as Net-X [22] and FreeMAC [23].

Virtualization of the WLAN interface enables several usage scenarios for wireless networking, some of these are:

- **Simultaneous Connectivity:** a wireless device can be connected to multiple wireless networks simultaneously. E.g., One virtual interface operates in STA mode to connect to an AP, while another virtual interface operates in an ad – hoc mode to create a peer-to-peer wireless network.
- **Wireless Relay/Extension:** a wireless client can extend the coverage area of the network by creating a second virtual interface in AP mode, allowing remote clients outside the basic operating range to relay data to the main AP.
- **Soft Handover:** a wireless client can use a second virtual interface to scan all available APs, while the first virtual interface is connected to the

wireless network. After selecting the new AP, a client can authenticate and associate with it without losing the connection with the current AP. In this scenario, we can avoid packet loss and delay times in real-time applications such VoIP and video streaming [2].

- **Multi-Streaming Service:** a mobile device can communicate with multiple APs operating on different channels, as the device has several virtual interfaces. The most stable connection becomes the main connection and others can become sub-connections. By this scenario, we can improve streaming performance such as multi-path streaming without relay server [24].
- **Wireless Mesh Network (WMN):** a multi-hop WMN is built through virtual interfaces created at some mesh nodes. In this case, a mesh node is configured to work in STA mode and acts as AP by creating a second virtual interface in AP mode. Thus, remote clients located outside the coverage range (wireless cell) can get access to the network via clients connected to any AP in the wireless cell [25].
- **Virtualized Environment:** a virtual machine can establish its own wireless LAN connection by creating a virtual interface in STA mode. In this case, multiple wireless connections are supported through one physical wireless LAN network interface.

4. VIRTUAL WLAN APPROACH

With the introduction of IEEE 802.11n and the increase in bandwidth, wireless LAN virtualization is required as an alternative approach for deploying multiple virtual wireless LANs with different authentication methods. Wireless LAN virtualization enables several virtual wireless networks to coexist on a common shared physical device. Multiple virtual interfaces can be created on top of the same radio resources, allowing the same functionality as in multi-radio solution.

All virtual interfaces operate concurrently without considering the physical nature of the wireless medium as well as physical management tasks. Each virtual interface abstracts a single wireless device and has its own wireless network and its own unique MAC address. From the application's perspective, the virtual wireless network behaves like wired Ethernet, but is wireless.

Using wireless LAN virtualization, a virtual interface can be configured to operate as an *access point* (AP) and also as *a station* (STA) device. A virtual AP is bound to a virtual network interface and each virtual AP independently keeps the configuration and service of the wireless network. In this way, several virtual APs can be configured on top of solely one physical wireless LAN device, as shown in Fig. 4.

A virtual AP acts as the master device in a virtual wireless network and operates in much the same way as physical AP, allowing wireless stations to communicate with each other by managing and maintaining a list of associated stations. In general, the virtual AP consists of two parts: *control plane* and *forwarding plane*. The control plane is concerned with the information that defines the functionality of the AP such as the SSID (Service Set Identifier), operation mode, and RF channel. While the forwarding plane defines the part of the AP, that uses a lookup table as a base to forward packets to its destination.

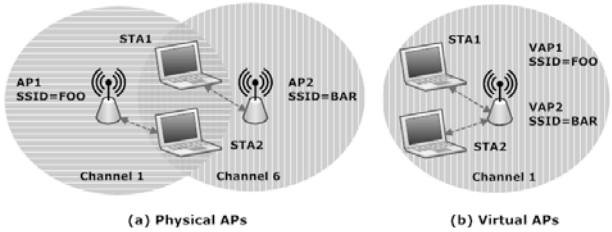

Fig. 4 – Physical and virtual APs

By integrating wireless LAN virtualization techniques into the hypervisor, the wireless LAN interface can be shared among several VMs. To each VM one or more virtual wireless interfaces can be assigned. As shown in Fig. 5, VIFs are configured to operate in one of the wireless operating modes, specifically the AP mode, and then can be assigned to various virtual networking components.

The main goal of this approach is to combine wireless network functionality into a common virtualized environment and to achieve performance levels comparable to the native hardware wireless LAN. A similar approach named virtual WiFi [3] has been taken to provide wireless LAN client functionality inside VMs. However, virtual WiFi approach is intended to support mobile client environments where the VM runs on the client device and has to be aware of the wireless interface to establish its own wireless connection.

The Virtual WLAN approach is suitable for virtualizing wireless LAN infrastructures, where multiple separate wireless LANs can be deployed on a shared physical infrastructures with different security mechanisms such as WPA and IEEE 802.11i. Since each virtual wireless LAN is logically separated, wireless LAN providers may use virtual WLANs to offer multiple services on the same physical infrastructure. Alternatively, virtual WLANs can be shared by multiple providers allowing each provider to offer separate services for their subscribers [1].

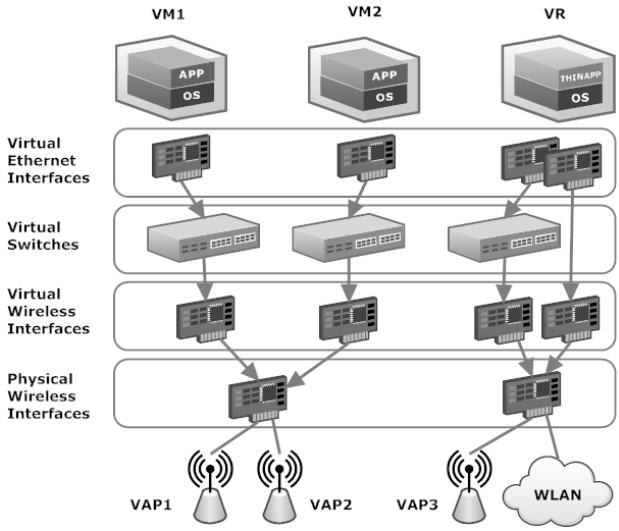

Fig. 5 – Virtual wireless LAN approach

This approach is based on the Atheros WLAN chipset which supports concurrent wireless connections sharing the same PHY layer of the wireless LAN device. This capability in wireless LAN devices is also referred to as multi-SSIDs, where each SSID is equivalent to a VLAN on a wired network. We extend multi-SSIDs capability to operate in the virtualization environments, where each virtual WLAN can have its own addressing, forwarding, routing, and security mechanism.

To emulate a physical AP, it is necessary to provide the emulation at different layers such as layer 2 (MAC), layer 3 (IP), and above. At the MAC layer, the behavior of a physical AP is being emulated by allocating a distinct MAC address and SSID to each virtual AP. At the IP layer, it is emulated by allocating a distinct IP address and potentially a Fully Qualified Domain Name (FQDN) to each virtual AP. In higher layers, the emulation can be carried out by providing each virtual AP with a unique authentication and accounting configuration such as (a shared key, or EAP methods with RADIUS authentication), or SNMP communities.

In our approach, a virtual wireless AP or router is constructed by configuring the VIF to operate in AP mode. This sets the main functionality of the wireless AP such as IEEE 802.11 operation mode and SSID. Once configured, the wireless interface is attached to a virtual switch to enable MAC forwarding similar to a physical AP. Then, the virtual AP interface is connected to a virtual router, in the same way as the virtual Ethernet interface, to enable IP forwarding and routing.

5. IMPLEMENTATION

The multi-SSID capability given by the Atheros chipset allows implementing multiple IEEE 802.11 networks on a single physical wireless card with Linux (Linux kernel version 2.6.33 and higher), since it includes a wireless driver supporting multiple VIF configurations.

The wireless driver for Atheros WLAN devices was initially developed by the madwifi project, and then became part of the Linux kernel. The implementation model of Linux kernel WLAN driver is currently based on SoftMAC wireless devices, where most of the MAC layer functionality is managed by the driver. For the time being, Linux kernel supports all wireless modes with PCI/PCI-Express Atheros WLAN devices only [26].

In order to implement our approach, we used a conventional PC with a wireless LAN card based on the Atheros IEEE 802.11n chipset. It had an Intel Core 2 processor with VT support, Gigabit Ethernet interface and 3 GB RAM. Ubuntu Linux has been chosen to host the virtualization environment for virtual WLAN approach. We used KVM as backend for virtualization and libvirt as frontend for managing VMs. With libvirt, there come two management tools: virt-manager as graphical user interface (GUI) and virtsh as command line interface (CLI).

The virtual wireless interfaces have been created using a CLI configuration utility in Linux named “iw”. Once created, the interfaces have been configured to function as virtual AP or virtual STA interfaces. It is essential for all VIFs to have a unique MAC address, which can be assigned with “ifconfig hw” command or “macchanger” utility.

A virtual AP functionality has been implemented using the hostapd daemon or background service. hostapd is an open source software for controlling wireless LAN authentication and association. It implements IEEE 802.11 AP management and provide support for several security mechanisms such as WPA, IEEE 802.11i, and IEEE 802.1X [27]. The virtual AP interface has been connected to a VDE switch to enable MAC forwarding similar to a physical AP.

For testing our approach, three virtual wireless routers have been hosted on the PC with a shared Internet connection. We created three virtual APs in IEEE 802.11g operation mode, and three virtual routers running Vyatta OS. Each virtual router had two virtual Ethernet interfaces. One of them was connected to the virtual AP interface and the other to the physical Ethernet interface using the Linux interface bridging feature. Each virtual router acted as a DHCP server and DNS forwarder for the virtual wireless LAN and each virtual AP broadcasted different SSIDs to distinguish the wireless networks. NAT functionality was also added to the virtual routers to maintain public IP addresses and to enhance wireless network security. Using these virtual routers, different wireless LAN clients could

access the Internet with different wireless LAN security mechanisms.

6. PERFORMANCE AND RESULTS

We have conducted some tests to understand the impact of the virtual software layer on wireless LANs. The objective of the tests was to compare and quantify the performance of both conventional and virtualized wireless networks. Testing WLAN performance primarily included two test metrics: *throughput* and *response time*. These performance metrics were used to evaluate the applicability of our approach for WLAN infrastructure virtualization since the virtual networks had to handle the same kind of traffic as conventional networks.

The throughput of WLAN is defined as the speed with which a user can send and receive data between the client and the AP. Throughput varies across the WLAN's coverage area. For this reason, we placed the test machines at close range to operate on the maximum available channel bandwidth. Theoretically, the maximum TCP rate of 802.11g network is 24.4 Mbps and the maximum UDP rate is 30.5 Mbps. The UDP throughput is higher than TCP throughput because there is less protocol overhead associated with UDP. Therefore, TCP throughput is the most relevant metric in our performance measurements.

To measure the throughput, we used IPerf and JPerf as the graphical interface. IPerf tool was used to measure TCP and UDP throughput in two directions: uplink direction (from the client to the virtual AP) and downlink direction (from the virtual AP to the client). To measure response times or latencies, we used ping. Ping is used to measure the round-trip time between the client and the virtual AP. In our test setup, IPerf was installed on two machines; the machine which hosts the virtual wireless routers functioned as IPerf server and the wireless client machine as IPerf client. IPerf was configured on the wireless client to run tests for 60 seconds in both directions and provided values in Mbps.

We performed the same test in both native and virtual environments. In the native hardware environment, the tests were performed between a remote client and host machine running three virtual APs without virtualization. In the virtual environment, the tests are performed between a remote client and a VM directly attached to the virtual routers. In this case, the wireless traffic passing through the virtual routers.

Fig. 6 depicts the throughput test results where all throughput results have been averaged over three measurements. The average downlink/uplink TCP throughput is 21.8/18.6 Mbps in the native hardware environment and 21.4/18.2 Mbps in a virtual environment. Latency test results show that the average round-trip time in native hardware environment is 1.1 msec and 2.1 msec in the virtual case. This latency overhead comes from the virtualization layer. The results show that our proposed solution achieves performance metrics comparable to the native hardware environment.

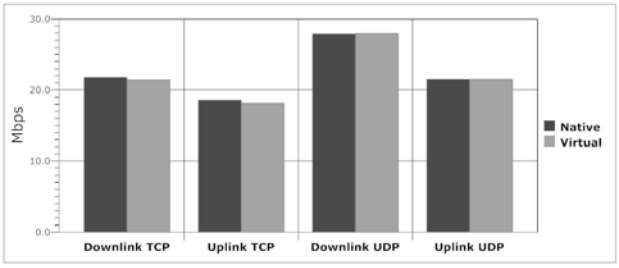

Fig. 6 – Throughput test results

7. CONCLUSION

In this paper, we introduced a virtual networking infrastructure using different virtualization techniques. Also, we proposed a viable approach to realize virtual WLANs by combining wireless LAN virtualization technique with open source virtualization platform.

Our approach adds wireless LAN functionally to virtualization environments. Summarizing some of the benefits, we can conclude that our proposed solution:

- Enables virtualized wireless LAN architectures.
- Builds wired and wireless networks without deploying physical infrastructure.
- Adds the wireless LAN management and control functions to virtualization environments.

For the future, it is planned to investigate performance measurement and optimization with the Xen open source hypervisor. Also, we will design a platform for virtual WLAN approach with different security infrastructures.

8. REFERENCES

[1] B. Aboba, Virtual access points, 2003, http://aboba.drizzlehosting.com/IEEE/11-04-0238-00-0wng-definition-virtual-access-point.doc.

[2] H. Coskun, I. Schieferdecker, and Y. Al-Hazmi, Virtual WLAN: Going beyond virtual access points, *Electronic Communications of the EASST*, 17, 2009.

[3] Lei Xia et al, Virtual WiFi: Bring virtualization from wired to wireless, *ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE)*, 2011.

[4] J. Lee and Y. Moon, Research on virtual network for virtual mobile network, *Second International*

Conference on Computer and Network Technology (ICCNT), pp. 98-101, 2010.
[5] J. Sahoo, S. Mohapatra, and R. Lath, Virtualization: A survey on concepts, taxonomy and associated security issues, *Second International Conference on Computer and Network Technology (ICCNT)*, pp. 222-226, 2010.
[6] P. Barham et al, Xen and the art of virtualization, *ACM Symposium on Operating Systems Principles (OSSP*), pp. 164-177, 2003.
[7] VMware. *Understanding Full Virtualization, Paravirtualization, and Hardware Assist*, 2007, http://www.vmware.com/files/pdf/VMware_para virtualization.pdf.
[8] A. Kivity, Kvm: The linux virtual machine monitor, *Ottawa Linux Symposium (OLS)*, pp. 225-230, 2007.
[9] I. Habib, Virtualization with kvm, *Linux Journal*, 2008 (166). http://www.linuxjournal.com/ article/9764.
[10] T. Abels, P. Dhawan, and B. Chandrasekaran, *An overview of xen virtualization*, http://www.dell.com/downloads/global/power/ps3 q05-20050191-Abels.pdf.
[11] B. Pfaff et al., Extending networking into the virtualization layer, *8th ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, 2009.
[12] Vmware, *VMware Virtual Networking Concepts*, 2007, http://www.vmware.com/files/pdf/virtual _networking_concepts.pdf.
[13] Vmware, *VMware vNetwork Distributed Switch*, http://www.vmware.com/files/pdf/VMware-vNetwork-Distributed-Switch-DS-EN.pdf.
[14] R. Davoli, Vde: Virtual distributed ethernet, *First International Conference on TRIDENTCOM*, 2005.
[15] J. Pettit, J. Gross, B. Pfaff, M. Casado, and S. Crosby, Virtual switching in an era of advanced edges, *2nd Workshop on Data Center – Converged and Virtual Ethernet Switching (DC-CAVES)*, 2010.
[16] Quagga, *Quagga: A routing software package for TCP/IP networks*, http://www.quagga.net/ docs/quagga.pdf.
[17] Vyatta Website, http://www.vyatta.org.
[18] J. Renato et al., Bridging the gap between software and hardware techniques for i/o virtualization, *USENIX Annual Technical Conference*, 2008.
[19] M. Anwer and N. Feamster, Building a fast virtualized data plane with programmable hardware, *ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures*, 2009.
[20] S. Tripathi, N. Droux, and T. Srinivasan, Crossbow: From hardware virtualized nics to virtualized networks, *ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures (VISA)*, 2009.
[21] R. Chandra and P. Bahl, Multinet: Connecting to multiple ieee 802.11 networks using a single wireless card, *IEEE International Conference on Computer Communications (INFOCOM)*, 2004.
[22] C. Chereddi, P. Kyasanur, and N. H. Vaidya, Net-x: A multichannel multi-interface wireless mesh implementation, *ACM SIGMOBILE Mobile Computing and Communications Review*, pp. 84-95, 2007.
[23] A. Sharma and E. Belding, Freemac: Framework for multi-channel mac development on 802.11 hardware, *ACM workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO)*, 2008.
[24] Sung-Won Ahn and Chuck Yoo, Network interface virtualization in wireless communication for multi-streaming service, *IEEE 15th International Symposium on Consumer Electronics (ISCE)*, 2011.
[25] Y. Al-Hazmi and H. de Meer, Virtualization of 802.11 interfaces for wireless mesh networks, *18th International Conference on Wireless On-Demand Network System and Services (WONS)*, 2011.
[26] Linux Wireless Website. http://linuxwireless.org.
[27] hostapd Website. http://hostap.epitest.fi/hostapd.

Ghannam Aljabari, *is a computer engineer in the field of networking and security who used to work as a Network and System Administrator in the Computer Center at Palestine Polytechnics University (PPU). He holds a BSc in Computer Systems Engineering from PPU and pursuing his master degree in Informatics at PPU in Hebron, Palestine. His research interests include virtualization, distributed systems and network security.*

Prof. Dr. Evren Eren, *graduated from the University of Bremen as an Electronics Engineer (Diplom Ingenieur) in 1988 and started at Krupp Atlas Elektronik, working within the marine division as a Software Engineer. In 1992 he changed to the Bremen Institute for Industrial Technology and Applied Work Science (BIBA), where he worked as as research scientist in EU funded projects. 1998 he obtained his PhD degree and moved to DETECON as Senior Consultant. Since 1999 he is professor at the University of Applied Sciences in Dortmund. His working and research areas encompass IT-security and networks.*