WiPi: A Low-Cost Heterogeneous Wireless Testbed for Next Generation Applications

Abdelhamid Attaby
Wireless Research Center
Egypt-Japan Univ. of Sc. and Tech.
Email: abdelhamid.rabia@ejust.edu.eg

Moustafa Youssef
Wireless Research Center
Egypt-Japan Univ. of Sc. and Tech. and
Alexandria University, Alexandria, Egypt
Email: moustafa.youssef@ejust.edu.eg

Abstract—**The high cost of setting up a wireless testbed prevents many institutes from equipping a lab with the required devices, especially in the developing countries. Remotely accessible testbeds provide a solution to this problem by allowing researchers to access the wireless testbeds through the Internet. In this work, we present the design, architecture and evaluation of WiPi: a low-cost remotely-accessible testbed that is built from heterogeneous devices and supports next-generation applications. WiPi efficiently utilizes available off-the-shelf computing nodes, such as standard laptops and Raspberry Pis, to increase the testbed resource utilization. Multiple features including users' isolation, resource pooling, ease of user experience, power efficiency, multiple application domains, efficient disk utilization, and disk protection are implemented as part of the testbed. Evaluation of WiPi using Software Defined Radios as the network interface over a mixed network of laptops and Raspberry Pis show that WiPi can support a different range of wireless networking applications, highlighting its suitability as a low-cost heterogeneous remotely accessible testbed.**

Keywords*—*Network experimentation, Wireless, Remotely accessible testbed, low-cost testbed.

I. Introduction

Wireless technologies flourish every day with new ideas and standards. The researchers experimenting with wireless technologies need to test their work and validate their results. Typically, different approaches can be used for this validation including simulations, emulation, and/or implementation on real devices. Simulations are considered the first choice for researchers due to their low-cost and ease of use. Simulations software such as ns-2 [4], ns-3 [6] and OPNET [2] provide modular and scalable tools for researchers to simulate the desired wired or wireless networks. Emulations, on the other hand, provide manageable and reproducible environments while mitigating the experiments with real applications. Emulations, e.g., CORE [1], EMPOWER [13] has gain researchers' interest recently. However, the experimenters need to execute and test their work on real devices under practical conditions to increase their research credibility. Both simulations and emulations, however, may not reflect reality as actual deployment on real devices.

Implementation on real devices is the optimal approach to capture the conditions in real scenarios. Nevertheless, the high cost and the overhead of setting up the hardware and software on real devices on scale prevent many researchers from validating their results on actual testbeds.

Recently, remotely accessible testbeds, such as CRC [5], CORNET [9], ORBIT [12] and NITOS [10], have gained momentum as they allow researchers to validate their results through pre-installed and open testbeds that can be accessed from anywhere over the Internet. These testbeds provide multiple nodes that are connected to RF devices in addition to extra software tools that can be used to validate the research. An abstract layer is used to manage the testbed, users, and deployed experiments. These leading testbeds, though allowing researchers to validate their protocols and systems in realistic environments, still have space for improvements to scale to new users, especially in developing countries where cost is the main concern.

In this paper, we present the WiPi remotely-accessible testbed which has been implemented with specific design goals including leveraging low-cost hardware that may cost tens of dollars, utilizing heterogeneous devices, resource pooling, and enabling next-generation applications. The testbed leverages commodity hardware devices such as standard laptops/desktops and Raspberry PIs (RPIs) as low-cost computing devices. Besides, it leverages low-cost RF nodes such as standard WiFi adapters and low-cost Software Defined Radios (SDRs) for communication. Moreover, it allows resource pooling to separate the computing nodes (i.e. laptops or RPIs) from the communication nodes (i.e. WiFi, low-cost SDRs, or USRPs) to efficiently utilize the available resources.

The WiPi architecture uses a controller server that runs the cOntrol and Management Framework (OMF) experiment controller that controls the testbed nodes [11]. In addition, it runs a web portal that manages researchers' reservations and authentications, a VLAN control module to isolate concurrent users, a power control module that controls powering on/off the Raspberry Pi nodes, and a disk imaging module to manage users' images. In addition, as part of its design, WiPi needs to handle a number of specific challenges related to the low-cost and heterogeneous hardware including users' isolation, disk images handling, ease of use, and power efficiency.

Evaluation of WiPi using heterogeneous devices and interfaces shows the throughput of the low-cost nodes can support a wide range of wireless networking applications, allowing WiPi to provide a low-cost remotely-accessible testbed.

The rest of this paper is organized as follows: Section II discuss related work. Our design goals and architecture are explained in Section III. We evaluate the WiPi testbed in Section IV. Finally, we conclude in Section V.

II. Related Work

Remotely accessible testbeds allow researchers to test and validate their work over real hardware devices. Many testbeds are public and target wired or wireless research field. In this section, we focus on the wireless testbeds that can be accessed remotely.

One of the most interesting wired and wireless testbeds is ORBIT [12]. It has a 20×20 grid of nodes, and each node is connected to a software defined radio device. The researchers can reserve the whole testbed and perform their experiments. ORBIT is using OMF to manage the controller server and other resources. To build a testbed like ORBIT, it requires high-cost nodes and devices.

CORNET [9] is another wireless testbed. The researchers interested in wireless fields and especially in the algorithms of Cognitive Radio (CR) and its applications, access CORNET to perform their tests and validate their results. The testbed still has a lack of disk imaging system, a reservation system, and others.

NITOS [10] present a testbed built from heterogeneous devices to wired and wireless researchers in different environments, e.g., indoor, outdoor and office. Similar to ORBIT[12], it is very costly to build a testbed like NITOS.

CRC [5] used OMF framework, Frisbee disk imaging system, a reservation system and other features. However, one node in CRC testbed costs about 1000 dollars while one node in WiPi testbed costs about few tens of dollars.

The WiPi testbed, as compared to testbeds mentioned above, considered the most cost-effective testbed. The low-cost of RPis and its ability to be connected with external sensors make the testbed flexible to extension and supportive to next-generation applications. It also presents added features such as a reservation system, disk imaging system, users isolation, device pooling and others.

III. The WiPi Testbed

This section covers the WiPi testbed including the testbed design goals, the proposed architecture and the future directions.

A. Design Goals

1) Low Cost: One of the main features in the testbed is its hardware cost. The low-cost of the element in the testbed make the final cost of the testbed is affordable. Single RPi node costs about \$25 - \$35, and it can work with SDR devices with some limitations. Figure 1 shows how a Raspberry Pi can be connected to a USRP1 via a USB cable. Laptops nodes can be formed from off-the-shelf laptops. A workstation can hold many services and software that operate the needed packages to run the testbed.

2) Heterogeneity: We utilized off-the-shelf heterogeneous devices to form a useful testbed that supports the wireless research community and inspires testbeds' builders with the ability to handle heterogeneous devices in the same testbed. The proposed testbed contains different kinds of devices including Raspberry Pis 3 Model B, laptops, and a workstation.

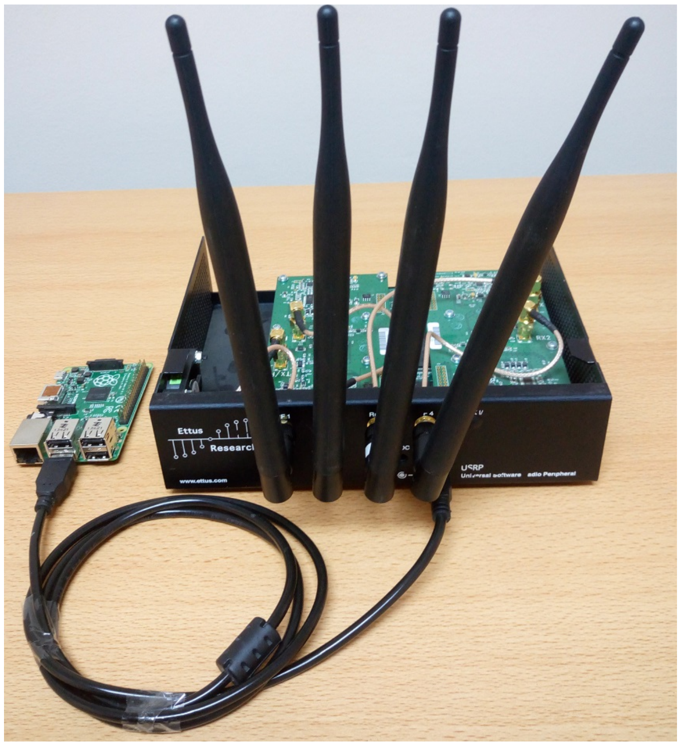


Fig. 1: Example of a Raspberry Pi node connected to a USRP1 via a USB interface.

TABLE I: Variance of Debian-based operating systems.

Node Type	RPi Node	Laptop Node	Controller Server
Operating System	Raspbian	Ubuntu	Debian

3) Better User Experience: The researchers may face difficulty when dealing with different operating systems. It is hard to users to learn different coding styles to deal with different operating systems. One of the main features in the proposed testbed is the homogeneity in the installed operating systems. We used a Debian-based operating systems that share the same Linux commands which in turn provides a better user experience. Table I shows the installed operating systems on the testbed nodes.

The limited number of USRPs and their high cost raises the need of efficient utilization of the high-cost devices. Other testbeds assign a USRP device to each node so that users can configure and use the USRPs from the connected nodes. We utilized high-cost USRPs using the concept of pooling, i.e. putting all USRP devices in a pool and assign a USRP only to the node upon request. The node-to-USRP connection is established only during the reserved time and based on the user's reservation. This new vision raises multiple challenges including USRPs and nodes isolation.

4) Dynamic Network Topology: The concurrent users and different USRP-to-Node assigning may conflict if there is no control over the network of testbed devices. The reserved nodes and devices have to be isolated from other concurrent users' ones. We group different users' devices and nodes by controlling the topology of the testbed network dynamically to

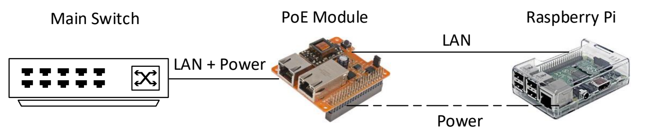


Fig. 2: PoE Module basic idea.

ensure the isolation of different reservations.

5) Power-Efficient Testbed: The low-power consumed by Raspberry Pi devices saves much power in the whole testbed. However, we control the powering operations of Raspberry Pis through Power over Ethernet (PoE) technology. Each Raspberry Pi is connected to a PoE module that can power on/off the RPi. The PoE module takes power from a PoE port of the main switch through LAN cable (Pin 4,5) and outputs the power to the power-in ports of RPi. Through a PoE switch and PoE modules, we can power on/off RPi nodes using scripts that are integrated with the reservation system. The system power on the reserved nodes only at the reserved time and turn off the nodes when the reservation ends or log out users at the end of their reservation times. Figure 2 shows the basic idea behind PoE modules of Raspberry Pi. PoE modules also force Raspberry Pis, when required, to reboot and initiate the process of loading user's image from the network. More details about the booting procedure of Raspberry Pis will be covered later.

6) Multi-Applications Domain: Besides the importance of the testbed in the wireless research, it can be extended to include more research fields. The Raspberry Pi's GPIO pins can be used to connect a different kind of sensors and attract researchers in the area of Internet of Things IoT.

7) Efficient Storage Utilization: We provide base images that can be loaded into the reserved nodes based on the user choice. After the user finishes his experiments, he needs to save back the changes made to the base image to the storage server. We used Frisbee [7] to handle disk imaging operations. Frisbee enables multi-cast in transferring disk images to targeted nodes. Its speed and scalability make the testbed efficient in using the storage.

8) Disk Protection: Users with root access can overwrite the SD card plugged into Raspberry Pis. They can affect the entire operation of the Raspberry Pi, even intentionally or by mistake. To avoid that, we force RPi3-B models to boot from network upon restart and ignore the SD card; then, the booted operating system will initiate the disk image writing procedure. For RPi1-B, the model doesn't support booting from the network; so, we can use u-boot[3] to add network booting feature to RPi1-B nodes.

9) Pooling:

B. Architecture

Figure 4 shows the architecture of the testbed. It consists of Raspberry Pis nodes installed with their PoE modules to control the nodes' power, in addition to, USRP devices that can be connected via LAN port and laptops nodes that are connected, in some of them, to USRP1 devices via USB port. All the nodes and USRP devices are connected via a managed PoE-support switch and a controller server. The controller server plays a set of roles and can be accessed by researchers at their reservation time with limited privileges. Details about the hardware and software used are covered in the next subsections.

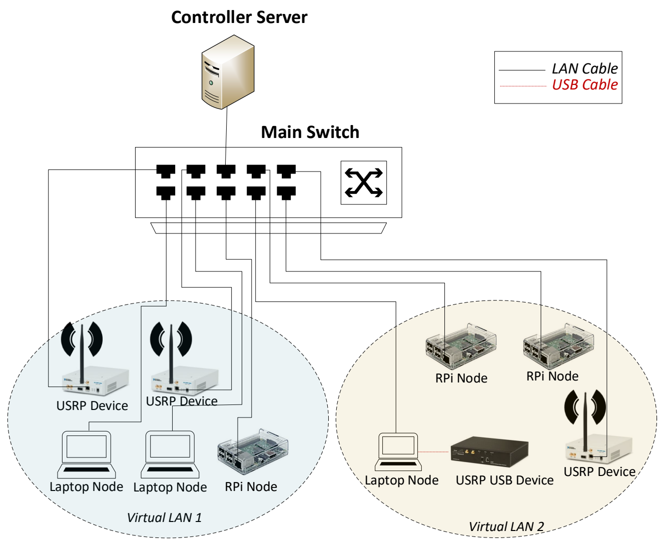


Fig. 3: Dynamic VLANs creation for resources pooling and users isolation.

1) Hardware: We implemented the WiPi testbed using the following hardware:

Raspberry Pi Nodes: The testbed includes RPi3-B nodes installed with their PoE modules. The testbed can be extended using other models of Raspberry Pis. There are few differences in their specifications and booting procedure that may affect their integration with the system. We show later how this can affect images loading procedure and how we solved it.

PoE Main Switch: A GigaEthernet PoE switch was installed to connect RPi nodes, laptop nodes, USRP devices and the controller server. The switch provides the necessary power to Raspberry Pi nodes through their PoE modules, also, it can be configured using its console port by VLANs control module and power control module. We apply the isolation using the concept of virtual local area network VLAN. The reserved nodes and devices are grouped in a new dynamically-created VLAN that prevent other users accessing the reserved devices. We developed a VLAN control module that automatically connects to and configures the main switch to add the ports assigned to the reserved nodes into a new VLAN during the reserved time. As shown in Figure 3, suppose that two independent users reserved different devices and nodes at the same time. Let user 1 reserved two USRP devices, two laptop nodes, and one Raspberry Pi node. Let user 2 reserved one laptop node connected to a USRP1 device through a USB connection, one USRP device and two Raspberry Pi nodes. The system will automatically detect the concurrency, and the controller server will run scripts that will access the main switch to create two VLANs, one for each user. The nodes and devices reserved by the first user will automatically be

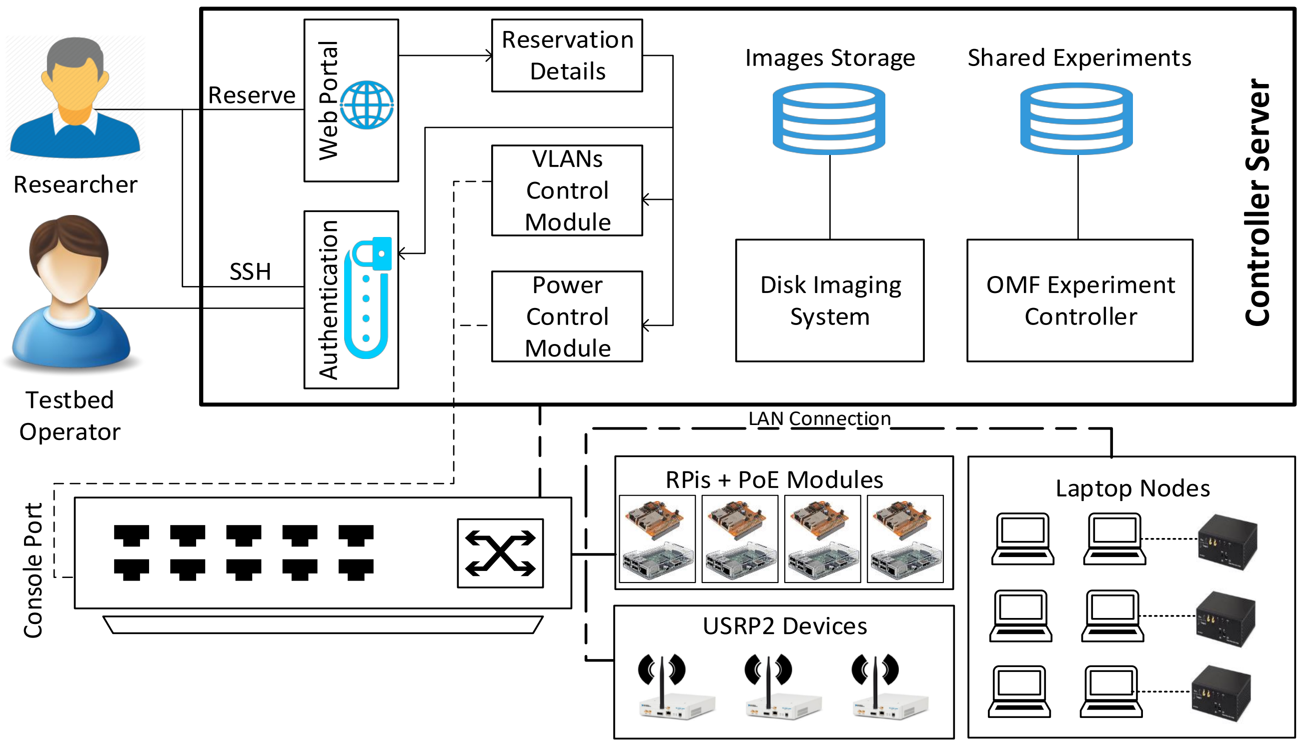


Fig. 4: Testbed architecture.

assigned to VLAN 1; similarly, the nodes and devices reserved by the second user will automatically be assigned to VLAN 2. Using this method, pooling and isolation can be achieved without any interference.

Laptop Nodes: The testbed includes laptops produced by various brands and specification. Some of the laptops are connected directly to USRP1 devices through USB. The researcher can reserve a laptop connected to a USRP1 device via USB cable or a standalone laptop. Standalone laptops can run various applications or act as USRP2 processing nodes; therefore, the testbed utilizes devices and nodes efficiently instead of dedicating a USRP2 device to each laptop or RPi.

USRP Devices: Software Defined Radio (SDR) devices with model number USRP1 and USRP2 are installed with their daughter boards. The high-bandwidth and high-dynamic range processing capability of the product provides researchers with powerful devices for testing their experiments and results.

Controller Server: We prepared a workstation to act as the controller server that manages the web portal, reservation system, experiments, VLANs control module, power control module and disk imaging software. The server can be fully controlled by testbed operators or partially accessed by researchers during their reservation times with limited privileges.

2) Software: The controller server and the nodes were prepared to accept different packages to facilitate the operation of the testbed. The main packages include:

Resource Controller and Experiment Controller: To provide an efficient framework to manage and measure different researchers' experiments, we used OMF[11] to manage the experiments and OML to measure the results. Using OMF, researchers can write the experiment description into a script that specifies the participated nodes and the required commands to be executed in each node at specific times. The script written by OEDL (OMF Experiment Description Language) can realize the experiment using a sequence of events to be executed on each node. During that, OML is collecting the results and storing them in a database to be recovered later. We used OMF because of its popularity and efficiency in experiments' management and measurement. OMF and OML source codes don't support Raspian operating system and aren't compiled for ARM processors. To integrate the required software with our testbed, we edited the source code and cross-compiled it to work over Raspberry Pis with Raspian operating system.

Disk Imaging: 5 minutes before the user reservation time, the system can loads base-images into nodes' disks. We provide base-images installed with common protocols and tools e.g., NS2[4], NS3[6]. After researchers finish their edits and software installations, they can save the entire disk image into the server's images storage. The saved image can be loaded next time by the same user. By this way, the users can't affect each other, and each user can resume from his final edits. To achieve that, we configure laptops and Raspberry Pis to boot initially from the network, pull their IP address from DHCP server and load from the TFTP server the assigned operating systems to them. We compiled a tiny operating system TinyOS [8] with the required Frisbee [7] client and disk imaging tools. The TinyOS kernel and files are loaded into the testbed nodes using TFTP server. After that, the Frisbee client call the Frisbee server to load the image of the specific user into that node.

TABLE II: Installed nodes specifications.

Type	Processor	Memory	LAN	USB
RPi3-B	1.2 GHz Quad Core	1 GB	10/100 BaseT Ethernet	USB 2.0
Laptop node (Example)	2.4 GHz Core i3	4 GB	10/100/1000 BaseT Ethernet	USB 2.0

TABLE III: Installed SDR devices specifications.

Type	LAN	USB	Max. Host Streaming Rate
USRP1	NA	USB 2.0	256 Mbps
USRP2 N210	Gigabit Ethernet	NA	800 Mbps

We can add other models of Raspberry Pi; Raspberry Pi 1 for example but the model doesn't support network booting. To overcome this problem, a u-boot operating system [3] has to be installed initially on the SD Card of Raspberry Pi 1 nodes to initiate TFTP booting procedure. Furthermore, the low-memory of Raspberry Pis (512 MB for RPi1 and 1 GB for RPi3) obstruct Frisbee client's operation during loading process. We overcome the problem by dividing the image into small size blocks and loaded each block separately.

Web Portal: The web portal allows both testbed operators and researchers to authenticate and access the system. Researchers have to reserve the required nodes and devices from the reservation system module of the web portal. Their reservation details are provided to both VLANs Control Module to separate the reserved items during reservation time and Power Control Module to power on the reserved Raspberry Pis 5 minutes before the reservation time and initiate user's disk image loading procedure. The user can access the controller server and run his experiments either through a web interface or SSH.

C. Future Directions

1) Mobility Nodes: Mobility nodes can be integrated with the testbed to add more support to testing facilities. A Raspberry Pi stacked over a mobile robot can act as a mobility node. More challenges have to be solved, e.g., the description language for the robot movement, the map and obstacles, controlling the power of mobile RPis, connection problems and others.

2) MIMO Support: Current testbed doesn't support Multiple In Multiple Out (MIMO) research. Several challenges have to be considered including MIMO connections and synchronization, scheduling, support of devices pooling and others.

3) Sensors Extensions: Motion, navigation, temperature and other sensors can be connected to the GPIO pins of a Raspberry Pi. Installing such sensors adds more support in the IoT research field.

IV. Evaluation

In this section, we evaluate the capabilities of low-cost devices when connected to SDR devices. The evaluation studies the throughput between nodes, either Raspberry Pis or laptops and USRP devices. Tables II,III summarise the specifications of the installed nodes and the SDR devices, respectively. The data can be streamed between the FPGA of a USRP device and a node using the host interface. We evaluated the streaming rate through *benchmark_rate* command that inputs the testing rate in Mega Samples per Second (MS/s), the sample type, the direction of the transfer and other options. We fixed the type of samples to be I/Q sample which is used by most applications and the data format to be 16-bit complex<float>. The total size of each sample is 32-bit. We monitored the number of samples that can be sent/received/drooped between nodes and different USRP devices in addition to the number of overflows/underflows detected, and then we calculated the actual rate. Figure 5 summarizes the results obtained using the two types of USRPs and the testbed nodes.

A. Raspberry Pi Nodes

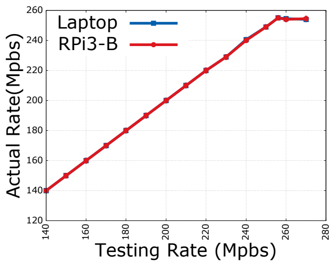
Using a Raspberry Pi as a software defined radio processing node has some limitations due to the limited capabilities of Raspberry Pis, however, if the experiment and the required processing are small, it can be handled by Raspberry Pis. We show here the throughput between Raspberry Pis and USRP devices. We connected Raspberry Pis with a USRP1 device via a USB cable and monitored the samples sent/received to/from USRP1 device in 10 seconds. Figure 5a,5b summarize the 95% confidence interval of the average of sending/receiving rate to/from a USRP1 device through USB2 connection. As shown in the figures, a Raspberry Pi 3-B can perfectly handle the small and high date rates sent/received to/from a USRP1 device up to the maximum host streaming rate of the USRP1 device (256 Mbps). Furthermore, due to the maximum connection rate of Ethernet technology (100 Mbps), a Raspberry Pi 3-B can't handle, on average, more than 93.5 Mbps practically when connected to an N210 USRP device through a LAN connection. Figure 5c,5d shows 95% confidence interval of the average of the actual rates that can be streamed between a Raspberry Pis and a USRP N210 device.

B. Laptops Nodes

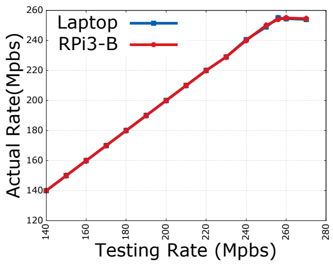
Nodes consist of laptops or laptops connected to USRP1 devices are considered the main processing unit to the researchers who are using the WiPi testbed. We evaluated the maximum throughput that a laptop can handle. Figure 5a,5b shows the relation between the testing rate and the actual rate in 10 seconds when a laptop is connected to a USRP1 device. The results show that a laptop can work as a perfect software defined radio processing node using a USB connection and up to the maximum allowed rate of USRP1 devices (256 Mbps). Figure 5c,5d summarize the same test when using a LAN connection to USRP N210 device; it shows that a laptop equipped with a GigaEthernet interface can handle date rates sent/received to/from an N210 USRP devices up to its maximum capabilities (800 Mbps).

V. Conclusion

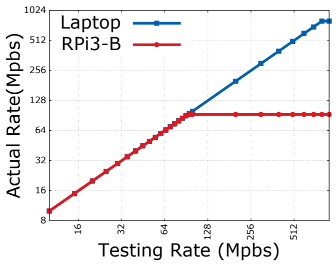
We presented the WiPi testbed as a low-cost testbed that can be accessed remotely to provide researchers with wireless nodes and devices to test their work and results; therefore, increase their research credibility. The testbed is implemented from heterogeneous devices and supports multiple features including powering control, homogeneous operating systems, resource pooling, users isolation using VLANs, multi-applications domain, disk protection and efficient disk



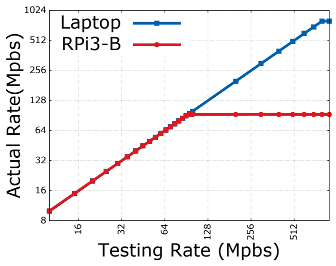
(a) Sending to USRP1 (USB).



(b) Receiving from USRP1 (USB).



(c) Sending to N210 (LAN).



(d) Receiving from N210 (LAN).

Fig. 5: Evaluating the data rate between the testbed nodes and the USRPs.

utilization. We also addressed the challenges we faced during implementation and how we handled them. The results show that Raspberry Pis can act as software defined radio processing nodes, in the case of small data rates, and laptops can process wireless physical operations perfectly. We are currently working on extending the testbed to include the mentioned future directions.

References

[1] Jeff Ahrenholz, Claudiu Danilov, Thomas R Henderson, and Jae H Kim. Core: A real-time network emulator. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7. IEEE, 2008.

[2] Xinjie Chang. Network simulations with opnet. In *Simulation Conference Proceedings, 1999 Winter*, volume 1, pages 307–314. IEEE, 1999.

[3] U Das. Boot–the universal boot loader. *Verfügbar unte r http://www. denx. de/wiki/U-Boot [23.09. 12]*, 2002.

[4] Kevin Fall and Kannan Varadhan. The network simulator (ns-2). *URL: http://www. isi. edu/nsnam/ns*, 2007.

[5] Samer S Hanna, Arsany Guirguis, Mahmoud A Mahdi, Yaser A El-Nakieb, Mahmoud Alaa Eldin, and Dina M Saber. Crc: collaborative research and teaching testbed for wireless communications and networks. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, pages 73–80. ACM, 2016.

[6] Thomas R Henderson, Mathieu Lacage, George F Riley, C Dowell, and J Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14(14):527, 2008.

[7] Mike Hibler, Leigh Stoller, Jay Lepreau, Robert Ricci, and Chad Barb. Fast, scalable disk imaging with frisbee. In *USENIX Annual Technical Conference, General Track*, pages 283–296, 2003.

[8] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, et al. Tinyos: An operating system for sensor networks. *Ambient intelligence*, 35:115–148, 2005.

[9] Timothy R Newman, An He, Joseph Gaeddert, Ben Hilburn, Tamal Bose, and Jeffrey H Reed. Virginia tech cognitive radio network testbed and open source cognitive radio framework. In *Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops, 2009. TridentCom 2009. 5th International Conference on*, pages 1–3. IEEE, 2009.

[10] Katerina Pechlivanidou, Kostas Katsalis, Ioannis Igoumenos, Dimitrios Katsaros, Thanasis Korakis, and Leandros Tassiulas. Nitos testbed: A cloud based wireless experimentation facility. In *Teletraffic Congress (ITC), 2014 26th International*, pages 1–6. IEEE, 2014.

[11] Thierry Rakotoarivelo, Maximilian Ott, Guillaume Jourjon, and Ivan Seskar. Omf: a control and management framework for networking testbeds. *ACM SIGOPS Operating Systems Review*, 43(4):54–59, 2010.

[12] Dipankar Raychaudhuri, Ivan Seskar, Max Ott, Sachin Ganu, Kishore Ramachandran, Haris Kremo, Robert Siracusa, Hang Liu, and Manpreet Singh. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 3, pages 1664–1669. IEEE, 2005.

[13] Pei Zheng and Lionel M Ni. Empower: A network emulator for wireline and wireless networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1933–1942. IEEE, 2003.