Computer Science and Information Systems 14(3):857–873 https://doi.org/10.2298/CSIS161230033X

Building a Lightweight Testbed Using Devices in Personal Area Networks

Qiaozhi Xu[1,2] and Junxing Zhang[1,#]

[1] College of Computer Science, Inner Mongolia University
Hohhot, China
ciecxqz@imnu.edu.cn,junxing@imu.edu.cn (#Corresponding author)
[2] College of Computer Science, Inner Mongolia Normal University
Hohhot, China

Abstract. Various networking applications and systems must be tested before the final deployment. Many of the tests are performed on network testbeds such as Emulab, PlanetLab, etc. These testbeds are large in scale and organize devices in relatively fixed ways. It is difficult for them to incorporate the latest personalized devices, such as smart watches, smart glasses and other emerging gadgets, so they tend to fall short in supporting personalized experiments using devices around users. Moreover, these testbeds commonly impose restrictions on users in terms of when and where to carry out experiments making them clumsy or inconvenient to use. The paper proposes to build a testbed utilizing users' devices in their own personal area networks (PANs). We have designed and implemented a prototype, which we call PANBED. Our experiments show that PANBED allows users to set up different scenes to test applications using a home router, PCs, mobile phones and other equipment. PANBED is light weighted with a size less than 16 KB and it has little impact to the other functions of the PAN. The experiment results also prove the realism, effectiveness, flexibility and convenience of PANBED.

Keywords: Testbed, Personal Area Network, PAN, Personal Network.

1. Introduction

Various networking applications and systems must be tested before the final deployment. Presently there are four commonly used test methods in networking and distributed system research: network simulation, overlay network, network emulation, and network testbed.

Network simulation is the method of simulating the operations of each network layer using a kind of software called simulator. Changes of system states caused by different events are recorded and modeled by the simulator. Network simulation is relatively simple, low-cost, controllable, repeatable, and relies on pure software environment to breaks the limitation of physical resources. Ns-2 [19], ns-3 [32], OMNet ++ [39], Atemu [30], TOSSIM [25], GloMoSim [8], SensorSim [28] are the most widely used simulation systems. However, a simulated environment can be quite different from the real physical environment, and it cannot capture changes of lower network layers in many cases, resulting in poor realism of experimental results.

A overlay network is a real network environment built on another existing network [4], which can test and evaluate the realistic performance of protocols and algorithms, such

as RON [7] and PlanetLab [12]. But overlay networks are high-cost, vulnerable to the impact of other network environments, and users are cumbersome to modify the network parameters, and unable to monitor the network behaviors which making the experiments and tests unrepeatable and difficult to control.

Network emulation is a trade-off method between network simulation and overlay network [44]. An emulation system achieves functions of a real system by using the software simulation and abstraction techniques on real devices and introducing the configurable parameters such as packet loss and link delay. Typical emulation systems are VMNet [45], Avrora [37], Dummynet [33], NSE [16], and ModelNet [38] and so on. An emulated environment is very close to a real one. It also possesses the repeatability of network simulation and realism of overlay network but it requires cumbersome manual configuration.

Current network testbeds typically incorporate simulation, emulation and overlay network into an integrated experimental platform of software and hardware. They are capable of producing repeatable and controllable scenes to reduce costs of setting up experiments. They are easy to use and offer different degrees of realism. The well-known network testbeds include Emulab [44], Kansei [14], MoteLab [43], GNOMES [42], GENI [9], Winlab [6], etc. However, these large-scale network testbeds tend to have some shortcomings: (i) A user must login to a testbed remotely to carry out his experiment, and the availability of devices is out of his control; (ii) it is difficult for these testbeds to incorporate the up-to-date or personalized devices, such as smart watches, smart glasses and other emerging gadgets, so they tend to fall short in supporting experiments aiming at the latest devices around users. (iii) The background traffic in both simulated and emulated systems is not realistic, which affects the realism of experimental results.

In order to overcome the shortcomings of large-scale network testbeds, this paper proposes to build a network testbed using users' devices in their own personal area networks (PANs). We have designed and implemented a prototype of this system, which we call PANBED. Our experiments show that PANBED allow users to set up different scenes to test applications using a home router, PCs, and mobile phones. The results demonstrate PANBED enable users to assess applications at their convenience using diverse, personal, up-to-date and low cost devices around them with little impact to existing PANs. As far as we know, PANBED is the first network testbed built on devices in a PAN.

To design and implement PANBED, we have identified and addressed the following six key challenges:

(1) Where to implement traffic shaping?

The traffic shaping in some testbeds is implemented by running DummyNet on the intermediate nodes (delay node). However, the same way cannot be adopted in the PANBED, because devices in a PAN are quite different from that in a testbed. Firstly, the number of devices in a PAN is limited. Most likely, there are only one home router and several devices. Secondly, not all devices in a PAN support DummyNet. Thirdly, these devices not only participate in experiments, but also complete original tasks for users, thus PANBED should change these devices as little as possible. For these reasons, we choose a home router to support the traffic shaping in the PANBED.

(2) How to implement traffic shaping?

There are two flow control modules, Netem and TC, in Linux that can shape the flow through a network card. OpenWRT [15] is based on the Linux kernel and often

used in embedded network devices. PANBED loads OpenWRT into the home router and implements the traffic shaping utilizing the Netem and TC modules.

(3) Is isolation of the control flow and data flow necessary on PANBED?

Some testbeds isolate the control flow from data flow by installing multiple network cards in experimental devices, and creating the control vlan and experimental vlans on switches, but devices in a PAN do not have such hardware conditions, so similar methods cannot be used in the PANBED. Devices in a PAN are all around users, and users know well about these devices. PANBED almost does not modify users' devices and only applies data flow control policies on the home router, so there is no isolation of the control flow from data flow.

(4) How to ensure the repeatability of experiments?

To ensure the repeatability of experiments and the consistency of experimental environments, many testbeds initialize devices with default or saved parameters before an experiment. However, in PANBED experimental devices are not dedicated. They need to complete their original tasks and cannot be frequently initialized, so it is difficult to ensure the repeatability of experiments in this situation. PANBED adopts two empirical approaches: (i) Since users know enough about the experimental devices, PANBED allows users to decide whether or not to initialize their devices, thereby improving the repeatability of experiments; (ii) before an experiment, PANBED collects and records system states of experimental devices, and provides secondary reference to users for the analysis of experimental results.

(5) How about realism?

In some testbeds there exist both real devices and simulated components such as NSE and DummyNet, but in PANBED, there are only real devices and it makes experimental results more realistic. In addition, the background traffic in many testbeds is not realistic, which affects the realism of experimental results. In PANBED, the background traffic is real, so the experimental results are more realistic.

(6) How does a user deploy her own application?

In most testbeds, a user remotely logins to experimental devices through SSH and deploys his application. In PANBED, all devices are around users, he can directly login and operate these devices.

We have designed and implemented PANBED based on the solutions to the above issues. Our experimental results show PANBED allows users to evaluate applications at their convenience, allows users to set up different scenes to test applications using a home router, PCs, and mobile phones and guarantees the realism of the experiment.

The remaining of the paper is organized as follows. The background and related work are given in Section 2. Section 3 and Section 4 describes the design and implementation of PANBED. The experiment and evaluation results are analyzed in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

The goal of PANBED is to provide users with a low-cost and flexible network testbed utilizing devices in users' PAN. The following introduces the related research works about the network testbed, OpenWrt and PAN.

2.1. Network Testbed

Various networking applications and systems must be tested before the final deployment. Tests based on a real environment are high cost, long time, difficult to control and repeat. Test results based on a simulation are inaccurate compared with that based on a real environment. So in recent years, many researchers have begun to study and build network testbeds [12], [44], [38].

These testbeds are relatively large in scale and their operation mode generally are: (1) In one or several physical spaces, devices are organized in a fixed ways and some resource pools are formed; (2) users login the web server of a testbed through Internet after registration and being agreed by the administrator; (3) users submit their experimental requirements which specifying the device type, operating system version, connection topology of these devices and parameters of these links, such as bandwidth, delay, and loss; (4) testbed servers parse these experimental requirements, allocate resources, and build network environments for users according to their requirements; (5) users login the assigned devices via SSH or telnet remotely and begin their experiments.

These testbeds provide users with controllable and repeatable experiment environments without any input from them, but, their device types are limited and it is time-consuming to introduce emerging devices into testbeds which limits their flexibility. PANBED proposed in this paper can build a testbed for users using devices in their PAN and enables user to assess applications at their conveniences using diverse, personal, up-to-date and low cost devices around them with little impact to the existing PAN.

2.2. OpenWrt Routers

OpenWrt is an embedded system based on Linux kernel and often used in network devices such as industrial devices, telephones, small robots, smart homes and routers, etc. The sofeware architecture of OpenWrt is shown in Fig. 1.

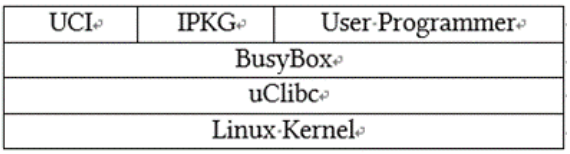

Fig. 1. Software architecture of OpenWrt [15] which shows OpenWrt is embedded a number of tools, such as uClibc, busybox and shell interpreter, etc. based on the basic Linux kernel.

Recently, OpenWrt is supported by more and more router vendors, such as 3Com, D-Link, TP-Link, Huawei, Netgear, etc. [3]. At the same time, many researchers also choose OpenWrt routers as a basic component in their researches for its openness and programmability. For example, Kai implemented a PPPoE traffic control system [46], Kim implemented the remotely intelligent management to an indoor lighting system so as to save energy [22], Kciuk realized the remotely control on robots in an intelligent buildings [21], Palazzi achieved the fast and smooth transmission of real-time flow and meanwhile ensured the high throughput of TCP applications [27], Lee designed a software-defined

wireless mesh network architecture SD-WMN [24], Serrano built a low-cost wireless network testbed [34], Reich designed a delay tolerance network testbed-MadNet [31].

2.3. Personal Area Network

PAN (Personal Area Network) refers to connecting personal terminal devices as a network by using variety of communication technologies, and nowadays, wireless personal area network (WPAN) is one of the main forms of PAN. Currently, researches on WPAN mainly focus on enhancements and improvements of the performance of WPAN such as throughput, energy consumption, coverage, data transmission rate and so on [35], [26], [11], [49], [50], [40], [23], some researchers also use WPAN to realize intelligent home, telemedicine and other purposes [17], [41], [29], [20], [13], [47], [48]. Up to now, there is no research to build a testing platform for users using devices in their PAN.

PANBED enables users to do testing with diverse, personal, latest devices around them and with little impact to existing PAN. As far as we know, PANBED is the first testbed to be built by utilizing devices within a PAN.

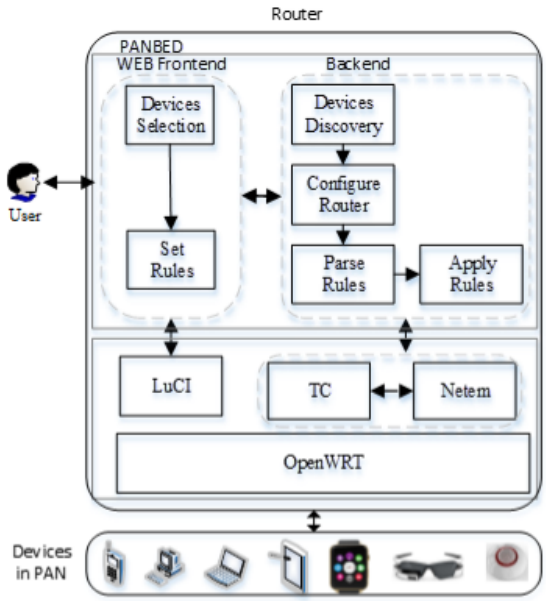

Fig. 2. PANBED architecture which is a light weight function based on the OpenWrt and is divided into frontend and backend.

3. PANBED Design

3.1. Overall Architecture

Usually there is a home router, several PCs, smart phones, tablets, sensors and other equipment in a PAN, and it is difficult and complex for users to do tests directly using these devices. PANBED provides users a convenient way to do testing under different network scenes using diverse, personal, and latest devices around them.

The architecture of PANBED is shown in Fig. 2. Users choose experimental devices and set experiment rules through the Web frontend. The backend completes functions such as discovering devices, configuring router, analyzing and applying rules, etc.

3.2. Frontend Design

PANBED provides a web access for users to facilitate their operations. Many testbeds set up a separate web server to deal with experimental requests. However, the number of devices within a PAN is limited, even no PC, so it is unrealistic to set up a separate web server in PANBED but to build the web service on the OpenWrt router.

Usually, users configure a router through a web page which is a web service based on uHTTPd [5]. uHTTPd is aimed towards being an efficient and stable server, suitable for lightweight tasks, commonly used with embedded devices and proper integration with OpenWrt's configuration framework [2].

For minimizing impact on the router and convenience of users, the web frontend of PANBED is embedded within the LuCI configuration page in a OpenWrt router with the template way, as shown in Fig. 3.

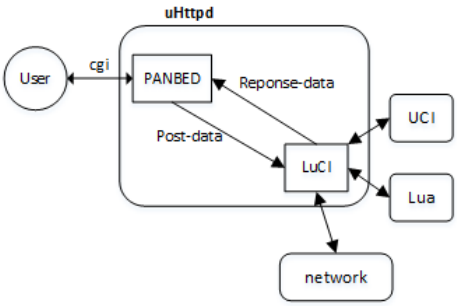

Fig. 3. Front-End Design of PANBED.

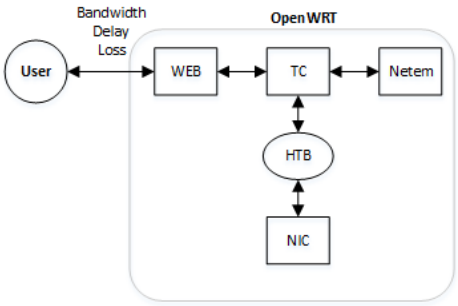

Fig. 4. Traffic Shaping of PANBED.

3.3. Backend Design

In PANBED, traffic control is the basis of backend's other functions, and repeatability of experiments is one of the important issues that PANBED needs to consider as a testbed.

Design of traffic control. In some testbeds, traffic shaping was implemented by DummyNet running on another Delay Node. DummyNet is a tool of FreeBSD system. In a PAN, there are a limited number of devices, PANBED cannot install and run DummyNet on all devices because these devices are not proprietary lab devices and may not support DummyNet; in addition, there may be no enough nodes to act as delay nodes; moreover, in order not to affect users' daily use, we should minimize the changes to these devices. For these reasons, traffic shaping in PANBED is implemented on the OpenWrt router.

OpenWrt supports two network emulation modules of Linux: netem [18] and TC (Traffic Controller)[10], [1]. Netem can simulate complex network transmission characteristics in a well-behaved LAN, such as variable bandwidth, delay, loss, repetition and reordering. TC controls the working mode of netem. TC supports classless and classful queue disciplines. The classless disciplines are relatively simple, and the date flow can be sorted, speed limited, and discarded, but cannot be differentiated fine-grained. The classful disciplines can implement fine-grained and differentiated traffic control by classifying packets with the classifier and filter.

In PANBED, a home router transmits experimental flow and non-experimental flow at the same time, but only control the experimental flow, so the classful disciplines are applied to achieve fine-grained traffic control. There are three types of classful queue disciplines: CBQ (based on class queuing), HTB (hierarchical token bucket) and PRIO (priority queue), but only HTB can control the flow fine-grained and easily, so we implement the traffic control using the HTB queue as shown in Fig. 4.

Repeatability of experiments. To ensure the repeatability of experiments, many testbeds initialized experimental devices before experiments starting. However, devices in a PAN are not special testing devices, and store a large amounts of users' data, it is not possible to initialize these devices frequently which poses challenges to the repeatability of experiments.

We take two empirical approaches to improve the repeatability: (i) Since users know enough about their experimental devices, PANBED allows them to decide whether or not to initialize these devices; (ii) PANBED records the system state of every experimental device before experiments starting, such as system version, CPU, memory usage, etc., and provides users a reference for analyzing of experiment results.

4. Implementation Details

4.1. Implementation of Frontend

The frontend is implemented by using the template technology through the OpenWrt LuCI configuration page. Users can view all devices connected to the router, select experimental devices and do their experiments through the web page.

4.2. Implementation of Backend

In order to enable users to easily build their personal testbed using devices around them, the backend needs to accomplish tasks such as device discovery, router configuration, rules parsing, rules applying and restore the PAN to the initial states after experiments.

Discovering devices. In home environments, most users use DHCP to allocate IP addresses, subnet masks, gateways, and DNS information to devices in a PAN. On a OpenWrt router, the information of devices connected to it is stored in dhcp.leases file and odhcpd file. PANBED discovers and displays all connected devices on the LuCI web page by parsing them.

Configuring the router. By default, a home router cannot directly control the flow using TC because for most of home routers, a LAN port doesn't be equipped a separate network interface adapter card (NIC). Usually, all wired LAN ports share a NIC, and all wireless devices share another NIC. Data exchanges among WAN, LAN and Wi-Fi go by bridge, and data exchanges among wired LAN ports don't pass through the NIC, as shown in Fig. 5 [2]. TC is a tool of network layer, and it can control the traffic only when the traffic passes through a physical NIC.

To solve the problem, vlan technology is used in PANBED. When users select N test devices, PANBED automatically creates N vlans on the router, and puts each device into a separate vlan so that data flow among these devices must pass through a physical NIC and make the TC take effect. In addition, PANBED creates routings for these vlans, because devices belonging to different vlans cannot communicate directly.

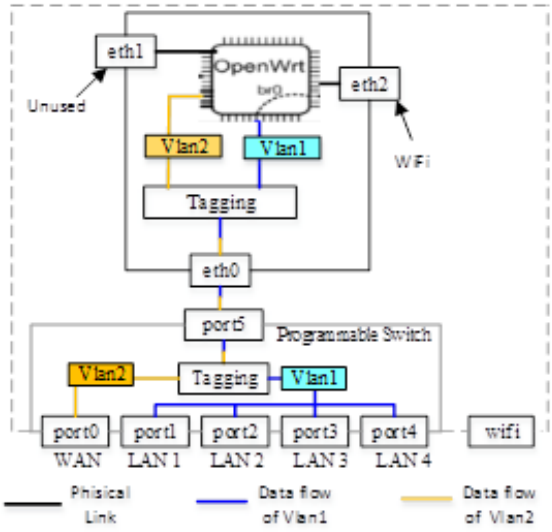

Fig. 5. Architecture of a Common AP which shows that all wired LAN ports share a NIC, and wireless devices share another NIC. The data exchanges among wired LAN ports don't pass through the NIC.

Parsing and applying rules. Links among testing devices specified by users are correspond to a series of rules. The format of each rule is as ¡source device, destination device,

bandwidth, delay, loss¿. PANBED parses and applies these rules by performing Algorithm 1.

```
Algorithm 1:
  (1)Creates N vlans on the Router;
  (2)Bonds every vlan with a specific interface,
      for example, vlan3 with eth0.3, vlan4 with eth0.4;
  (3)Allocates N experiment devices into N Vlans respectively;
  (4)Set IPs= {IPd1, IPd2 IPdn}; d1, d2,, and dn refers to
      experiment device 1, device 2, , and device n.
  (5)Read Rules
  (6)For each rule[i] in Rules
    Creates a htb sub-class with handle i and sets up the bandwidth limiter;
    Sets up netem for configuring delay and loss for sub-class i;
    Sets up the filter for filtering data flows of meeting conditions;
    Writes all above to a shell script file;
  (7)Run the script file;
```

Restoration of PAN environment. After finishing experiments, PANBED removes all experimental rules, vlans and routings on the router and restores the PAN to original states.

5. Evaluation

Users just need to upload several script files less than 16KB to a home router to utilize PANBED to do experiments. The following illustrates the realism, effectiveness and convenience of PANBED with a use case.

5.1. Use Case

With the enhancement of smart devices, mobile applications based on crowdsourcing are more and more. Assuming that a crowdsourcing application requires smart devices such as mobile phones, smart bracelets, watches and glasses to periodically report position, temperature, humidity and noise. For not influencing users' experience, the application can store the data in a file for a while until the file size exceeds a threshold, then upload it to the server.

A suitable threshold is important for better users' experience and it is related to many factors, among which the network condition is an important factor. When users are in a good Wi-Fi environment, the threshold can be set larger, but when users are in a mobile network environment with bad signal, too large threshold will increase the upload delay, even cause fail.

In order to improve users' experience, the developer wants to test a suitable threshold at different network conditions, and make the application adjust the threshold automatically according to the current network condition.

The current testbeds' supporting for smart devices are limited, some of them use virtual machine, and some of them provide real android smart phones, but their versions are old and their hardware condition is limited, so there exist some problems to complete similar testing tasks described above on these testbeds.

Certainly, the developer also can use devices around him to do the testing directly, but the processes are too complex. Firstly, the developer should be good at the Linux and network technology such as cross compile of OpenWrt, routing, vlan, traffic control and other technologies. Secondly, the developer should know the network configurations of each experimental device, configure the testing rules and restore them after experiments by manually.

PANBED builds a convenient experimental environment for testers utilizing devices around them. Using PANBED, testers can do testing easily and needn't to care about the configuration of each device and the implementation details of the underlying.

5.2. PANBED Setup

One of the important advantages of PANBED is portability and economy. We purchase an OEM OpenWrt router which works in 2.4GHz, with four 5dBi high omnidirectional antennas, supports IEEE 802.11b / g / n, IEEE 802.3 and IEEE 802.3u protocols, with a maximum wireless speed of 300Mbps, with one adaptive WAN port of 10/100M, four adaptive LAN port of 100/1000M, with wireless security basic features, with motherboard chipset of MT7620N, and costs about 65 Yuan RMB.

PANBED is light weighted. Firstly, it is very easy to install PANBED on an OpenWrt home router which only requires users to upload several script files to the router by SHH. The size of these files is less than 16KB, and they provide users all functions of PANBED. The total time of loading PANBED is less than 2 minutes.

Secondly, it is very simple for users to do testing on the PANBED. Users log on the router through the web browser such as http://192.168.1.1/. Then they can choose devices and do testing easily. Fig. 6 shows our devices in the PAN, including one PC, two Android phones, one Android smart glasses and one android smart bracelet. All devices are in network 192.168.1.0/24, connect to the router by Ethernet or Wi-Fi, can communicate with each other, and access the Internet.

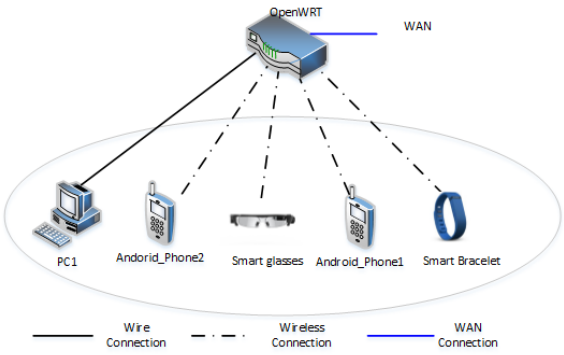

Fig. 6. Devices in our PAN including an OpenWrt home router, a PC, two android mobile phones, a smart glass and a smart bracelet, and they are in a same vlan initially.

5.3. Experiment Results

For completing the testing tasks described in the use case and verifying the effectiveness of PANBED, we implement a client app based on the android platform and a sever program of language C. The client periodically collects the position, temperature, humidity and noise of current environments and writes to a file. The client will send the file to a server when the file size is larger than the threshold set by the user. After uploading, the client returns the time taken by the uploading operation.

In following experiments, we choose android mobile phones as experimental devices for convenience, but PANBED is not confined to them, and those who support Wi-Fi and TCP/IP protocols can all be supported by PANBED.

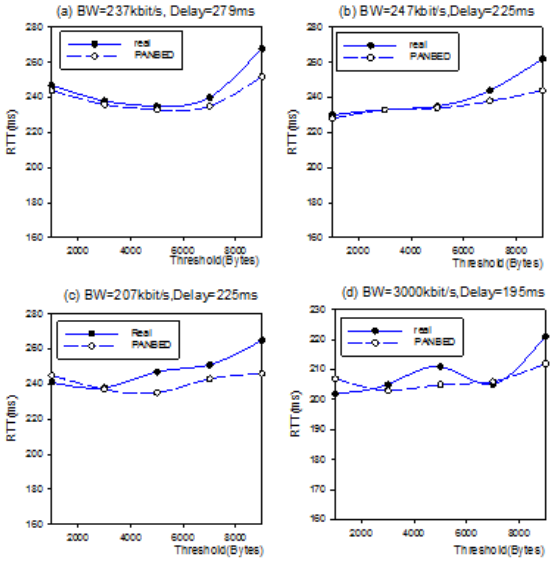

Fig. 7. Comparing of the results between PANBED and the real environment in four different network conditions which shows the values of RTT in PANBED are very similar with the real environment, and the difference is related to the dynamics of the network in some extent.

Realism of PANBED. Firstly we validate the realism of PANBED. We run the client on the android phone1 and run the server respectively on a remote PC and on the PC1 shown in Fig. 6. Then we measure the time taken by the uploading operation in two situation. In order to ensure the realism, we measure the network link between the android phone1 and the remote PC in real time with iperf [36] before the uploading, then emulate a same network link on the PANBED. The comparing results of PANBED and real situation in four different network conditions are shown in Fig. 7(a) - Fig. 7(d). We find that the experiment results on PANBED are essentially in agreement with that in the real environment and it proves the realism of PANBED.

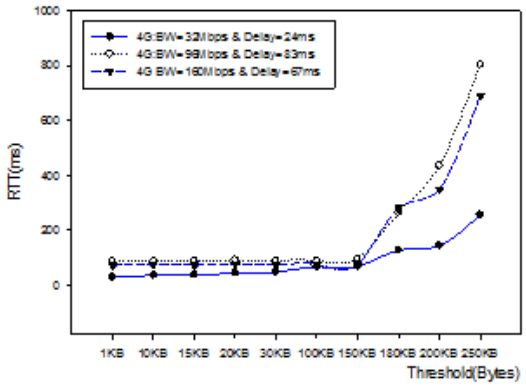

Fig. 8. RTT at Different Network and Different Threshold which shows the PANBED could simulate different network scene effectively.

Effectiveness of PANBED. To resolve the problems described in the use case and verify the effectiveness of PANBED, we emulate three different network links on PANBED and measure the uploading time at different threshold: (1) 4G network at 32Mbps bandwidth and 24ms delay; (2) 4G network at 96Mbps bandwidth and 83ms delay; (3) 4G network at 160Mbps bandwidth and 67ms delay. The results are shown in Fig. 8. We find, for the general mobile network, when the file size is less than 150K Bytes, the time spent on uploading file is relatively stable, but when the file size exceeds 180K Bytes, the time taken by uploading file will grow at speed of two times, three times, and even 4 times. It proves that PANBED can effectively solve the similar experimental requirements described in the use case.

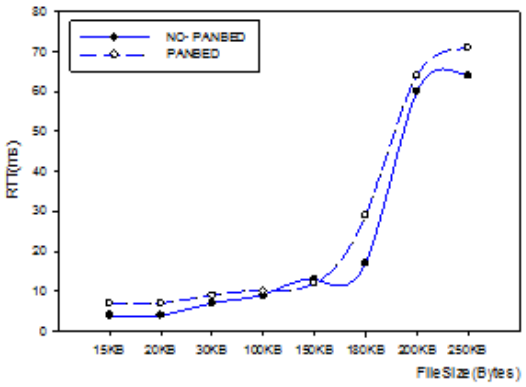

Fig. 9. Downloading time of android phone2 in two situations which shows the impact of doing experiment in the PAN is slight to other non-experimental devices.

Impact to other functions of PAN. The impact of PANBED to other non-experimental devices is very slight. We measure the time consumed by the android phone2 (shown in Fig. 6) which downloads files in case of not running PANBED and running PANBED on the router. When running PANBED, android phone1 and PC1 are selected as experimental devices, the link between them is set as BW=3Mbit/s and delay=19ms. The android phone1 sends data to PC1 continually. The results is shown in Fig. 9 which illustrates that the time taken by the android phone2 to download files is similar under two conditions and that proves it has little influence for other non-experiment devices to run PANBED on the router.

Fig. 10 displays the impact of PANBED on the memory and CPU of the router in three cases: (i) PANBED is not installed on the router; (ii) PANBED is installed but no testing task; (iii) PANBED is installed and works. The network scene is set as BW = 56 Kbit/s, Delay = 10ms, Loss = 0.10. The results are shown in Fig. 10 which prove that installing PANBED has little influence on the router and the PANBED is light weighted.

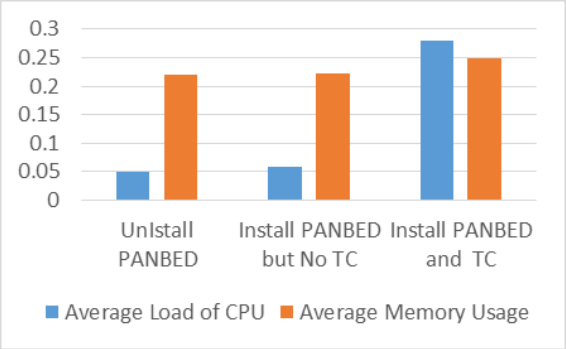

Fig. 10. Performance impact of PANBED to the router which shows the PANBED is a lightweight function embedded into the OpenWrt and its impact to the router is acceptable.

6. Conclusion

The paper describes the design and implementation of the prototype of PANBED, which build a small-scale personal testbed for users utilizing devices in their own personal area networks (PANs). The experiment results show that PANBED allows users to set up different network scenes to test applications easily using a home router, PCs, mobile phones and other devices. PANBED is light weighted with a size less than 16 KB and it has little impact to other functions of a PAN. The experiment results also prove the realism, effectiveness, flexibility and convenience of PANBED. PANBED can be used as a supplement to some existing testbeds and enable users to assess small applications at their convenience using diverse, personal, latest and low cost devices around them.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China (Grant No.61261019), the Inner Mongolia Autonomous Region Natural Science Foundation

(Grant No.113113), the Program of Higher-level Talents of Inner Mongolia University, the Inner Mongolia Autonomous Region Natural Science Foundation (Grant No.2012MS0930), and the Inner Mongolia Autonomous Region Higher Education Institutions Scientific Research Project (Grant No.NJZY12032).

References

1. Network traffic control in openwrt. https://wiki.OpenWrt.org/doc/howto/packet.scheduler/packet.scheduler, accessed June, 2017
2. Openwrt network interfaces. http://wiki.openwrt.org/OpenWrtDocs/NetworkInterfaces?action=attachFile&do=get&target=ASUS-Internals-default-sm.png, accessed January, 2017
3. Openwrt supported devices. https://wiki.OpenWrt.org/toh/start, accessed June, 2017
4. Overlay network. http://en.wikipedia.org/wiki/Overlay_network, accessed June, 2017
5. Web server configuration (uhttpd). https://wiki.OpenWrt.org/doc/uci/uhttpd, accessed June, 2017
6. Wireless information network laboratory. http://www.winlab.rutgers.edu, accessed June, 2017
7. Andersen, D., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient overlay networks. ACM SIGCOMM Computer Communication Review 32(1), 66–66 (2002)
8. Bajaj, L., Takai, M., Ahuja, R., Tang, K., Bagrodia, R., Gerla, M.: Glomosim: A scalable network simulation environment. UCLA computer science department technical report 990027(1999), 213 (1999)
9. Berman, M., Chase, J.S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R., Seskar, I.: Geni: A federated testbed for innovative network experiments. Computer Networks 61, 5–23 (2014)
10. Brown, M.A.: Traffic control howto. Guide to IP Layer Network (2006)
11. Chillara, V.K., Liu, Y.H., Wang, B., Ba, A., Vidojkovic, M., Philips, K., de Groot, H., Staszewski, R.B.: 9.8 an 860μw 2.1-to-2.7 ghz all-digital pll-based frequency modulator with a dtc-assisted snapshot tdc for wpan (bluetooth smart and zigbee) applications. In: Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International. pp. 172–173. IEEE (2014)
12. Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., Bowman, M.: Planetlab: an overlay testbed for broad-coverage services. ACM SIGCOMM Computer Communication Review 33(3), 3–12 (2003)
13. Drake, J.D., Brian, J.M.: System and method for enabling a viewable pan id over a wireless personal area network (Jan 12 2016), uS Patent 9,237,511
14. Ertin, E., Arora, A., Ramnath, R., Naik, V., Bapat, S., Kulathumani, V., Sridharan, M., Zhang, H., Cao, H., Nesterenko, M.: Kansei: a testbed for sensing at scale. In: Proceedings of the 5th international conference on Information processing in sensor networks. pp. 399–406. ACM (2006)
15. Fainelli, F.: The openwrt embedded development framework. In: Proceedings of the Free and Open Source Software Developers European Meeting (2008)
16. Fall, K.: Network emulation in the vint/ns simulator. In: Computers and Communications, 1999. Proceedings. IEEE International Symposium on. pp. 244–250. IEEE (1999)
17. Gutirrez J, Villa-Medina J F, N.G.A.e.a.: Automated irrigation system using a wireless sensor network and gprs module. IEEE transactions on instrumentation and measurement 63(1), 166–176 (2014)

18. Hemminger, S., et al.: Network emulation with netem. In: Linux conf au. pp. 18–23 (2005)
19. Issariyakul, T., Hossain, E.: Introduction to network simulator NS2. Springer Science & Business Media (2011)
20. Katsaounis, G., Tsilomitrou, O., Manesis, S.: A wireless sensors and controllers network in automation a laboratory-scale implementation for students training. In: Control and Automation (MED), 2014 22nd Mediterranean Conference of. pp. 1067–1073. IEEE (2014)
21. Kciuk, M.: Openwrt operating system based controllers for mobile robot and building automation system students projects realization. In: Research and Education in Mechatronics (REM), 2014 15th International Workshop on. pp. 1–4. IEEE (2014)
22. Kim, C.G., Kim, K.J.: Implementation of a cost-effective home lighting control system on embedded linux with openwrt. Personal and ubiquitous computing 18(3), 535–542 (2014)
23. Kim, D.H., Bae, K.: System level approach for low energy consumption in wireless personal area networks. In: Consumer Electronics (ICCE), 2016 IEEE International Conference on. pp. 520–521. IEEE (2016)
24. Lee, W.J., Shin, J.W., Lee, H.Y., Chung, M.Y.: Testbed implementation for routing wlan traffic in software defined wireless mesh network. In: Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on. pp. 1052–1055. IEEE (2016)
25. Levis, P., Lee, N., Welsh, M., Culler, D.: Tossim: Accurate and scalable simulation of entire tinyos applications. In: Proceedings of the 1st international conference on Embedded networked sensor systems. pp. 126–137. ACM (2003)
26. Noh, J.Y., Kim, M.K., Yim, C.H., Han, K.S., Choi, K.H., Yu, J.H., Kim, M.S.: Packet transmission system based on wireless personal area network and method thereof (Oct 7 2014), uS Patent 8,855,090
27. Palazzi, C.E., Brunati, M., Roccetti, M.: An openwrt solution for future wireless homes. In: Multimedia and Expo (ICME), 2010 IEEE International Conference on. pp. 1701–1706. IEEE (2010)
28. Park, S., Savvides, A., Srivastava, M.B.: Sensorsim: A simulation framework for sensor networks. In: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems. pp. 104–111. ACM (2000)
29. Paul, B., Marcombes, S., David, A., Struijk, L.N.A., Le Moullec, Y.: A context-aware user interface for wireless personal-area network assistive environments. Wireless Personal Communications 69(1), 427–447 (2013)
30. Polley, J., Blazakis, D., McGee, J., Rusk, D., Baras, J.S.: Atemu: a fine-grained sensor network simulator. In: Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on. pp. 145–152. IEEE (2004)
31. Reich, J., Misra, V., Rubenstein, D.: Roomba madnet: a mobile ad-hoc delay tolerant network testbed. ACM SIGMOBILE Mobile Computing and Communications Review 12(1), 68–70 (2008)
32. Riley, G.F., Henderson, T.R.: The ns-3 network simulator. Modeling and tools for network simulation pp. 15–34 (2010)
33. Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols. ACM SIGCOMM Computer Communication Review 27(1), 31–41 (1997)
34. Serrano, P., Bernardos, C.J., de La Oliva, A., Banchs, A., Soto, I., Zink, M.: Floornet: deployment and evaluation of a multihop wireless 802.11 testbed. EURASIP Journal on Wireless Communications and Networking 2010, 8 (2010)
35. Shrestha, B., Hossain, A.Z.E., Camorlinga, S.G., Krishnamoorthy, R., Niyato, D.: Method and system for allocation guaranteed time slots for efficient transmission of time-critical data in ieee 802.15. 4 wireless personal area networks (Mar 10 2015), uS Patent 8,976,763
36. Tirumala, A., Qin, F., Dugan, J., Ferguson, J., Gibbs, K.: Iperf: The tcp/udp bandwidth measurement tool. http://dast. nlanr. net/Projects (2005)

37. Titzer, B.L., Lee, D.K., Palsberg, J.: Avrora: Scalable sensor network simulation with precise timing. In: Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on. pp. 477–482. IEEE (2005)
38. Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P., Kostić, D., Chase, J., Becker, D.: Scalability and accuracy in a large-scale network emulator. ACM SIGOPS Operating Systems Review 36(SI), 271–284 (2002)
39. Varga, A.: Omnet++. Modeling and Tools for Network Simulation pp. 35–59 (2010)
40. Vatti, R.A., Gaikwad, A.N.: Throughput improvement of high density wireless personal area networks. In: Computational Intelligence and Communication Networks (CICN), 2014 International Conference on. pp. 506–509. IEEE (2014)
41. Wang, Y., Wang, Q., Zheng, G., Zeng, Z., Zheng, R., Zhang, Q.: Wicop: Engineering wifi temporal white-spaces for safe operations of wireless personal area networks in medical applications. IEEE transactions on mobile computing 13(5), 1145–1158 (2014)
42. Welsh, E., Fish, W., Frantz, J.P.: Gnomes: A testbed for low power heterogeneous wireless sensor networks. In: Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on. vol. 4, pp. IV–IV. IEEE (2003)
43. Werner-Allen, G., Swieskowski, P., Welsh, M.: Motelab: A wireless sensor network testbed. In: Proceedings of the 4th international symposium on Information processing in sensor networks. p. 68. IEEE Press (2005)
44. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. ACM SIGOPS Operating Systems Review 36(SI), 255–270 (2002)
45. Wu, H., Luo, Q., Zheng, P., Ni, L.M.: Vmnet: Realistic emulation of wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems 18(2), 277–288 (2007)
46. Zhang, K.: The Design and Implementation of An OpenWrtbased PPPoE Traffic Control System. Ph.D. thesis, Nankai University,Chian (2014)
47. Zhang, W., Han, S., He, H., Chen, H.: Network-aware virtual machine migration in an overcommitted cloud. Future Generation Computer Systems (2016)
48. Zhang, W., Li, X., Xiong, N., Vasilakos, A.V.: Android platform-based individual privacy information protection system. Personal and Ubiquitous Computing 20(6), 875–884 (2016)
49. Zheng, G., Hua, C., Zheng, R., Wang, Q.: Toward robust relay placement in 60 ghz mmwave wireless personal area networks with directional antenna. IEEE Transactions on Mobile Computing 15(3), 762–773 (2016)
50. Zhu, Y.H., Chi, K., Tian, X., Leung, V.C.: Network coding-based reliable ipv6 packet delivery over ieee 802.15. 4 wireless personal area networks. IEEE Transactions on Vehicular Technology 65(4), 2219–2230 (2016)

Qiaozhi Xu received the B.S. degrees in computer science and techonology from Inner Mongolia Normal University of China in 2000 and the M.S degree from Nanjing Normal University of China in 2005.She joined Inner Mongolia Normal University of China since 2005. She is currently working towards her Ph.D. in Inner Mongolia University since 2013. Her research interests include network testbed,computer network and cloud computing.

Junxing Zhang is a professor in the College of Computer Science at the Inner Mongolia University. He is also the Director of the Inner Mongolia Key Laboratory of Wireless Networking and Mobile Computing. He received a B.S. degree in Computer Engineering from the Beijing University of Posts and Telecommunications, a M.S. degree in Computer Science from the Colorado State University, and a Ph.D. degree from the University

of Utah. His research interests include network measurement and modeling, mobile and wireless networking, network security and verification, etc. Prof. Zhang was awarded the title of "Grassland Talent" by the government of the Inner Mongolia Autonomous Region in 2010. He has published over 40 papers in various internationally recognized journals and conferences, and led several national and provincial research projects. He also served as a peer reviewer for several international journals and conferences, such as IEEE Transactions on Mobile Computing, Wireless Networks, and ICNP.

Received: December 30, 2016; Accepted: August 8, 2017.