CSCE 664 Wireless and Mobile Systems Project Paper Report

Yukun Zeng, Bingqian Jiang
Department of Computer Science and Engineering
Texas A&M University
College Station, TX 77840
Email: {yzeng, jiangbingqianj}@tamu.edu

***Abstract*—We consider the problem of scaling mobile cloud benchmark on pay-as-you-go cloud computing platforms like AWS and Google Cloud. The intuition behind such an idea is that in scalability test of mobile cloud platforms, a large cluster of physical mobile devices are hard to manage and impractical in terms of infrastructure cost. In this paper, we propose to virtualize mobile clouds by leveraging on-demand public cloud to avoid unnecessary cost. More specifically, we directly virtualize mobile devices on top of Xen in stead of doing nested virtualization on the user-level OS commonly offered by cloud providers. The major advantage of such a direct virtualization approach is that it could eradicate the overhead of nested virtualization used in other virtualization solutions like Ravello Systems. However, one drawback of virtualizing mobile cloud for test purpose is that the network connection (virtual I/O in this case) between different virtual devices is totally different from real world scenario where mobile devices are connected through wireless networks, typically WLAN. To obtain a more realistic virtualized environment, we further introduce a virtual router that sits on top of the virtual I/O and is responsible for forwarding, dropping, retransmitting data packets to emulate the wireless network behavior on virtual I/O. Through preliminary experiments, we have show significant differences between virtualized and actual wireless network and proved the necessity of the virtual router for more realistic mobile cloud virtualization.**

***Index Terms*—Virtual Router, Wireless Network Emulation, Virtualized Networking, Virtualization, Mobile Cloud**

I. Introduction

A. Mobile Cloud

With the development of computing capability, we can now achieve performance simular to personal computer on a portable mobile device. This on the one hand eased our life by providing more complex and useful applications on mobile devices while it also created some dilemmas where massive computing resources fitted into phones, smart bands, etc are left idle and wasted most of the time. People have proposed many mobile cloud solutions that leverage the idle computing resource on those devices and use them to fill in high computing resource demand elsewhere. A lot of efforts have been put into mobile cloud and people have developed different platforms to solve different problems by leveraging idle mobile computing resources, we refer readers to [1] [2] [3] for more information about the emerging mobile cloud computing trend. In this paper, we will focus on a mobile cloud platform called Mobile Storm [4], which is a state-of-the-art platform for distributed stream processing and it is a previous work by LENSS lab.

B. Scalibility Benchmark

Naturally we can think of scalability as a important metric in evaluation of such platforms, however it's unpractical to probe the platform scalability using dedicated mobile device clusters as it's too expensive and even harder to manage than server clusters. To solve such a problem, we proposed to benchmark those platforms by leveraging the on-demand computing resources provided by state-of-the-art cloud computing platforms like AWS and Google Cloud, i.e., we could build mobile cloud by virtualizing any amount of devices based on our demand thereby testing the platform scalability. This approach does not only fit into the benchmarking of mobile cloud platforms but also is suitable for scaling different tests like probing the max load of an application server by adding more and more mobile devices as clients. However, new problems emerge if we are proceeding this way. In most common scenarios of mobile cloud, devices are usually connected through wireless networks, the backend can be either one server or a cluster and those servers are usually connected by wired networks. A typical example of the Mobile Storm cluster can be found in Fig. 1 where we have multiple mobile devices connected to each other as well as the master server through wireless router. However, things get totally different if we tries to virtualize the Mobile Storm cluster in public clouds. As shown in Fig. 2, the virtual mobile devices communicates with each other using virtual I/O. Different virtual devices might be running on the same physical machine or different physical machine, thus the virtual I/O can be either I/O of certain machine or wired network between different machines. Anyhow, such connectivity between virtual devices differs a lot from the actual scenario and we have to find a way to emulate wireless network in the virtualized environment so that we can benchmark in a realistic and scalable fashion.

C. Existing Solutions

Though there is a huge trunk of research papers in the field of cloud computing, virtualization as well as mobile clouds, there are few works dedicated to scaling mobile cloud benchmark by virtualizing mobile devices on public clouds, most highly relevant systems are commercialized and

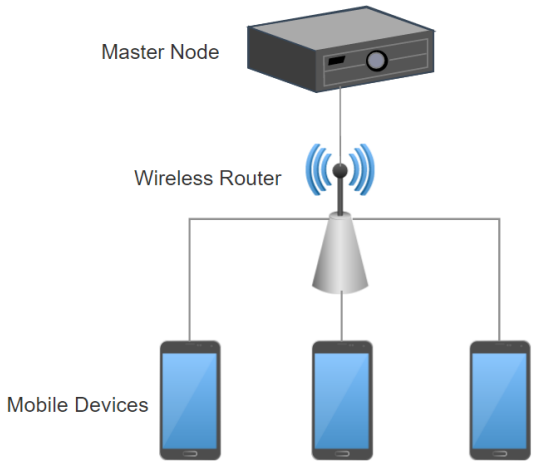


Fig. 1. Networking in Actual Mobile Storm Cluster

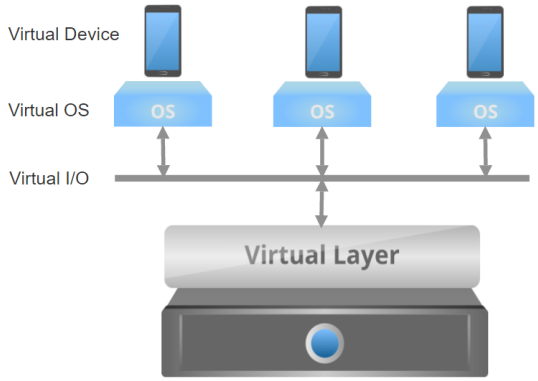


Fig. 2. Networking in Virtualized Mobile Storm

available now on market place of AWS. Ravello Systems and Genymotion are two commercial platforms that enable the scalability benchmark for mobile cloud. Ravello Systems enables us to run any virtual machine on any cloud without making any changes through adding a new hypervisor called HVX on top of VMs provided by public cloud platforms. Breaking down the stack of Ravello's solution, as shown in 3, we can see that we can further run hypervisor on VMs provided by AWS and then run any Virtual Machine on top of the hypervisor. The solutions provided by Genymotion incorporate two aspects, one is to emulate Android x86 OS on top of a host OS in a near-native manner (basically running Android x86 on VirtualBox with all hardware acceleration), the other is to emulate Android OS on demand in public cloud. Genymotion on Demand could similarly provide easy pay-as-you-go Android device virtualization while it's close source and its internal mechanism is rarely discussed. While both solutions can be used for scaling mobile cloud benchmark, none of them can restore the actual wireless network scenario thus cannot guarantee realistic benchmark.

D. Observation and Thoughts

As shown in Fig. 2, the network of virtual Mobile Storm cluster is based on virtual I/O which is totally different from

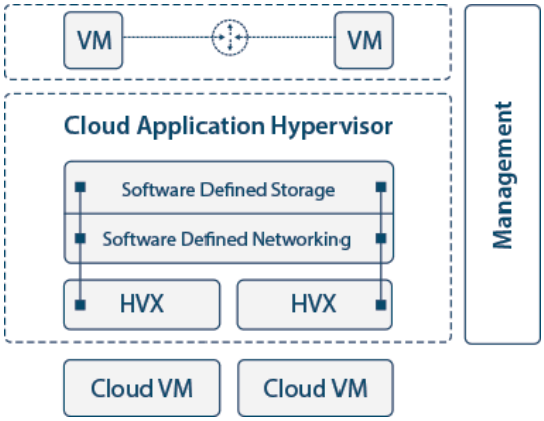


Fig. 3. Destack Ravello System Architecture

the wireless network behavior in actual scenario. We assume the virtual I/O is totally reliable. To acquire a wireless-alike environment in the virtualized cluster, our thought is that we might add a router (the router term might not be exactly accurate) on top of virtual I/O to control the traffic. The virtual router is responsible of forwarding data packets to target virtual devices like a wireless link that might generate packet loss, delay, retransmission, basically we wanna incorporate all features of a wireless link as data packets pass through the virtual router.

E. Contribution

We expect the contributions of this paper as follows:

- Propose a new approach of virtualizing mobile cloud directly on top of the Xen virtualization layer, i.e., be able to run android virtual devices cluster on Xen environment, this might also require efforts in virtualized networking as we need to figure out a way for building the interconnectivity between different virtual devices.
- Develop a virtual router sitting on top of the virtual I/O to forward data packet and control the transmission between different virtual devices, the final goal of the virtual router is to emulate a wireless network behavior on virtual I/O so that we obtain a realistic environment when virtualizing mobile devices cluster on public clouds.

II. Related Works

Current solutions for virtualizing mobile devices on cloud mainly include Genymotion [5] and Ravello Systems. They can both provide pretty good performance for virtualized devices and are highly scalable, but none of them takes common wireless communication paradigms between mobile devices into consideration. This is far from ideal as it is proved in [6] the wireless network differs a lot from wired network. Both wireless and wired network simulation or emulation have been studied thoroughly in the last decades [7] [8]. Network virtualization can be realized directly through virtualization systems like VMWare, VirtualBox or Xen, common networking provided by those virtualization systems include NAT, Host-only and bridged. Related works in network virtualization area

include [9] [10]. However, few researches are dedicated in wireless network emulation in virtualized environment, we need a physical wireless network card on the host machine to acquire wireless behavior for VMs, which is impractical in public cloud. Among all related works, we focus on the following researches as they are the most related ones to our topic:

A. Empower: a network emulator for wireline and wireless networks

Empower[7] is a distributed network emulation system. It is able to facilitate the emulation of either wireline or wireless networks. In the case when network topology is critical to the underlying network protocol, the emulator could provide specific mechanisms to emulate network topology. Empower can also generate user-defined network conditions and traffic dynamics at packet level. However, Empower is specialized in dealing with emulation for wire and wireless networks, which does nothing with cloud service computing. Therefore, we need to move forward for other applications which would fit our concern more.

B. HVX: Virtualizing the Cloud

HVX [11] is a virtualization platform that enables complete abstraction of underlying cloud infrastructure from the application virtual machines. HVX allows deployment of existing VMs into the cloud without any modifications, mobility between the clouds and easy duplication of the entire deployment. HVX can be deployed on almost any existing IaaS cloud. Each instance of the HVX deployment packs in a nested hypervisor, virtual hardware, network and storage configuration. Combined with image store and management APIs, the HVX can be used for the creation of a virtual cloud that utilizes existing cloud provider infrastructure as the hardware rather than using physical servers, switches and storage.However, HVX's virtualization still exhibits the problem that it is unable to simulate the real world wireless communication, which may have higher latency and packets loss rate than wired connection, which HVX adopt at its end. On the other hand, HVX could be referred as running virtualization on VM, which would significantly slow down the speed. Thus, we may need to propose a better solution to optimize discussed problems.

C. Genymotion on Demand

Genymotion [5] is a new cloud-based virtualization platform aimed at Android developers. Running on Amazon Web Services, Genymotion On Demand offers access to the full Android operating environment online. However, Genymotion on Demand is a closed source software whose source code is not published. Therefore, we currently could not be guaranteed whether performance of Genymotion on Demand could satisfy our requirements.

D. Xen

Xen [12] is an x86 virtual machine monitor which allows multiple commodity operating systems to share conventional hardware in a safe and resource managed fashion, but without sacrificing either performance or functionality. This is achieved by providing an idealized virtual machine abstraction to which operating systems such as Linux, BSD and Windows XP, can be ported with minimal effort. The virtualization approach taken by Xen is extremely efficient: it allows operating systems such as Linux and Windows XP to be hosted simultaneously only with a negligible performance overhead — at most a few percent compared with the unvirtualized case.

The advantages of virtualization with Xen comparing to other solutions are as follows:

- Independence on hardware and the support for different operating systems (both commercial and open source OSs).
- Neither performance nor security and functionalities are sacrificed in Xen to accomodate each other.
- Xen offers resource isolation and performance guarantees for VMs running on it without risk of DoS.
- It is now open sourced as Xen Project [13] in collaboration with Linux Foundation and enjoys technical support from a huge community.

Due to those advantages, Xen soon gains popularity with modern public cloud services like AWS. In this paper, we adopt XenServer[14], an open-source virtualization platform that has been accepted by current industry based on Xen Project hypervisor, as our environment for virtualizing Android devices to build a scalable mobile cloud. We will briefly discuss other available mobile cloud virtualization approaches as well as the XenServer setup we used in our paper in the following sections.

III. Proposed Solution

A. Virtualizing Mobile Devices in Cloud

Ravello systems provided solution to run any virtual machine on any cloud without changes to operating systems through nested virtualization, however there will be additional overhead in the 2nd layer of virtualization in terms of processor and memory speed, I/O, network, etc. Moreover, normal customers of public clouds are serviced directly from the virtualized OS level (i.e., an operating system like Windows or Linux is immediately installed per user's request) and they might not have access to nested virtualization as it's a low level setting in Xen which is not commonly exposed by cloud platforms for security considerations. Here in this paper we propose to virtualize Android system directly on top of Xen, which can greatly reduce overheads mentioned above and this is practical as long as Android image is supported in the cloud. More specifically, we use XenServer as the hypervisor for managing resources of the physical machine and install Android directly on VMs without any intermediate Linux or Windows system.

B. Wireless Emulator on Virtual I/O

As we have mentioned before, we can test the scalibility of a mobile cloud platform easily by leveraging virtualization techniques on public clouds, however, such an approach cannot provide a realistic wireless networking environment as the communication between different virtual devices goes through the virtual I/O, which is either based on system I/O or wired network between different host servers. To better emulate the actual wireless networked mobile cloud platform, we propose a wireless network emulator on top of virtual I/O to virtually control the delivery of data packet between different virtual devices to acquire a network transmission pattern similar to real wireless network system. As shown in Fig. 4, the virtual wireless network emulator lies on the virtual I/O layer and is responsible for coordinating the network transmission between different virtual devices. At the current stage, we expect the emulator to be an application on Android, it might exposes some ports to receive and forward data packets in the virtualized network, it works by dropping packets and do retransmission in a wireless manner.

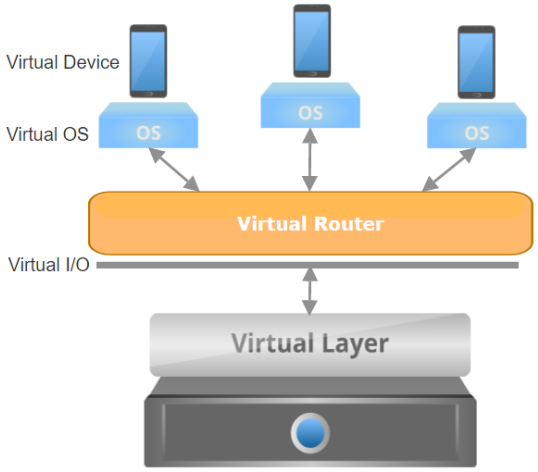


Fig. 4. Virtual Wireless Network Emulator

C. Expected Chanllenges

Our project is highly challenging in that it incorporates problems in both virtualization and networking, basically we need to find proper approach to achieve the following goals:

- Improving virtual device performance: virtual machines are typically much slower than physical machines with the same hardware and software configuration, even with the hardware speedup provided by newly introduced Intel VT-x or AMD-V CPU, which makes the benchmark harder. To reduce the virtualization overhead, we propose to virtualize mobile devices at the lowest level possible and we adopt Android for x86 to acquire better support for desktop CPUs like Intel Core series (oppsed to typical Android device CPU like ARM).
- Networking in virtualized environment: as we mentioned above, we need to virtualize mobile device cluster using XenServer, however, to the best of our knowledge so far, native network is not provided directly by the virtualized Android system on XenServer, we might have to deal with device driver and figure out a way to network virtual devices in either the same machine or LAN.
- Wireless network emulation: though a huge volume of researches have been put into wireless network emulation and simulation, here we expect many challenges since we are doing emulation on Android platform and the whole system lies in a virtualized environment, the final implementation of wireless emulation should be a totally different story comparing to previous work.

IV. Experimental Setup

To validate our argument that the virtualized networking is totally different from actual wireless environment applied in mobile cloud, we did a series of preliminary experiments to capture the characteristics of virtual network and actual wireless network. In this section we present the setup of our virtualized environment as well as methodologies we adopted in preliminary experiments.

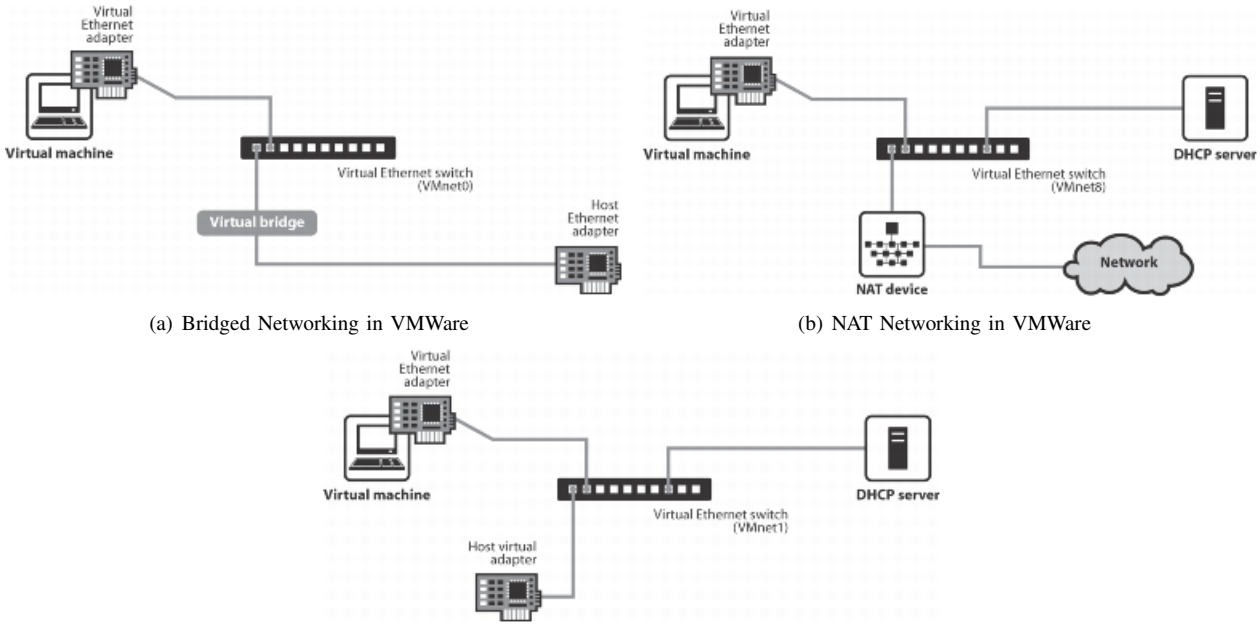
A. XenServer Virtualization Environment

1) Hardware Configuration: As we have mentioned before, Xen is independent on the underlying hardware and supports various operating systems, however one thing worth mention is that Xen does require a standard 64-bit x86-based system to support paravirtualized linux, a standard 64-bit x86-based system with Intel VT or AMD-V for running Windows. In our case, we failed to virtualize Android system on an old AMD server (no AMD-V support) but we succeeded in running both Linux and Android on two Intel Core 2 platform (with Intel VT support). Eventually we set up 3 Android VMs across two Intel-based PCs: a Dell OptiPlex 755 PC with Core 2 Duo CPU E8400@3.00GHz and 4GB RAM, running 1 Android instance; a HP DC7900 PC with the same CPU but 8GB RAM, running 2 Android instances. Be noticed that if not otherwise indicated, we use a standard Android instance of the following configurations defaultly:

- **Virtual Hardware:** 2 vCPUs and 2GB RAM, 8GB internal storage and a Virtual Network Interface.
- **System:** Android x86 5.1 Lollipop rc1 (64 bit).

The physical Android machines we adopted in the preliminary experiments are two Huawei Y538 smartphones, with Quad-core Snapdragon 210 processor@1.1GHz, 1GB RAM, running Android 5.1.1 (32 bit). The server (referred as master node in mobile cloud) we used is a Y570 laptop with Core i7 CPU and 8GB RAM, running Windows 10.

2) Android x86 on XenServer: One important reason that we choose Android over iOS or other mobile OSs is that Android is an open-source OS with the support from Google and a huge community. However, Android is developed for mobile devices at the first place and it has certain requirements on hardware configurations. Though native Android system runs pretty fast on smartphones and tablets with mostly ARM-based CPUs, it's a nightmare to run it on desktop CPUs like Intel and AMD. In order to speed up Android on desktop



(a) Bridged Networking in VMWare

(b) NAT Networking in VMWare

(c) Host-only Networking in VMWare

Fig. 5. VMWare/VirtualBox Networking Mechanisms at a Glance

hardware, the community have ported Android to x86 platform by modifying the linux kernel and HALs (Hardware Abstraction Layer) of native Android, known as Android x86[15]. It is shown in some benchmarks that Android x86 outperforms native Android by over 100x with similar hardware configurations[16]. For the sake of Android VM performance, we instead adopted Android x86 and installed it on several VMs on XenServer for virtulizing mobile cloud.

3) Networking: Before introducing the network setup in our experiments, we briefly introduce the virtualized network paradigms in existing virtualization systems. The commonly used desktop virtualization solution, VMWare Workstation[17] and VirtualBox[18], adopted similar networking mechanisms and they provide 3 solutions for VM networking:

- **Bridged:** In bridged networking, a VM will be assigned unique identity on the same external network that the host machine lies in. Take VMWare Workstation as an example, this is achieved by bridging the Virtual Ethernet Switch to the physical Ethernet adapter on host as shown in Figure 5(a).
- **Network Address Translating (NAT):** VMs configured in NAT mode will form a internal LAN and they share the same network identity with the physical machine. In this case, VM will connect to the virtual Ethernet switch and be assigned an private net IP address by a virtual DHCP server. For access to external network, VM sends packets to the host and the packets will be forwarded after NAT is done. A simple demonstration of the NAT mechanism is shown in Figure 5(b).
- **Host-only:** Host-only mode provides a private network connection between the VM and the host using a similar virtual Ethernet switch, as shown in 5(c). The only difference between Host-only and NAT is that no NAT will be done on the host in Host-only mode.

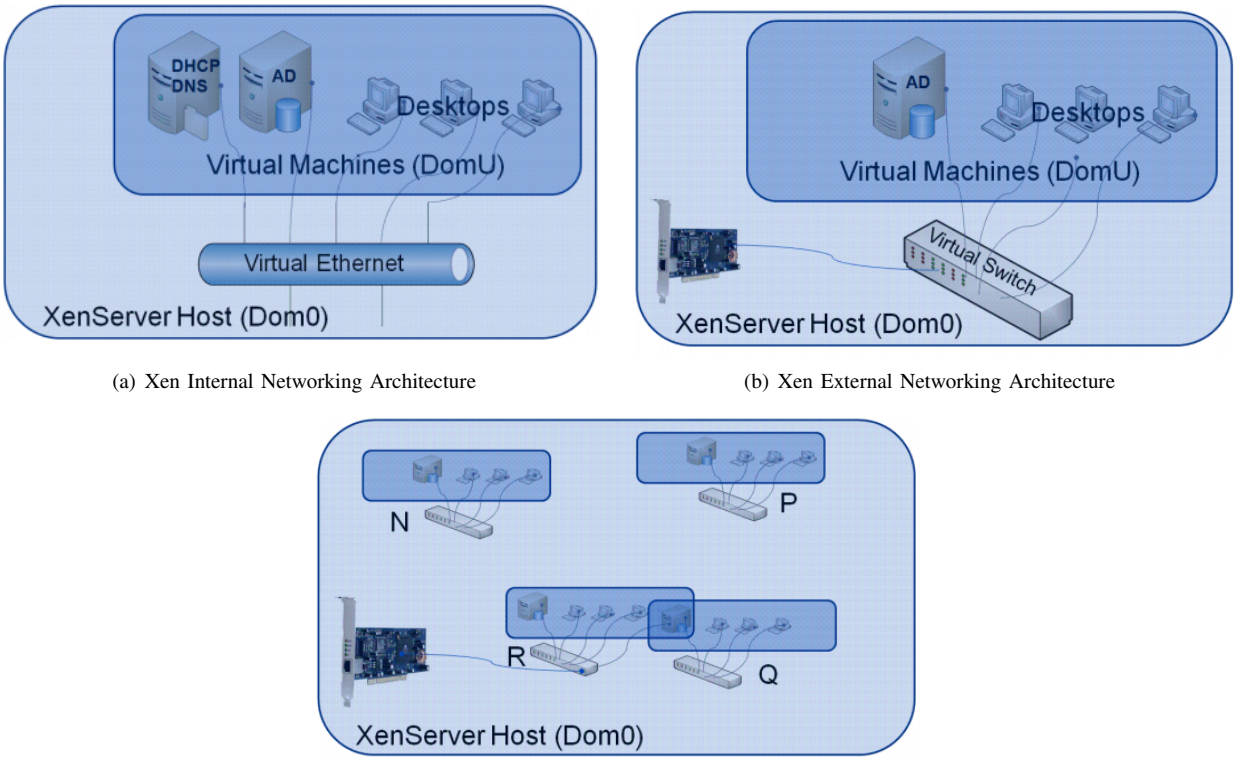
In Xen, every VM has a virtual NIC (Network Interface Controller), all of which are connected to the virtual network controlled by the XenServer host, also be noticed that the virtual network utilities provided by XenServer works like a physical Ethernet segment at layer 2 of the OSI, posing the nonrealistic networking between VMs as we discussed. Network solutions similar to Host-only and Bridging are also provided in Xen, however they are referred differencely as internal network and external network, here in our discussion we focus on those network modes provided in XenServer:

- **Internal Networking:** Internal Networking is sometimes also referred to as Single-Server Private Network, in which VIFs (virtual interface) of different VMs are connected by a virtual Ethernet and VMs do not have access to external network.
- **External Networking:** The only difference between external networking and internal networking is the accessibility to external network (i.e., under internal networking VMs have the same network accessibility as the host does). This is made available by adding a virtual switch between the virtual Ethernet and the PIF (physical interface).

An illustrative comparison between two network paradigms in Xen is shown in Figure 6.

As our goal in the preliminary experiments is to show the great differences between virtualized network and actual wireless connections between smartphones, we set up two Local Area Networks (LANs) respectively, one as a network connected by a wireless router with a subnet virtualized using XenServer across Android VMs, the other is a simple wireless LAN across master node and physical Android smartphones.

- **Virtualized Networking:** we have two xenservers and the master node connected to a wireless router through wired network, 3 virtual Android devices are virtualized



(a) Xen Internal Networking Architecture

(b) Xen External Networking Architecture

(c) Xen Routing between Internal Network and External Network

Fig. 6. Xen Networking Mechanisms at a Glance

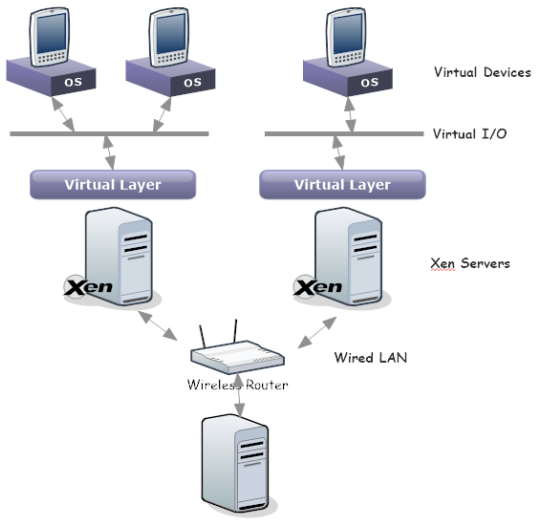


Fig. 7. Experimental Setup for Virtualized Mobile Cloud

8.

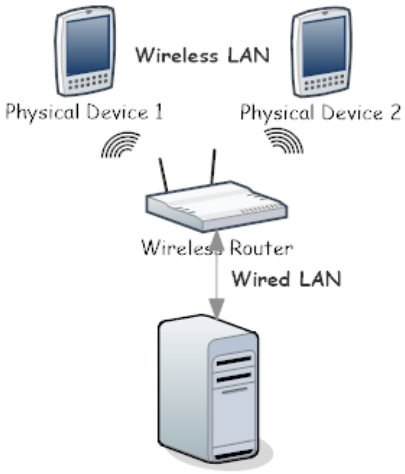


Fig. 8. Experimental Setup for Actual Wireless Mobile Cloud Paradigm

on xenservers thus they are connected to each other either by system I/O or by wired network, a figure that illustrates the system and network architecture in this case is shown in Figure 7.

- **Physical Networking:** in experiments of actual wireless networking, we have two Huawei devices connected to the wireless router through WiFi, the master node is again connected to router by wired network as showin in Figure

B. Benchmarking Network on Android

To characterize the features of both virtualized network and actual wireless network, we need some proper benchmarks to stress the network and probe important metrics in the data transmission process, in this section we first review the current existing solutions for benchmarking network on

Android platform and then propose the methodologies we applied in our experiments.

1) Literature Review: Though Android network benchmarking gains few attentions from the academia, there are a huge trunk of network tools in Google Play to characterize mainly the wireless network performance of Android. Aruba Utilities[19] includes a number of tools useful for characterizing and troubleshooting wireless LANs, some of which can work under any WLAN network while others are designed specifically for Aruba products. For our concern, Aruba Utilities has a Wi-Fi monitor showing the Wi-Fi environment such as current access point, and dynamic signal strength. It includes Android versions of iPerf[20] (a cross-platform tool for active measurements of the maximum achievable bandwidth on IP networks), Ping, DNS and mDNS, which all offer network test functionality. The measurements are written to a plain-text log file and various csv report files, which could be saved for later use. JPerf[21] is a graphical front end for the popular network testing tool iPerf. JPerf provides a quick test for WAN or LAN connection to determine the maximum network throughput. Test results will be automatically graphed and presented in a easy-to-read format. Moreover, it can also be used to detect packet loss, delay, jitter, and other common network problems. Other network tools and speed test APPs include WiFi Speed Test[22], which provides real-time monitoring of data transmission speed between specified server and client; he.net[23], developed based on iPerf and provides GUI for network diagnostics, gathering and visualizing network performance/status; etc. However, The above mentioned application are all based on iPerf, except for that they do introduce some new features like user-friendly GUI (jPerf), IP Calculator (he.net) and link coverage measurements (Aruba Utilities). A common disadvantage of them is that they by default call Android WifiManager API to set up the network connection and perform their tests only under Wi-Fi environment. In our virtualized environment or even in the actual scenario (as Ethernet is also supported in Android), the network for benchmarking doesn't necessarily has to be wireless. Therefore, in the wired network scenarios like virtualized mobile cloud, those APPs can simply crash when trying to setup the connection. To correctly benchmark both the performance of wired and wireless network, we propose to simply port iPerf on Android and execute it at the terminal without any GUI.

2) Categorizing Transmissions in Mobile Cloud: In mobile cloud as shown in 2, we have a master node who serves like a manager of the cluster and the mobile devices are usually referred as slave nodes. Consider centralized task scheduling and task allocation, the most widely used cloud management paradigm, in which the master node collects all heartbeats from slaves and allocates tasks to them based on the current load information of slaves, typical network transmissions in this management scheme can be categorized as follows:

- **Master-to-Slave:** a Master-to-Slave transmission happens when the master node transmits scheduling decisions and task input data to slaves through the wireless link. In such transmissions, though the reliability can be guaranteed by using TCP, the instability of wireless link might lead to high latencies in delivering scheduling decisions, low bandwidth would also result in higher response time as the input data cannot be transmitted timely.
- **Slave-to-Master:** typical Slave-to-Master transmission includes the regular heartbeat from slave to master node (which is sometimes referred to as status report), in some cases, the execution results might also be returned to master node through the wireless link. Late arrivals of heartbeat might lead to wrong scheduling decisions while delay of returning data lengthens the average response time.
- **Slave-to-Slave:** one thing in common among various distributed and cloud computin g platforms is the interaction between different slaves. The intercommunication between those slaves in the cluster can be transmissions for intermediate results, job/task migrations per scheduling requests, etc.

Apparently from the discussions above, the performance of mobile cloud will be greatly influenced by the network. However, in the virtualized mobile cloud, either the highly reliable wired network or virtual I/O between VMs on the same server can correctly reflect the properties of a realistic wireless network. In the next section, we empirically prove this argument and show different characteristics between virtualized and actual wireless network in both 3 types of transmissions mentioned above.

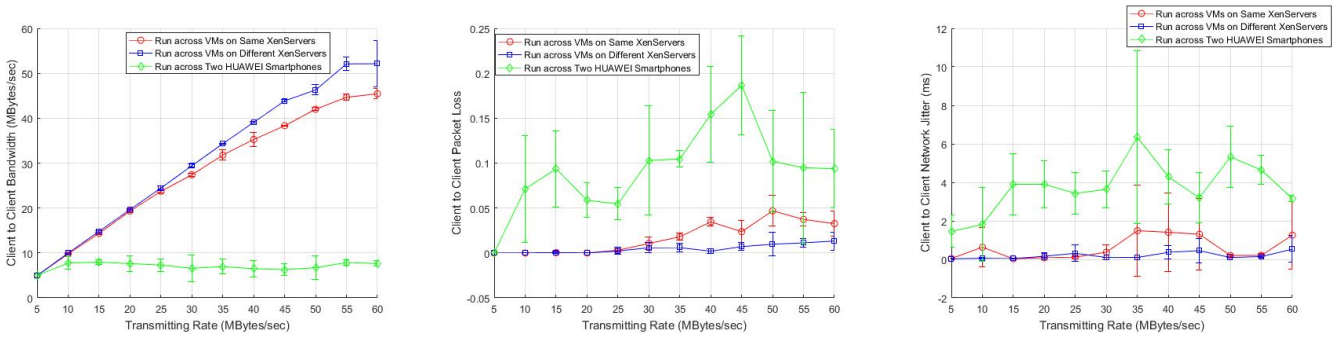
V. Preliminary Observations

A. Experiment Methodologies

Recall that the purpose of our experiment is to find out the differences between virtualized and actual wireless network in terms of common network metrics. As discussed in the last section, there are 3 different types of network traffics and we definitely want to characterize them under both virtualized environment and actual environment. We propose to use bandwidth, packet loss and jitter as metrics for describing the network since they might significantly influence the performance of mobile cloud as follows:

- **Bandwidth:** the bandwidth of a certain transmission category is important in the sense that it could precisely reflect the long-term performance and capacity of a network, which directly determines the capacity of mobile cloud platform. For example, in distributed stream processing cluster, the system throughput is upper bounded by the bandwidth of the wireless network.
- **Packet Loss:** in mobile clouds that adopt unreliable transmission protocols, packet losses of the network directly lead to unreliability of the platform while in other mobile clouds packet losses are the major contributor of network jitters.
- **Jitter:** jitters in the network might result in delays of scheduling decisions from the master node, underutilization of downstream computing resources as well as wrong scheduling decisions as mentioned before.

A common strategy to probe the network capacity and other characteristics is to gradually stress the network and collect metrics data for analysis. Following the physical and virtual



(a) Slave-to-Slave Bandwidth (b) Slave-to-Slave Packet Loss (c) Slave-to-Slave Jitter

Fig. 9. Network Characteristics Comparison for Slave-to-Slave Communication

environment settings proposed in section IV, for each transmission category we set the receiver as the iPerf server and the sender as the iPerf client and then stress the network by sending data at a transmitting rate ranging from 5 Mbps to 60 Mbps (5 Mbps as interval) in 10 seconds and record the corresponding metrics we obtain. We conduct experiments in both virtualized and wireless environments and repeat the experiments for 3 times in different time of a day. The final result we acquire is a tendency diagram (increasing transmitting rate) with error bars to mitigate the variance influences and observe fluctuations of network status under different networking scenarios.

B. Slave-to-Slave Network Characteristics

The metrics we measured under different traffic load for Slave-to-Slave transmissions are shown in Figure 9. Be noticed that this test differs from all other experiments in that we have three groups of data for comparison. The underlying reason is that for Slave-to-Slave transmission in virtualized environment, we also have two scenarios:

- Data transmissions happening between VMs across different XenServer.
- Data transmissions happening between VMs on the same XenServer.

From the bandwidth tendency shown in Figure 9(a), we can see that at the very beginning, the bandwidth of all three scenarios increases with the transmitting rate. However, immediately as the transmitting rate exceeds 5 Mbps the actual wireless network seems to be overloaded and the following bandwidth (no matter how much the transmitting rate increases) stay around 8 Mbps. For both other two kinds of network, their maximum bandwidth is much higher than the actual scenarios. Internal VM network and across-server VM network respectively yields maximum bandwidth around 45 Mbps and 52 Mbps. Furthermore, by analyzing jitters and packet losses, we can also get a basic understanding of the high unstability in wireless network comparing to its counterparts. For short, the wireless link in the actual scenario yields much lower bandwidth, produces a lot more jitters and packet losses, and is highly unstable.

C. Master-to-Slave Network Characteristics

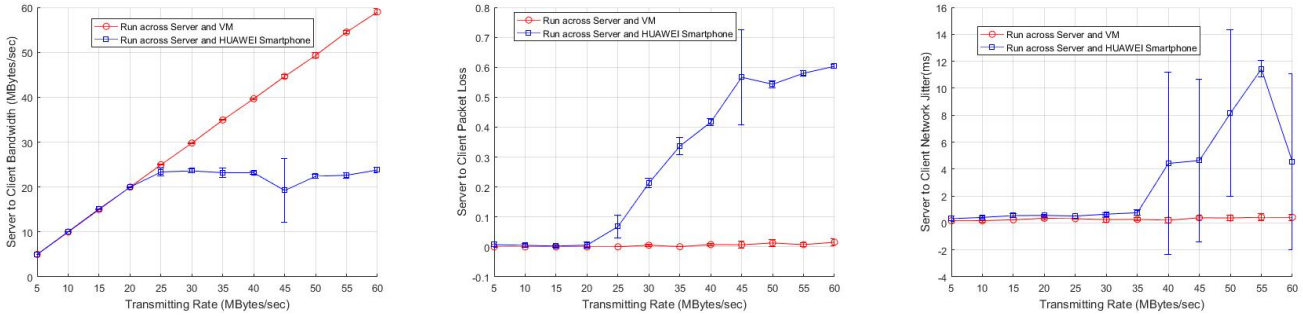
The metrics we measured under different traffic load for Master-to-Slave transmissions are shown in Figure 10. Apparently the figure shows a much better bandwidth for actual Master-to-Slave network, around 25 Mbps, this makes sense due to the fact that only one device is occupying the wireless channel in this scenario which means much less interferences, collisions and backoffs. However, we also noticed the increasingly high packet loss rate and jitters in this experiment are due to overloading of wireless link, while currently we cannot find a proper explanation for why both packet losses and jitters in this scenario exceed the Slave-to-Slave case. Again, the wireless link is highly unstable if we look at the error bars.

D. Slave-to-Master Network Characteristics

The metrics we measured under different traffic load for Slave-to-Master transmissions are shown in Figure 11. Through the measurement figures, we can see that the results look similar to the Slave-to-Slave case. The only one thing worth mention here is that the Slave-to-Master bandwidth lies in between the max bandwidth of Master-to-Slave and Slave-to-Slave. On the one hand, we don't have much interferences in this case, therefore we can transmit data faster than Slave-to-Slave. On the other hand, it's common for an actual wireless router to have lower uplink bandwidth than downlink, thus the network performance here is inferior to the Master-to-Server case.

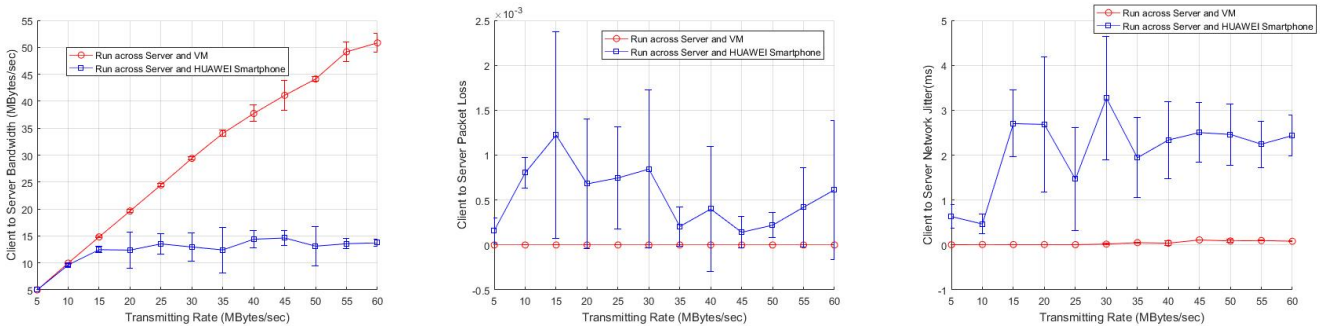
VI. Conclusion

So far in this paper, we have established two mobile cloud systems, one based on efficient direct virtualization while the other is an actual mobile cloud connected through wireless link. A detailed review and analysis of networking in virtualized environment are also provided. For network benchmarking, iPerf 2 is poted to Android platform and several tricks are applied before it executes successfully. The major contribution of our paper by now is an empirical analysis of network differences between virtualized and actual wireless scenario. Through a series of experiments, we have shown that virtualized environment is nonrealistic for testing mobile clouds since it enjoys much higher bandwidth and is really stable in terms of jitter and packet losses, comparing to a common wireless link.



(a) Master-to-Slave Bandwidth (b) Master-to-Slave Packet Loss (c) Master-to-Slave Jitter

Fig. 10. Network Characteristics Comparison for Master-to-Slave Communication



(a) Slave-to-Master Bandwidth (b) Slave-to-Master Packet Loss (c) Slave-to-Master Jitter

Fig. 11. Network Characteristics Comparison for Slave-to-Master Communication

VII. Future Work

A. Bridging Wireless to Ethernet

As we have already covered in the literature review of network benchmarking tools on Android, most of those APPs are based purely on WiFi connection due to the dominance of it on mobile devices. This exposes a major problem in our virtualization approach for scaling mobile cloud tests. Most APPs available on Android platforms are like those tools and Ethernet connection adopted in our virtualization approach is not commonly supported. Therefore, we propose to bridge the wireless network interface (wlan0) of virtualized Android system to the Ethernet interface (eth0) so that the wireless traffic can be automatically forwarded to Ethernet. In this way, we will be able to run almost all applications or mobile cloud platforms without any changes and the compatibility of our method will be greatly improved.

B. More Reliable Experiments

For compatibility considerations, the preliminary experiments are conducted based on iPerf 2.0.5, an obsolete version that is no longer maintained by developers. We discovered several problems in this version of iPerf like we cannot set transmitting rate in MBytes and the TCP-related utilities are limited. We expect to port newer iPerf 3 into experiments which incorporates more functionalities and could be used to capture more features of the network. Also, currently we are testing the network in an environment with many interferences, we expect a great probability that the uncommonly high packet losses and jitters in Figure 10 are caused by those interferences. Additionally, though repeated experiments are carried out, we are still looking for new methodologies that involves a lot more tests for recognizing the wireless network performance pattern.

C. Characterizing and Emulating Wireless Network

We expect a large amount of long-term and short-term experiments in the future so that we might be able to capture the bandwidth/jitters/packet losses distribution of wireless link and find a way to emulate it in virtualized environment.

References

[1] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[2] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[3] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future generation computer systems*, vol. 29, no. 1, pp. 84–106, 2013.

[4] Q. Ning, C.-A. Chen, R. Stoleru, and C. Chen, "Mobile storm: Distributed real-time stream processing for mobile clouds," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE, 2015, pp. 139–145.

[5] M. Wilson, "Genymotion On Demand is a cloud-based virtualized Android emulator for the enterprise," https://betanews.com/2017/02/27/genymotion-on-demand-android/, 2017, [Online; accessed 20-March-2017].

[6] S. Cen, P. C. Cosman, and G. M. Voelker, "End-to-end differentiation of congestion and wireless losses," *IEEE/ACM Transactions on networking*, vol. 11, no. 5, pp. 703–717, 2003.

[7] P. Zheng and L. 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*, vol. 3. IEEE, 2003, pp. 1933–1942.
[8] G. Judd and P. Steenkiste, "Using emulation to understand and improve wireless networks and applications," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 203–216.
[9] A. Menon, A. L. Cox, and W. Zwaenepoel, "Optimizing network virtualization in xen," in *USENIX annual technical conference*, no. LABOS-CONF-2006-003, 2006.
[10] N. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications magazine*, vol. 47, no. 7, 2009.
[11] Oracle, "HVX: High performance nested virtualization," https://www.ravellosystems.com/technology/nested-virtualization, 2015, [Online; accessed 20-March-2017].
[12] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 164–177.
[13] Xen, "The xen project, the powerful open source industry standard for virtualization." https://xenproject.org/.
[14] XenServer, "Xenserver | open source server virtualization," https://xenserver.org/.
[15] Android-x86, "Android-x86 – porting android to x86," http://www.android-x86.org/.
[16] Nuxeo, "How to speed up the android emulator by up to 400%," https://doc.nuxeo.com/blog/speeding-up-the-android-emulator/.
[17] VMWare, "Vmware workstation," http://www.vmware.com/products/workstation.html.
[18] Oracle, "Virtualbox," https://www.virtualbox.org/wiki/VirtualBox.
[19] A. Networks, "Aruba utilities," https://play.google.com/store/apps/details?id=com.arubanetworks.arubautilities.
[20] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: The tcp/udp bandwidth measurement tool," *htt p://dast. nlanr. net/Projects*, 2005.
[21] T. Brethour and K. Gibbs, "Jperf version 1.0 the iperf front end," *The Board of Trustees of the University of Illinois*, 2003.
[22] Z. Pallagi, "Wifi speed test for android guide," https://pzoleeblogen.wordpress.com/2013/11/26/wifi-speed-test-for-android-how-to/.
[23] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu, "iperf – the ultimate speed test tool for tcp, udp and sctp," https://iperf.fr/.