Software Defined Wireless Access for a Two-Tier Cloud System

Sina Monfared, Hadi Bannazadeh, Alberto Leon-Garcia
Department of Electrical and Computer Engineering
University of Toronto

{Sina.Monfared, Hadi.Bannazadeh, Alberto.Leong}@utoronto.ca

***Abstract*—In this paper we introduce an architecture for wireless access on two-tiered heterogeneous clouds. We first discuss the concept of Software Defined Infrastructure (SDI), and role of Software Defined Radio in SDI, and review relevant literature. Next we present the architecture design for a Software Defined Wireless Access module in detail and provide measurements for the implemented solution. While the architecture introduced is capable of handling protocols such as LTE and WiFi, GSM is used for in our implementation.**

I. Introduction

We present an architecture for creating a converged wireless access module that can be deployed in two-tier clouds. This wireless access network can implement various wireless protocols such as WiFi, LTE, GSM, HSPA, FM radio, Bluetooth, etc. Software Defined Radio (SDR) transceiver units have advanced to cover a wide range of frequencies, covering all such protocols [1] [2].

Our goal is to make the access network for the cloud as programmable as possible. The key to achieve this is virtualization of the networking resource. Hence our approach is to redefine the functionalities of the access network so that they can be implemented in an infrastructure that has been virtualized.

II. SOFTWARE DEFINED INFRASTRUCTURE

SDI is a new conceptual architecture that for supporting applications through virtualization and integrated management of converged heterogeneous resources within a multi-tiered cloud [3]. Its objective is to provide high level abstraction and programmability for both cloud applications and network functions. The type of resources in a heterogeneous cloud include computing resources, programmable hardware (FPGAs), and networking resources. SDI allows the combination of the concepts of Software Defined Networking (SDN), Cloud Computing, and Software Defined Radio (SDR) to introduce an environment where applications can be deployed quickly and with great flexibility on an infrastructure that includes heterogeneous resources.

In general, multi-tiered clouds include large remote data centers and smart edge nodes, the latter being closer to the end-user. Having such closer smart edges helps address issues such as latency and security. The smart edge may also be used to increase the bandwidth efficiency of content distribution, by caching packets of popular content.

The design of the control and management system SDI is crucial to providing programmability and flexibility in the platform. Conventionally, there has been separate modules for the control and management of the cloud and network resources. In SDI, however, we combine the two, resulting in a single control/management module. This enhances the overall platform to have higher performance and flexibility, as the centralized control module can plan resource scheduling and modifications more efficiently. This approach leads to reductions in operational expenses, as there will be automated control and management of the virtualized resources.

A. SDI Architecture

Figure 1 shows the components of the SDI Resource Management System (RMS). The RMS includes various Resource Controllers (RCs) to provide support for different types of resources. The RCs are all connected and managed by two higher level modules, SDI Manager and the Topology Manager.

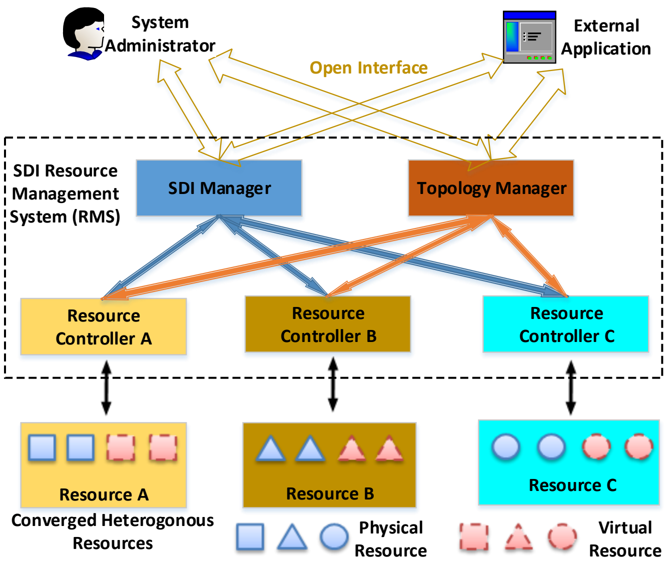

Fig. 1. SDI Components

The Topology Manager monitors the resources and stores particular parameters of interest, such as connection status, bandwidth, CPU utilization, etc. The SDI Manager takes the role of deploying applications, consulting with Topology Manager when needed. Both provide open interfaces for the user, which is either a developer or the application itself.

B. SAVI Testbed

The Canadian Smart Applications for Virtual Infrastructure (SAVI) Testbed includes an implementation of the SDI architecture [3]. The SAVI Testbed has a datacenter as its Core node, and it also includes Smart Edge nodes at seven universities. These nodes provide virtualized resources.

Figure 2 shows a high level representation of SAVI SDI RMS. The RMS uses Openstack (as the controller for the cloud computing resources) and an Openflow-based platform (to implement the control of networking resources). The two are governed by a higher level SDI Manager, code-named Janus.

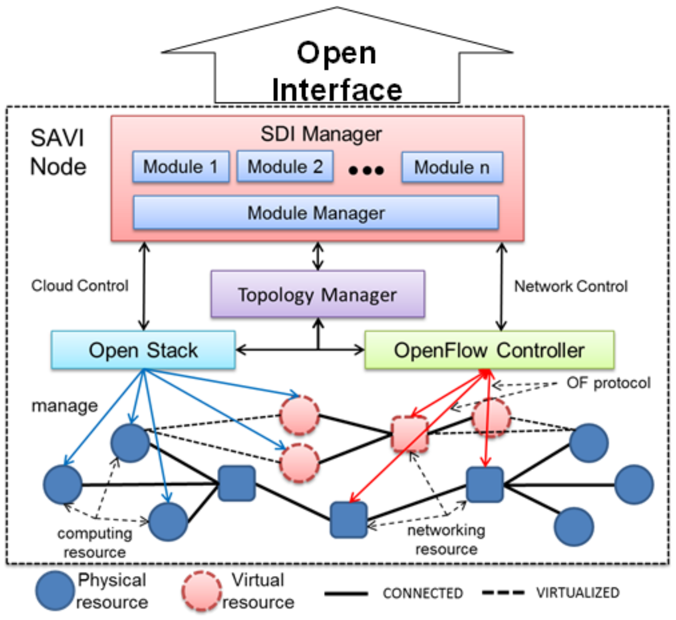

Fig. 2. SAVI Implementation of SDI RMS

The Topology Manager (code-named Whale) holds information on physical and virtual resources in SAVI as well as interconnections between them. It uses RESTful APIs of Cloud Controller and SDN Controller and other management protocols such as SNMP and IPMI to gather these information.

III. Related Work

The mobile provider market has been facing new challenges. From the mobile operators perspective, the demand for mobile Internet bandwidth keeps growing, resulting in growing amortized costs, while the profit per customer is marginalized, by decreasing prices as a result of competition. Hence, the mobile operators have to expand their Radio Access Networks (RAN) by investing in new hardware for their Base Stations while keeping the expansion cost minimal. Furthermore, they need to deploy new resources to meet new standards and remain competitive, and deploying new proprietary hardware is known to be a slow task [4].

These challenges have led to innovations using virtualization. Virtualized resources are not only faster to deploy and more scalable, but they introduce new perspectives such as energy optimization and carbon footprint reduced networking. RANs have had a vertical architecture, involving purpose built boxes for Base Stations. The Base Stations are not used all the time, with quite a high load variance between peak and surge times. While utilization patterns are mostly predictable (e.g. people go to work at downtown during the day and come back to suburbs at night), the conventional Base Stations need to be on around the clock and ready for the maximum load possible. Having the flexibility to adapt their operation as well having less active sites without a reduction in delivered performance are key strategies for mobile operators to reduce the costs.

This is where virtualization and centralized signal processing can come into play. The evolution of Base Stations has led to the virtualization of resources. As shown in Figure 3, there are at least two methods proposed for breaking down the Base Station into separable components. Both solutions have the Antenna, Power Amplifier (PA), Low Noise Amplifier (LNA), TX/RX (Transmitting and Receiving), and Digital front end at the distributed sites. These modules together are called a Remote Radio Unit (RRU), also known as Remote Radio Head (RRH). The primary task of the RRH is to deal with sending and receiving signals, and the aim is to reduce the computational capacity such that the signal processing can be centralized as much as possible.

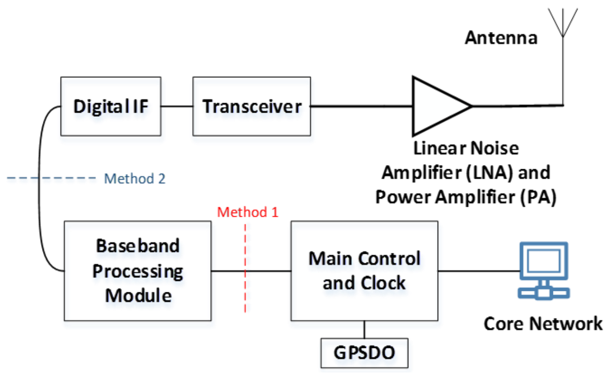

Fig. 3. Baseband Unit and Remote Radio Head

One method involves doing the baseband processing as part of RRH, while the other would leave it for the baseband unit (BBU). The advantage of doing baseband processing as part of RRH is that this reduces the bandwidth requirements between the RRH and BBU. The disadvantage is that upgrade costs will be higher for the distributed baseband processing solution.

Within the BBU, the rest of signal processing is performed, including synchronization with a GPS clock. The BBU may include other centralized functions performed on the edge of the Core network, and so may require high speed processors.

A significant factor in separating the Base Station into the BBU and RRH is the delay between the two. While the recent protocols have higher data rates, they also have more stringent time delay budgets. LTE requires the sum of the propagation, transporting, and processing delays between the BBU and RRH to be lower than 3ms [5]. Hence, LTE requires the BBU and RRH to be located within 40 km of each other, having a link capacity of at least 10Gbps between the two [6].

Potential benefits of using Software Defined Networks to manage cellular data networks have been discussed in [7] [8] and demonstrated in [9]. They include scalability, finer granularity in control, simplified routing, and rapid topology adaptation upon need. It has further been demonstrated that SDN based architectures can efficiently amalgamate various wireless protocols [10].

The SAVI Testbed provides an SDI environment for experimentation. SAVI can run centralized experiments with physical resources away from each, even with hundreds of miles of distance. Centralized management of the cloud and its network allow for programmability of the network and allocation of computing resources to meet delay requirements. In particular the VMs in Smart Edges can be located closer to users and include certain delay sensitive signal processing on the edge of the network. Finally, SAVI's feature of smart mobile edges can be exploited to provide wireless access for an area in case of an emergency situation. This is described in more detail in the section below.

IV. High Level Design

The overall architecture implements the wireless access as a modular SDR system. The functionalities of the wireless access are divided such that they can be realized in software and implemented across the cloud. The design relies on having a multi-tier cloud, with Smart Edges and Core datacenters. This is needed to divide the modules of the Base Station into those that need to located closer to user (and hence implemented in a distributed fashion) and those that can be centralized on a more computationally capable backbone datacenter (Core).

The first sub-module that the wireless signals are passed to is the physical layer TRX (Transceiver / Transmitter and Receiver). From a software perspective, this is a dumb module that translates the given digital input onto an electromagnetic signal meeting certain requirements. The main parameter to be specified for such module is the frequency range, while other parameters of interest can include output signal power, type of modulation, etc. For TRX, we use low power transmitters that meet the power efficiency and sustainability objectives outlined below.

The TRX is implemented along with signal amplifiers (Low Noise Amplifier and Power Amplifier), as well as Digital Intermediate Frequency (IF) module [4]. The two stages of amplifier are used such that one initially amplifies the signal without introducing a significant noise coefficient (LNA), and a second one increase the power of the signal and noise altogether (PA), preparing it for final transmission. The Digital IF module is in charge of first moving the signal to an intermediate frequency, and then again further down converting it to be used by the baseband module.

All three components mentioned so far (TRX, Amplifiers, and Digital IF) are implemented within a single component of the system, SDR unit. The SDR units are available off-the-shelf, and provide the layer 1 and 2 functionalities (physical layer and data link layer) for wireless communication.

The next module is the baseband signal processing module. This module performs Digital Signal Processing (DSP) on an already digitized and down converted signal outputted from the Digital IF. The DSP includes sampling, noise filtering, clock and data recovery, and decision circuits. We implemented this module within the distributed Base Station (on the Smart Edge close to the SDR unit). Implementing it in a centralized fashion (on backbone datacenters) would pose difficulties meeting the bandwidth and delay requirements.

The SDR Software on the Smart Edge (in charge of Baseband processing) reaches out to the backbone datacenters (Core) via the Wide Area Network, providing it with datagrams. At that point, the Core datacenters could perform functions such as session initiation, subscriber / network registry, etc.

Figure 4 shows the overall architecture of SAVI Wireless Access. SDR units function similar to the minimized Base Station, with limited processing capability. Their modules include Antenna, PA, LNA, TRX, and Digital IF. Wireless Access for SAVI would not only include Cellular service, but also WiFi, as most Software Defined Radio (SDR) units offer coverage for a wide range of radio spectrum.

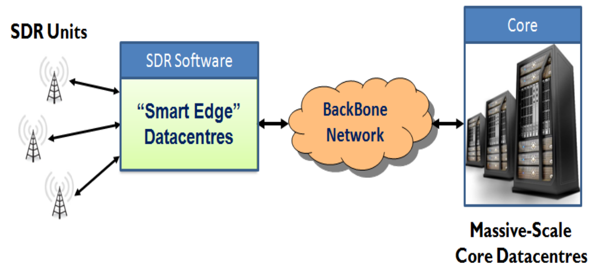

Fig. 4. High Level Architecture for Wireless Access

Some cellular and WiFi operations are delay sensitive (such as TX and RX), while some are not (such as IMS / packet routing operations). SAVI Smart Edges are used to implement the SDR Software so that the round trip time (RTT) is minimized for the delay sensitive operations. The SDR Software on the Smart Edge performs the Baseband signal processing to reduce the bandwidth requirements between the Smart Edge and SDR units. Also, there are constraints on the channel bandwidth between the Core and Smart Edges, depending on the application to be implemented.

V. Design Objectives

The design objectives includes reliability, guaranteed Quality of Service (QoS), platform power consumption, environmental sustainability, and flexibility of having mobile Base Stations. Most of the objectives can be met in the current SAVI Testbed. However, some require a bigger infrastructure.

By reliability, we mean reliable delivery of mobile services, including LTE, 3G, GSM, and WiFi. This is reflected in percentage of successful calls, maximum duration of calls without interruption, maximum number of calls per SDR unit, etc.

By using low power transmitters that cover smaller areas, we can make our distributed Base Stations more power efficient and environmentally sustainable. Of course, this increases the total number of Base Stations needed for a wide area, but the centralized resources can be reallocated to the busier Base Stations, and the VMs within the less busy Base Stations can be downsized and/or migrated.

Finally, having mobile smart edges, we may provide mobile Base Stations for an area facing a disaster. We may leave the centralized functionalities to the core network, so long as we ensure we provide sufficient bandwidth between the smart edges and the core datacenter. Also, it would be useful to have automated orchestration to deploy resources on the fly. SAVI has a project dedicated to orchestration using overlay networks, namely Virtual Network Overlay (ViNO). Using ViNO, we could quickly deploy predefined topologies.

VI. Detailed Design

As a proof of concept, we implemented the proposed architecture for GSM cellular service on the SAVI testbed. The Core node and the Smart Edge were located in the University of Toronto.

For the SDR Unit, we used Ettus USRP-N210. This device is programmed using its Ethernet interface [11]. There are two ways to provide synchronization / discipline clock for this device, one using an internal GPSDO (GPS Disciplined Oscillator) kit, the other inputting the clock using SMA interface. We used the former as we implemented the Base Station with a single SDR. Of course, we could have implemented several SDR Units within on Base Stations, should we wish to cover several protocols / ranges of frequencies at the same time, and several SDR units input a single GSPDO signal, using the SMA interfaces.

We used OpenBTS (Open Base Transceiver Station, developed by Range Networks [12]) for the SDR Software. OpenBTS is designed as wireless access point software dedicated for GSM. It defines mobile phones as SIP (Session Initiation Protocol) end users, and implements VOIP (Voice Over IP). OpenBTS is open source, and that permitted us to modify certain parts of the code for it to meet our architectural requirements.

The physical and data link layer components of the design are demonstrated in Figure 5. The main modules are Ettus USRP-N210 (for SDR unit) and OpenBTS (for SDR Software).

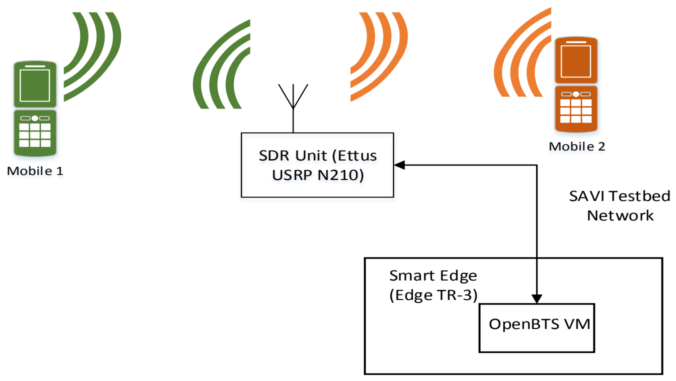

Fig. 5. Layer 1 and 2 Components

For GSM, one of the centralized functions implemented on Core is the SIP Soft switch. Soft switch can be thought of as a telephone switch, matching phones IMSI (International Mobile Subscriber Identity) with Extension (i.e. phone number) and setting up calls using SIP protocol. IMSI is the unique ID that are on phone and get registered on the Base Station. IMSI can be thought of as a MAC Address for cellular phones, except it is an identifier of the phone on the radio network than over the Internet. For the Soft switch, we chose Asterisk real-time [13], a modified version of Asterisk, with faster database read/write capabilities. The default port for SIP communication between the Soft switch and OpenBTS is UDP port #5060.

Two other centralized functions of OpenBTS are Smqueue (store-and-forward text messaging server) and Subscriber Registry (SIPAuthServer). Smqueue enables Short Message Service (SMS) by storing the message once sent and forwarding it once the destination becomes available. Subscriber Registry is the SIP endpoint registration as well as SIP Authorization. This is needed to reflect location changes (pairing IMSIâĂŹies with the IP Address of the Smart Edge VM that the phone is registered on). Smqueue uses UDP port #5063 while uses UDP port #5064. These two are provided as part of the OpenBTS platform and do not require separate code. However, they do need some tuning as by default they are not configured to run on a VM separate than that of OpenBTS transceiver (i.e. the module that does baseband processing).

Each of the Software components mentioned so far has its own dedicated database. In particular, OpenBTS.db stores SDR Unit parameters such as GSM Frequency and IMSI registration RegEx (regular expression for IMSI of phones to be permitted to register). Furthermore, sipauthserver.db stores IP address and IMSI pairs and smqueue.db stores short messages waiting to be delivered. Finally, sqlite3.db is used by Real-Time Asterisk to store pairs of extension (phone number) and IMSI.

Figure 6 demonstrates the interactions between the OpenBTS components and Asterisk. Unlike the architecture introduced in the related work section, the GPS clock synchronization happens at the SDR Unit itself. This is due to the fact such local synchronization is more exact and less prone to RTT.

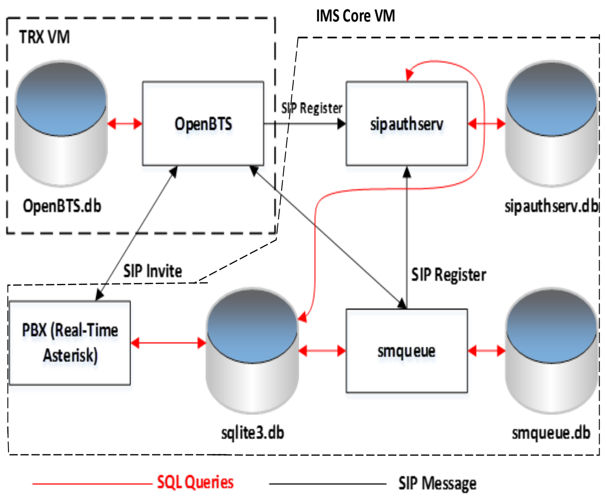

Fig. 6. Software Components

VII. MEASUREMENT AND VERIFICATION

A. Test Setup

As mentioned earlier, we needed to have at least two VMs, one running the SDR Software (OpenBTS) and the other running the centralized functionalities. The setup included 2 VMs, one named “TRX”, implemented on the Smart Edge (Edge TR-3) and the other named “IMS Core” implemented at the backbone datacenter (Core). Together, the TRX and SDR Unit represented the distributed Base Station, while IMS Core represented the centralized core. The specifications for the two VMs are listed in Table I. The Smart Edge and code nodes were connected through the backbone network, which includes several 1GE and 10GE Openflow switches.

OpenBTS was installed on both TRX and IMS Core. However, on TRX, only the transceiver components (performing baseband signal processing) were activated, and it was tuned to leave the rest of the tasks (communicating

TABLE I
VM Specification

VM Name	Operating System	RAM Size	# of VCPUs	Disk Size	Location
TRX	Ubuntu 12.04.2	8 GB	4	80 GB	TR-3
TRX	Ubuntu 12.04.2	8 GB	4	80 GB	TR-3

with the Soft switch, Smqueue, and Subscriber Registry, which were all installed on IMS Core) to the remote OpenBTS (on IMS Core). Hence the remote OpenBTS just takes the output of baseband signal processed and does the necessary communications with other components accordingly.

There were slight code modifications as well as database parameters tuning to have these components work with each other, as by default they are designed to all operate within the same machine (i.e. OpenBTS Transceiver and control modules are within the same machine by default).

We used two Android low level smartphones for the handsets to make the calls. The system was implemented and tested, and its operation was verified. The operation included echo test (provided by the OpenBTS software, a service where everything said is repeated back to give a sense of delay), Short Message Service (SMS / Text Message), and VOIP phone calls between the two cell phones.

B. Measurement

The key parameters of interest are the following:

- Call connection reliability
  - Percentage of successful call connections (i.e. not dropped)
  - Length of successful call
- Bandwidth usage for calling and SMS (in Bytes/Sec)
- Round Trip Time (RTT)
- Call Setup Time

In terms of successful connecting of calls, we found the system to complete 93% of calls (over 56 test calls, only 4 were unsuccessful, in one case getting a busy signal, and in 3 cases the system simply dropping the call without connecting at all). However, once a call connection was made, we found that the call would not disconnect due to network issues. In particular, we recorded two 24 hour long phone conversations. For that, we used two computers playing two different 1 hour long recorded calls repeatedly into earphones, which were located close to handsets.

The measurements for total bandwidth usage in IMS Core are reflected in Figure 8. This measurement is for a total of 14 hours of call, using various ranges of voice and music frequency. Each sample reflects the average traffic rate over a 2 second window. We used bwm-ng and iftop programs to record and compare these data. The averages for the total, outbound, and inbound traffic are

8705.953 Byte/s, 8774.400 Byte/s, and 17480.35 Byte/s. The standard deviations for the total, output, and input traffic are 49.2, 87.75, and 114.12 Byte/s.

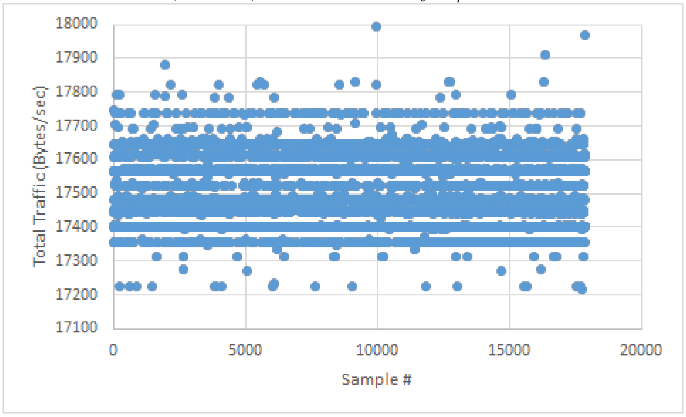

Fig. 7. Total Traffic of IMS Core over 10 Hours

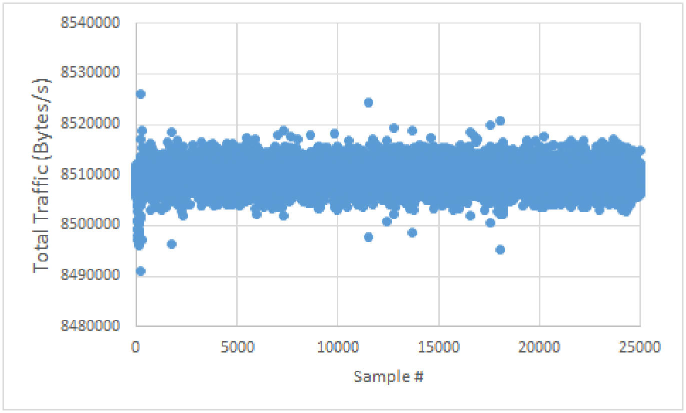

Fig. 8. Total Traffic of TRX over 14 Hours

For round trip time (RTT), the value of interest was that of TRX pinging the SDR unit, as that would show the minimum delay that a VOIP packet would face in case of echoing and that would hence represent the network delay. Performing 1000 ping tests while 2 handsets were in active call, we found the maximum, average, and standard deviation for RTT to be 1.692 ms, 1.417 ms, and 0.161 ms. These numbers are well below the 3ms delay budget required for LTE (of course these do not reflect processing and framing delays which should be taken into account). The other parameter that is not as significant but could be considered is the RTT between TRX and IMS Core, which could potentially affect the initial call setup time. For 1000 pings. We found the maximum, average, and standard deviation for that RTT to be 1.708ms, 1.227ms, and 0.122ms.

For call setup times, 50 call response times were measured. We defined the call setup time as the delay in between the time the one cellular device would dial and send the call request to server (i.e. call button is pressed) to the time the call shows up on the second device (i.e. the second phone rings). We observed that on average, Handset #1 experienced shorter delays than handset #2, which could relate to technical issues within the handset itself. Overall, the mean and standard deviation for call setup time were 6.6 and 4.3 seconds.

VIII. Conclusion

We have demonstrated how SDI, using the SAVI Testbed, can create wireless access networks in software. We made the case for smart edges for separating SDR functionalities that are delay sensitive and need to be closer to the user and those that can be centralized. We introduced an architecture that can be scaled up and down and deployed on the fly. As a proof of concept, we detailed an implementation of the proposed architecture for a GSM application and provided measurements for it, arguing that the architecture is also capable of implementing other wireless protocols such as LTE and WiFi which will be our focus for future work. More utilization of SDI capabilities including hand-over is another area that we would like to explore.

References

[1] F. Daneshgaran and M. Laddomada, "Transceiver front-end technology for software radio implementation of wideband satellite communication systems," *Wireless Personal Communications*, vol. 24, no. 2, pp. 99–121, 2003.

[2] S. Bang, C. Ahn, Y. Jin, S. Choi, J. Glossner, and S. Ahn, "Implementation of lte system on an sdr platform using cuda and uhd," *Analog Integrated Circuits and Signal Processing*, vol. 78, no. 3, pp. 599–610, 2014.

[3] J.-M. Kang, H. Bannazadeh, H. Rahimi, T. Lin, M. Faraji, and A. Leon-Garcia, "Software-defined infrastructure and the future central office," in *Communications Workshops (ICC), 2013 IEEE International Conference on.* IEEE, 2013, pp. 225–229.

[4] C. Mobile, "C-ran: the road towards green ran," *White Paper, ver*, vol. 2, 2011.

[5] B. Ullman. (2013) Designing an arm-based cloud ran cellular/wireless base station. [Online]. Available: http://www.embedded.com/design/connectivity/4425740/Designing-an-ARM-based-Cloud-RAN-cellular-wireless-base-station

[6] J. Davies. (2014) Mobile fronthaul optimized for cloud ran. [Online]. Available: http://www2.alcatel-lucent.com/techzine/mobile-fronthaul-optimized-cloud-ran

[7] J. Kempf, B. Johansson, S. Pettersson, H. Luning, and T. Nilsson, "Moving the mobile evolved packet core to the cloud," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on.* IEEE, 2012, pp. 784–791.

[8] M. Dayananda and J. Priyanka, "Managing software defined radio through cloud computing," in *Advanced Communication Control and Computing Technologies (ICACCCT), 2012 IEEE International Conference on.* IEEE, 2012, pp. 50–55.

[9] L. E. Li, Z. M. Mao, and J. Rexford, "Toward software-defined cellular networks," in *Software Defined Networking (EWSDN), 2012 European Workshop on.* IEEE, 2012, pp. 7–12.

[10] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *Network Operations and Management Symposium (NOMS), 2014 IEEE.* IEEE, 2014, pp. 1–9.

[11] E. Reseach. (2014) Usrp n210. [Online]. Available: https://www.ettus.com/product/details/UN210-KIT

[12] Range Networks Incorporated. (2014) Openbts. [Online]. Available: https://wush.net/trac/rangepublic

[13] Voip-Info.org LLC. (2014) Asterisk realtime. [Online]. Available: http://www.voip-info.org/wiki/ view/Asterisk/+RealTime