C-RAN: A Step Towards a 5G Wireless Network

Submitted by:
Zachary Allin, Akash Patel, Luciano Taranto, Mihail Stantchev, Jerry Wasdyke

Team Project Number: S15-001

Advisor:
Professor Dario Pompili

May 1, 2015

Submitted in partial fulfillment of the requirements for senior design project

Electrical and Computer Engineering Department
Rutgers University, Piscataway, NJ 08854

Abstract

The goal of the project is to implement a testbed for the Cloud Radio Access Network (C-RAN). Derived from the research of Professor Dario Pompili and his graduate students, this testbed will be a small-scale implementation of C-RAN using Universal Serial Radio Peripheral (USRP) devices in conjunction with the open source softwares, OpenBTS and OpenAirInterface. With this testbed, we show that dynamic provisioning of centralized computing resources can improve the energy efficiency and resource utilization of the cellular network.

Table of Contents

1. Introduction

The core and current technology of the cellular wireless world is known as the Radio Access Network (RAN). In this architecture, there are two major components - the Base Stations (BSs) and the user equipment. The BSs are composed of antennas for handling communications to and from user equipment as well as computing resources for processing cellular traffic. Each BS has a set amount of computing resources (CPU, RAM, etc.) and each BS is responsible for using these resources to process its own traffic. Unfortunately, this technology does not satisfy the enhanced demands of today’s users.

A common issue seen in today’s network infrastructure is the change in the number of active users at different locations during variable times of the day, namely the “tidal effect” [1]. The most common explanation is that there is a significant amount of people that transition from downtown office areas during working hours to the residential areas during non-working hours. In the currently used network architecture, as users relocate, the nearest BS to which they connect changes causing variations in the utilization of BSs. Due to this overpopulation of certain BSs and underutilization of others, resources (CPU, RAM, I/O, etc.) are not optimized.

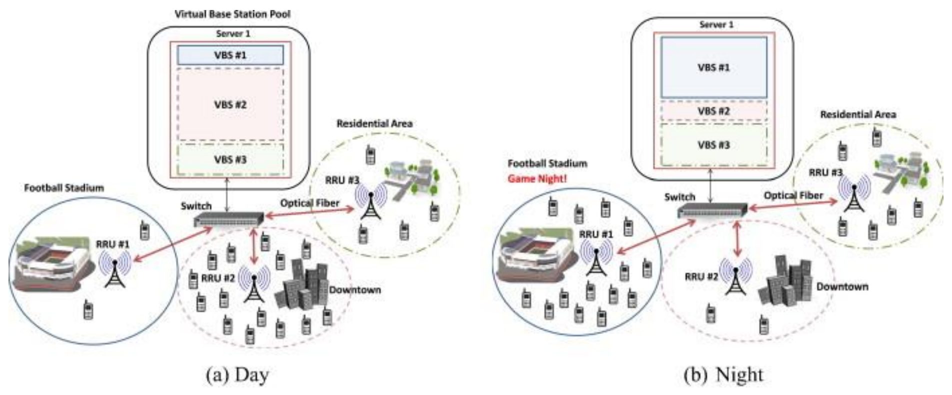

Figure 1: Tidal Effect

In this report, we introduce C-RAN (Cloud Radio Access Network) as a remedy to the tidal effect allowing for the more efficient allocation of resources through the concept of dynamic allocation and provisioning. This allows for the shifting of resources on demand as they are needed depending on the number of active users at different cells in the network, which can be done through the virtualization of BSs. Virtualizing the BSs removes the processing component of BSs in the field and brings the processing to a centralized location. Instead of having BSs in the field that are their own computers, we investigate having just antennas (remote radio heads) in the field that bring the cellular data back to a centralized location where multiple BSs are located in a virtualized environment that allows for the pooling of computing resources. In our experiment, we set up a Virtual Base Station (VBS) to investigate the potential benefits of moving to the C-RAN architecture.

2. Methods/Results/Approach

2.1. Methods

The project was initially subdivided into three major objectives: 1) initializing and setting up the BS in a virtualized environment, 2) establishing communication between the BS and the user equipment through the Universal Software Radio Peripherals (USRPs), and 3) performing experiments to profile the current technology as a basis for comparison against the new system. The first major choice of the design was which software platform to use for the BS. There were a few candidates to pick from - OpenBTS (2G), OpenBTS-UMTS (3G), and OpenAirInterface (4G LTE) among others. OpenBTS allows for mobile phones to be used in a Voice over IP (VoIP) implementation of a GSM network. OpenBTS-UMTS adds Universal Mobile Telecommunications System (UMTS) for 3G functionality. For more advanced experimentation, OpenAirInterface would be used to deploy mesh/ad-hoc LTE networks.We chose OpenAirInterface right from the start as it provided us with the latest standard (4G LTE) to improve upon. We could have chosen any of the others, but a major part of the excitement and drive for the project came from working with the latest standard to improve those technologies. Additionally, OpenAirInterface contains a great deal of other functionality such as simulation and emulation of cellular interfaces which would assist us with profiling the current technology. The platform we chose to run OpenAirInterface on was a virtualized system running Ubuntu 14.04.2. We chose this operating system for its compatibility and support for OpenAirInterface. This operating system was to run virtualized within VirtualBox on a high power workstation computer. The reason for choosing VirtualBox was primarily related to cost - it is a free system and a major player in the world of virtualization. The computer we ran this on was a Dell workstation with a 6-core Intel Xeon processor and 32GB of RAM. This would allow us to simulate a “datacenter” with a large amount of resources for running multiple BSs. The next major design choice was to choose the USRPs for communicating between the BS and the user equipment. For the USRPs, we chose the Ettus B210 boards as those boards were already proven to work well with OpenAirInterface from previous work done by the community. The user equipment (UE) we chose to use as the devices in our network were gathered from the group members themselves. The major factor in that decision was that the devices owned by the group members were unlocked allowing for usage on our test network.

After the initial design of the system was solidified and we began work on the project, we found that there were some significant issues with the way our system was designed which required a major evolution of the design throughout the life of the project. The first major issue we ran into was actually building the OpenAirInterface platform on our development system. We struggled with both missing dependencies as well as flaws within the source code which prevented us from building the software. We spent approximately two weeks attempting to trace the errors throughout the source code to resolve the issue. As we were working on that, a new version of the software was released as the OpenAirInterface developers had also been working on those issues. With this new version, we were able to build the software and begin testing the simulation capabilities. The next major setback occurred when we attempted to get communication going between the user equipment and the BS through the USRPs. The BS struggled to communicate with the USRP boards which prevented us from getting our devices on the network. After some investigation, we discovered that VirtualBox provided no USB 3.0 support which severely limited the data transmission capabilities between OpenAirInterface and the hardware boards. Every time we tried to establish communications between the software and the boards, OpenAirInterface would timeout while waiting for a response from the boards. We resolved this issue by transitioning our virtualization platform from VirtualBox to VMware Workstation which does provide USB 3.0 support. At this point, we found that the software was able to communicate with the boards but we were still unable to get the user equipment to join our test network. Some further research using the information provided on the OpenAirInterface TWiki led us to cause of the issue. The SIM cards we had purchased had been pre-programmed and we had not been provided with all of the values needed to establish relations between the device using the SIM card and the BS. We discovered that LTE communication worked on the principle of authentication between the SIM cards and the BS. SIM cards used in an LTE network are programmed with a secret key (Ki) which is also known to the BS. When a device using an LTE SIM tries to connect to the BS, it needs to authenticate using the shared Ki value. OpenAirInterface called for the SIM cards to be programmed for a specific Ki value which was pre-programmed into itself but the cards we had were not programmed with that value. We explored the possibility of changing the Ki value in the OpenAirInterface software and rebuilding it but we were unable to get the Ki value of the SIM card from the vendor we purchased the cards from. This led to a major design change in our system. In order to get communication working using the SIM cards we already had, we needed to change to a protocol that did not require authentication. The next best protocol that worked without this authentication is GSM. This precipitated a transition from OpenAirInterface (4G LTE) to OpenBTS (2G). After this change, we were able to set up a test network in our lab environment and get the user equipment we had to connect to that network.

Although we encountered many setbacks related to implementation choices during our design, the overarching issue was the time constraint. There were initial issues with our group finding an advisor and selecting a topic that would provide a useful contribution to a real-world problem that caused us to start late. This meant that we were operating at a severe disadvantage with regards to time and this was the contributing factor to many of the setbacks that were previously mentioned. If we had not been under such a constraint, we would have been able to perform more research in the beginning which would have given us a better starting design for implementation.

This project required an intimate knowledge of cellular network technology and its intricacies. As none of us had much of an understanding in this subject area, communication between the various components in a network was heavily researched. We utilized a variety of resources including, but not limited to, YouTube videos, mailing lists, equipment data sheets, documentation, and prior work done in the field. This resulted in a great learning experience for all of the members in the group.

2.2. Use of Standards

Our project involved the application of several different standards in both software and hardware.

In software, we built our VBSs on top of Linux (Ubuntu 14.04 LTS) guest virtual machines (VMs). The team used VirtualBox and the VMware Workstation virtualization solutions. These were both installed on a Windows 7 host. As mentioned in section 2.1, both OpenAirInterface and OpenBTS were used on the VBSs to emulate a BS. To collect system data, the sar (system activity report) command was used and the data collected was then processed in MATLAB.

The USB 3.0 standard was used to connect the USRP boards to the VBS. Initially, we attempted to use the LTE standard with the OpenAirInterface. However, after changing to OpenBTS, the GSM standard was used. To remain compliant with the rules and standards set forth by the Federal Communications Commission (FCC), the network was broadcasted on the 900 MHz band at low power. The 900 MHz band is not used in the United States thus our testbed did not interfere with other broadcasted signals. Initial testing of the USRP boards was done by using the open source software, GNU Radio. This allowed for communication between USRPs by transmitting and receiving signals for testing purposes.

2.3. Experiment/Product Results

To determine the benefits of virtualizing the BSs we compared the CPU power consumption of our approach of running multiple BSs in parallel in a virtual environment with shared resources against a standard, not virtualized, approach in which each BS is a separate entity with its own resources. In case one and two of our experiment, we examined two traditional BSs running OpenBTS. One operated under heavy load while the other was under light load. The heavy load BS, case one, was handling six concurrent calls and one hundred sent text messages. This load caused a peak CPU utilization of 92% and an average CPU utilization of 77% for the two threads. The light load BS, case two, was handling a single call which caused a peak CPU utilization of 17% and an average utilization of 10%. Each BS was given two threads to process data. Case three and four involved two virtual BSs running OpenBTS on the same machine with shared resources. In these cases, since the BSs are virtual, we were able to adjust the resources given to each BS. We gave three threads to the BS under heavy load, case three, and one thread to the BS under light load, case four. Case three caused a peak CPU utilization of 67% and an average CPU utilization of 57%. Case four caused a peak CPU utilization of 27% and an average CPU utilization of 19%. The results of these experiments are shown below in Figure 2.

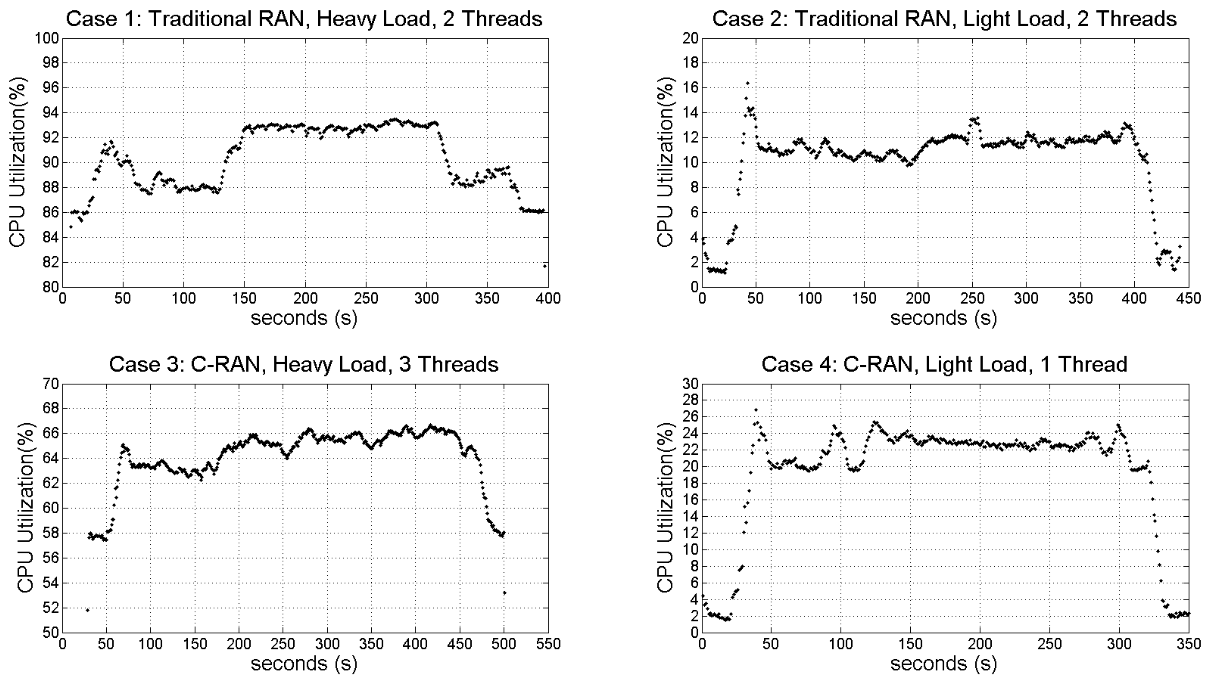

Figure 2: Experimental Results showing CPU Utilization

The processor used for the experiments is the Intel Xeon E5-1650 which has 12 threads. Minimum power draw for the processor is 63 watts and maximum power draw 122 watts (Kennedy). Power draw increases linearly as CPU utilization increases (Ellison). The slope of this curve is (0.59watts)/(% total CPU utilization) with a starting point of 63 watts at 0% CPU utilization. The total CPU utilization equals to CPU utilization * threads used/total threads in processor. Based on this information and the measured CPU utilization we can calculate the power consumption of the processor. For case one and two, each BS has its own CPU and power consumption is calculated individually. For case three and four, one processor is able to run both base stations.

$$Case\ 1{:}\ 0.59 * (77.47) * (2/12) + 63 = 70.62W$$
$$Case\ 2{:}\ 0.59 * (10.13) * (2/12) + 63 = 64.00W$$
$$Case\ 3 + 4{:}\ 0.59 * ((57.04) * (3/12) + (18.85) * (1/12)) + 63 = 72.34W$$

Traditional RAN

	Power Usage	CPU Utilization	Resource Utilization
Case 1	70.62W	77.47%	1.23GB, 2 Threads
Case 2	64.00W	10.13%	1.26GB, 2 Threads

C-RAN

Case 3	72.34W	57.04%	1.13GB, 3 Threads
Case 4		18.85%	1.12GB, 1 Thread

Using virtual BSs decreased CPU power consumption for the same amount of traffic from 134.62W to 72.34W. This is a 46.26% decrease in power consumption. This massive decrease is due to the fact that in tradition RAN each BS must have its own dedicated processor while in C-RAN multiple virtual BSs can be ran by 1 processor. Additionally maximum CPU utilization was kept much lower in C-RAN. Keeping CPU utilization below 100% prevents the CPU from being the bottleneck of the BS.

3. Cost and Sustainability Analysis

The testbed that was implemented was expensive. The entire design cost approximately $9144. The breakdown can be seen as follows:

Item	Unit Cost	Quantity	Total
Dell Precision Tower 5810	$3,360.00	1	$3,360.00
USRP B210	$1,100.00	3	$3,300.00
Board Mounted GPSDO	$625.00	3	$1,875.00
LP0965 Antenna	$45.00	6	$270.00
Sierra Wireless AC313U	$50.00	5	$250.00
Test SIM Cards	$13.00	5	$65.00
SMA Male to Male Cable	$4.00	3	$12.00
SMA Male to Male Cable	$12.00	1	$12.00
		Total	$9,144.00

The rather high cost of our experiment may prevent some from reproducing our design and experiment. Particularly students in undergraduate research would have increased difficulty in acquiring the funds or specialized hardware. Our team was fortunate enough to work with Professor Pompili and his graduate students who had proper grant funding through the National Science Foundation (NSF) to purchase all of the necessary equipment.

If a scaled version of our testbed was to be implemented in the real world outside of our testing laboratory, the cost would scale appropriately. One of the hopes of our project is that we can reduce the future cost of installing and hosting new cellular towers. The implementation centralizes the necessary computational resources thereby reducing the cooling and power needs and costs at each cell site. As these are major factors in cellular network expenditure, we see the overall costs of maintaining and managing a cellular network decrease.

Another benefit also has both economic and environmental implications. Currently upgrades to the cellular network architecture essentially requires a complete overhaul of the network equipment used. Creating a cloud based network allows for ease of upgradability to future cellular standards. The transition costs to newer technologies would be significantly reduced as only the software running the network would have to be upgraded whereas the hardware could remain untouched.

4. Conclusions/Summary

Though we did not accomplish all of our original goals, this project gave us great insight in the process of setting up and using USRPs and cellular software to create mobile networks. Throughout the semester the team encountered many challenges, including but not limited to learning the necessary background information on cellular network architecture, learning how to use the USRPs, figuring out how to properly build OpenAirInterface and OpenBTS, and learning how to get everything to fit together.

Our experiments with OpenBTS proved that setting up a C-RAN network will provide a significant reduction in CPU power consumption while also allowing the network to reallocate resources from BSs with low loads to BSs with high loads. Future work for our project would involve purchasing LTE compatible sim cards and setting up our BSs to use OpenAirInterface. OpenAirInterface will give us the capability of profiling 4G data utilization of our network which we can relate to the power consumption of the CPU. Additional improvements can be made by upgrading VMware Workstation to VMware vSphere. vSphere would allow us to dynamically reallocate resources for our BSs on the fly without having to first shut down the virtual machines. vSphere would also give us better profiling tools for measuring resources utilization of the BSs.

5. Acknowledgments

We would like to acknowledge Professor Dario Pompili and his PhD graduate students, Tuyen Tran and Albofazl Hajisami, for their continued assistance and support throughout this entire project. We would also like to acknowledge EURECOM as the authors and maintainers of the OpenAirInterface Project.

6. References

[1] D. Pompili *et al.*, “Dynamic Provisioning and Allocation in Cloud Radio Access Networks (C-RANs),” *Ad Hoc Networks Elsevier*, vol. 30, pp. 128-143. Jul. 2015

[2] B. Ellison. (2009) *The Problem of Power Consumption in Servers* [Online]. Available: https://software.intel.com/sites/default/files/m/d/4/1/d/8/power_consumption.pdf

[3] P. Kennedy. (2012, Jul. 31) *Intel Xeon Processor E5-1650 Sandy Bridge-EP Xeon Review 6C/12T 3.2GHz* [Online]. Available: http://www.servethehome.com/intel-xeon-e51650-sandy-bridgeep-xeon-review-6c12t-32ghz/

[4] EURECOM. (2015, Feb. 05) *OpenAirInterface* [Online]. Available: https://twiki.eurecom.fr/twiki/bin/view/OpenAirInterface