FROM 2.5G TO 5G: ENHANCING ACCESS AND PERFORMANCE FOR MOBILE USERS

by

Talal Ahmad

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September 2019

Professor Lakshminarayanan Subramanian

Dedication

To my late father, my mother, siblings and wife, with affection.

Acknowledgements

My family always supported my education. My parents (Nasira Parveen and Zulfiqar Ahmad) went above and beyond their financial capacity to put me in good educational institutions. My older brother Fazal Rabbani also worked hard to help cover my college tuition. I will be forever indebted to him not just because of the financial support but also because he was the one who convinced me to pursue a degree in computer science at FAST-NUCES after my rejection from LUMS.

I would like to thank my teachers during my undergraduate years, in particular Aamir Wali and Sobia Tariq Javed who taught computer programming and digital logic design in the second semester. These two classes together in one semester made me fall in love with the field of computer science. I desperately needed this because my first semester at FAST-NUCES was my worst semester in terms of grades as I was not interested in computer science at all at that time. Over the years there were other great teachers at FAST. I would like to acknowledge Ms. Saira Karim in particular because I interacted with her a lot over the years as she taught the algorithms class and was my informal final year project advisor.

I would like to acknowledge the guidance I got from Fahad Pervaiz regarding graduate school admissions. He was the one who got me started me on research in Umar Saif's NEWT Lab at LUMS after my junior year. For a student who always wanted to get into LUMS for undergrad, this was no less than a divine intervention that put me on course for PhD. Umar's focus on doing high quality research was always inspiring and I will be forever grateful for his advice and support.

In initial years at NYU I was lucky to get advice from Aditya Dhananjay, Sunandan Chakraborty and Shankar Kalyanaraman. Yasir Zaki was a wonderful collaborator over the years and helped immensely in papers we wrote together. The friendship of Ashwin Venkatraman kept me grounded over the years. Varun Chandrashekran often provided the necessary camaraderie for late night shenanigans; a memorable escapade was when we walked from 715 Broadway in Manhattan to Juliana's in Brooklyn to have pizza. Other people who I had the pleasure of knowing at NYU were Shiva Iyer, Trinabh Gupta, Sebastian Angel, Cheng Tan, Kate Boxer and Edwin Reed-Sanchez.

The time I spent at Microsoft Research was also very memorable. I had the good fortune to have excellent mentors in the form of Ranveer Chandra, Ashish Kapoor and Eric Horvitz. The co-interns also made it an excellent learning experience. Some of the co-interns were Deepak Vasisht, Vasuki Narasimha Swamy, Zerina Kapetanovic, Amy Kumar and Mathew Lentz. I thank Microsoft Research for this opportunity.

Islamic Center at NYU (ICNYU) was an important part of my life at NYU. It provided me with an endless list of friends who kept me going in hard times during the program. Sana Riaz was my first friend at IC and we still stay in regular touch. Over the years I had other great friends. Waqid Munawar Volli was always available to listen to my problems. Sharing my apartment with Afraz Khan lead to a year of extremely positive and nurturing conversations which made me a better human being. Georgia Halliday's wisdom often prevented me from making stupid mistakes in my personal life and I will be forever grateful. Maryam Saleem was always there for all her friends, regardless of whatever state she was in. Jasrin Jalal always went above and beyond in caring for her friends. Aziza Gauda always had a unique and refreshing perspective on everything. Mazhar Ladji might actually be in love with every girl he meets, but he was definitely the life of every gathering. Saad Hasan was our resident dentist, religious scholar and DJ all mushed into one person. Hiba Husayni was always refreshingly honest and straightforward. Shatha Muhsineh was the fitness guru and was always very helpful and encouraging to me. Zara Nizar was one of the nicest people and she went out of her way for not just friends but even strangers. Some other great folks I had the good fortune of knowing were Hana Hamdi, Noorul Hana Anwar, Noorul Ain Akmal, Mohammad Fazdly, Sana and Humayra Mayat, Amira Shouman and the director of ICNYU Khalid Latif.

I have been blessed with great teachers throughout my life. It would be next to impossible to name them all but each and every one of them played a role in making me who I am today. I would like to acknowledge them all here.

Last but not the least, I would like to acknowledge the guidance I got from my advisor Prof. Lakshminarayanan Subramanian. He was very kind to me over the years and helped me become a better version of myself.

Abstract

This dissertation has two overarching themes: i) enhancing access to connectivity for mobile users in rural contexts and ii) enhancing transport layer performance for mobile users.

More than half of the world's population faces barriers in accessing the Internet. A recent ITU study estimates that 2.6 billion people cannot afford connectivity and that 3.8 billion do not have access to it. To enhance access I have worked on two projects: Wi-Fly and GreenApps. Wi-Fly is a new connectivity paradigm designed for regions without Internet coverage that enables communication between lightweight Wi-Fi devices on commercial planes and ground stations. Through empirical experiments with test flights and simulations, we show that Wi-Fly and its extensions have the potential to provide connectivity in the most remote regions of the world. In GreenApps, we look at how localized cellular applications can be built for rural communities on top of software-defined cellular base stations. We deployed the GreenApps platform on rural base stations for communities in Ghana and Nicaragua, and supported multiple localized applications for rural communities.

Enhancing transport layer performance over cellular networks is critical to improve end-to-end application performance for mobile users. Cellular networks have unique challenges that make conventional transport protocols unsuitable for these environments. In the past few years, several new delay-based congestion control algorithms have been developed with complex nonlinear control loops for cellular contexts. While these protocols have shown promise, it has been extremely challenging to analyze and interpret the behavior of these algorithms especially under highly variable network conditions. In the Model-Driven Interpretable (MDI) congestion control work, we provide a model-driven framework to reason about the behavior of such congestion control algorithms. Our modeling approach simplifies a congestion control algorithm's behavior into a guided random walk over a two-dimensional Markovian state space(a Markov model). We show that the model of a congestion control algorithm can give key insights into its convergence and performance. More recently, we also looked at how to learn early signals of congestion in highly varying 5G channels. In particular we worked with Wi-Gig traces collected at 60 GHz and showed that it is possible to learn highly accurate early congestion signals using delay features observed at end hosts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation has two overarching themes: i) enhancing access to connectivity for mobile users in rural contexts and ii) enhancing transport layer performance for mobile users. In this chapter we give an introduction to the contributions we make in this dissertation.

1.1 Enhancing connectivity access for mobile users in rural contexts

More than half of the world's population faces barriers in accessing the Internet. A recent ITU study estimates that 2.6 billion people cannot afford connectivity and that 3.8 billion do not have access. We looked at two different ways of enhancing connectivity – providing widespread connectivity through airplanes and building a ground-up cellular ISP and enabling localized applications.

Widespread Connectivity: Recent proposals for providing low-cost connectivity include fielding of drones and long-lasting balloons in the stratosphere. We propose a more economical alternative, which we refer to as *Wi-Fly*, that leverages existing commercial planes to provide Internet connectivity to remote regions. In Wi-Fly we enable communication between lightweight Wi-Fi devices on commercial planes and ground stations, resulting in connectivity in regions that do not otherwise have low-cost Internet connectivity. Wi-Fly leverages existing ADS-B signals from planes as a control channel to ensure that there is a strong link from every plane to the ground stations. We also propose that ADS-B signals can help ground stations to intelligently wake up and associate to the appropriate plane. For our experimentation, we customized two airplanes to conduct measurements. Through empirical experiments with test flights and simulations, we show that Wi-Fly and its extensions have the potential to provide connectivity to the most remote regions of the world at a significantly lower cost than existing alternatives. Wi-Fly was joint work with Microsoft Research Redmond and was published as a paper in Hotnets 2017 [1]. Refer to chapter 2 for more details.

Ground-up connectivity: As part of this dissertation we also worked on *GreenApps*, a ground-up platform that enables off-grid, near offline and highly available cellular applications in rural contexts. The GreenApps platform enables rapid development and deployment of almost-always available applications that can be executed locally on top of open source cellular base stations. The GreenApps platform has been deployed in two rural regions in Kumawu, Ghana and Pearl Lagoon, Nicaragua and supports different community-centric applications. It was published as a paper in the ACM SIGCAS [2] and in Mobicom 2017 Smartobjects workshop [3]. Refer to chapter 3 for more details.

1.2 Enhancing transport layer performance for mobile users

Enhancing transport layer performance over cellular networks is critical to improve end-to-end application performance for mobile users. This is an important problem to address because cellular networks are a primary mode of connectivity for a large number of rural users in developed and developing countries. In this dissertation, we looked at understanding congestion control over cellular channels by leveraging the Model-Driven Interpretability framework. We also propose that we can learn congestion state for 5G networks and use it to improve congestion control.

Model-Driven Interpretability: Several new congestion control algorithms are being developed with complex non-linear control loops. It is hard to analyze and interpret the behavior of these algorithms especially if the underlying network is highly variable (e.g., cellular links). We introduce a novel modeling approach that allows us to simplify a congestion control algorithm's behavior into a guided random walk over a two-dimensional state space. Using our modeling framework, we were able to analyze and interpret the behavior of three different algorithms (Verus, BBR and Cubic) with completely different control loops on cellular channels. We demonstrate that the model versions of these protocols successfully approximate the throughput and delay characteristics across variable network conditions. We show that the model of an algorithm can give key insights into convergence and performance of an algorithm. This work is under review at this time. Refer to chapter 4 for more details.

Learning Congestion State: Millimeter wave (commonly known as mmWave) is enabling the next generation of last-hop communications for mobile devices. But these technologies cannot reach their full potential because existing congestion control schemes at the transport layer perform sub-optimally over mmWave links. In this chapter, we show how existing congestion control schemes perform sub-optimally in such channels. Then, we propose that we can learn early congestion signals by using end-to-end measurements at the sender and receiver. We believe that these learned measurements can help build a better congestion control scheme. We show that we can learn Explicit Congestion Notification (ECN) per packet with an F1-score as high as 97%. We achieve this by doing unsupervised clustering using data obtained from sending periodic bursts of probe packets over emulated 60 GHz links (based on real-world WiGig measurements), with random background traffic. We also describe how the learned values of ECN can be utilized for rate estimation. This work is scheduled to appear in the proceedings of ACM mmNets 2019 [4]. Refer to chapter 5 for more details.

Chapter 2

Wi-Fly: Widespread Opportunistic Connectivity via Commercial Air Transport

4.4 billion people around the world live without Internet connectivity [5]. Most of the world's offline population (64 percent) live in impoverished rural settings, where poor electric power and communications infrastructure, and lack of hardware for base stations and end-user devices impede Internet adoption[6]. More than 80 percent of people who are currently disconnected from the Internet are younger than 55 years old, and more than 42 percent are younger than 25 years old. Another recent study [7] has shown that reducing the cost of Internet access can help connect a significant population of the world (see Figure 2.1). In order to address the critical challenge of connectivity, several small scale[8, 9, 10] and large scale[11] efforts have been aimed at connecting the disconnected people around the globe. Unfortunately, not much has changed over the last decade. The small

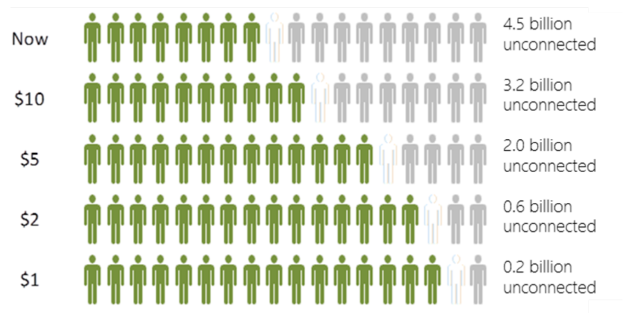


Figure 2.1: Impact of reducing Internet access costs on global connectivity. The vertical axis on the left shows the cost of access.

scale, community-based solutions help people but they require funding and ongoing technical support, which are both hard to provide, given the cost of existing solutions.

In this chapter, we focus on the promise of leveraging commercial air transport system as a means of providing connectivity on a continental scale. At any given moment, there are over 10,000 aircraft flying globally and covering large swaths of geographical area. For our study we consider the possibility of utilizing all these aircrafts as access points that population hubs on the ground can connect to. By the virtue of the long distances these aircraft travel, we have shown that it is possible to connect large areas where connectivity has previously been unfeasible.

Our approach, called Wi-Fly, aims to leverage various technological components of existing infrastructure so that connectivity can be provided at low costs. Wi-Fly uses Wi-Fi frequencies and radios to communicate between the aircraft and ground stations. In our framework, ground stations (communication hubs) listen for ADS-B transmissions from aircrafts to learn when they are nearby, and where they will be in the near future, and power up and fine-tune their Wi-Fi hardware accordingly. Such a capability helps increase the throughput of the system by determining the best aircraft to communicate with when multiple planes and ground stations are operating.

Wi-Fly aims to provide connectivity to delay-tolerant applications by leveraging the existing Internet available in commercial planes using Ku and Ka bands. The goal of Wi-Fly is to support two primary scenarios. First, we seek to enable people to communicate with each other using emails, instant messages, and by downloading popular content. Second, we seek to provide communication to "things", such as sensors that would otherwise be disconnected in remote regions, e.g. in forests, deserts and agricultural tracts.

This work is a significant departure from Google's recently proposed Project Loon and Facebook's Aquila efforts. Instead of creating custom aerial vehicles, we aim to keep the costs down by utilizing existing infrastructure. Another key benefit of our approach is that Wi-Fly is much easier to deploy on a large scale. Finally, the design of our system is much more flexible and robust; consequently it has the potential to adapt to various aircraft scheduling constraints. In summary, the key contributions of this work are:

- Solution to the connectivity problem that leverages existing air transport infrastructure.
- Using ADS-B to detect the presence of aircraft and build a predictive model of connectivity.
- Real-world experiments with real aircraft data that shows the promise of the framework.
- Real-world deployment on general aviation aircraft that demonstrates empirical performance.

2.1 Motivation

In order to assess the efficacy of our connectivity model, we built a simulator using flight data from the website FlightAware [12], along with experimental data collected using our platform, detailed in Section 2.4. In May 2016, we downloaded the information of all planes flying across a single day over the continental US and Africa regions. The data was downloaded in the form of snapshots of currently flying airplanes at a given time. Each snapshot was downloaded approximately 5 to 10 minutes apart. For each flight that was in the snapshot, we noted the plane height, speed, and exact location in the region. In order to make sure that we were only using commercial planes that were in the air (as opposed to those parked at airports), we made sure that every plane under consideration was at an altitude above 20,000 feet, and had an airspeed of over 200 Knots[1].

As a preprocessing step, we studied the data carefully and worked to remove all potential discrepancies. We particularly looked at flights' unique IDs across snapshots and made sure that their locations were different, so as to ensure that we only considered flights that are constantly moving. Thus, we observed that several flights which were stationary at airports were being erroneously reported as flying by the FlightAware API. These flights were removed from the data. As mentioned earlier the data comprised a single day. We used a single day's data because flights follow a similar pattern for most days.

Along with gathering FlightAware data of planes in air, we also conducted experiments using a custom built measurement platform detailed in Section 2.4. In our experiments, we placed a commodity high-power Wi-Fi router which had

[1]Knot is unit of speed equaling 1.15078 miles per hour

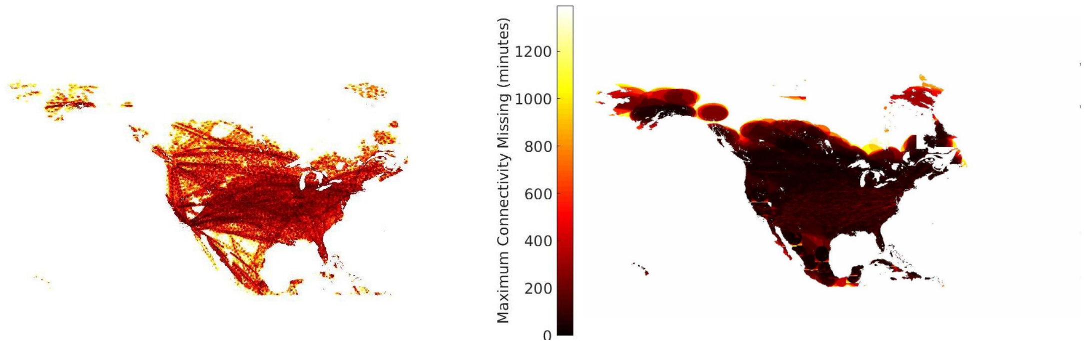


Figure 2.2: Coverage map for continental US with zero antenna gains (left) vs with 10 dbm gains at receiver and transmitter (right) using regressed parameters.

a measured transmission power of 25dbm [2] in a modified single engine airplane, as shown in Figure 2.6. We placed the AP in the wingtip of the airplane, and allowed us to avoid making any structural changes to the plane. Note that the wingtip is made of fiberglass, a non-conductor. We attached dipole antennae with an estimated gain of less than 5 dbm in the plane and collected beacon packets sent by the router at a ground station. While the plane was in flight, we also collected a time-synchronized GPS log of the flight which gave us the exact distance between the receiver and the transmitter at all times.

2.1.1 Wireless propagation parameters

We chose to use the free space propagation model in the simulator as supported by previous studies[13, 14]. While multi-path models have also been used, they were based on special receivers (for example parabolic receivers following the airplane[15]), and in special terrain settings [13]. For the line of sight component independent of the terrain, the use of free space propagation is precedent. The

[2]This is significantly lower than the legal limit.

free space propagation model is summed up by the Friis equation:

$$\frac{P_r}{P_t} = G_t G_r \frac{\lambda^2}{16\pi^2 R^2}$$

In the above equation, P_r and P_t denote the receive and transmit power respectively. G_r and G_t denote the receive and transmit gain. λ and R denote the wavelength and distance between transmitter and receiver respectively.

In order to fit a path loss model based on our experiments, we assumed that R is raised to a variable γ, and we estimated this parameter γ using experimental data we collect.

We use the received power (gathered from the measurement platform in Section 2.4), antenna gain of receiving and transmitting antennae, and applied curve fitting to calculate the value of the parameter γ. For wavelength calculation, we use the value of center frequency of channel 1 (2412 MHz), which the router was using to transmit the packets. In order to have diversity, we use two different receiving stations with two different receiving antennae. We use MATLAB for applying regression for estimation of γ.

We collect the data by flying the plane (in a circular pattern) at various altitudes, and using two different laptops to decode the packets. We record the packets received, and post-process them with the flight log from the plane. The estimated value of γ using this data was 2.12.

2.1.2 Coverage Results

After estimating propagation parameter γ, we use the FlightAware data of the planes to estimate which areas will get coverage in different regions around the

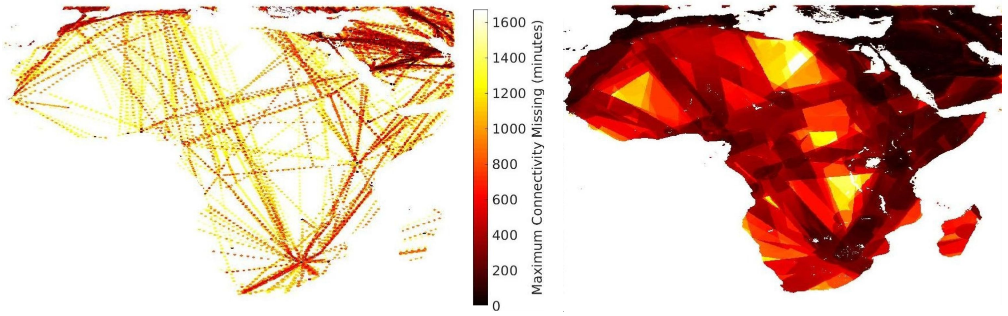


Figure 2.3: Coverage map for Africa with zero antenna gains (left) vs with 10 dbm gains at receiver and transmitter (right) using regressed parameters.

world. For these coverage simulations, we assume a transmit power of 2W. We use different values of transmitter and receiver antenna gain. We also use a gain of 0 dBi or 10 dBi in order to show beam-forming gains. We calculate SNR using thermal noise for the bandwidth of the system. We assume connectivity if the SNR was more than 0 dB. Figure 2.2 and Figure 2.3 show the regions of America and Africa respectively, that can be covered by Wi-Fly, and the maximum amount of time that each region would *not* have Internet access during a day. We evaluate the potential coverage with and without antenna gain.

As we see in these figures, most regions can be provided Internet access with overhead commercial aircrafts, with most of the covered regions losing connectivity for at most 10 hours. This map was made operating under the assumption that all the flights are using our system to provide coverage. As we expect the number of flights to increase with time, the coverage for this map will only increase, further motivating the design and study of such a system.

For the Africa maps shown in Figure 2.3, we also did a population coverage analysis based on publicly available population maps [16]. We calculate that 80% of the population requires less than 10 hours of maximum connectivity loss based on the coverage shown by high gain transmitter and receiver on the right in Figure 2.3. We also calculate that even with the conservative parameters used in the map on left in Figure 2.3, 50% of the population in the region is connected during the course of the day.

2.2 Wi-Fly Architecture

Wi-Fly leverages commercial air transport to provide Internet access to remote areas. The goal of the Wi-Fly system is to support two primary scenarios. First, we seek to enable people to communicate with each other using emails, instant messages, blogs, etc., and also download popular content such as news and weather updates. Second, we seek to provide connections to "things", such as sensors that would otherwise be disconnected in remote regions, e.g. in forests, deserts, and agricultural tracts, to enable access to sensed data.

Planes in Wi-Fly have local Internet access either using a satellite connection over the Ku [17, 18] or Ka bands [19], or from another ground station [20]. Nearly 70 airlines around the world offer Wi-Fi in flight. Worldwide 39% of airline passengers already have an internet connection, 68% of which is high quality Internet [21]. This percentage is on the rise with US taking the lead with 80% of flights having Wi-Fi [21]. Furthermore, governments of many countries are exploring opening up additional spectrum [18, 22] that will enable airplanes to provide even higher bandwidth Internet connectivity in the near future. In an alternate approach, the methods we propose can also be fielded in the absence of plane connectivity via special satellite or ground stations; planes employing the local communication with the ground stations can cache and sync content when they land at airports.

Wi-Fly extends the connectivity in the planes to various kinds of ground stations including connectivity hubs for people and sensors. Each plane is equipped with antennas, and a transceiver that communicate with special Wi-Fly ground stations. This equipment is in addition to the other transceivers that are used on the plane. Although excess weight could be a potential concern, our experimental setup adds only a small additional weight of three lbs, which is acceptable, per our private communications with commercial flight companies.

The ground stations also have a transceiver and an antenna pointed at the sky. In our current design, aimed at high bandwidth applications, Wi-Fly requires a phased array antenna to maximize the coverage of the horizon, while also providing high gain. Since the antenna configuration of each station is cumbersome, the Wi-Fly system for broadband connectivity uses an alternative technology such as Open Cellular platforms[9, 10, 8], TV white spaces[23], Wi-Fi access points, point to point links [24], or GPRS, for the eventual last mile connectivity. For IoT applications, that have a much lower throughput requirement, Wi-Fly modulates the signal over a narrow bandwidth that has a much lower noise floor. Hence, the antennae do not need as much gain, and therefore each sensor has an attached omni-directional antenna, and each sensor also directly communicates with the plane.

To achieve good connectivity between the ground stations and the planes, Wi-Fly needs to solve three main challenges:

- First, to ensure good throughput between the plane and the ground station, Wi-Fly needs to dynamically adapt its wireless parameters. Using Wi-Fi, or any existing ground communication system as is, leads to poor throughput. Existing Air-to-Ground systems, such as the ATG4 used by Gogo Wireless use a much slower EVDO technology.

- Second, Wi-Fly needs to scale to multiple airplanes and ground stations. This requires the design of a unique media access control (MAC) protocol. In contrast to Wi-Fi or cellular systems where the clients are mobile, in Wi-Fly, the airplanes (base stations) are mobile. Furthermore, we seek to do an optimization that considers multiple parameters: the clients need to associate with the plane that is close enough to provide a sufficient signal, that is also least lightly loaded, and that promises association for the longest period of time.
- Third, Wi-Fly ground stations need to support a low-power mode to allow IoT devices to communicate with the planes. Having them always on would consume a great deal of power, and heavily duty cycling the sensors might miss communication opportunities with an overhead plane.

2.3 ADS-B Assisted Control Channel

The challenges highlighted at the end of the previous section require techniques for ground stations to learn about the presence of nearby aircraft. If a ground station is aware of all the planes, they can connect with a correct plane to maximize throughput, and if there is no plane, they can deactivate the base station to save power and reactivate at a later time when there is a plane available.

One approach could be to download a schedule of plane flights. However, planes may be delayed and may deviate from scheduled flight paths (e.g., due to wind and/or weather). Another approach is to perform active polling, where ground stations periodically probe for incoming planes. However, such an approach adds latency in establishing the link, and introduces extra overhead messages on the channel.

To circumvent these challenges, Wi-Fly uses a control-channel–based approach. However, instead of using a separate radio on the plane to implement the control channel, the Wi-Fly control channel uses existing ADS-B (Automatic Dependent Surveillance - Broadcast) [25] signals sent by planes. ADS-B signals are sent on 1090 MHz or 978 MHz, with a 50 KHz or 1.3 MHz bandwidth respectively, and encoded using PPM. Planes use these signals as an alternative to secondary RADAR. The aircraft transmits its latitude, longitude, speed, bearing, pressure, altitude, callsign, etc. in separate messages of 10 bytes each, with a total packet size of 112 bits.

Each packet in ADS-B is identified using some data fields known as Downlink Format (bits 1 to 5) and Type Codes (bits 33 to 37). Wi-Fly proposes sending of some extra packets from the plane using unused type codes carrying useful information like current load and available capacity. We call our technique ADS-B Assisted Control Channel, using which the aircraft can send small amounts of data to the ground stations. We use the Downlink Format 17 which is meant for broadcast and caps the maximum packets sent to a very few packets per second. This enables Wi-Fly to have a lightweight, push-based approach, where planes signal their presence without clogging the data channel. We would like to note that we collected ADS-B data over several weeks in different cities and realized that there are several unused Type Codes, in particular Type Codes 5-8, 25-27 and 30 are not being used.

We note that ADS-B signals operate in a lower frequency, and are sent at a higher power. Hence they propagate farther than the data (transmitter in 2.4

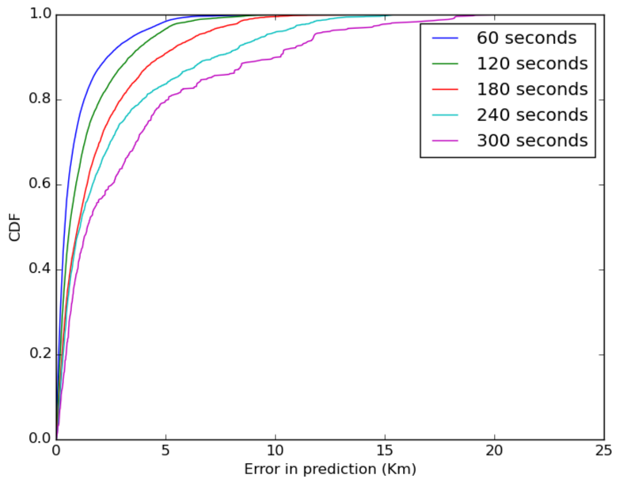


Figure 2.4: Errors in prediction of planes locations at various times in the future based on ADS-B data. Different lines indicate different times into future for which predictions were made and error calculated based on actual plane location.

GHz). Such a range mismatch between the control and data channel has been shown to cause inefficiencies in protocol design. For example, when the control channel is used as a contention medium [26], then many more devices will hear the transmission on the control channel, and backoff, while they might never have heard the packets on the data channel. However we don't have this problem in Wi-Fly because ground stations never transmit on ADS-B channel and just use one way packets to learn the exact location and heading of the plane.

We use the ADS-B assisted control channel to help address the challenges discussed in the previous section. ADS-B packets provide telemetry data like location, speed and heading of planes which we use to calculate how long a plane will be able to connect to a base station. In figure 2.4 we show the accuracy of this approach using real ADS-B data. Using location prediction and current load on the plane (which is sent by leveraging unused Type Codes) the base station on ground chooses to connect with the plane that has the least load and will stay

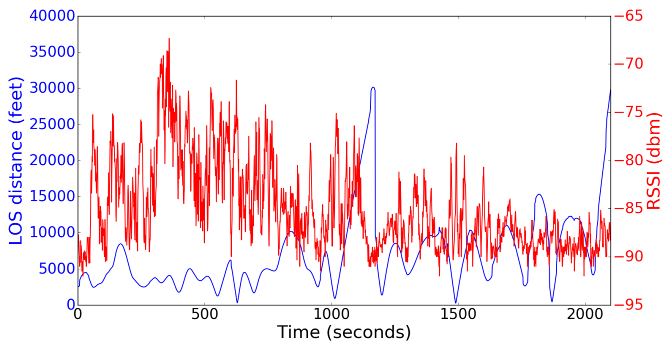


Figure 2.5: Correlation between distance and RSSI of packets at a per second granularity during flight experiments.

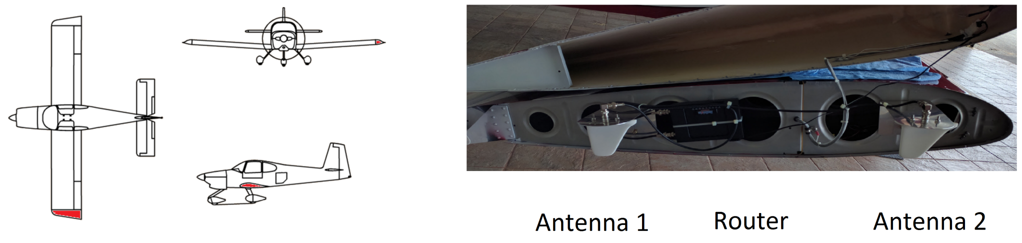


Figure 2.6: Instrumentation of the plane.

connected for the longest time hence increasing the throughput of the system. The ADS-B signal is also used as a wake-up signal to re-activate base station on ground when a plane comes around.

2.4 Measurement Platform

We instrument a general aviation aircraft to carry the equipment. Specifically, the wing tips of the aircraft are hollow and built out of fiberglass and provides ample space for the antennae and the Wi-Fi router to be installed. Figure 2.6 shows the details of the installation. The red color on the aircraft depicts the wing tips where the equipment was installed. In our installation, we used an off-theshelf RadioLab router and 2.4 GHz Wi-Fi antenna that were cased in a custom aviation form factor. The antennae were mounted on steel plates, which provided the required ground plane, and were attached to the aircraft's wing ribs. We made sure that we have maximum separation between the antennae while not doing any enhancements to the shape of the wing. The aircraft was also equipped with a Mode-C transponder, and during the test flight, the Air Traffic Control (ATC) was contacted, who then assigned a squawk code for the Mode-C broadcast.

On the ground, we collect data at two different channels. The ADS-B and Mode C telemetry channel at 1090 MHz and Wi-Fi channel at 2.4 GHz. The telemetry data was collected using a USRP N210 with an SBX daughter card, and a 1090 MHz helix antenna. The mode-C squawk code assigned by the ATC allowed us to distinguish the packets sent by the test aircraft from other broadcasts. The Wi-Fi channel data was collected using 2 Wi-Fi receivers on the ground. One receiver was a Lenovo T430 Laptop with the built in Wi-Fi chip, and the second receiver uses the external Intel 5300 Wi-Fi card connected to a HP Pavilion laptop over the express card slot.

2.4.1 Collecting Channel State Information (CSI)

We use the modified Intel 5300 firmware and driver [27] to collect CSI information from air to ground. This tool collects CSI for various kinds of packets, but the biggest challenge we faced was how to keep the base station associated with the AP in the plane. In order to keep the plane associated, we used a replicated setup of client and AP on the ground and in the aircraft. We used a CSI collecting laptop on ground which was connected to an AP on ground. The AP on ground pretended to be the AP in air, this meant that the AP on ground had the same

IP, MAC, BSSID and Wireless channel as the AP in the plane's wingtip. The only difference was that the AP on ground did not send any 802.11n High Throughput (HT) packets for which the modified Intel 5300 card collects CSI. In the plane we put a replicated client which looked like the CSI client on ground but it did not have the Intel 5300 card to collect CSI. The client in the plane was constantly associated with the AP in the plane and was pinging the plane while the client on ground was constantly associated with the server on ground.

This replicated setup was done in the hope of getting the 802.11n HT packets that were intended for the client in the plane, and to collect them at the client on the ground, and consequently gather CSI from them. We were able to collect CSI packets in this manner consistently during the flight, giving us a deeper understanding of the channel from the plane to the ground. We use the data collected for throughput simulations shown in earlier sections. For a single flight, the correlation between distance from the receiver and RSSI value of all received packets is shown in Figure 2.5.

2.5 Summary

The key takeaways of this chapter are:

- Wi-Fly is a communications methodology and architecture aimed at providing opportunistic connectivity in remote and under-served regions around the world.

- Wi-Fly leverages ADS-B assisted control channel to provide low powered base stations for connectivity with commercial planes.

- Wi-Fly provides coverage maps based on actual air-to-ground measurements in ISM bands.

- Wi-Fly gives design and architecture of a system to Contrary to other solutions because it leveraging of the existing infrastructure of commercial air transport.

Chapter 3

GreenApps: A Platform For Cellular Edge Applications

Despite significant advances in the penetration of cellular networks in developing regions in the past decade, the growth has been predominantly centered in urban regions, with relatively low rural presence [28, 29, 30, 31, 32, 33]. The conventional cellular connectivity model has not proven to be economically viable for rural settings due to lack of stable and reliable grid power [6] and due to lack of reliable backhauls and data services [34, 35]. The situation is further exacerbated by the fact that most rural areas have low population densities which translate to lesser revenue for commercial cellular providers.

This chapter presents the design, implementation and deployment of *GreenApps*, a platform that leverages programmable base stations to deliver off-grid, near offline mobile applications that empower rural communities. Recent advances in hardware [36, 37, 38, 39] and open-source software [40, 9] provide a unique opportunity to deploy programmable cellular networks in rural contexts; however, most deployments [33, 41, 42] of such base stations have focused primarily on providing basic communication services like calls and SMS. Heimerl et al. [33] demonstrated the viability of independently run, locally operated cellular networks. Similarly, Rhizomatica [41] has deployed tens of community-run GSM cellular networks in Oaxaca, Mexico with a short-term experimental spectrum license. Zheleva et al. [42] deployed Kwiizya, a system for enabling basic calls and SMS-based service in Zambia. However, there has been very limited work on leveraging these programmable base stations to enable new community-based mobile applications at the edge.

We present our experiences in designing, implementing and deploying GreenApps in hard to reach rural regions in Ghana and Nicaragua. We provide technical details of how we built our platform, as well as details of the development process for our applications within GreenApps. We have deployed our complete end-to-end platform, including applications, in rural regions, in both countries. We also present an evaluation of key components of the GreenApps platform. GreenApps is currently serving populations in Nicaragua and Ghana, and we have preliminary evidence on user satisfaction with our platform. A wide-scale and in-depth study of the impact of our platform on the livelihood of locals in these regions is part of our future work.

3.1 Building a VC-ISP

Cellular networks have achieved significant penetration levels in developing regions during the past decade. Much of this growth has been predominantly urban-centric with relatively low rural presence [30, 31, 32], and unfortunately existing cellular connectivity model is not economically viable for rural settings for a multitude of fundamental challenges in rural contexts:

Power: Rural regions lack stable and reliable grid power. Cellular networks are inherently power hungry in rural settings, and seek to blanket large areas, often relying on diesel-powered generators for power, which is a highly expensive proposition [6].

Lack of Financial Incentives: While cellular networks incur high cost of capital and operational expenses, the demand often does not match the cost due to low user-densities and low purchasing power. For years, low demand and high infrastructure costs have been locked in a vicious cycle. Bharti Airtel and Reliance Communications, two of the largest cellular service providers in India, announced their intent to pull out of rural markets due to unfavorable economics [43].

Reliability: Maintenance of rural, wireless networks is difficult due to a myriad of reliability issues, power-related problems, and a lack of local expertise to solve them. Lack of clean power is known to frequently trigger device failures in rural networks [32].

In this section, we present the design, implementation and deployment of *Virtual Cellular ISPs (VC-ISP)*, a new ground-up cellular network architecture that enables third-party vendors to offer cellular coverage and mobile services to rural areas in an extensible manner. The basic building of the VC-ISP network architecture is a Virtual Cellular Node (VCN) which are open, programmable cellular base station platforms [37, 9, 44] that enable easy deployment of low-power software defined base stations that function efficiently over an IP backplane. Given that VC-ISPs are designed for areas with limited cellular coverage, the second building block of the VC-ISP model is the use of highly directional backhauls to extend backhaul connectivity to the VCNs. In this work, we assumed that a backhaul is available and then we looked at how to build the communication mechanism used by the VCNs to form a cellular network.

The VC-ISP network architecture is built upon on the existing work on community cellular networks[33, 42, 45, 41], which can be viewed as stand-alone installations of virtual cellular nodes. In VC-ISP architecture, we aim to address the following question: *How do we enable a distributed and potentially disjointed collection of open, programmable cellular base stations to act in unison as a single virtual cellular ISP?* Given the lack of incentives for conventional cellular providers to provide rural connectivity, the VC-ISP model empowers third-party providers to build a cellular network comprising of several distributed VCNs connected only by an Internet backhaul to the cloud. In addition, due to the programmable nature of individual VCNs, we show how VC-ISPs can provide new forms of distributed mobile edge services where computation, storage and application state can be moved to the mobile edge.

We demonstrated the effectiveness of the VC-ISP model using a real-world 3-node VC-ISP deployment system, which includes a solar-powered cell tower installation in rural Ghana. Our Ghana based installation also supports different types of distributed mobile services (as outlined earlier) that have been used by rural users within the community. We also evaluated the effectiveness of specific aspects of the VC-ISP architecture using emulation and simulation experiments. In summary, we believe that VC-ISP represents a radical departure from the conventional approach of designing cellular networks, and could enable ground-up innovation, allowing several budding rural entrepreneurs to launch competing cellular network services in rural localities while seamlessly co-existing with existing operators.

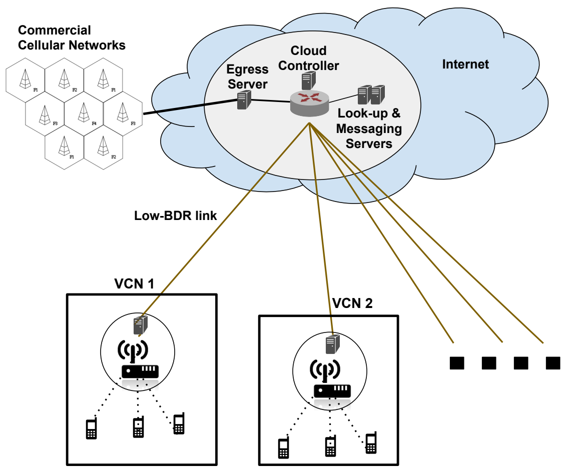


Figure 3.1: VC-ISP Architecture

3.2 VC-ISP Architecture

VC-ISP is a new model for designing and rethinking the evolution of cellular network providers. The network architecture comprises of a distributed collection of software-defined VCNs powered by emerging open-source cellular base-station platforms, which together provide the abstraction of a single virtual network with the entire gamut of features offered by a traditional cellular provider. Using this architecture, we wish to extend coverage beyond existing boundaries, moving a wide variety of services to the extreme edge. The VC-ISP network architecture aims to enable third-party vendors to offer cellular ISP services in a decentralized manner without requiring modifications to mobile devices, and allowing cooperation with traditional cellular networks.

Conventional cellular networks continue to use highly inflexible and expensive platforms with complex control-plane protocols. This makes the network nearly impossible to configure or manage. On the contrary, VCNs enable us to program and easily integrate a wide range of other middlebox services and functionalities.

The strawman architecture is illustrated in Figure 3.5; a distribution of VCNs provide access to a geographically disjoint set of regions and collectively provides the abstraction of an unified VC-ISP. We envision all of the nodes to be connected to the Internet using various types of backhaul technologies such as wireless backhauls, satellite links, balloons, leased lines, telephony, fiber etc. Owned by a single party, a centralized cloud controller would interconnect all nodes and is in charge of administering this cohesive network. This controller resides at an appropriately placed geographic location to normalize communication latency. To achieve such an architecture, we discuss design specifications and the high level goals we wish to achieve. To summarize, the architecture should be able to:

- *Goal 1:* Operate in regions where there is high latency, low bandwidth and intermittency.
- *Goal 2:* Reduce processing at the core and migrate relevant application processing to the edge to further reduce latency.
- *Goal 3:* Avoid placing heavy reliance on conventional sources of power (including fossil fuels).

Addressing these goals would require us to tackle several research challenges. The first and foremost challenge involves the exploration of a variety of operational conditions. Additionally, majority of developing regions have poor cloud infrastructure and a weak backhaul, resulting in low Bandwidth-Delay Ratio (BDR) regimes. Traditionally designed applications running in these regimes face several issues including frequent timeouts, operational bandwidth insufficiency and various other inconsistencies. Dealing with these issues and obtaining *Goal 1* requires rethinking application requirements. In VC-ISP mobile clients are connected to a *local* server placed in the proximity of the base station. This local server has complete autonomy in decision-making for a certain class of events but also must abide to directives from the central controller. Therefore, the local server ensures greater responsiveness to clients. The VC-ISP will provide a separate intermittency-aware substrate where all messages are tagged with session and communication group specific identifiers that enable nodes to store and deliver messages to their corresponding recipients in a delay-tolerant manner. Mobile applications in VC-ISP can also leverage this substrate to specify how to handle application specific messages in an intermittency condition.

This centralized cloud infrastructure provides mechanisms for the now unified (nodal) network to communicate with non-member networks. This infrastructure comprises of 2 key components:

- *Egress nodes*, which are responsible for forwarding traffic between either VCNs or between VCNs and the conventional cellular network.
- *Lookup and messaging servers*, which play a critical role in enabling the VC-ISP to offer connectivity and a variety of cellular services across nodes.

Each VCN in the network operates independently and communicates with other VCNs using the egress nodes, and the lookup and messaging servers. We partially obtained *Goal 2* as the local servers helped facilitate this process in most cases because they have a cached version of the naming and addressing table. Because these servers also have administrative autonomy, they are able to act on the behest of the centralized controller, and can provide additional services for applications at the extreme edge. Different types of edge services will be explained in subsequent sections.

The modular nature of the various components required to build the VC-ISP ensures minimal power consumption. Therefore, we obtain *Goal 3* by predominantly powering the nodes using solar panels. By building a compact, mobile base station unit, we are able to avoid the reliance on the unstable grid power in these regions.

3.3 Naming & Addressing in VC-ISP

To successfully utilize the various services provided by cellular networks, even from outside the network, it is imperative to have an efficient scheme for device naming and addressing. We wished to achieve three goals using our identity management service, namely (a) flexibility in identities held, (b) seamless user experience, and (c) uniqueness.

3.3.1 Flexibility

A user can have an authenticated identity which permits it to use services offered by the VC-ISP. However, this identity should not facilitate communication with external users (outside the VC-ISP). Similarly, a user can have another identity whose sole purpose is to enable communication with the outside world. To this end, the VC-ISP issues two types of identities: *(a) Local identities* that are only usable/valid inside the VC-ISP framework, and *(b) Global identities* which make it possible for devices outside the framework to connect with those inside. It is the duty of the identity management service to multiplex the functionalities accordingly based on the nature of the identity.

Users with local identities can communicate with others within the same VCN or in the same VC-ISP. In contrast, global identities provide all the functionalities supported by local identities and enable connection both to and from outside the VC-ISP. The importance of this form of flexibility stems from the fact that providing access to an identity outside the network has an associated cost. Globally identifiable names need to be purchased (or leased on a monthly basis) from organizations selling VoIP services, like Twilio.

3.3.1.1 Generation & Dissemination

Unique identifiers are maintained, regulated and generated (sequentially, in the order in which requests are received) by a centralized server separate from the controller, henceforth referred to as a name server. These identifiers (or names) are flat and carry no location information. The flexibility offered in naming can best be utilized when the architecture is optimally designed. To achieve the required performance, we propose two different implementations, namely:

1. Server-Client Model: A single, centralized name server issues both global and local identities as requested. If a new users enters any node, the registration requests are routed to this name server in the order of entrance. In the case of a cellular device that connects through the OpenBTS interface, the request to generate a name comprises of the IMSI number (which is globally unique and non-mutable). This information is then used in the standard GSM challenge-response protocol, when the phone associates with the network. The response from the central name server is a unique identifier. This server keeps track of each identity issued (name) and its current location (address), maintaining consistent state information. This information is required to route calls and messages correctly. In addition, it records the IMSI number of the SIM card to which the identity was issued. If the user moves to a different node, the server still knows that an identity has been issued and only the location of the user needs to be updated. To reduce communication latency, this information is replicated at the local servers at a nodal level and updated on occurrence of a cache-miss.

2. DHT-based Model: In this model name/location mapping requests are answered by a cloud controller chosen by a predefined hash function. These cloud controllers collectively form a DHT for managing identities and application level data storage. This DHT-based model used by several authors [46, 47] deals with failures and provides faster access even when there is a skew in users between different nodes by evenly dividing storage and processing.

Intermittency aware lookup: As explained earlier, the VC-ISP architecture provides an eventually consistent view of the entire name space to the local server at a nodal level. If a user wants to connect with another user, it first contacts the local server of the node it currently is in. On occurrence of a cache-miss at this node, the required information is retrieved using suitable cache-update protocols [48] in a lazy manner. In the absence of a backhaul to the central name server, information at the local server at the nodal level is not consistent but once the backhaul is available consistency is achieved eventually.

3.3.2 Seamless User Experience

In both of the above mentioned models, naming and addressing information is cached at the local servers at each node. Though not complete, caching ensures a reduction in communication latency in the case of lookups, therefore improving the end-user experience.

Interoperability: It is crucial for the architecture to support the following: (a) communication within the VC-ISP, (b) communication across VC-ISPs, and (c) communication between the VC-ISP and other non-member networks (in both directions). Communication with the architecture has been explained in previous sections, where information obtained from the combination cache hits and misses on local servers and the centralized name server suffices to locate a given identity and ensue communication. Communication across VC-ISPs necessitate a federation manager, an entity responsible for aggregating both names and addressing information across all VC-ISPs. At the expense of a lookup to this manager, members of a particular VC-ISP can locate users at another VC-ISP. In our current hierarchy, the federation manager is located at a level above the cloud controller and central name server in order to provide a better view of the organization below. To better understand the third case, consider a scenario where a client with a global identity wishes to contact someone outside the node who is not part of the VC-ISP i.e a contact in a non-member network. Upon entering the contact's information, the identity management system in the local server generates a cache miss. A subsequent check of the central name server also results in a miss as the contact's naming and addressing information belongs to a third party telecommunication organization. On observing both misses, the local server redirects the communication request to the cloud server, which further forwards this request to the VoIP service provider from whom the global identity was purchased. It is now the responsibility of the service provider to appropriately route the communication request. Thus, any client with a global identity can communicate with a contact outside the VC-ISP architecture without entering any additional information and vice-versa. It is important to note that clients with local identities can effectively communicate with other clients within the VC-ISP using a similar mechanism of cache and central name server lookup to locate the address of an identity. The key difference is that the calls are routed within the VC-ISP architecture using OpenBTS and not a VoIP service provider.

3.4 Implementation and Evaluation of VC-ISP

In this section, we describe in greater detail how we implemented different parts to evaluate performance of the VC-ISP system.

3.4.1 Cellular And VC-ISP Simulator

We simulated a 3-level VC-ISP hierarchy in Python, to compare and contrast its performance with conventional cellular networks in the case of random failure events. The first level in our hierarchy was the cloud controller, which is assumed to never fail. The second and third levels are regular nodes (VCNs). This can be visualized as a tree starting at the cloud controller.

Simulation Procedure: The nodes at the second level contribute 60% of the simulated traffic, while the nodes at third level make up the remaining 40%. At periodic time intervals, random failure events occur, comprising of link failures between the two nodal levels or link failures between a node and the cloud. Thus, we are able to estimate the traffic that is dropped because of the corresponding failure event and the component it affects. Using Monte Carlo simulations, we were able to plot the average amount of failed/dropped traffic. This experiment was repeated after varying the number of nodes in various levels of the hierarchy. In the case of conventional cellular networks, it is assumed that the serving gateways and packet gateways are located in the cloud, and traffic is a value proportional

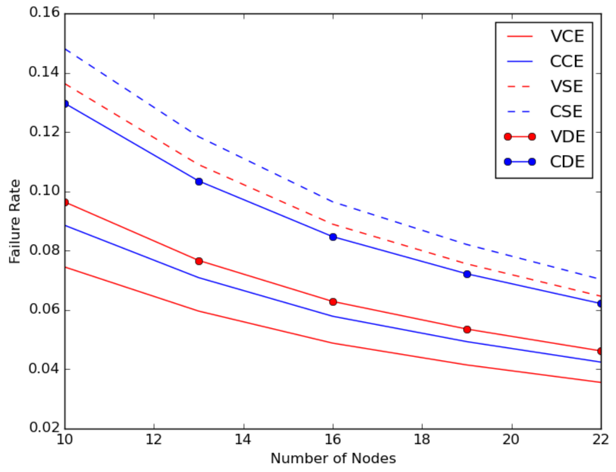


Figure 3.2: VCISP vs Conventional Cellular

to the total number of nodes.

In addition to this simulator, we implemented the two identity management models to compare them across the time taken to converge in issuing identities with different volumes of user activity. We have created a 10-node AWS deployment with nodes in different geographic localities to simulate multiple cloud controllers. The locations of the AWS datacenters include California, Oregon, Virginia, Singapore, Frankfurt, Sydney, Tokyo, Ireland and Sao Paulo. Each of the nodes run a Python-based HTTP server, and HTTP `GET` requests are used to simulate inter-node communication.

3.4.2 Simulator Evaluation

We discuss the results of the implementation in §3.4.1, where Figure 3.2 contrasts VC-ISP and conventional cellular networks. The latter have serving gateways and packet gateways far from the actual base stations. Once the link to the base station is down everything breaks. In the analogous case, while the link

Metric	Definition
VCE	VC-ISP Call Error
CCE	Cellular Call Error
VSE	VC-ISP SMS Error
CSE	Cellular SMS Error
VDE	VC-ISP Data Error
CDE	Cellular Data Error

Table 3.1: Metrics used in Figure 3.2 and their definitions.

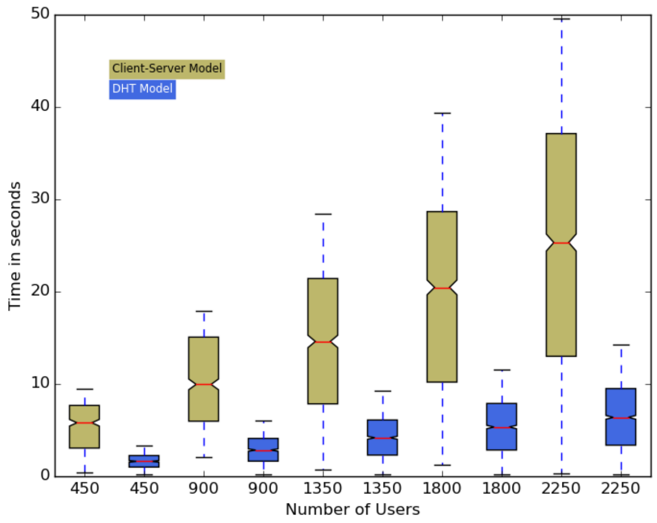


Figure 3.3: Latency for issuing identities: Single server vs 10-node DHT

between the cloud controller and the node is down, the node continues to provide local access. Therefore VCE, VSE and VDE (Table 3.1) values are always lower than their corresponding cellular counterparts. This general trend suggests that lower error rates are a consequence of a larger number of sites and users. This increases the total calls, ergo reducing the impact of a single failure (simulated within each time interval).

The box and whisker plot in Figure 3.3 compares the client-server model to the DHT-based model in terms of issuing a new global identity to a user. Issuing a new global identity entails issuing a serialized HTTP query to Nexmo. In the DHT-based model, the requests were generated by multiple processes in the node at Frankfurt while other nodes were responsible for processing the request and forwarding it to Nexmo. A synthetic hash function evenly divides the requests to the corresponding nodes that process it, without any location bias. In the client-server model, similarly generated requests were both initiated and responded to by the local host server on the Frankfurt node. It can be seen that the DHT outperforms the single server, even when the number of users is low. This is true as the cumulative processing power in a 10-node DHT exceeds that of a single node server.

3.5 Deployment

We built and evaluated a VC-ISP with three nodes, located in Kumawu (Ghana), New York City (USA) and Abu Dhabi (UAE).

VCNs: We used two hardware platforms as Base Transceiver Station (BTS) nodes. The first was a 1 Watt RapidCell from Range Networks [37], and the second was a

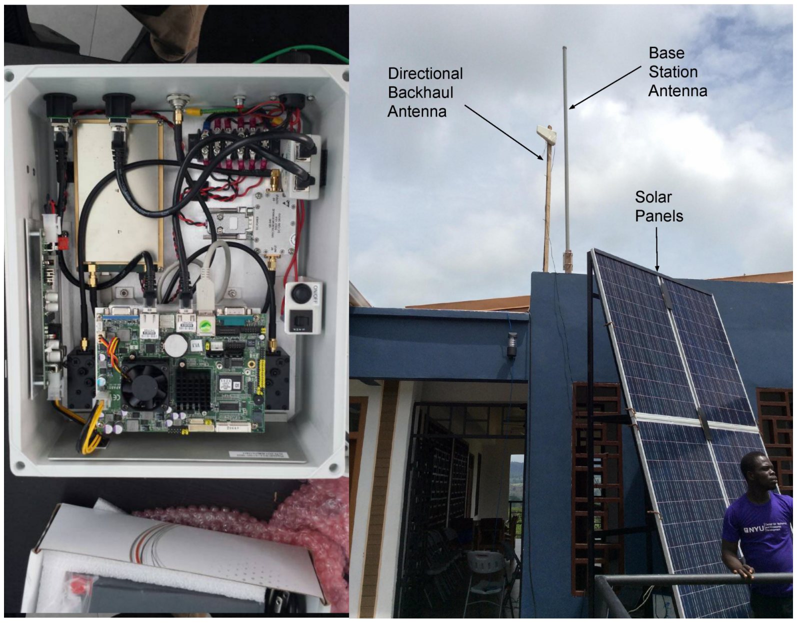


Figure 3.4: Our VCN in Kumawu Ghana. This VCN was deployed in a region at the edge of existing commercial cellular coverage and was connected to other VCNs using the naming and addressing mechanism explained above. It was also powered by solar panels.

10 mWatt USRP B100 from Ettus Research. We positioned one USRP B100 BTS in Abu Dhabi, another in New York, and finally positioned the RapidCell in Kumawu, Ghana (Shown in Figure 3.4). Nodes ran OpenBTS along with smqueue and sipauthserve which deal with SMS and user authentication respectively. Freeswitch was used for routing calls, messages and data. We placed a PC next to the BTS setup and referred to it as the local server. This PC hosted the local cache. It is important to note that even without this PC, the OpenBTS combined with smqueue, sipauthserve and Freeswitch is able to provide the functionality of a VCN.

Cloud Controller: We used an AWS EC2 instance hosted in Frankfurt, because the location is roughly equidistant from New York, Abu Dhabi and Kumawu. The instance had a memory of 1 GiB and was allocated a single vCPU. Our EC2 instance was running Ubuntu 14.04, and all application support on the cloud was

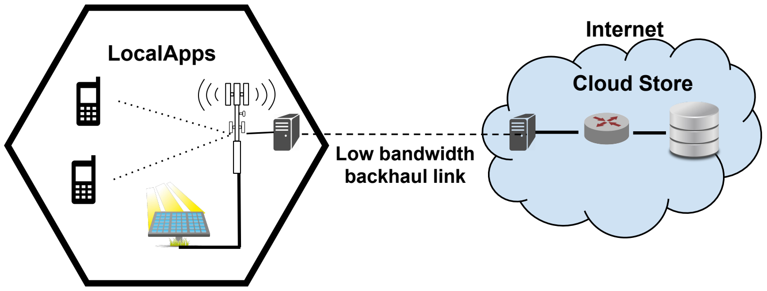


Figure 3.5: GreenApps System Architecture. GreenApps architecture is described in detail in section 3.6

written using Python. The specs of the instance suffice for application support as it met the minimal required processing.

Experience: We successfully used our 3-Node VCN to conduct calls and text messages between mobile devices connected to two different VCNs with acceptable latency and decent call quality. We also conducted several calls and SMS to users connected to conventional cellular networks. As explained earlier for calls and SMS routing, a local cache was first checked and when the location was not in the same VCN, the call was then routed to the cloud controller. The cloud controller then routed the calls to the appropriate VCN. In cases where the network was not available the text messages and voice messages were saved locally at the node until the network was available.

3.6 GreenApps Architecture

Our GreenApps platform is shown in Figure 3.5. GreenApps consists of the following 4 main building blocks: (1) an inexpensive open-source GSM base station; (2) a local server; (3) new mobile applications that we built for specific communities; and (4) an Internet backhaul link, whenever available, for opportunistically synchronizing local apps with the cloud store. The main goal of GreenApps is to satisfy the essential needs of local communities via mobile applications that can operate under extreme rural conditions with no power, and no cellular or Internet connectivity. Under the condition where an Internet backhaul link is available, GreenApps exploits this opportunity to provide services distributed across multiple cellular base stations.

3.6.1 Enabling Edge Application Support

GreenApps supports SMS, voice and data-centric applications locally using an approach which is completely different from how applications are built in conventional cellular networks. In the conventional cellular architecture, messages are routed from base stations to the Internet-facing edge in the core network (after traversing various entities in the core). The Internet-facing edge then routes messages to specific data centers where applications are hosted. While this architecture is designed to be consistent and scalable, it requires reliable links from the base station to the Internet-facing edge in cellular core network. Contrary to this our goals are to reduce dependence on backhaul and to simultaneously provide application support for applications to run locally at the base station. To achieve this, we need to route messages in a scalable manner and prevent any unintentional blocking of critical user communication in this process. We introduce hooks in the routing software at the base station in order to route messages to applications based on the destination short code and send them to the appropriate application engines.

Our platform leverages the hooks and already available APIs to build useful applications in base stations that use the OpenBTS or Osmocom stacks. For each application, the local server coupled with the base-station runs the applications locally and the messages from clients are forwarded by the base station to the correct application. To route an SMS message to a particular application based on the application short code, we used different tools for the OpenBTS and Osmocom software stacks. The OpenBTS stack usually works with `smqueue` to route all messages. Instead we routed the messages to Freeswitch `mod_sms` module specially designed to handle SMS traffic. `mod_sms` enables us to route the messages using a chat-plan defined through XML to route SMS based on the destination phone number. In the Osmocom stack, the `RCCN` library is used to route messages in conjunction with osmobsc. All received messages are routed to a local python server which routes appropriate responses back into the network using the same library. We modified the code of `sms.py` in `RCCN` to route the messages intended for applications (based on the destination short code) to our local application server. Given the current operational constraints of the OpenBTS and Osmocom stacks, we believe a single compute server suffices for running several applications without any performance penalties.

Across both the stacks, we leverage the Asterisk PBX setup for enabling IVR server applications. The Asterisk software is installed locally and all calls to the IVR application short code are bridged to Asterisk. The users can listen to a menu of options and navigate by pressing various keys. Enabling this feature is a standard functionality available in Freeswitch and any other PBX software.

3.6.2 Identities and Distributed Applications

A key building block for supporting different forms of applications in GreenApps is an identity management layer. GreenApps supports three basic types of identities: network identities, user identities and application identities. In GreenApps, the common user identity is the the International Mobile Subscriber Identity (IMSI) which is physically written on a SIM card. The network identity is the identity issued by the network against the IMSI. This is typically the phone number of the user, in our case this refers to a local phone number identifiable within the base station.

In GreenApps, distributed applications can be built by using application specific identities, which refer to unique information state of the application that may be tied with one or a group of user identities. The identity space corresponding to an application can be viewed as a hash table or a key value store. For example, consider the case where two users wish to perform a BUY/SELL transaction in GreenApps across multiple cellular sites connected via a server in cloud; each item transacted is associated with a unique application identity and each buy/sell bid is a mapping between an application identity and a set of user identities. The complete key/value store of such identities in an application forms the application state. The application state is manipulated by simple put/get operations when the application is running locally, but if an application is running in a distributed manner across multiple base stations, then the local state across each base station needs to be selectively synchronized with the cloud state. To achieve this, we provide primitives for distributed applications running across multiple base stations.

3.6.3 Synchronization Primitives

In GreenApps we provide some primitives for synchronization between cellular nodes and a cloud based server. These primitives are available to all applications using a python library. These primitives provide a flexible and convenient way to upload data to the server and enable distributed applications when the backhaul is available. Each primitive gives the application a choice of synchronization in a slightly different way based on urgency of synchronization for a particular application functionality.

1. `SLOWPUT(Identity i, Type t, Data d):`

GreenApps is designed to provide local applications but it is designed to provide synchronization with the cloud if there is a need. To this end, for each application identity the application logic is provided a method to define synchronization requirement. The Internet link between the base station and the cloud has a low bandwidth-delay ratio and is typically intermittent. This raises questions regarding what identities need to be synchronized to ensure high availability to a user while having consistency in the application state at the cloud. First, to ensure high availability of identity lookup, GreenApps supports quick local authentication of user identities; pre-registered users and two users within the same base station can communicate with each other without the need for backhaul connectivity to the cloud. Second, we provide synchronization support for each of the application identities so that users can pick and choose the consistency over availability where necessary by ensuring the data is saved on the cloud before it is saved locally.

In the event, the GreenApps base station has a backhaul link with moderate bandwidth, GreenApps can enable lazy synchronization of application state between the local server instance and a cloud instance of the same application to enhance end-to-end reliability. GreenApps supports an application agnostic primitive *SLOWPUT*, which is a bandwidth-aware mechanism to synchronize data between the local node hosting applications and a cloud data store. SLOWPUT provides a controllable way to synchronize application state to the cloud. We achieve this by maintaining a common queue across various applications. This queue contains different data items(from different applications) that need to be uploaded to the cloud and uses a fair queuing mechanism to allocate the bandwidth between competing requests.

We used Memcached [49] to provide the functionality of lazy synchronization. Memcached is a high-performance, distributed memory object caching system which can be used for general applications. A queue is maintained using a linked list of different objects in cache. An item is added to the queue if an application executes the SLOWPUT primitive. There is a separate program that is always running and keeps uploading the items in the queue if the bandwidth is available. The items are uploaded on a first come, first served basis. The function call for SLOWPUT is as follows: `SLOWPUT(Identity i, Type t, Data d)`. In this call, user `i` communicates marshalled data `d` for application `t`.

`2.  FASTGET(Identity i, Type t, Data d):`

This primitive is used by the local application to send and receive data from the server in an *active* manner. The parameters include a user `i` communicating data `d` for application `t`. This primitive is usually called as a result of an application function when there is a higher urgency needed for consistency in application state. The communication is not queued, as in `SLOWPUT` primitive. Instead, this primitive is executed instantaneously, provided the the backhaul link is available. For example, any inconsistencies in the case of a banking transaction are grave, and a strong consistency has to be ensured even if the user does not get a response immediately. Such applications use the `FASTGET` primitive and marshall the transaction in the data parameter, forwarding it to the central server of the banking application.

3.7 Applications

In this section, we describe three applications we deployed using GreenApps in Ghana and Nicaragua. These applications are used for various useful functions in the communities like dissemination of information, entertainment and communication.

3.7.1 Fishline

Fishline is an SMS-based application that enables small business owners to send out mass advertisements over SMS to people in the community. This application has been built and is running in the Pearl Lagoon GreenApps platform. To motivate the need for Fishline we conducted 6 qualitative interviews in Pearl Lagoon. People interviewed included 2 artisan fisherman, 2 deep water fishermen, and 2 owners of fishing cooperatives. We chose various people associated with the fishing community because that is how most people earn a livelihood in the area. This cross section of fishermen represents the 3 basic types of individuals working in the local fishing economy. We asked them questions about demographics, their fishing practices, and the economics of Pearl Lagoon. All the interviewees suggested interest in the idea of an application to advertise fish for sale. Fishing cooperatives also saw a use for the Fishline, since they would be able to get messages earlier before fishermen are on shore, and can prepare for large catches as they arrive. Fishline is an entirely local application and no synchronization primitives were used in the Fishline application.

3.7.2 P2P Transactions Marketplace in Ghana

We built a P2P transactions application that uses an SMS-based interface to enable traders to sell commodities to customers in the community. We designed our application in a way where farmers and traders could look up the prices that others are offering for the product before deciding whether they want to sell or buy from a particular user. There are three kinds of messages sent, SELL, BUY and SEARCH. The SELL message is of the format,"SELL <commodity> <amount> <price>". For example, if farmer wants to sell 5kg of corn for 10 cedi[1], then, the message "SELL corn 5kg 10" is sent via SMS, and this message is saved in a database along with the user's number who sent it. Subsequent people who are interested in buying or looking up the price can do so using a BUY or SEARCH SMS. This application was deployed in our Ghana setup.

Whenever the messages match a short code for P2P application, the message is routed to the HTTP server hosting the application code. The lazy `SLOWPUT` is used to send data to the server on sale of a commodity. Therefore when a user tries to sell, we acknowledge the sell has been received locally(by sending a message back to seller) and transfer the sell to the cloud in a lazy manner. In case the user is interested in purchasing a commodity, `FASTGET` is used, and the data returned is sent to the user over one or more text messages. This is because we want strong consistency and do not want to give user a response immediately without ensuring that we are actually able to update the record in the cloud.

[1]cedi is the local currency in Ghana

3.7.3 IVR-Based Social Media

Traditional IVR systems have required access to either a cloud based service or have relied on a local server with a cellular dongle connecting to a local cellular network service. The key benefit of the GreenApps platform powered IVR application is to enable an off-line and off-grid IVR application. By attaching the server functionality to the base station, the application server functionally locally resides on the programmable base-station without the need for backhaul connectivity.

The key building block for this application is the local code that glues the IVR to the cellular system. The important function is the `ReceiveHandler` running at local server, which is invoked using a HTTP request on entering the IVR extension. After the initial HTTP request, the call is bridged from base station to Asterisk IVR server running locally. The Asterisk server logs each input of user as entered. If the user is recording a message, the `ReceiveHandler` is executed after its completion. The `ReceiveHandler` uses the lazy `SLOWPUT` primitive to send the recorded file to the cloud component for lazy synchronization.

3.8 Application Performance and Experiences

We briefly outline some of the early implementation and deployment experiences in running these applications.

Marketplace Application: As a basic setup we had two base stations, one in Ghana and the other one in New York. We developed an Android application that enables phones to automatically `SELL` and `BUY` items in a periodic manner. We had 4 phones sending `SELL` messages and 3 phones sending `BUY` and `SEARCH` messages. 5 of the 7 phones were connected to the BTS in Ghana and the others were connected

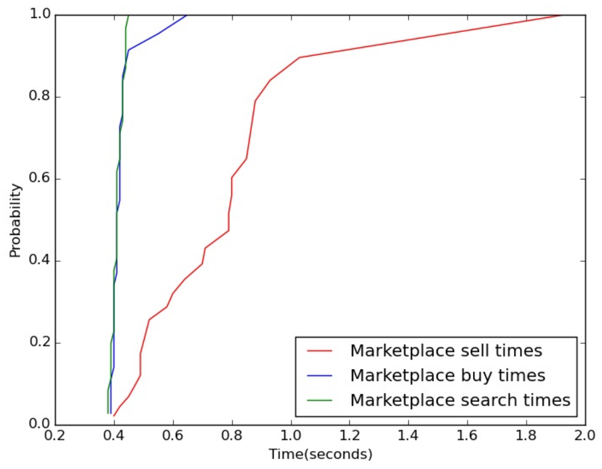
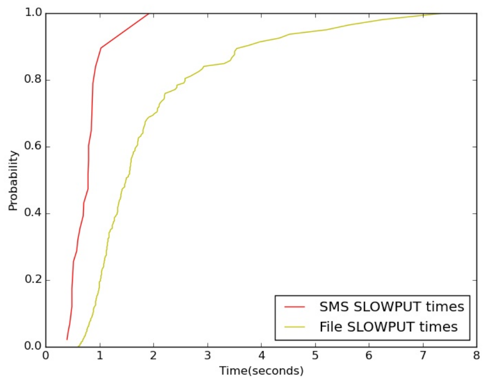


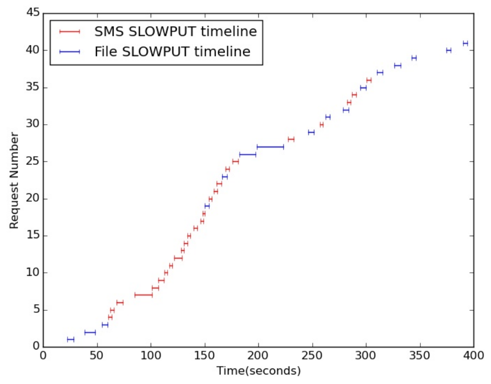
Figure 3.6: CDF of time it takes for the different functions in the P2P Marketplace application

to the base station in New York. Simultaneous with the above application we made calls manually to the IVR application and ran a script at local server to generate data. This helped us simulate the local component receiving data from IVR and sensing applications.

We report the performance of the P2P transactions application in Figure 3.6. The graph above shows the end-to-end latency between the local application and server for the `BUY`,`SELL` and `SEARCH` operations. It is important to recall that the `SELL` operation translates to the `SLOWPUT` primitive, while `BUY` and `SEARCH` translate to `FASTGET` primitives. It can be observed in Figure 3.6, there is a clear distinction between the time it takes for `SELL`, compared to `BUY` and `SEARCH`. This is expected because `SELL` requests are queued and processed in a lazy manner while the BUY and SEARCH are executed using the urgent FASTGET primitive. It is important to note that no failures were experienced and all messages received at base stations were transferred to the server. At a user level user got SMS back for all the functions they performed. They got SMS for `SELL` immediately because



(a)



(b)

Figure 3.7: CDF of time taken for the `SLOWPUT` of file vs SMS and a view of the queue in the P2P Marketplace application

the local component of the application gives the response immediately and then executes the `SLOWPUT` operation while the response for `BUY` operation is sent only after response from cloud to maintain consistency.

IVR Application: File upload is a common operation of the voice-based applications. Figure 3.7a shows the CDF of the time it takes for different `SLOWPUT` requests on the New York-based node. Figure 3.7b shows the queue and how requests are processed for the `SLOWPUT` primitive for the base station in Ghana. Figure 3.7 shows that SMS-based applications will work faster than the file-based applications as SMS-based messages are smaller than files. It also shows that file-based applications can slow down the SMS-based applications. This is true as both are placed in a common queue for processing, and servicing larger files is more time consuming than servicing smaller SMSs.

3.9 Summary

The contributions of this chapter can be summarized as below:

- We present implementation and deployment of Virtual Cellular ISPs, a new bottom-up cellular network architecture that enables connectivity in uncovered regions. Unlike the conventional monolithic cellular network architecture, the VC-ISP model enables a distributed collection of open, programmable cellular base stations to communicate using a novel intermittency-aware naming and addressing mechanism. We use a simulator and real world deployment of VC-ISP across 3 different countries.

- We present GreenApps which provides useful abstractions over nodes in VC-ISPs.

- GreenApps was leveraged to build real-world applications which were used by different rural communities.

Chapter 4

Model-Driven Interpretable Congestion Control

Several new congestion control protocols such as Sprout [50], PCC [51], Verus [52] and BBR [53] have demonstrated significant performance gains against traditional TCP variants under different network settings. A common recurring theme across many of these protocols is to use *delay-based signals* as a way to measure the network congestion state and react quickly to varying network conditions. While there have been a broad array of research that have studied the dynamics of loss-based congestion control protocols [54, 55, 56], we still lack a principled framework for understanding the dynamics of delay-based protocols.

In this chapter, we propose *Model-Driven Interpretability (MDI)*, a new framework that aims to enhance our ability to interpret the behavior of delay-based congestion control protocols in highly variable network channels. Given any protocol, our MDI framework uses empirical data on the protocol's performance on variable network channels to train a stochastic two-dimensional discrete-time Markov model to represent the behavior of the protocol. In essence, using empirical behavior of a protocol across diverse network conditions, MDI converts a protocol into a stochastic random walk in a Markovian state space. Each state transition in this state space is determined by the delay variation feedback from the network. The MDI version of a protocol aims to achieve two key properties:

1. The MDI version of a protocol aims to closely approximately the statistics (e.g., mean and variance) of the throughput and delay distributions of the original protocol.
2. The MDI version of a protocol aims to track the temporal behavior of the original protocol, i.e., how the protocol reacts to variations in network conditions.

We note that achieving these two properties for a broad array of protocols is a non-trivial task. Many recent protocols with complex control loops, by definition, do not obey a Markovian memoryless process. In the MDI framework, this notion of protocol memory is implicitly captured in the definition of the state space, transition probabilities in this space and the stochastic random walk using delay feedback. While the state space does represent a significant approximation to the original protocol, we show that in practice, the MDI version of several protocols successfully approximates the behavior of the original protocols.

MDI makes it easy for us to reason about properties of congestion control protocols. Examples of these are:

1. *Understanding Protocol vs Network Variability Interactions:* MDI can be used as a tool to measure how state space transitions vary: (i) across protocols under the same network condition; (ii) across network conditions under the same protocol.

2. *Reasoning about Convergence and Fairness:* By representing a protocol in a Markovian state space, we can apply conventional Markov model mixing time concepts and compute the mixing time of a Markov chain and the corresponding stationary distribution across the state space. Across all the four protocols, we show that the empirically measured statistical distribution of a protocol closely mirrors the stationary distribution of the MDI version.

To evaluate MDI, we have developed MDI versions of three different protocols: Verus, BBR and TCP Cubic. We compare the behavior of the original protocols and the MDI versions of these protocols in realistic network environments using real-world cellular traces in 3G and 4G networks. Across most of these scenarios, we demonstrate that MDI closely approximates throughput and delay statistics of the original protocol as well as temporally tracks the behavior of the protocol. We hope that a key longer-term benefit of the MDI approach is to leverage the large body of statistics literature on Markov models and random walks to understand the stability, dynamics and adaptivity of delay-based protocols.

4.1 Related Work

Congestion control for cellular networks: The most widely used TCP variants, is Cubic [57] but it is known to perform poorly in cellular networks [50]. This has led to newer delay-based congestion control protocols like Sprout [50] and Verus [52] that were specifically designed for the context of cellular and other variable network channels. While Sprout focuses on the problem of reducing self-inflicted queuing delays and uses packet inter-arrival times to detect congestion, Verus was designed to create a balance between the packet delays and the throughput. BBR [53] was recently proposed by Google and has shown good performance. BBR uses the round trip propagation time and bandwidth of the bottleneck link to find the optimum operating point for congestion control.

Applying machine learning to congestion control: The new congestion control protocols being proposed have complex control loops, which makes them harder to understand in the context of different network conditions. The recent development of congestion-control protocols that employ machine learning (e.g., Remy [58], Vivace [59] and Indigo [60]) has only compounded this issue (e.g., some of Remy's congestion control protocols employ rule tables with more than 100 rules). Weinstein et. al. (Remy) [58], Sivaraman et. al. [61] and Pötsch [62] have provided different methodologies to model non-linear congestion control from a theoretical perspective.

Analyzing TCP behavior: TCP and its variants have been thoroughly studied using modeling and analytical techniques [63, 54, 55]. In a recent work called ACT [56] we see the concept of a guided random walk in the state space of implementation variables to find regions where the algorithm should never go, thereby indicating the existence of a possible bug in the implementation. This approach of automated model-guided method is also followed by others as well [64] to explore the variable space in the implementation of a congestion control algorithm. Our modeling approach also uses a random walk but our state space is limited to a delay and window variable and our goal is to not reach unreachable points, but to guide the model to follow the native algorithm it is modeling.

Emulation and testing environments: Network emulators like Dummynet [65], NetEm [66], Mininet [67] and MahiMahi [68] are useful in emulating real world links in order to train any model to learn the behavior of a congestion control algorithm. We used MahiMahi because it provided a simple shell based interface to emulate many different link conditions using real cellular network traces. In addition to this, MahiMahi can be used to test application layer and kernel based implementations of congestion control protocols.

4.2 MDI Design

The main idea behind MDI is to build a model that reflects the statistical properties of the modeled protocol with the goal of providing a more intuitive and predictable understanding of the protocol behavior. At an abstract level, MDI assumes that any congestion control protocol can be modeled by the relationship between the current and the next state, where each state is a tuple that represents the relative change in the network delay and the sending window size. These states are used to construct a two-dimensional discrete-time Markov chain that can mimic the behavior of a specific congestion control protocol.

4.2.1 Modeling Delay based Control

Consider a protocol P that uses delay variations as a congestion signal. One can imagine such a protocol maintains a recent history of delay observations, using which it estimates the next sending window or rate. Let us consider an epoch to represent the unit of time that a protocol uses to make a decision; this epoch can be a variable period of time or maybe fixed for specific protocols.

The challenge in constructing a Markov model representation of a protocol P is in determining the appropriate state space and mapping the protocol actions to transitions between the states. The most straightforward approach is to map the absolute values directly by describing a state as (d_i,w_i) where d_i and w_i are the experienced delay and the sending window respectively in epoch i. For brevity, we use d and w (without the epoch subscript i) to abstractly represent the observed delay and window parameters. While a two-variable state space using (d, w) is simple, it may not be rich and generic enough since it may not be sufficient to capture the variations in these parameters. If one were to represent the state space using a history of delay measurements and window measurements, the state space representation can be much richer, but correspondingly much harder to accurately learn. In fact, for each additional dimension of representation in the state space, we need an order of magnitude more training data to learn the state transitions of a protocol.

To make the state space representation richer, in addition to the delay d, and window w, we chose to use two additional parameters: (1) the relative change in the delay across neighboring epochs (captured by $\alpha(d)$); (2) the relative change in the window across neighboring epochs (captured by $\beta(w)$).

While these four parameters do provide a much richer representation of the transitions of a protocol, the amount of training data required to accurately train the protocol transitions in the 4-dimensional space is at least two orders of magnitude more than the training data required in the original (d, w) space. To strike the right balance between state complexity and state richness, we chose to condense these four parameters into two composite parameters represented by $\alpha(d) \cdot log_{10}(d)$ and $\beta(w) \cdot log_{10}(w)$. We chose these functions to derive a compact representation of the window and the delay transition state space while considering their relative transitions across epochs.

4.2.2 Discrete-time Markov Model States

A discrete-time Markov model of a protocol is represented in the form of a state-transition probability matrix. The matrix describes the probabilities of going from one state to any other state at each transition. The state space and the transition probabilities are obtained through training by running the protocol on a predefined set of network configurations. We refer to this as the training phase of the Markov model. In order to capture the protocol behavior, the matrix should include as many states as possible that can be observed during the training phase. Let us consider an epoch to represent the unit of time that a protocol uses to make a decision; this epoch can be a variable period of time or maybe fixed for specific protocols. A particular state within the transition matrix is defined in the form of tuples with value pairs of $(\hat{d_i}, \hat{w_i})$, where $\hat{d_i}$ is calculated using the value of packet delays in the current epoch (d_i) and the previous epoch (d_{i-1}). Similarly, $\hat{w_i}$ is calculated using the value of sending window in the current epoch (w_i) and the previous one (w_{i-1}) as follows:

$$\hat{d_i} = \left[\left(\frac{d_i}{d_{i-1}}\right) - 1\right] * log_{10}(d_i) \tag{4.1}$$

$$\hat{w_i} = \left[\left(\frac{w_i}{w_{i-1}}\right) - 1\right] * log_{10}(w_i) \tag{4.2}$$

We use the tuple $(\hat{d_i}, \hat{w_i})$ to represent a state because it captures a 4-dimentional space helping us train the MDI model with less training data. $\hat{d_i}$ captures the value of delay and the relative change in delay while $\hat{w_i}$ captures the value of congestion window and the relative change in window.

4.2.3 Model Derivation

To derive the transition matrix representing the model of the protocol, we use a protocol emulation strategy in a constrained network environment similar to Remy [58] where a user is constrained by a network model and a background traffic model. Consider a network emulation environment where one can execute the protocol P under a variety of network conditions and background traffic. Here, we perform an array of network emulations by varying the network traces, packet loss rates and RTT to emulate constrained real-world scenarios. For each network emulation of the protocol P for a specific user, we observe all possible state transitions of $(\hat{d}_i, \hat{w}_i)$, with $\hat{d}_i$ ranging from $\hat{d}_{min}$ to $\hat{d}_{max}$, and $\hat{w}_i$ ranging from $\hat{w}_{min}$ to $\hat{w}_{max}$. Using these state transitions a two dimensional Markov chain is created.

The transformation of continuous delay and window values to discrete $\hat{d}$ and $\hat{w}$ values for representation in model has to capture the essence of the underlying protocol. This transformation is controlled using a few hyper-parameters. The range of possible $\hat{d}$ and $\hat{w}$, and the quantization of this range are some of these hyper-parameters. We chose appropriate values of these hyper-parameters through rigorous experimentation and we hope to automate this process in future work.

4.2.4 State transition in the MDI model

Assume that a protocol P adjusts the congestion window as a function of delay feedback. Assume that a user executing the protocol P currently has the following values: the current sending window w_i, and the previous epoch delay feedback d_{i-1}. In order to decide on the value of the next window w_{i+1}, the user first has to observe the current delay feedback d_i. The congestion control protocol P

decides on the next window w_{i+1} based on the following factors: the protocol P, the prior window w_i, and the delay variations. Upon observing d_i, P which was at state $(\hat{d_i}, \hat{w_i})$ now has the value of $\hat{d_{i+1}}$ in the model space of the protocol. From training of the model we know the possible values of $\hat{w_{i+1}}$ and the probabilities of ending up in those values. In order to decide which $\hat{w_{i+1}}$ is chosen in the next epoch, MDI randomly samples a value of $\hat{w_{i+1}}$ from the probability distribution for that $\hat{d_{i+1}}$. This process is essentially a table lookup followed by a probabilistic decision within the state transition probability matrix.

The key assumption that MDI makes while modeling a protocol is that the state transition from $(\hat{d_i}, \hat{w_i})$ to $(\hat{d_{i+1}}, \hat{w_{i+1}})$ can be captured by a guided Markov model with two basic properties. First, the delay feedback determines the direction of the window change (increase or decrease), in essence *guiding* the direction of state change. Second, the delay variations of d_i and d_{i+1} have an inherent randomness that influences the protocol's choice of the next window w_{i+1}. The guided Markov assumption is clearly an approximation of the original protocol behavior.

4.2.5 Evaluating MDI matrix

The model representation provides a lookup table of a protocol which can be plugged into an implementation of Algorithm 1 as a replacement for congestion control loop at transport layer to replicate the behavior of the protocol being modeled. The produced output of the algorithm in a test setting can be used to empirically compare its performance against the native algorithm. Lines 3 to 8 in Algorithm 1 deal with states that are outside the transition matrix. The constant multipliers help multiplicatively increase/decrease the window variable to bring the delay variable up/down to values represented within the bounds of

Algorithm 1 MDI pseudocode

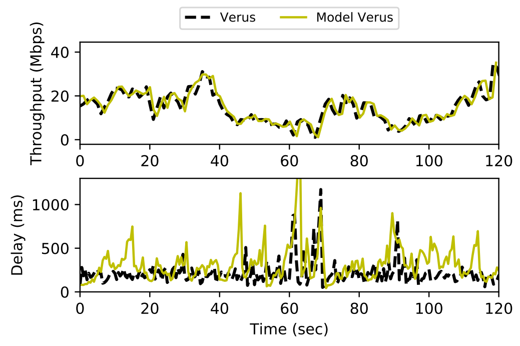
```
1: while TRUE do
2:     Compute $\hat{d}_{i+1}$ from ACKs
3:     if $\hat{d}_{i+1} < \hat{d}_{min}$ then
4:         (Increase $\hat{w}_{i+1}$ using constant multiplier $c_1 > 1$)
5:         $\hat{w}_{i+1} \leftarrow \hat{w}_i * c_1$
6:     else if $\hat{d}_{i+1} > \hat{d}_{max}$ then
7:         (Decrease $\hat{w}_{i+1}$ using constant multiplier $c_2 < 1$)
8:         $\hat{w}_{i+1} \leftarrow \hat{w}_i * c_2$
9:     else
10:        Lookup matrix for possible values of $\hat{w}_{i+1}$
11:        $\hat{w}_{i+1} \leftarrow$ Randomly sample next $\hat{w}_{i+1}$
12:        Compute new congestion window given $\hat{w}_{i+1}$
13:        $\hat{d}_i \leftarrow \hat{d}_{i+1}$
14:        $\hat{w}_i \leftarrow \hat{w}_{i+1}$
15:    sleep(epoch)                                    ▷ epoch set to 20 ms
```

the model. In Lines 10 to 14, we are just doing a lookup in the transition matrix and picking up a $\hat{w}_{i+1}$ based on probabilities of different possible values of next state based on previous state $(\hat{d}_i, \hat{w}_i)$ and the $\hat{d}_{i+1}$ calculated using delay feedback in the current epoch. We devised this algorithm for evaluating how accurately our model captures the native algorithm.

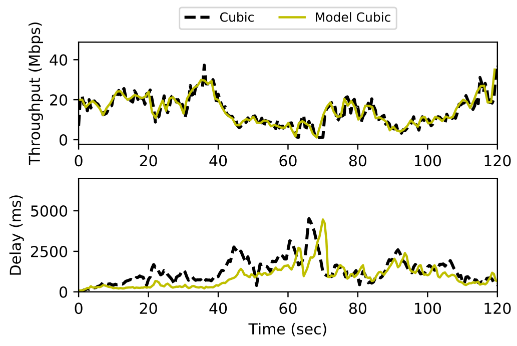
4.3 Model vs. State-of-the-art Protocols

4.3.1 Evaluation methodology and setup

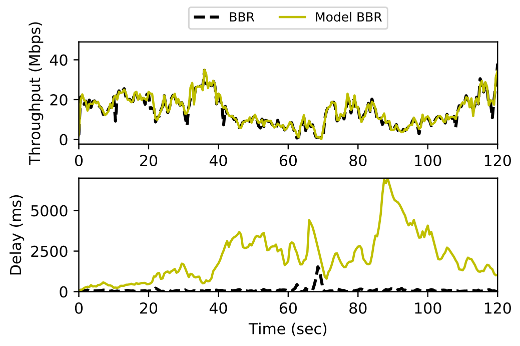
In this section, we evaluate and compare the performance of the generated MDI model in contrast to the original/native version of the protocol. We evaluated 3 different state-of-the-art congestion control protocols: Verus, Cubic, and BBR. The protocol training was done using two major sources: i) a large set of cellular channel traces derived from six long unique cellular traces our team collected in



(a) Verus



(b) Cubic



(c) BBR

Figure 4.1: MDI vs. original algorithm performance over 4G Verizon test.

UAE and Germany, ii) emulated links from Pantheon [60]. In ii), we increased the number of emulated links by changing delay, loss rate and router queue size for the original emulated links. There were 100 5-minute long traces in the cellular dataset, and 42 5-minute long traces in the Pantheon dataset of emulated links. In order to replay these traces, we used a cellular network emulator. The network emulator consists of a server and a client connected through MahiMahi [68]. MahiMahi replays the traces to emulate network links with different bandwidth, delay and loss characteristics. Using this emulator we ran each of the congestion control protocols and collected logs of the protocol behavior under every trace. Recall that for the training we had to save the congestion window of each of these protocols and then correlate them to the delay they experienced.

The same setup was used for the testing scenarios when evaluating the model version of the protocol and comparing it to the original native version. To test the trained models, we developed a simple sender that took as input the model and made a decision on choosing the next congestion window (as in Algorithm 1) based on the state transition matrix representing the model and the experienced delay over an epoch interval of 20 ms. This value represents the frequency at which an algorithm makes a decision to update congestion window. Although we fixed this value across algorithms, it should ideally be tuned automatically for the algorithm being modeled. We have left this for future work. As mentioned in § 4.2.4, we use the observed delay to do a guided random walk in the transition matrix. If the observed delay lies outside the boundaries of the state space, then we employ constants c_1 and c_2 as described in § 4.2.5 and Algorithm 1. We used values of 1.1 and 0.95 for c_1 and c_2 respectively. The probability of using these boundary conditions instead of making a decision using the model is defined as the *escape probability*. This is equivalently the probability that the observed delay lies outside the boundaries of the state space.

We used a set of cellular traces for testing. These testing traces were taken from a number of previously published papers:

- 4G Verizon: taken from [50] and represents a recorded channel over Verizon's 4G network in the US. The median capacity of the trace was 9.79 Mbps.

- 3G Etisalat: taken from [52] and represents a recorded channel over the Etisalat's 3G network in the UAE while driving on a highway with 120 km/h. The median capacity of this trace was 15.68 Mbps.

Table 4.1: Pearson Correlation Coefficients (PCC) for both delay and throughput, between the original protocol and MDI version for each quantity, and the escape probabilities, for each algorithm and test trace.

4G Verizon			
Algorithm	Delay PCC	Throughput PCC	Escape Prob.
Verus	0.62	0.92	11.4%
Cubic	0.75	0.92	1.43%
BBR	0.37	0.96	1.09%
3G Etisalat			
Verus	0.32	0.86	20.8%
Cubic	0.52	0.91	1.59%
BBR	0.47	0.92	1.47%
Rapidly Changing Network			
Verus	0.23	0.85	22.9%
Cubic	0.51	0.95	1.39%
BBR	0.33	0.93	1.59%

- Rapidly changing network: inspired by [51], this trace represents a network with a highly fluctuating channel, where the capacity varied randomly every 5 seconds. The median capacity of this trace was 9.99 Mbps.

4.3.2 MDI: Tracking the original protocols

We ran the different protocols across multiple traces in the three test network scenarios. For each protocol, we present results on how the throughput of the congestion control scheme and its delay varies over a snippet of 120 seconds. We present these results in Figure 4.1 and in Table 4.1. Figure 4.1 shows the graphs for all three protocols and how they compare to their MDI model versions temporally when testing on the Verizon 4G trace. For all of the test traces we report empirical numbers of Pearson Correlation Coefficient (PCC) and the escape probabilities in Table 4.1.

The key observations are four-fold. First, across all these protocols, the MDI version is able to accurately track the throughput which demonstrates the power of MDI to accurately model the throughput behavior of these protocols. Second, for these protocols, MDI tracks the delay behavior approximately in some cases. For Cubic and Verus protocols, the delay statistics (mean and variance) match well between the MDI versions and the native protocols. Third, for BBR, we notice a discrepancy between the observed delays in the MDI and the native versions. We note that achieving perfect temporal agreement in the observed delay between the model version and the native version of a protocol is a very difficult property to achieve. For specific protocols like BBR, the mismatch in the delay properties comes at the expense of a strong correlation in the throughput patterns between the model and the original version of the protocol. Finally, even though we have omitted the corresponding plots for the other network scenarios (3G and rapidly changing networks) due to lack of space, these observations hold across all the three types of network scenarios.

Verus tracking results shown in Figure 4.1a show that there is a good correlation in both throughput and delay. The delay experienced with MDI Verus is a little more jittery, but it is important to note that MDI Verus keeps the end-to-end delay low (see y-axis compared to Cubic), just like native Verus. Temporal tracking results of Cubic and MDI Cubic are shown in Figure 4.1b. The MDI version of Cubic is able to approximately capture the bufferbloat dynamics of Cubic in terms of delay. MDI Cubic has very similar delay compared with native Cubic. Based on the temporal tracking results for MDI BBR in Figure 4.1c, we observe that MDI BBR has very similar throughput across time but the delay experienced by MDI BBR is much higher than the delay experienced by native BBR. We believe this may be due to a poor choice of hyper-parameters in the model representation for BBR. We also believe that the Markovian assumption of BBR behavior may not be entirely correct.

In summary, we observe that MDI does have the ability to accurately track the throughput behavior of all the protocols across highly variable network conditions. MDI can approximately match the mean and variance of the delay statistics for Verus and Cubic. For BBR, achieving temporal agreement in delay statistics has been challenging for the model. To put this in context, achieving perfect temporal agreement (for both throughput and delay) across diverse network settings is an arduous if not an impossible task.

4.3.3 Effect of Network on MDI matrix

We trained the Markov models under a set of varying network conditions as a means of understanding the impact of such variations on the output models. As mentioned earlier, we had two separate datasets we used for training—the cellular traces and emulation links from Pantheon. We wanted to see how dependent is the MDI model on the actual traces that are used for training.

Figure 4.2 shows the sparsity structure of the MDI matrices trained for each different network condition across all the protocols. The top row shows models trained using the cellular traces and the bottom row shows the models trained using the Pantheon emulated links. Each subplot in the Figure is a “spy” plot of the MDI matrix (referring to the MATLAB command `spy`) which puts a black dot in locations with nonzero values and white everywhere else. The values on the x-axis and y-axis of the sub-plot are a condensed state space because the original state space is much bigger.

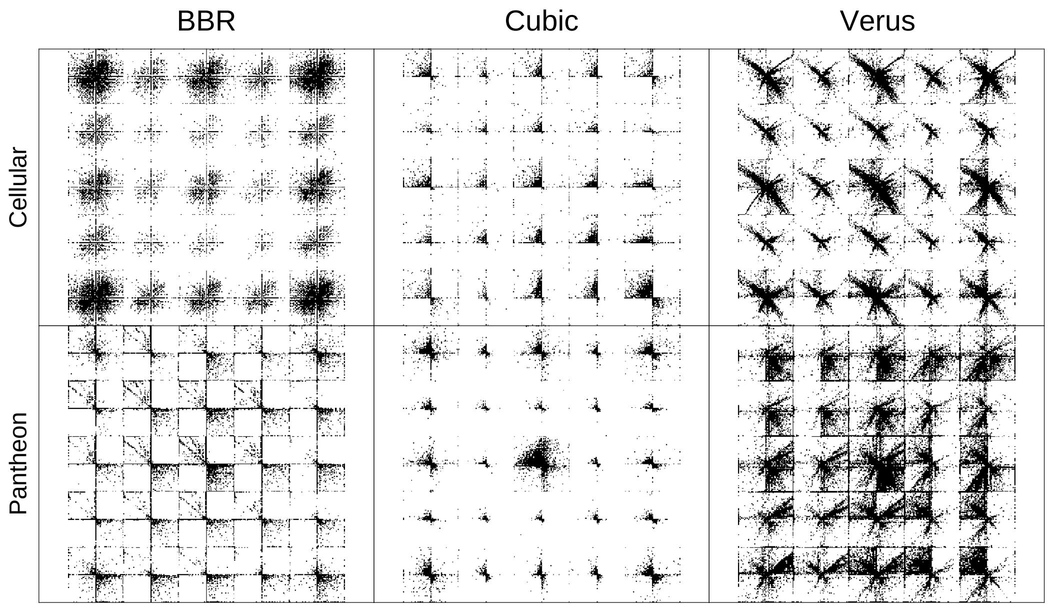


Figure 4.2: Matrix sparsity structure (nonzero entries shown by black dots) for matrices trained with different training data. The top row was trained using the cellular traces while the bottom row was trained by using Pantheon traces.

As can be seen, there is a considerable difference in the pattern of the matrix, which indicates that the trained MDI models do depend on the network conditions. At the same time, we also notice that there are considerable differences across the models for Cubic, BBR and Verus for a given network condition type. This tells us that these models do capture the protocol behavior itself instead of merely adapting to the network condition type. The effect of network condition on the model is different for different protocols. We note that there is a considerable similarity between the matrices for MDI Cubic for both network conditions, while for MDI BBR, there seems to be much higher dependence of the model on the network condition. The two matrices for MDI Verus also exhibit some similarity. This qualitative similarity in matrices across both network conditions for both MDI Cubic and MDI Verus leads us to believe that there possibly exists a more general model that works across different network conditions, and a deeper investigation of this remains part of future work.

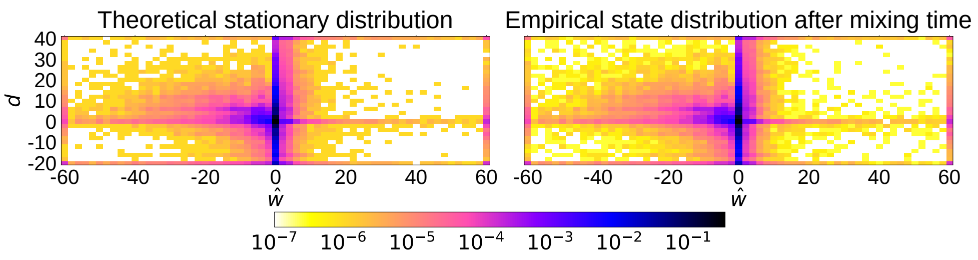
4.3.4 Model vs Protocol: Distribution Convergence

Using our Markov formulation, we are able to provide convergence guarantees as strong as the original protocols, using properties of convergence of Markov chains. Before presenting our results we briefly review some basic notations and definitions regarding Markov chains and convergence.

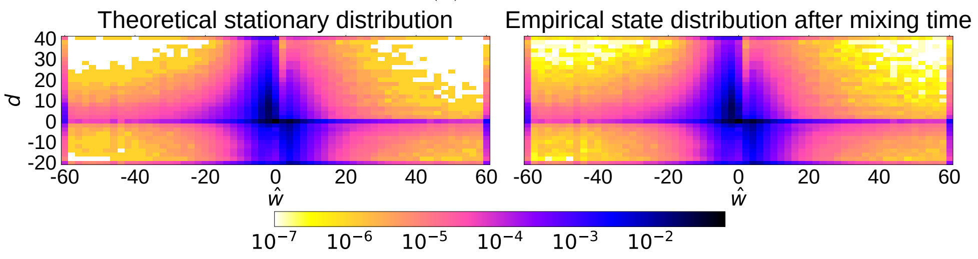
Markov chains and Mixing times: Every Markov chain can be represented as a transition matrix P, where the entry p_{ij} represents the probability of transitioning to state j from state i. Suppose $\mu^{(t)}$ is row vector that represents a probability distribution over the state space at a time t. Then at $t+1$, the distribution over the state space is given by $\mu^{(t+1)} = \mu^{(t)}P$. If the initial distribution at $t = 0$ is given by $\mu^{(0)}$, then we have from above that $\mu^{(t)} = \mu^{(0)}P^t$. The limiting distribution λ is the limit of $\mu^{(t)}$ as $t \to \infty$. If a unique limiting distribution exists, then it equals the *stationary distribution*, which is the row vector π, such that $\pi P = \pi$. It is computed as the left eigenvector of the transition matrix corresponding to the largest eigenvalue [69].

The *mixing time* of a Markov chain, t_{mix}, is the time t to convergence from an initial distribution $\mu^{(0)}$, i.e., when the probability distribution $\mu^{(t)}$ over the state space is sufficiently "close" to the stationary distribution π that they are indistinguishable from one another. Any random walk process in a finite Markov space is associated with a finite mixing time [70]. The mixing time is a measure of time to "convergence", or equivalently the time beyond which the distribution over the state space remains stationary. In order to obtain the most conservative estimate, we define the mixing time as the maximum of the times to convergence starting from all possible initial states.

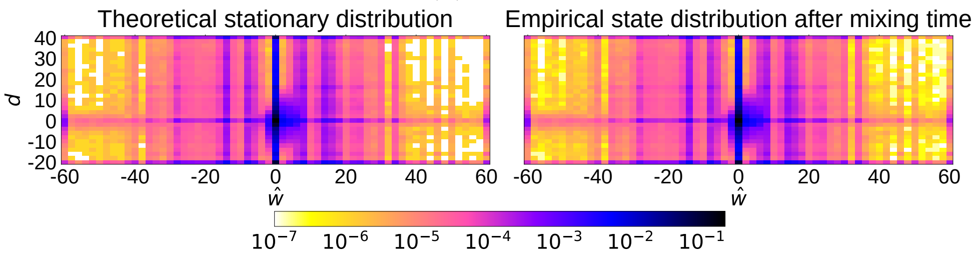
Observations: In our context, the state space comprises of the Cartesian product of 31 states in the delay space $\langle \hat{d} \rangle$ and 61 states in the window space $\langle \hat{w} \rangle$, a total of 1891 $(\hat{d}, \hat{w})$ tuples. For each protocol, we computed the mixing time as the maximum of times to convergence beginning from each of the states. If the start state is i, then the initial distribution $\mu^{(0)}$ is a one-hot vector, with 1 at the location corresponding to state i and 0 everywhere else. Then, at every iteration t, we compute $\mu^{(t+1)} = \mu^{(t)}P$, and declare convergence at time t_{mix} when the maximum element-wise difference between $\mu^{(t_{\text{mix}})}$ and $\mu^{(t_{\text{mix}}+1)}$ is less than a certain defined threshold (ϵ). We compute mixing times for three different thresholds: 10^{-3}, 10^{-5} and 10^{-7}. The last is chosen as it approximately equals the machine epsilon for 32-bit float.



(a) Cubic



(b) Verus



(c) BBR

Figure 4.3: Comparison between the theoretical stationary distribution of the Markov chain model (left) that is trained on the training set of traces, vs the empirical frequency distribution of the states after mixing time (right) for the three protocols.

The heatmaps in Figure 4.3 show the theoretical stationary distribution computed using the Markov chain transition matrix trained over training data, compared with the empirical distribution of states *after convergence* (i.e., the mixing time) of the original protocol over training data. The heat-maps are displayed over the two-dimensional $(\hat{d}, \hat{w})$ state space. The fact that these distributions match very closely is a very strong result that our Markov model versions of the protocols faithfully replicate the original protocols in terms of convergence properties. Ta-

Table 4.2: Mixing times (in RTTs) and maximum difference of probabilities in the empirical distribution of the states (Q) in the testing set after mixing time, and those in the theoretical stationary distribution (P) from the Markov model trained on the training set.

	t_{mix} (10^{-3})	t_{mix} (10^{-5})	t_{mix} (10^{-7})	$\max \lvert P - Q \rvert$
BBR	23	48	74	0.0002
Cubic	15	30	45	0.0004
Verus	21	46	71	0.0002

ble 4.2 shows the mixing times (in RTTs) obtained from the transition matrix. We note that the mixing time values are vastly different across the different protocols.

To further ascertain the closeness between the empirical distribution of the states and the stationary distribution as shown in figure 4.3, the table also additionally shows a maximum absolute element-wise difference between the two probability distributions, which is very low, of the order 10^{-4}, for all the three protocols. We also computed the Kullback-Leibler Divergence [71] as a measure of how well the empirical distribution of states after the mixing time approximates the stationary distribution. The values were very close to zero, of the order 10^{-6}, which indicates the distributions were nearly identical.

4.4 Summary

The contributions of this chapter can be summarized as below:

- We introduce Model-Driven Interpretability(MDI) framework which is used to model congestion control algorithms in highly varying networks.
- MDI models congestion control protocols using a Markov chain-based framework that preserves the temporal behavior of the original algorithm and

approximately reconstructs its throughput and delay characteristics.

- We show how MDI models converge to the same state distributions as the empirical algorithms which confirms that models are accurate.
- The MDI model representation essentially provides a lookup table of a protocol which can used as a drop-in replacement to replicate the behavior of a congestion control protocol in highly varying network settings.
- We hope that this Markov modeling approach provides a new lens for understanding the performance of congestion control algorithms on highly variable networks.

Chapter 5

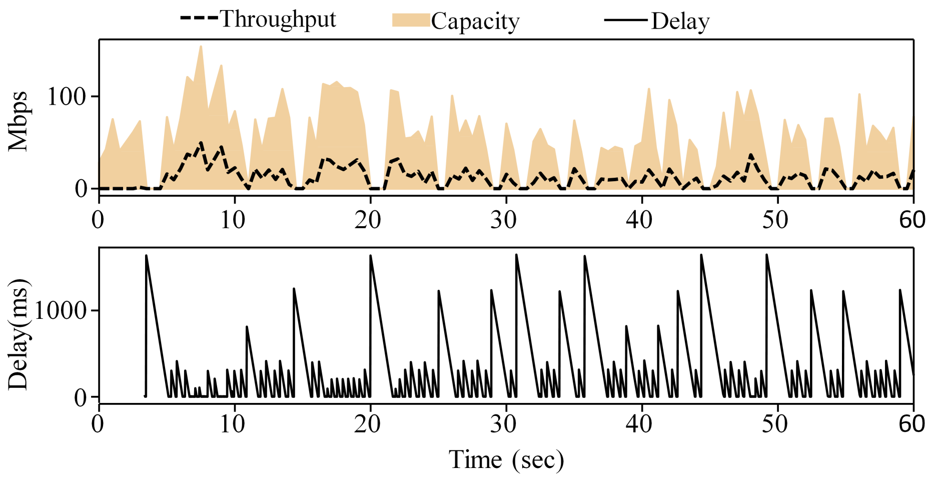
Learning Congestion State For mmWave Channels

Millimeter wave (commonly known as mmWave) is enabling the next generation of last-hop communications for mobile devices. But these technologies cannot reach their full potential because existing congestion control schemes at the transport layer perform sub-optimally over mmWave links. In this chapter, we show how existing congestion control schemes perform sub-optimally in such channels. Then, we propose that we can learn early congestion signals by using end-to-end measurements at the sender and receiver. We believe that these learned measurements can help build a better congestion control scheme. We show that we can learn Explicit Congestion Notification (ECN) per packet with an F1-score as high as 97%. We achieve this by doing unsupervised clustering using data obtained from sending periodic bursts of probe packets over emulated 60 GHz links (based on real-world WiGig measurements), with random background traffic. We also describe how the learned values of ECN can be utilized for rate estimation.

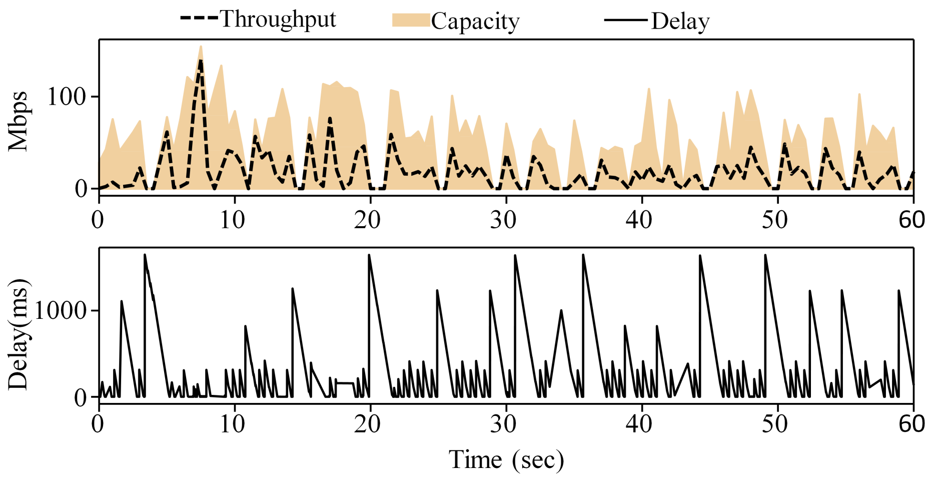
Millimeter wave (mmWave) radio bands like 60 GHz band, 28 GHz band and 39 GHz have unique properties like high oxygen absorption, diffraction and penetration losses. On the other hand, we already have new short-range wireless specifications that are poised to power the next generation of access networks e.g Wireless Gigabit (commonly known as WiGig [72]) and 5G new radio (commonly known as 5G-NR [73]).However, the protocols available at transport layer do not fully harness the potential of these lower layer specs.

We make two high level contributions in this chapter: *i)* we show that existing congestion control schemes fail over mmWave, and *ii)* we show that we can learn early signals of congestion from the network using end-to-end feedback, which can then potentially be used to build a sketch of a novel congestion control algorithm.

Congestion Control over 5G: The congestion control schemes of the past are not completely equipped to handle the highly volatile mmWave channels. The *de facto* congestion control protocol is TCP Cubic for most of the traffic. TCP Cubic treats packet loss as the signal for congestion in the network. When we introduce link outages and capacity variations, which are fairly common in mmWave channels, Cubic fails. When bottleneck buffers are large, loss-based congestion control (like Cubic) keeps them full, causing bufferbloat, and when the buffers are small, loss-based congestion control can further lower throughput by multiplicative decrease on packet loss [53]. There have been new congestion control schemes that have been proposed to overcome Cubic's limitations. We also evaluated the performance of these schemes over mmWave. We show in § 5.2 that the newer congestion control schemes such as Verus [52] and Sprout [50], which are supposed to overcome TCP's limitations in highly varying channels (cellular), also prove ineffective.



(a) Sprout throughput and delay over WiGig



(b) Verus throughput and delay over WiGig

Figure 5.1: Throughput and delay over WiGig. The top plot shows the fluctuating channel capacity (light brown shaded region), the actual observed throughput (black dotted line), and the bottom plot shows the observed one-way delay in the same period.

Learning Congestion State over 5G: Explicit Congestion Notification (ECN) is a bit that is marked in the packet by certain routers and switches when the buffer occupancy exceeds a certain threshold. This feedback eventually reaches the sender, which can then modify its sending rate as in DCTCP [74]. Unfortunately, most routers and switches do not mark the ECN bit, and even if this function is available, it is not always enabled in the network [75]. Hence, we attempt to answer the following question: *Can we learn the ECN signal for each packet using end-to-end measurements at the sender and receiver?* We show that we can indeed learn ECN marker with an F1-score as high as 97% at a per packet level. We discuss how this can be used to create a better congestion control scheme for mmWave links.

5.1 Background and Related Work

The millimeter wave bands have the potential of enabling high throughput communication with the additional caveat of rapidly fluctuating links over few wavelengths. Due to the high carrier frequency, mmWave communications suffer from huge propagation loss, reducing the coverage area of base stations. In addition to this, due to weak diffraction ability mmWave communications are sensitive to blockage by obstacles such as humans and furniture [76]. Therefore, line of sight and non-line of sight communication can experience significantly different channel throughput.

TCP variants are known to suffer from a number of inefficiencies in these kinds of scenarios. For example, TCP Cubic suffers from the bufferbloat problem due to large window sizes, which results in extremely high packet delays in cellular networks [50, 52]. Another issue that TCP fails in addressing in cellular networks are packet losses that are not linked to congestion; Hu et. al. [77] identified packet reordering as a common occurrence in cellular networks and being mistaken as a congestion related loss.

Shortcomings of legacy TCP over highly varying cellular channels have led to newer delay-based congestion control protocols like Sprout [50] and Verus [52] that were specifically designed for cellular. While Sprout focuses on the problem of reducing self-inflicted queuing delays and uses packet inter-arrival times to detect congestion, Verus was designed to create a balance between packet delay and throughput. Recently, PCC Vivace [59] has been shown to react well to changing networks while alleviating the bufferbloat problem. PCC Vivace leverages ideas from online (convex) optimization in machine learning to do rate control.

Congestion control has been shown to improve with explicit feedback from network, the examples of this feedback can range from available bandwidth at a time to Explicit Congestion Notification (commonly known as ECN) marker [74, 78, 79]. These protocols yield great result but require specific in network hardware to provide feedback to the sender. Therefore, it has been difficult to deploy them at larger scales. In our approach, we leverage un-supervised learning to learn this ECN feedback from network and give a sketch of how it can be used to develop a congestion control algorithm. There has been other work on learnability of congestion control [61] that are more general, while we specifically try to learn an ECN signal. There has also been prior work on using machine learning to improve congestion control for wireless channels [80]. Comparing our approach against such similar works will be part of future work.

5.2 Congestion control over 5G

We explored the performance of two different algorithms, Verus [52] and Sprout [50], using a setup similar to Pantheon [60] local tests. We chose these two as they both are designed for highly varying cellular channels and react quickly to changing channel conditions. We used a Mahimahi [68] linkshell to emulate an mmWave link from a trace file that was generated using a real WiGig router leveraging the approach shown in §5.4.1. The underlying conditions used to produce the highly varying trace were random movements of humans between a WiGig sender and a receiver radio. The trace used was 60 seconds long as we wanted to emulate a long running data flow. There was no background traffic so as to see the full potential of these algorithms.

Figure 5.1 shows the performance of these algorithms in terms of achieved throughput and one-way queuing delay experienced as we ran a single long running flow. The average capacity of the trace was 51.6 Mbps. The queue used in these experiments was a standard droptail queue and the size was set to the bandwidth delay product (BDP). To calculate the BDP, we used the maximum observed capacity value of the link since the link is highly varying. We set the link propagation delay to 20 ms using the Mahimahi delayshell.

Having similar design goals, both of them succeed in keeping the average delay reasonably low, at 84 ms for Sprout and 191 ms for Verus. However, neither is able to keep up with the highly varying channel in terms of throughput, and consequently the channel ends up being grossly underutilized in both cases. Sprout is able to maintain an average throughput of only 11.3 Mbps (21% utilization), while Verus does slightly better at 17.1 Mbps (33% utilization). Sprout is not aggressive in ramping up because it tries to keep the delay low. Verus is more aggressive than Sprout but is still not aggressive enough for mmWave links.

While delay-based algorithms like Verus and Sprout have been shown to work on certain highly varying links, clearly they have their limitations, and those limitations become even more stark at the range of mmWave, where channel fluctuations are extremely rapid. Thus we realize that we need more than a purely delay-based approach to congestion control in mmWave scenarios. In the past, other types of feedback, such as an Explicit Congestion Notification (ECN) marking in the network have been used to devise improved congestion control mechanisms[74]. However, since such signals require specialized hardware, we explore the ability to predict such early signals of congestion using a purely data-driven approach, from end-to-end delay and packet arrival time information observed at the sender.

5.3 Learning ECN

We showed in the last section that end-to-end delay feedback takes us only so far in the case of highly fluctuating network conditions, and thus we need a different kind of feedback from the underlying network. In-network congestion feedback like an Explicit Congestion Notification (ECN) [81] can deliver extra information with each packet received and has been shown to improve performance of congestion control algorithms in specialized network environments [74]. ECN uses two bits in the IP header to mark congestion in ECN-enabled routers. The high-level idea is that ECN-enabled routers can set the Congestion Experienced (CE) codepoint (this is when both the ECN bits have a value of 1) in the IP header of packets instead of doing active queue management. For example, a Random Early Detection (RED) queue [82] marks the CE codepoint in packets with a probability p if the queue size increases beyond a threshold. In case of DCTCP, CE codepoint is marked by a router if the queue reaches certain threshold.

We propose to learn the ECN value for each packet using an unsupervised learning approach. For our setup we assume the ECN is marked 1 with a probability of 100% if the router buffer is filled more than 50%. So all of the packets correlating with that buffer state are assumed to have ECN 1.

We have a simple setup similar to § 5.2 where we have a sender and receiver communicating with each other over a WiGig channel. The sender wishes to send data at the maximum rate possible without causing congestion in the network. In addition, there exist some background traffic in the link, which is unknown to the sender. We send probe bursts of packets into the network and try to use the measurements from these probes to predict the value of ECN for each packet. In particular we use the RTT and inter-arrival time (IAT) of each packet to do the prediction. While the RTT is readily available at the sender, the IAT is not, and so we need to make some assumptions to compute the IAT.

The sender sends a packet with a sequence number, and the receiver sends an acknowledgment back for that sequence number. We assume that it is possible to piggyback the IAT value in the acknowledgment, and that the link between the receiver and sender is not saturated and error-free. When sender receives an acknowledgment for a sequence number, it calculates the RTT and saves the inter-arrival time sent by the receiver. It then uses these values and same data from previous sequence numbers to estimate whether the current packet was in a congested buffer.

5.3.1 Learning algorithm

Given a particular kind of network and some random background traffic, we use small bursts of probe packets at periodic intervals to learn the value of the ECN at time t_i. We assume to have some limited information available from n number of earlier times t_j, where $j < i$, when previous packets belonging to the probe were received. The amount of historical data stored is bounded by memory constraints.

As mentioned before, we rely on two quantities for each historical packet at an earlier time t_j: the Round Trip Time (d_j) of a previous packet, and its inter-arrival time $t_j - t_{j-1}$. The ECN bit, $\hat{b}_i$ at time t_i, is essentially modeled as a mapping from a multiple of these two quantities computed at several past time instances, to 0 or 1 labels (congestion or no congestion). This is a 2-class classification problem, for which supervised learning algorithms, such as Support Vector Machines (SVM), can be applied. However, from a practical viewpoint, the training information required for supervised learning algorithms may not be available at the sender. For that reason, we exploit unsupervised-learning solutions. We used the K-means clustering algorithm to study its potential to learn congestion and no congestion for each transmitted packet.

Given that we have information stored for n past probe packets received, an important parameter in our prediction model is the *history length* (H). This is the number of past occurrences of received probe packets to use as input for prediction. Congestion builds up in a router over a period of time, which can be very small or very large. Therefore, we believe it is important to experiment with varying the history length in order to avoid either over-fitting or under-fitting the model. The longer the history length, the larger the amount of past packet information that is used in the prediction, and consequently the larger the number of parameters to estimate. Hence using a history greater than 1 may also help to ensure that the problem is not over-determined thus potentially improving learning performance.

5.4 Evaluation and Results

5.4.1 Experimental Setup

To collect the WiGig traces, we used the NETGEAR Nighthawk X10 wireless router, which implements the 802.11ad standard [83] for 60 GHz channels. We used an Acer TravelMate P648 which comes with a WiGig enabled network card. We started a UDP sender at a Linux machine connected to the wireless router over Ethernet, and used it to send UDP packets at a very fast rate to the IP address of the laptop. A receiver at the laptop recorded the timestamp every time a packet was received. We used the log from the laptop to record the inter-arrival times between the packets. This time was used to create a trace file that is afterwards used as the channel trace in the Mahimahi [68] linkshell emulator. The trace files contains the number of slots that are available in any given millisecond to send a packet. The linkshell in Mahimahi acts as a controlled router that queues packets and sends them at the desired rate as dictated by the trace file. The traces gave us the ability to run several algorithms and compare the performance across traces. The same approach has been used by Verus [52] and Sprout [50] in the past.

We collected the WiGig traces under different scenarios. We show these traces here to demonstrate how a WiGig channel looks like under different scenarios. There were sometimes long unexpected disconnects in the collected traces. To make sure that these long disconnects were not affecting our whole analysis, we

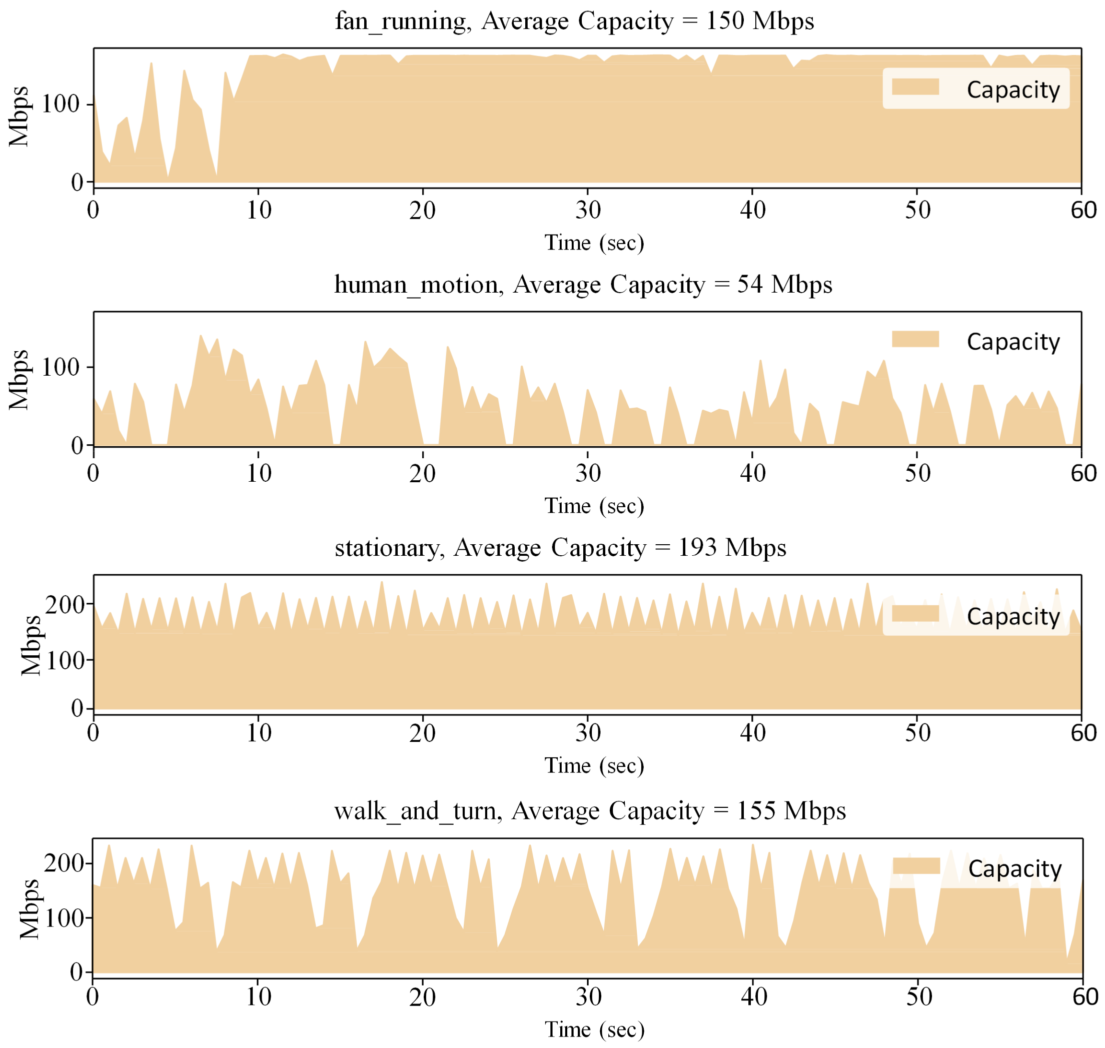


Figure 5.2: Capacities of various channels

removed disconnects from traces if they were longer than 3 seconds. Figure 5.2 shows the capacities of the 4 traces we used. The traces are described below.

- The *fan_running* trace shows the capacity when we kept the laptop and router on a desk inside a room to collect a long stable trace. The slight instability in this trace can be due to a running fan that was between the router and the laptop. The initial instability can be attributed to the human who started the trace collection script before moving away from the laptop.
- The *human_motion* trace shows the capacity of the link where we introduced human interference in between the router and the receiving laptop. As can be observed, the channel capacity is greatly affected by the attenuation due to the human body.
- The *stationary* trace shows the capacity of the trace where the laptop was placed next to the router. The y-axis shows that this is the highest capacity trace with an average capacity of 193 Mbps. This trace was collected in a different indoor environment compared to the previous ones (*fan_running* and *human_motion*).
- The *walk_and_turn* trace shows the channel capacity when a person holding the laptop moves towards the WiGig router and then turns. This trace was smaller in length so it was concatenated multiple times to produce a longer trace to emulate a long running flow.

After collecting these traces, we emulated these links in the Mahimahi linkshell emulator [68], where a realistic propagation delay of 20 ms was added. Each time we emulated a link, we sent packet probe bursts of length 10 packets at periodic intervals with random Poisson traffic in the background. These packet probe bursts were sent by a sender. We varied two parameters – the interval between two consecutive probe bursts (*burst interval*), and the mean bit rate of the background traffic. We show results with two burst interval values, 5 ms and 10 ms, and the mean value of background traffic rate was between 620 and 680 Mbps. These high sending rates were chosen in order to reduce the skew between number of congestion instances and number of no-congestion instances in the training data ("class imbalance"). With background traffic rate set in the ballpark of the channel capacity, the amount of skew was as high as 1:1000, which resulted in bad performance by the ML algorithm in predicting instances of congestion correctly. Hence we kept increasing the rate until the amount of skew became acceptable. The worst skew in the current setup was approximately 1:10. We collected comprehensive data to analyze the clustering performance for each link by emulating it several times.

The dataset for the clustering model was built in the following way. From the output trace of the emulation for 60 seconds, we computed the ECN at every millisecond since the start, from the queue occupancy (recall from §5.3 that we mark ECN as 1 in that millisecond if the buffer occupancy was 50% or more). At every millisecond, for history length H, the RTT and IAT were taken for H immediately previous instances of packet arrivals as the input features to predict congestion at the current time. The total number of features were thus $2H$ for history length H. All such samples were collected in a dataset for each value of H and channel type. A single dataset had approximately 25000 feature-label pairs; the exact number depended on the history length H.

5.4.2 Learning congestion using clustering

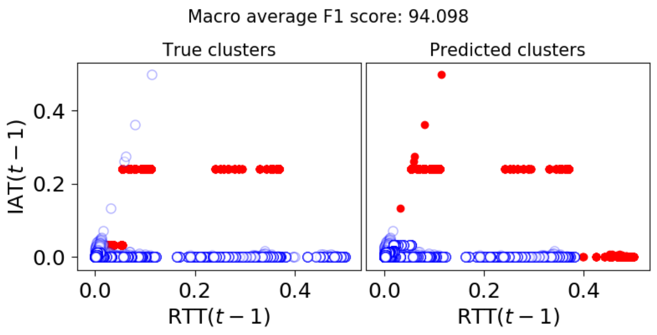
After analyzing the datasets obtained using the procedure described above, we made two observations. First, the RTT and IAT samples were on different scales. Secondly, most of the samples were confined to small ranges of values. Hence we applied the following preprocessing steps to the data before running the clustering.

1. The feature values were transformed by a simple exponentiation, so that those having small values remain small, while the ones larger get separated from the rest. The idea of this transformation was to effect better separation of those samples indicating congestion from the rest.

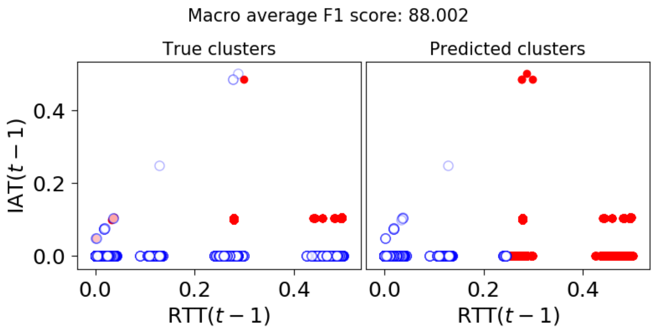
2. Outliers detected using a fixed deviation from the mean were removed. We experimented with deviation of σ, 3σ and 5σ, where σ represents the standard deviation. Outliers correspond to those samples that have large feature values, and are thus more easily separable.

3. Finally, the resulting dataset was scaled so that RTT and IAT samples become comparable. Each feature was scaled to be in $[0, 1]$. Additionally, we assign RTT and IAT different relative weights, β and $1 - \beta$ respectively, where $\beta \in [0, 1]$.

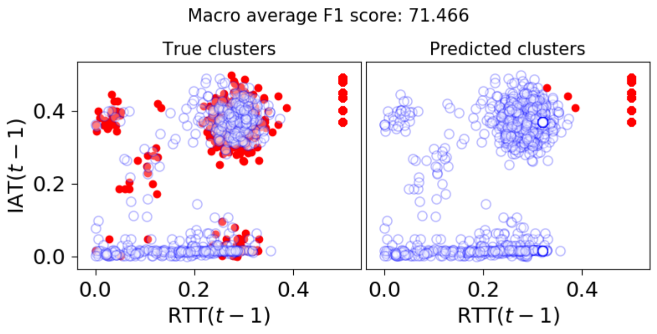
We ran the K-Means clustering with $K = 2$ separately for each burst interval, history length (from 1 to 4) and each channel type. We also experimented with the weight β in steps of 0.1 from 0 to 1. For each setting, we set aside 25% of the dataset for testing. Once a clustering model was learnt on the training set, it was applied on the testing set. One of two resulting clusters was assigned to 0 and other to 1. The assignment was done in such a way as to maximize amount



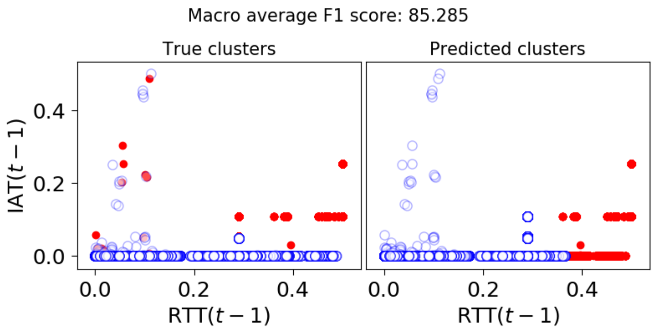
(a) *fan_running*



(b) *human_motion*



(c) *stationary*



(d) *walk_and_turn*

Figure 5.3: Illustration for equal weighting of RTT and IAT for all channel types for history $H = 1$ and minimal outlier removal. The red markers indicate ECN=1, and blue markers ECN=0. The illustration is to show some of the natural separation in the feature space.

Table 5.1: Percentage of the average F1-scores of unsupervised learning

Channels	Burst Interval 5 ms History 1	2	3	4	Burst Interval 10 ms History 1	2	3	4
fan_running	94	94	94	94	94	94	94	94
human_motion	96	97	97	97	94	94	94	94
stationary	97	97	97	97	96	97	97	97
walk_and_turn	95	95	95	95	94	94	94	94

of overlap with the ground truth. Following this, we are able to study clustering performance using the standard metrics used for quantifying classification performance – *precision* and *recall*. The *precision* for a class C is the fraction of samples correctly classified as class C to all the samples classified as class C, and the *recall* is the fraction of samples correctly classified as C to all the samples that are in fact class C. Compared with the *accuracy*, which is simply the fraction of number of correctly classified samples to the total number of samples, which can be misleadingly high when there is significant class imbalance as in our case, the precision and recall better capture the false positive and false negative rates in the classification. We report the *F1-score* in our results below, which is the harmonic mean of the precision and recall [84]. The F1-score is a popular and widely used metric in the field of information retrieval for measuring classification performance in many applications, as it captures both the precision and the recall [85].

Table 5.1 shows the best prediction result for each channel, history length and burst interval. Each reported F1-score in the table above is the average of two F1-scores, one for each of the two classes ECN=0 and ECN=1. Since our objective is to analyze the potential performance of using an unsupervised method for ECN prediction, each entry is the maximum score across all values of β. Figure 5.3 illustrates clustering for history $H = 1$ when feature space is two-dimensional and when $\beta = 0.5$. We note that we observe F1-scores higher than 71% even in a default setting where there is no relative weighting.

The clustering algorithm, in all cases, is able to learn the ECN with high accuracy. In addition, we can observe that, for our datasets, similar performance can be achieved irrespective of the history length. Finally, the results also indicate that the scenario with smaller inter-burst interval (i.e. 5 ms) brings about slightly better results. This last observation is due to higher congestion when the interval between traffic bursts is reduced, which in turn reduces the imbalance between the numbers of samples in each of the two classes, ECN=0 and ECN=1.

5.4.3 Using learned ECN

We have shown that we can learn the ECN signal for each packet with a reasonable accuracy. This learned signal can be used as an indicator of congestion building up in the network. We wanted to see if our learned signal can give us a congestion window that highly correlates with the capacity in the network at that time. To this end, we started with the DCTCP algorithm, which is designed specifically for data centers and uses the ECN. We decided to replace the true ECN signal with the learned ECN signal in the control loop of the algorithm. If the learned ECN can give us a congestion window that correlates well with the channel capacity, we can say that learned ECN can be used as a replacement for the true ECN. A high positive correlation would mean that we have a sending rate that increases and decreases proportional to the channel capacity.

For this simulation we assumed that we are in the congestion avoidance phase. Recall that congestion avoidance is the phase after slow start and TCP behavior is Additive Increase and Multiplicative Decrease (AIMD), for no-congestion and congestion respectively. In congestion avoidance phase, DCTCP does additive increase in the case of no congestion, while doing a fractional decrease based on ECN in the case of congestion. The equation for fractional decrease is: $\text{cwnd} = \text{cwnd} \times \left(1 - \frac{\alpha}{2}\right)$, where α is the weighted fraction of ECN marked packets in the last RTT. When α is close to 0 (low congestion), the window is only slightly reduced. When α is zero, the DCTCP rate only increases by the Maximum Segment Size (MSS), which is not desirable for mmWave channels but we still use it to demonstrate how learned ECN signal can be used.

We start our simulation with an arbitrary congestion window post slow start phase. After that, we calculate congestion window using the fraction of learned ECN packets in a time window (simulating RTT). We then compute the cross-correlation of the congestion window with the actual channel capacity. We observed a maximum Pearson correlation coefficient of 0.622 between the congestion window and channel capacity. This means that the congestion window shows decent correlation with the channel capacity and can be used to track channel capacity over time. DCTCP is not the perfect fit for mmWave channels because it has been designed for data centers but the correlation value indicates that the learned ECN may be a powerful signal for congestion control. We believe that we can get an even higher correlation if we replace the additive increase with a more aggressive ramp-up function. Determining the exact function that would work for mmWave links remains part of future work.

5.5 Summary

In this chapter we make the following contributions:

- We attempt at learning congestion state of a highly varying mmWave channel, and we believe that these results are very encouraging.
- We use an unsupervised learning approach so that learning can be done in an on-line manner
- We also looked at how this learned ECN signal can be used for building a congestion control algorithm.

Chapter 6

Conclusion

In this chapter we draw conclusions from our work on enhancing connectivity and transport layer performance. We also discuss potential future work.

6.1 Enhancing connectivity access for mobile users in rural contexts

We presented Wi-Fly, a communications methodology and architecture aimed at providing opportunistic connectivity in remote and under-served regions around the world. We demonstrated the efficacy of this system using simulations based on real-world measurements using instrumented aircrafts. Wi-Fly leverages ADS-B assisted control channel to provide low powered base stations for connectivity. Contrary to other solutions, Wi-Fly promises to be low-powered and inexpensive via its leveraging of the existing widescale infrastructure of commercial air transport.

The GreenApps project help enable localized communications in rural communities. The Fishline application was used in Nicaragua by various users to advertise through SMS in the local community. Several advertisements were sent out using the service and many users responded back to advertisements. The P2P marketplace application has been tested with cohorts of farmers and traders from the local community to perform BUY/SELL transactions. The IVR-based social media application was inspired by the success of voice-based citizen journalism platforms like Polly [86] and shows how one can build such voice-based applications to function effectively in rural contexts in an intermittency-aware manner. Overall, this project lead to actual impact on ground.

6.1.1 Discussion and future work on enhancing connectivity

Spectrum considerations for Wi-Fly: In the US and many other countries, the frequency range 2400-2483.5 MHz (2.4 GHz band) is authorized for use by devices that do not require individual licenses. Depending on the country, these devices are referred to as unlicensed devices, license-exempt devices, class licensed, etc. With some changes to the wording based on the country, the general rule is that unlicensed devices cannot cause harmful interference to–and cannot claim protection from interference from–any device operating in the 2.4 GHz and adjacent bands. There is typically a minimum set of technical rules for these unlicensed operations, which serves to create a low barrier for innovation as well as a low compliance cost. Furthermore, in the International Table of Frequency Allocation, mobile and fixed services are co-primary in this spectrum range, and there is no prohibition on aeronautical mobile use. The 2.4 GHz band is home to technologies such as Bluetooth, Zigbee, and Wi-FiTM Certified devices (Wi-Fi) that are compliant with different WLAN standards developed by IEEE 802.11. Additionally, we do not see any technical reasons why unlicensed LTE-based technology cannot be made to operate in the band if operators so choose.

Changing aircraft routes for maximizing coverage in Wi-Fly: An interesting possibility with the proposed framework is that we can consider modest re-routing of aircraft based on the demand for connectivity. For example, bandwidth requirements can vary depending upon various factors such as time-of-day, special events, population density, etc. Due to the mobile nature of the access points (aircraft), it should be possible to dedicate more resources to areas that need better connectivity at minimal costs. In fact, similar ideas have been explored in the context of weather prediction using aircraft [87]. Similar to that work, we can envisage an analysis where we can determine best actions (aircraft re-routes) to take so that it maximizes *value-of-connectivity*.

Incentivizing Wi-Fly adoption: For Wi-Fly to work as a potential means for providing affordable Internet connectivity for people and things in some of the most remote and geographically challenging areas in the developing world, we require a globally reproducible model for subsidization of airliners willing to adopt Wi-Fly. We see the feasibility of business models where Internet service providers and governments can partner with the airlines to provide connectivity based on regional routes of planes. We need such partnerships to catalyze Wi-Fly adoption.

Open questions about Wi-Fly: More work is needed to identify the best approach to be used for PHY layer. This includes exploration of MIMO based approach, antenna design at receiver and sender while taking into account economics of fuel efficiency and plane design.

6.2 Enhancing transport layer performance for mobile users

MDI showed that congestion control algorithms can be approximated with a Markov chain-based framework that preserves the temporal behavior of the original algorithm and approximately reconstructs its throughput and delay characteristics. Furthermore, using this framework provides a set of additional benefits such as protocol convergence analysis. The model representation essentially provides a lookup table of a protocol which can used as a drop-in replacement to replicate the behavior of a congestion control protocol in highly varying network settings. We hope that this Markov modeling approach provides a new lens for understanding the performance of congestion control algorithms on highly variable networks.

Our results on Learning Channel State for mmWave Channels are a first attempt at solving the problem of learning congestion state of network, and we believe that these results are very encouraging. The fact that we are able to use unsupervised learning and obtain such high accuracy shows the immense potential of this approach to replace or augment traditional ECN and other congestion notification methods. In our experiments, we also tried other ML algorithms such as logistic regression and SVM, but found that a simple unsupervised learning approach like K-means performs the best. In future work we intend to explore the use of reinforcement learning, as well as looking at the exact control loop for congestion control and online ECN learning.

6.2.1 Discussion and future work for enhancing transport layer for mobile

Tuning parameters for MDI model: There are several configurable parameters in our training procedure viz. the epoch size of the Markov chain, the upper and lower bounds of the state space, and the quantization granularity of the state space. As part of future work, we anticipate automatically searching for the optimal value of these parameters—analogous to hyper-parameter tuning in machine learning. We expect this to improve the fidelity with which the MDI protocol reconstructs the original protocol.

A better understanding of fairness of MDI protocols: We would like to assess whether the MDI protocol and the original protocol compete fairly when run on the same network. In addition, we would like to investigate how the fairness within an ensemble of senders running the MDI protocol compares to the fairness within an ensemble of senders running the original protocol. Finally, we would like to investigate if different senders starting from arbitrary initial states converge to their fair shares in an MDI protocol once the mixing time has elapsed.

Alternatives to Markov chains: We have proposed Markov chains as a representation that can capture the behavior of a variety of different congestion control algorithms. In light of the observations for BBR, we ask – are there other possible representations that we could use instead (e.g., neural nets)? How do they compare with Markov chains, both in training time and fidelity of reconstruction of the original protocol's behavior?

Bibliography

[1] Talal Ahmad, Ranveer Chandra, Ashish Kapoor, Michael Daum, and Eric Horvitz. Wi-fly: Widespread opportunistic connectivity via commercial air transport. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pages 43–49. ACM, 2017.

[2] Talal Ahmad, Edwin Reed-Sanchez, Fatima Zarinni, Alfred Afutu, Kessir Adjaho, Yaw Nyarko, and Lakshminarayanan Subramanian. Greenapps: A platform for cellular edge applications. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, page 45. ACM, 2018.

[3] Talal Ahmad and Lakshminarayanan Subramanian. Virtual cellular isps. In *Proceedings of the 3rd Workshop on Experiences with the Design and Implementation of Smart Objects*, pages 35–40. ACM, 2017.

[4] Talal Ahmad, Shiva Iyer, Luis Diez, Yasir Zaki, Ramón Agüero, and Lakshminarayanan Subramanian. Learning congestion state for mmwave channels. In *Proceedings of the 3rd ACM Workshop on Millimeter Wave Networks and Sensing Systems*, mmNets '19, New York, NY, USA, 2019. ACM.

[5] 4.4 billion people around the world still don't have internet. http://wapo.st/2jGAe2o. Accessed: 2017-01-05.

[6] Talal Ahmad, Shankar Kalyanaraman, Fareeha Amjad, and Lakshmi Subramanian. Solar vs diesel: Where to draw the line for cell towers? In *Proceedings of the Seventh International Conference on Information and Communication Technologies and Development*, page 7. ACM, 2015.

[7] Richard Thanki. The economic significance of licence-exempt spectrum to the future of the internet. *White Paper*, 2012.

[8] Kurtis Heimerl, Shaddi Hasan, Kashif Ali, Eric Brewer, and Tapan Parikh. Local, sustainable, small-scale cellular networks. In *Proceedings of the Sixth International Conference on Information and Communication Technologies and Development: Full Papers-Volume 1*, pages 2–12. ACM, 2013.

[9] Openbts. http://openbts.org/. Accessed:2017-01-05.

[10] Osmobts. https://projects.osmocom.org/projects/osmobts. Accessed:2017-01-05.

[11] Milton Mueller. Universal service and the telecommunications act: myth made law. *Communications of the ACM*, 40(3):39–47, 1997.

[12] Flightaware. https://flightaware.com/commercial/flightxml/. Accessed: 2017-01-05.

[13] J Kunisch, I de la Torre, A Winkelmann, M Eube, and T Fuss. Wideband time-variant air-to-ground radio channel measurements at 5 ghz. In *Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP)*, pages 1386–1390. IEEE, 2011.

[14] David W Matolak. Air-ground channels & models: Comprehensive review and considerations for unmanned aircraft systems. In *Aerospace Conference, 2012 IEEE*, pages 1–17. IEEE, 2012.

[15] Michael Rice, Adam Davis, and Christian Bettweiser. Wideband channel model for aeronautical telemetry. *IEEE Transactions on aerospace and Electronic systems*, 40(1):57–69, 2004.

[16] Nasa sedac. http://sedac.ciesin.columbia.edu/. Accessed: 2017-01-01.

[17] Row 44. https://www.linkedin.com/company/row-44-inc. Accessed:2017-01-05.

[18] Ying Chen, Mark P van der Heijden, and Domine MW Leenaerts. A 1-watt ku-band power amplifier in sige with 37.5% pae. In *Radio Frequency Integrated Circuits Symposium (RFIC), 2016 IEEE*, pages 324–325. IEEE, 2016.

[19] Fly-fi. http://www.jetblue.com/flying-on-jetblue/wifi/. Accessed:2017-01-05.

[20] Gogo. https://www.gogoair.com/. Accessed:2017-01-05.

[21] routehappy 2017 wi-fi report. https://www.routehappy.com/insights/wi-fi/2017. Accessed:2017-01-05.

[22] Gogo gets FAA approval for 70 mbps in-flight wi-fi service, but most rollouts coming in 2016. http://www.fiercewireless.com/wireless/gogo-gets-faa-approval-for-70-mbps-flight-wi-fi-service-but-most-rollouts-coming-2016. Accessed:2017-01-05.

[23] Sidney Roberts, Paul Garnett, and Ranveer Chandra. Connecting africa using the tv white spaces: from research to real world deployments. In *Local and*

Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on, pages 1–6. IEEE, 2015.

[24] Rabin K Patra, Sergiu Nedevschi, Sonesh Surana, Anmol Sheth, Lakshminarayanan Subramanian, and Eric A Brewer. Wildnet: Design and implementation of high performance wifi based long distance networks. In *NSDI*, volume 1, page 1, 2007.

[25] Martin Strohmeier, Matthias Schafer, Vincent Lenders, and Ivan Martinovic. Realities and challenges of nextgen air traffic management: the case of ads-b. *IEEE Communications Magazine*, 52(5):111–118, 2014.

[26] Prabhjot Kaur, Moin Uddin, and Arun Khosla. Throughput analysis for opportunistic spectrum access among unlicensed devices. In *Communications (NCC), 2012 National Conference on*, pages 1–5. IEEE, 2012.

[27] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Tool release: Gathering 802.11n traces with channel state information. *ACM SIGCOMM CCR*, 41(1):53, Jan. 2011.

[28] Rabin Patra, Sergiu Nedevschi, Sonesh Surana, Anmol Sheth, Lakshminarayanan Subramanian, and Eric Brewer. WiLDNet: Design and Implementation of High Performance WiFi Based Long Distance Networks. *Proceedings of NSDI 2007*, 2007.

[29] Bhaskaran Raman and Kameswari Chebrolu. Design and Evaluation of a new MAC Protocol for Long-Distance 802.11 Mesh Networks. In *ACM MOBICOM*, August 2005.

[30] Vijay Gabale, Bhaskaran Raman, Kameswari Chebrolu, and Purushottam Kulkarni. Lit mac: Addressing the challenges of effective voice communication in a low cost, low power wireless mesh network. In *Proceedings of the First ACM Symposium on Computing for Development*, ACM DEV '10, pages 5:1–5:11, New York, NY, USA, 2010. ACM.

[31] Vijay Gabale, Ashish Chiplunkar, Bhaskaran Raman, and Partha Dutta. Delaycheck: Scheduling voice over multi-hop multi-channel wireless mesh networks. In *COMSNETS*, 2011.

[32] Sonesh Surana, Rabin Patra, Sergiu Nedevschi, Manuel Ramos, Lakshminarayanan Subramanian, Yahel Ben-David, and Eric Brewer. Beyond Pilots: Keeping Rural Wireless Networks Alive. *Proceedings of NSDI 2008*, 2008.

[33] Kurtis Heimerl, Shaddi Hasan, Kashif Ali, Eric Brewer, and Tapan Parikh. Local, sustainable, small-scale cellular networks. In *Proceedings of the Sixth International Conference on Information and Communication Technologies and Development: Full Papers-Volume 1*, pages 2–12. ACM, 2013.

[34] Aditya Dhananjay, Ashlesh Sharma, Michael Paik, Jay Chen, Trishank Karthik Kuppusamy, Jinyang Li, and Lakshminarayanan Subramanian. Hermes: data transmission over unknown voice channels. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 113–124. ACM, 2010.

[35] Michael Paik, Ashlesh Sharma, Arthur Meacham, Giulio Quarta, Philip Smith, John Trahanas, Brian Levine, Mary Ann Hopkins, Barbara Rapchak, and Lakshminarayanan Subramanian. The case for smarttrack. In *Informa-*

tion and Communication Technologies and Development (ICTD), 2009 International Conference on, pages 458–467. IEEE, 2009.

[36] GSM LiteCel by Nuran Wireless. http://nuranwireless.com/products/gsm-litecell/. Accessed: 2017-01-01.

[37] Range networks. http://www.rangenetworks.com/. Accessed: 2017-01-01.

[38] Sysmocom. https://www.sysmocom.de/. Accessed: 2017-01-01.

[39] Facebook opencellular. https://code.facebook.com/posts/1754757044806180/introducing-opencellular-an-open-source-wireless-access-platform. Accessed: 2017-01-01.

[40] Osmocom. https://osmocom.org/. Accessed: 2017-01-01.

[41] Rhizomatica community base station. http://rhizomatica.org/. Accessed: 2015-11-05.

[42] Mariya Zheleva, Arghyadip Paul, David L. Johnson, and Elizabeth Belding. Kwiizya: Local cellular network services in remote areas. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 417–430, New York, NY, USA, 2013. ACM.

[43] RCom, Bharti seek early exit from rural telephony scheme. In *The Economic Times (02/07/2011).*

[44] Xin Jin, Li Erran Li, Laurent Vanbever, and Jennifer Rexford. Softcell: Scalable and flexible cellular core network architecture. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 163–174. ACM, 2013.

[45] Abhinav Anand, Veljko Pejovic, Elizabeth M. Belding, and David L. Johnson. Villagecell: Cost effective cellular connectivity in rural areas. In *Proceedings of the Fifth International Conference on Information and Communication Technologies and Development*, ICTD '12, pages 180–189, New York, NY, USA, 2012. ACM.

[46] Aditya Dhananjay, Matt Tierney, Jinyang Li, and Lakshminarayanan Subramanian. Wire: a new rural connectivity paradigm. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 462–463. ACM, 2011.

[47] Changhoon Kim, Matthew Caesar, and Jennifer Rexford. Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises. In *SIGCOMM '08*.

[48] Thomas Roberts Puzak. Analysis of cache replacement-algorithms. 1985.

[49] Memcached. https://memcached.org/. Accessed: 2017-01-01.

[50] Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan, et al. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI 13)*, pages 459–471, Lombard, IL, 2013.

[51] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira. PCC: Rearchitecting Congestion Control for Consistent High Performance. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI 15)*, pages 395–408, Oakland, CA, USA, 2015.

[52] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. Adaptive congestion control for unpredictable cellular net-

works. In *Proceedings of the ACM SIGCOMM 2015 Conference*, pages 509–522, London, UK, 2015.

[53] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: Congestion-Based Congestion Control. *Queue*, 14(5):50:20–50:53, October 2016.

[54] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling tcp throughput: A simple model and its empirical validation. *ACM SIGCOMM Computer Communication Review*, 28(4):303–314, 1998.

[55] Neal Cardwell, Stefan Savage, and Thomas Anderson. Modeling tcp latency. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, volume 3, pages 1742–1751. IEEE, 2000.

[56] Wei Sun, Lisong Xu, Sebastian Elbaum, and Di Zhao. Model-agnostic and efficient exploration of numerical state space of real-world {TCP} congestion control implementations. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 719–734, 2019.

[57] S. Ha, I. Rhee, and L. Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008.

[58] K. Winstein and H. Balakrishnan. TCP Ex Machina: Computer-generated Congestion Control. In *Proceedings of the ACM SIGCOMM 2013 Conference*, Hong Kong, China, 2013.

[59] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. {PCC} vivace: Online-learning congestion control. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pages 343–356, 2018.

[60] Francis Y Yan, Jestin Ma, Greg D Hill, Deepti Raghavan, Riad S Wahby, Philip Levis, and Keith Winstein. Pantheon: the training ground for internet congestion-control research. In *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, pages 731–743, 2018.

[61] A. Sivaraman, K. Winstein, P. Thaker, and H. Balakrishnan. An Experimental Study of the Learnability of Congestion Control. In *Proceedings of the ACM SIGCOMM 2014 Conference*, Chicago, IL, USA, 2014.

[62] Thomas Pötsch. *Future Mobile Transport Protocols: Adaptive Congestion Control for Unpredictable Cellular Networks.* Advanced Studies Mobile Research Center Bremen. Springer, 2016.

[63] A. Wierman and T. Osogami. A unified framework for modeling tcp-vegas, tcp-sack, and tcp-reno. In *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.*, pages 269–278, Oct 2003.

[64] Samuel Jero, Endadul Hoque, David Choffnes, Alan Mislove, and Cristina Nita-Rotaru. Automated attack discovery in tcp congestion control using a model-guided approach. In *Proc. of Network and Distributed System Security Symp., San Diego, CA, USA*, pages 1–15, 2018.

[65] Marta Carbone and Luigi Rizzo. Dummynet revisited. *Computer Communication Review*, 40(2):12–20, 2010.

[66] Stephen Hemminger et al. Network emulation with netem. In *Linux conf au*, pages 18–23, 2005.

[67] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. Reproducible network experiments using container-based emulation. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 253–264. ACM, 2012.

[68] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. Mahimahi: Accurate record-and-replay for {HTTP}. In *2015 {USENIX} Annual Technical Conference ({USENIX}{ATC} 15)*, pages 417–429, 2015.

[69] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.

[70] David Aldous. Random walks on finite groups and rapidly mixing markov chains. In *Séminaire de Probabilités XVII 1981/82*, pages 243–297. Springer, 1983.

[71] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[72] WiGig Alliance. Wigig white paper: defining the future of multi-gigabit wireless communications. *Wireless Gigabit Alliance, Beaverton, Ore, Tech. Rep*, 2009.

[73] S-Y. Lien, S-L. Shieh, Y. Huang, B. Su, Y-L. Hsu, and H-Y. Wei. 5g new radio: Waveform, frame structure, multiple access, and initial access. *IEEE communications magazine*, 55(6):64–71, 2017.

[74] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center tcp (dctcp). In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, pages 63–74, New York, NY, USA, 2010. ACM.

[75] Brian Trammell, Mirja Kühlewind, Damiano Boppart, Iain Learmonth, Gorry Fairhurst, and Richard Scheffenegger. Enabling internet-wide deployment of explicit congestion notification. In *International Conference on Passive and Active Network Measurement*, pages 193–205. Springer, 2015.

[76] Yong Niu, Yong Li, Depeng Jin, Li Su, and Athanasios V Vasilakos. A survey of millimeter wave communications (mmwave) for 5g: opportunities and challenges. *Wireless Networks*, 21(8):2657–2676, 2015.

[77] Zhenxian Hu, Yi-Chao Chen, Lili Qiu, Guangtao Xue, Hongzi Zhu, Nicholas Zhang, Cheng He, Lujia Pan, and Caifeng He. An in-depth analysis of 3g traffic and performance. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, AllThingsCellular '15, pages 1–6, New York, NY, USA, 2015. ACM.

[78] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. *ACM SIGCOMM computer communication review*, 32(4):89–102, 2002.

[79] Matthew Caesar, Donald Caldwell, Nick Feamster, Jennifer Rexford, Aman Shaikh, and Jacobus van der Merwe. Design and implementation of a routing control platform. In *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 15–28, Berkeley, CA, USA, 2005. USENIX Association.

[80] Pierre Geurts, Ibtissam El Khayat, and Guy Leduc. A machine learning approach to improve congestion control over wireless computer networks. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '04, pages 383–386, Washington, DC, USA, 2004. IEEE Computer Society.

[81] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit congestion Notification (ECN) to IP. RFC 3168, Network Working Group, September 2001.

[82] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on networking*, 1(4):397–413, 1993.

[83] IEEE Standards Association et al. Ieee std 802.11 ad-2012: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 3: Enhancements for very high throughput in the 60 ghz band. Technical report, ISO/IEC/IEEE 8802–11: 2012/Amd. 3: 2014 (E), 2014.

[84] C. J. Van Rijsbergen. *Information Retrieval.* Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[85] Steven M. Beitzel. On understanding and classifying web queries, 2006.

[86] Agha Ali Raza, Mansoor Pervaiz, Christina Milo, Samia Razaq, Guy Alster, Jahanzeb Sherwani, Umar Saif, and Roni Rosenfeld. Viral entertainment as a vehicle for disseminating speech-based services to low-literate users. In *Proceedings of the Fifth International Conference on Information and Communication Technologies and Development*, pages 350–359. ACM, 2012.

[87] Ashish Kapoor, Zachary Horvitz, Spencer Laube, and Eric Horvitz. Airplanes aloft as a sensor network for wind forecasting. In *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, IPSN '14, pages 25–34, Piscataway, NJ, USA, 2014. IEEE Press.