UNIVERSITY OF CALIFORNIA

Los Angeles

Software-Defined Mobile Cloud

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Computer Science

by

Ian Ku

2014

ABSTRACT OF THE DISSERTATION

Software-Defined Mobile Cloud

by

Ian Ku

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2014

Professor Mario Gerla, Chair

With the growth of mobile users and the ever-increasing amount of data exchange, mobile network systems must be able to handle the explosion in application traffic and service requirement of users. By utilizing the growth in capabilities of mobile platforms such as vehicles, Mobile Cloud brings together Mobile Cloud Computing (MCC) and wireless networks to interconnect mobile devices and become a cloud-like service provider. In the first part of the thesis, we demonstrate the benefits of the Mobile Cloud by introducing a flexible experiment structure that utilizes virtualization and resource sharing to allow different protocols to be compared in the same mobility pattern and channel conditions in Vehicular Ad-hoc Networks (VANETs). A virtualized environment is setup on each node where the combination of Xen and gentoo software is used to create multiple virtual guests.

However, even with the benefits brought by virtualization and resource sharing, Mobile Cloud network architectures still lacks in flexibility, making it difficult to change and re-configure network behavior, and new services hard or costly to deploy. Also, without flexibility to apply network policies, it is difficult to adapt Mobile Clouds to changing conditions, resulting in a lack of automation and traffic differentiation hard to enforce. In the

second part of the thesis, we introduce Software-Defined Mobile Networks, using software defined paths, topologies, virtual networks to improve and complement current Mobile Cloud services. Our approach focuses on two concepts: The first is virtualization, where we use the benefits of Mobile Cloud to share resources to provide utility and services over a network. The second is Software-Defined Networking (SDN), where we apply the emerging SDN technology to take the Mobile Cloud one step further by bringing flexibility, programmability, and control. We first provide a background on SDN and how SDN is used for wireless and mobile, including several prototypes we built to show how such systems can operate in real-world situations. We then propose designs for Software-Defined Mobile Networks architectures, with an emphasis on Software-Defined Vehicle Network (SDVN) architecture and its operational mode to adapt SDN to VANET environments. We present the required core components to build our SDVN system, including variations that are required to accommodate different wireless environments, such as mobility and unreliable wireless link conditions. We also present the benefits of a SDVN and the services that it can enable by building and evaluating multiple SDVN features, and comparing it with traditional VANETs and Mobile Ad-hoc Networks (MANETs).

Mobile Cloud enables new services with virtualization and resource sharing, and Software-Defined Mobile Networks takes Mobile Cloud further by adding network programmability, flexibility, and control. We show that by combining the two ideas together, we can now form the Software-defined Mobile Cloud.

The dissertation of Ian Ku is approved.

Danijela Cabric

Milos Ercegovac

Leonard Kleinrock

Mario Gerla, Committee Chair

University of California, Los Angeles

2014

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would first like to thank my adviser, Professor Mario Gerla, for guiding me and providing me with all the valuable insight and guidance. I would also like to thank all of my colleagues and peers in Network Research Lab: Jui-Ting Weng, You Lu, Lung-Chih Tung, Eduardo Cerqueira, Xiao Li, and many others for their help and advice. Also, I would like to thank my colleagues in Orange Silicon Valley: Christos Kolias, Bertrand Weber, Dachuan Yu, and many others for their insights. Finally, I would like to thank my family, for without their love and support this work would not have been possible.

VITA

2004	B.S., Computer Science, National Tsing Hua University, Taiwan
2006	M.S., Computer Science, National Tsing Hua University, Taiwan
2010	M.S., Computer Science, UCLA, Los Angeles, California
2008-2010	Teaching Assistant, Computer Science, UCLA.

PUBLICATIONS

1. I. Ku, Y. Lu, R. L. Gomes, F. Ongaro, E. Cerqueira, M. Gerla. “Towards Software-Defined VANETs: Architecture and Services”, 2014 13th IEEE IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), 2014.
2. I. Ku, Y. Lu, and M. Gerla. “Software-Defined Mobile Cloud: Architecture, Services, and Use Cases”, 2014 10th International Wireless Communications and Mobile Computing Conference (IWCMC), 2014.
3. J.T. Weng, I. Ku, and M. Gerla. "Surveillance service on the open mobile cloud", Wireless On-demand Network Systems and Services (WONS), 2013 10th Annual Conference on. IEEE, 2013.
4. J.T. Weng, I. Ku, G. Pau, M. Gerla. “Running Consistent, Parallel Experiments in Vehicular Environment”, Journal of Communication 2013.
5. I. Ku, C. Kolias, S. Catanzariti, “SmartFlow: Intelligent Mobile Offloading”, Demo in Orange Labs Salon de la recherche, France, 2012.
6. I. Ku, C. Kolias , S. Catanzariti, “Openflow and the ‘Remote Control’ App”, Poster and

Demo in Open Networking Summit, 2011.

7. I. Ku, J.T. Weng, E. Giordano, G. Pau, M. Gerla, "Running consistent, parallel experiments in vehicular environment." Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on. IEEE, 2011.

Chapter 1. Introduction

1.1 Towards the Mobile Cloud

In order to keep up with traffic growth, mobile networks must not only optimize the current resources but also add new components/technologies that increase the capacity. Having witnessed the phenomenal burst of research in cloud computing, the idea of Mobile Cloud is bringing together Mobile Cloud Computing (MCC) and mobile wireless networks to interconnect mobile devices and become a cloud-like service provider.

Cloud computing is about the sharing of resources to provide remote services over a network. Today, what we refer as the "cloud" usually means the internet cloud and the edge cloud, as shown in Figure 1.1. The internet cloud provides services through infrastructure through a data center model, while the edge cloud provides services through access. What researchers mean by Mobile Cloud Computing is actually about access to the internet cloud through mobiles.

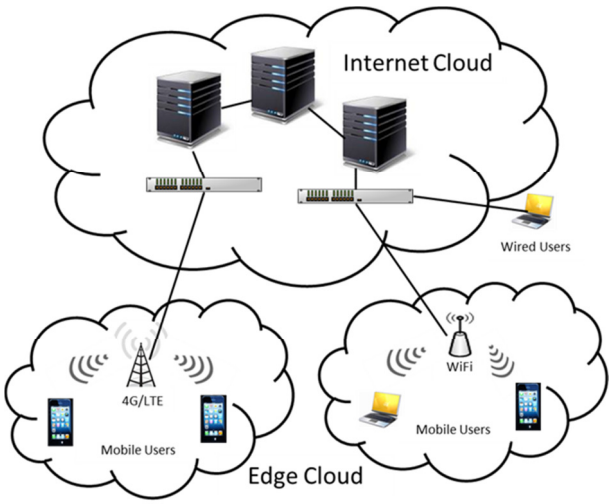


Figure 1.1 Classic Cloud Overview

Mobile platforms, such as vehicles and smartphones, have increasingly grown in capabilities, including computing, storage, and sensor abilities. These end devices not only are the consumers of services from the internet cloud, but are now capable of becoming cloud-services providers. The edge cloud is starting to leverage this, and by taking one step further, the Mobile Cloud is about extending the cloud into mobile wireless and peer-to-peer networks. The Mobile Cloud will stimulate research beyond the scopes of traditional Internet Clouds or Mobile Computing technologies.

One example of the Mobile Cloud would be the Mobile Cloud Intelligent Transport Services [5]. There has been much work on how Mobile Ad-hoc Networks (MANETS) and Vehicular Ad-hoc Networks (VANETs) can be used to provide a wide range of services, including both safety and non-safety related applications. Examples include safety services, traffic management services, and surveillance services. As vehicles become more and more powerful, they can group together to form the mobile vehicular cloud, using Vehicle-to-vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications [1] and virtualization to provide sharing of resources and offer mobile vehicular cloud services. Figure 1.2 shows the extended cloud architecture.

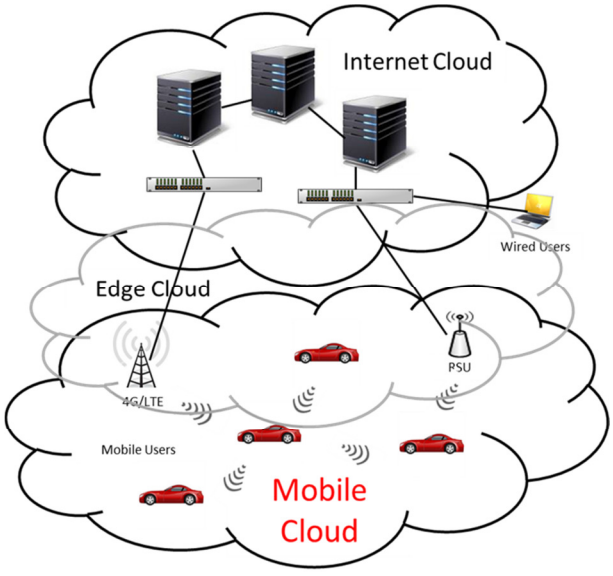


Figure 1.2 Extending Cloud into Mobile

1.2 Mobile Cloud Case Study: Parallel experiment platform for VANET

In the first part of the thesis, we present a case study on one of the services enabled by the Mobile Cloud: Parallel Experiment Platform for VANET. We show how the Mobile Cloud can utilize resource sharing and virtualization to provide new services.

The dynamic nature of VANETs makes it challenging to compare real mobile experiments. This is because realistic mobility pattern for mobile nodes is hard to control, performing experiments requires lots of resources, and it is almost impossible to reproduce mobility pattern in sequential experiments.

To address these challenges, we introduce a Parallel Experiment Platform for VANET in chapter 2, a testbed where multiple experimental configurations can run simultaneously on the same node, and thus experience identical network conditions. The testbed exploits Xen and Gentoo to provide a virtualized environment at every node. In the virtualized environment, multiple virtual machines, each using an independent experiment configuration (ex. Network protocol), run in parallel sharing the same physical resources.

By using our Parallel Experiment Platform, we show multiple benefits. (1) Fewer physical resources required by using a virtualized environment. (2) By using virtual machines to run different experiments simultaneously, each set of experiments encounters identical network conditions and produces comparable results. (3) The virtualized environment is easier to control, resulting in more consistent experiments and results that are easier to interpret. We compare two well-known Ad-hoc routing protocols, AODV and OLSR, to demonstrate our platform. Experiments confirm that our testbed generates consistent and comparable results that are consistent with those published in previous studies.

1.3 Building a Software-Defined Mobile Cloud

Have demonstrated the benefits of the Mobile Cloud through our case study, we realize that current systems still face many challenges, especially in flexibility. In specific, changing and reconfiguring existing network behavior is a very difficult operation, which results in difficulty or high cost to deploy new services. Network devices are often complex devices that require individual configuration to change network behavior. For example, in our case study, a reconfiguration requires recalling all vehicles and nodes. Also, without flexibility to apply network policies, it is difficult to adapt mobile networks to changing conditions, resulting in a lack of automation and traffic differentiation hard to enforce. This issue creates challenges in the deployment of Mobile Cloud applications and services. Therefore, open and flexible mobile architectures are key requirements to allow experimenters to test their solutions in productive environments, as well as to improve the management of network resources, applications, and users.

In the second part of the thesis, we introduce, Software-Defined Mobile Network, using software defined paths, topologies, virtual networks to improve and complement current Mobile Cloud services. Our approach focuses on two concepts: The first is virtualization, where we use the benefits of Mobile Cloud to share resources to provide utility and services over a network.

The second is introducing Software-Defined Networking (SDN) [2]. Nowadays, SDN has emerged as a flexible way to control the network in a systematic way, with OpenFlow [3][4] as the most commonly used SDN protocol. The flexibility of SDN makes it an attractive approach that can be used to satisfy the requirements of MANET/VANET scenarios. Applying SDN principles to MANET/VANETs will bring the programmability and flexibility that is

lacking in today's distributed wireless substrate, while simplifying network management and enabling new V2V and V2I services.

There are many potential benefits of SDN, many of which are already demonstrated or under study in wired-SDN systems, which can also be used in a wireless and mobile SDN system. Examples include intelligent routing, where routing is not based on only source and destination, and traffic differentiation. Client side techniques such as WAN optimization and adaptive streaming can create potential unfairness if all traffic is treated equally. With the introduction of SDN, flow-based prioritization allows traffic to be treated differently at the forwarding plane to deliver the required QoS while maintaining fairness. Enabling SDN in wireless networks can bring the programmability and flexibility that is lacking in today's distributed wireless substrate while simplifying network management and enabling new services.

While wireless and mobile deployment of SDN has recently, begun, its scope has been primarily focused on carrier backbones and access networks. OpenRoads [6] envisions that users will move between wireless infrastructures. CloudMAC [7] proposes virtualized access points. The Wireless & Mobile Working Group (WMWG) [8] in ONF focuses on wireless backhaul, cellular Evolved Packet Core (EPC), and unified access and management across enterprise wireless and fixed networks (e.g., campus Wi-Fi).

Other works on wireless SDN include OpenFlow in wireless mesh environments [9], OpenFlow in smartphone as an application [10], OpenFlow in wireless sensor networks [11], SDN in heterogeneous networked environments [12], and SDN for handover management in heterogeneous networks [13]. However, there remain many possibilities and challenges that have not yet been fully addressed. We look to explore these potentials by introducing SDN into the Mobile Cloud.

In chapter 3, we describe background on SDN technology, and the first step to wireless and mobile SDN, radio access network SDN. In specific, we describe UCLA's SDN campus deployments, use cases on SDN for the mobile packet core and SDN for WiFi access networks, and then present two SDN prototypes that we built to show how such systems can operate in real-world situations. We then in chapter 4 propose designs for Software-Defined Mobile Networks architectures, with an emphasis on Software-Defined Vehicle Network (SDVN) architecture and its operational mode to adapt SDN to VANET environments, as SDN in MANETs/VANETs is one of the potential wireless infrastructures that can support Software-Defined Mobile Networks. We present the required core components to build our SDVN system, including variations that are required to accommodate different wireless environments, such as mobility and unreliable wireless link conditions. We also present the benefits of a SDVN and the services that it can enable by building and evaluating multiple SDVN features, and comparing it with traditional VANETs and Mobile Ad-hoc Networks (MANETs). Based on our studies, we show how SDVN can be used to bring network programmability, flexibility, and control into the Mobile Cloud.

Chapter 2. Running Consistent, Parallel Experiments in Vehicular Environment

2.1 Introduction

Technology of MANETs has been studied for years [14]. The main advantages of MANET include easy deployment, distributed control, bandwidth efficiency and no need for infrastructure. Less than a century after automobiles became affordable for the general public, millions of vehicles travel along highways and streets around the world. Inspired by MANET, VANET is an emergent technology designed to improve safety, comfort, and convenience for vehicle drivers and passengers.

The use of networking vehicles has attracted considerable attention from both research community and automotive industry and has spawned initiatives such as Inter-Vehicle Communication Systems (IVC) [15], Intelligent Transportation Systems [16]-[18] etc. VANETs enable Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communications. These new communication schemes open a new wide set of possible applications together with very challenging novel research topics.

VANET applications can be divided into three categories: emergency/safety messaging, geo-advertising, and data sensing. Different types of applications have very different message propagation and delay requirements. Messages from emergency/safety applications are typically generated from an accident or collision source, and require delivery with the shortest delay. Because message importance decreases as the distance to the accident source increases, the hop count requirement for safety applications is generally only 1-hop or 2-hops. Geo-

advertising targets vehicles within a given zone and direction since the information is only relevant to vehicles in that area. Messages are generally initiated from infrastructure and routed to the mobile destination through geo-routing protocols. The last type of application is to exploit vehicles as sensors. The main advantage of vehicle sensing is the unlimited power support and good coverage offered by the vehicles. Therefore, it is considered as a good platform to collect quasi real-time data such as traffic status, weather condition, or pollution. Besides VANET applications, vehicle users may still require the Internet access for personal usage such as email, P2P file transfer, or streaming video.

Due to various applications in VANET requiring multihop support, researchers have come up with several routing protocols for data dissemination. [31]-[34]. These results show the importance of routing to VANET. However, individuating the most appropriate protocol requires extensive comparison studies. Unfortunately, comparing VANET protocols is anything but trivial, as discussed below.

Before the introduction of testbeds, VANET applications and protocols were evaluated via simulation. Through simulation tools such as ns-2, opnet, or qualnet, researchers were able to obtain results where the performance of various applications and protocols can be gauged. However, even the best of simulation tools cannot simulate the exact physical conditions of the real world, and thus results can be unsatisfactory when compared with real world experiments. There are three main modeling challenges that VANET researchers face : traffic (mobility) pattern; radio propagation models, and environmental network interference.

As the mobility pattern plays a very important role in VANET, it would be unrealistic to assume that mobile nodes are moving according to simple mobility schemes such as random walk, or random direction. A common approach is to separate the synthesis of mobility from the network simulation: first generate the entire node mobility trace through either road traffic simulators, public transportation schedule, or real time logs, and then feed the obtained trace

as input to the network simulator. For example, in [19], the traffic log is first generated by a scalable traffic simulator (MMTS). However, the problem of this approach is that the radio propagation model in VANET cannot be simply extracted from city map and motion trace.

The general approach to model radio propagation is to use well-established statistic models. Since buildings and other vehicles also interfere with the radio signal, real channel conditions are much more complicated. In [35], the authors provide a new signal propagation model to estimate signal coverage through existing maps. However, attenuation caused by dynamic factors such as trucks or obstacles, is still hard to reproduce. Unlike mobility models, it is very costly in terms of memory resources to record the channel conditions between every node pair and use them in simulation. Vehicular testbeds have been used to measure real channel conditions [36]. However, when running vehicular experiments, the mobility pattern changes from experiment to experiment, leading to inconsistent results.

The third challenge is external interference that maybe caused by surrounding wireless networks. This varies randomly in an urban environment and it is extremely difficult to measure and reproduce in a simulator.

From the above we conclude that simulation results are unlikely to provide an accurate comparison of vehicular protocols because of the difficulty of modeling motion, propagation and interference in a realistic manner. The only safe method then, seems to be testbed measurements. Unfortunately, here another equally difficult challenge awaits us: experiment reproducibility. Suppose we run first routing algorithm A (say, AODV). Then, 10 minutes later, we run algorithm B (say OLSR). Can the results be fairly compared? Generally not, because in 10 minutes the external interference may have changed, the motion pattern of the various vehicles involved in the experiments may have changed (for example, due to unpredictable traffic lights), and the radio propagation may have changed (say, due to mobile obstacle beyond our control).

Thus, realism is definitely the big advantage of testbeds (with respect to simulation platforms). But now the problem of inconsistency of environmental conditions emerges. This is the challenge we address. Namely, we propose to implement a VANET testbed that is set up to compare different protocols in the exact same topology and channel conditions. The testbed contains multiple mobile nodes, and each node runs multiple virtual machines in parallel. We run a different protocol in each machine. We show that in this way a realistic, fair side-by-side comparison of the different protocols can be made.

The rest of the chapter is organized as follows: In section 2.2, related work is introduced. Section 2.3 describes our system platform. The virtualization overhead is evaluated in section 2.4. Two protocols, AODV and OLSR are used to evaluate our platform, and the results are reported in section 2.5. Section 2.6 summarizes this chapter.

2.2 Related Work

The Wireless Signal Propagation Emulator developed at CMU [40] accurately emulates wireless signal propagation in a physical space. In fact it can be trained to replicate a number of different representative environments, including urban vehicular scenarios. The emulator takes in the signals generated by wireless network cards through the antenna port, subjects the signals to the same effects that occur in a real physical space (e.g. attenuation, multi-path fading, etc.), and feeds the combined signals back into the wireless cards. The wireless emulator forms the basis for a wireless testbed that supports highly realistic experiments, while also being fully repeatable and easy to control. The emulator, however, has some limitations in reproducing arbitrary motion patterns. In addition, although the propagation scenario is very realistic, it is still artificially created, as opposed to measured in real life.

Orbit [20] is a testbed that combines an indoor radio grid emulator and an outdoor field trial network. This testbed is available for use either via remote or on site access. The indoor radio emulator consists of 400 802.11 radio nodes in a 20x20 grid with a wired control channel to each node, and the outdoor field trial network consists of 50 nodes. Each wireless node is equipped with two 802.11 wireless interfaces. As to mobility support, the outdoor testbed isgrounded, and the indoor emulator only supports virtual grid mobility. For example, in [21] the authors present a comparison between AODV and OLSR, performed through the ORBIT indoor testbed. We use MAC level filtering to block the connection between two neighbor nodes, and create node connectivity that is similar to mobile nodes. The work provides some initial observations and indicates that AODV performs better than OLSR in terms of stability.

The department of computer science at University Uppsala has opened to the community the Ad-hoc Protocol Evaluation Testbed (APE Testbed) [22]. APE is an encapsulated execution environment with tools for post test-run data analysis. It is like a small linux package with Ad-hoc configuration and network traffic analysis tools. The package can be installed in either windows or linux environment to perform Ad-hoc experiments and display the result with GUI. In [23], Lundgred et al. used APE to evaluate the performance of AODV and OLSR with up to 37 nodes along indoor hallways and athletic fields. Their results show AODV performs better than OLSR when mobility is high. However, to the best of our knowledge, there has not any mobile or VANET experiment using APE testbed.

Several vehicular experiments and testbeds have been proposed in the past [24]-[29]. However, none of them compared the behavior between different protocols. A general problem is that controlled mobility in vehicle experiments is almost impossible unless the scenario is extremely simple.

Many academia facilities have mesh network testbeds that use AODV or OLSR to perform layer 3 routing. Some of the mesh testbeds are deployed in real environment, such as MIT

RoofNet, Berlin Roof Net, and Mesh Networking from Microsoft Research [37]-[39]. These systems provide experimental results in real world channel condition, but lack node mobility.

Various simulation studies compared the difference between AODV and OLSR, before any comparison could be performed on real world testbed. Among these researches, [30] is a simluation study that based on VANET scenario. We compare AODV with OLSR through the Vehicular Mobility Model (VMM) they proposed. VMM considers both road and obstacles (macromobility) and the point of view of the driver (micromobility).

To sum up, Figure 2.1 shows the different approaches to compare different routing protocols. Simulation can capture the node mobility well, but fails to provide realistic channel conditions (propagation and interference). On the other hand, the testbed approach provides a convincing channel condition, but cannot support complicated mobility at the same time. In this chapter, a novel approach is used to perform parallel experiments. Thus, the mobility pattern does not need to be reproduced in order to compare different routing protocols.

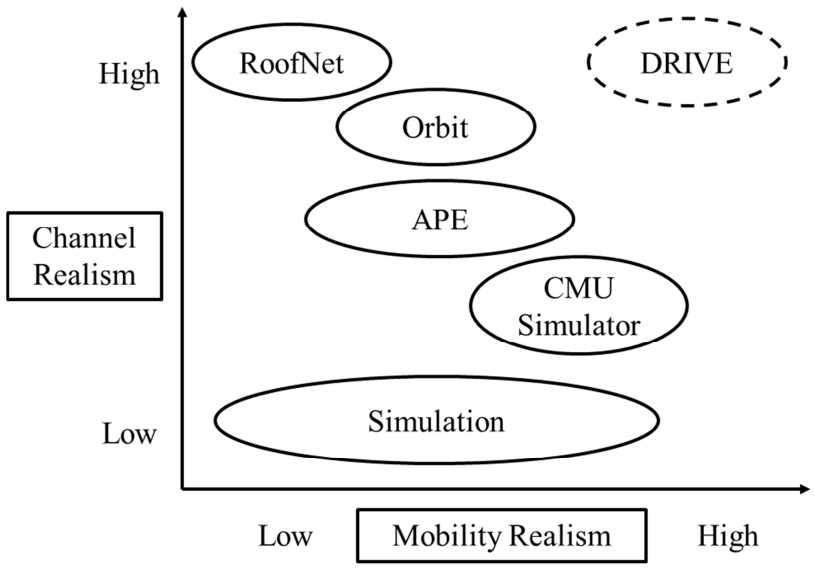


Figure 2.1 Tradeoff between mobility and channel realism

2.3 Overview of Experiment Platform

The In this section we describe our experiment platform. The main purpose of this study is to provide the community with an experimental platform that allows the evaluation of different protocols and applications under the same environmental conditions. We propose to run several experiments together in one single run. Even if mobility pattern and channel conditions cannot be reproduced, experiments running at the same time still experience the same environment condition. The idea is to have multiple virtual machines running on the same mobile node, and each of them runs its own protocol and application, while sharing the same wireless card to communicate with the rest of the network. In the following we first describe the hardware platform and subsequently the software setup.

2.3.1 Hardware Platform

Figure 2.2 Node setup on a vehicle

Our network nodes are common commercial laptops with an Intel Core 2 Duo CPU, 2GB of RAM and 120GB hard drive. Each laptop is instrumented with a Ubiquiti SRC wireless card with Atheros 802.11 wifi chipsets (AR5004). The Atheros 802.11 wifi chipset is supported by the open source linux madwifi driver [45], that allows many customized settings

including fixed channel selection, transmission power, and monitor mode. For our experiments all the wireless cards are in Ad-hoc mode, using channel 1 only. The transmission power is set to the hardware supported maximum (19dbm).The wireless card is connected to a magnetic mount antenna with 8dB nominal gain. Each laptop is also equipped with a GPS receiver to be able to track the position of the nodes during the experiments. Figure 2.2 shows an example of a node setup on a vehicle.

2.3.2 Software Platform

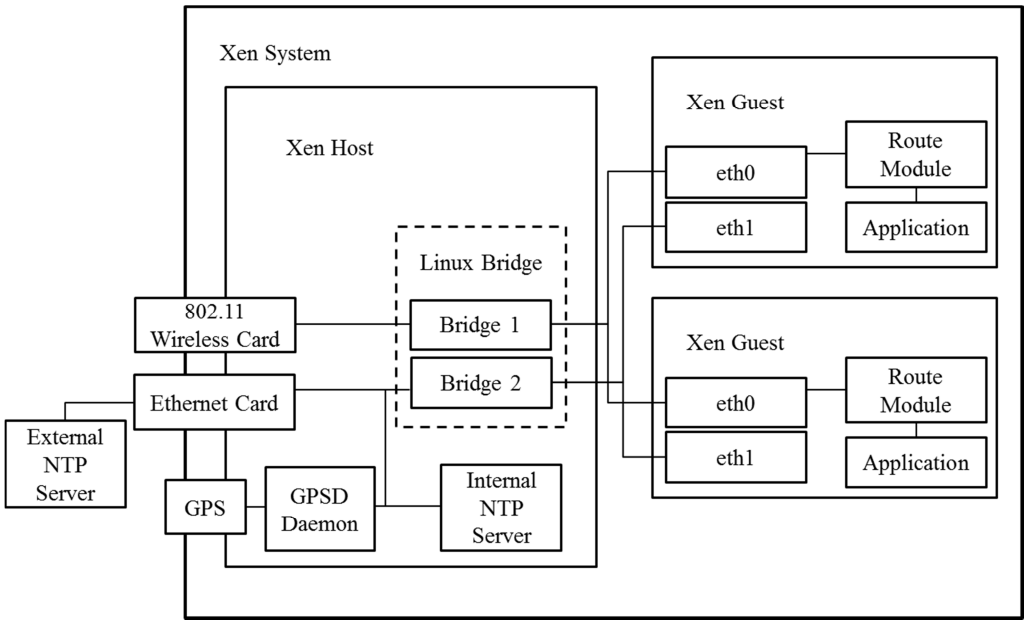


Figure 2.3 Software Platform

Each node is installed with the linux Gentoo distribution (kernel version 2.6.21) patched with Xen. Xen is an open source industry standard virtualization environment that allows several virtual machines (Xen guests) to share the same hardware (see Figure 2.3). Each virtual machine can run a different operating system with different protocols and applications. In addition, the Xen platform provides the ability to connect the guest operating system to the host operating system (Xen host) through a virtual Ethernet bridge. This bridge internally connects to one virtual network card for each Xen guest. Therefore, every Xen guest shares

the same outgoing link to the physical wireless card. The Xen guests use this bridge to communicate with the outside world.

In addition to the first bridge, we setup a second bridge that allows communication between host and guest without introducing additional overhead that might affect performance. In order to perform better time synchronization, we use the second bridge to run network time protocol (NTP) and correct the time drift between Xen host and guests clocks. We also use of the second bridge to obtain geo information. The Xen host is connected to the GPS device by gpsd [49]. Gpsd is an open source daemon that interfaces with the GPS device and provides a socket interface for retrieval of the location information. Through the Ethernet bridge, Xen guests can access GPS information through the gpsd daemon. This allows the use of application and protocols that require GPS information to be used in our experiment system.

The most important components on Xen guests are the routing module, the synchronization tool, and the network traffic generator application. Since each Xen guest is anindependent system, it can run its own operating system and routing module.

2.4 Xen Overhead Evaluation

Our virtualized approach requires the sharing of resources and introduces additional overhead over a regular linux system. The overhead introduced by the system can be categorized into two ways:

- **Virtualization Overhead:** the virtualization platform adds an extra software processing layer between applications and hardware. In addition, hardware resources are used to run the virtual machines. Therefore, the delay between the generation of a packet at application layer and the time it is sent out of the hardware interface is longer than for a regular linux system with no virtualization.

- **Sharing Overhead:** When multiple virtual machines transmit at the same time, they contend for the same physical medium. This contention causes lowers the maximum throughput the hardware can achieve and introduces extra delay due to the enqueueing of packets.

In this section we investigate on the impact of such overhead factors and find the limitation introduced by our virtualization system.

2.4.1 Virtualization Overhead

In order to evaluate the overhead introduced by the virtualization platform we performed a set of experiments to compare the performance of a native linux system and our virtualized environment. For the virtualized environment, we setup two Xen guests on the same machine. We then activate a constant UDP data flow that generates packets every 150ms destined to another wireless receiver, only from one of the Xen guests, while the other Xen guest remains idle. Figure 2.4 shows the end-to-end delay for both the native linux system and the virtualized platform for an increasing packet size. A first observation is that the data transfer on the virtual environment suffers from higher latency. Moreover, an increase of the packets size causes an increase of the delay for both systems. However, the virtualized environment suffers a higher increase. This can be explained by the fact that every packet generated by Xen guest needs to be moved in memory more times than in the native system before it reaches the outgoing interface. Nevertheless, the delay increase for the virtual environment is in the range of tens of milliseconds. Compared to the delay introduced by external factors such as multi-hop paths and channel contention, the overhead is negligible. In our field tests, our UDP traffic generator sends out 50 bytes per 150 ms. With this network load the overhead of the two systems is comparable.

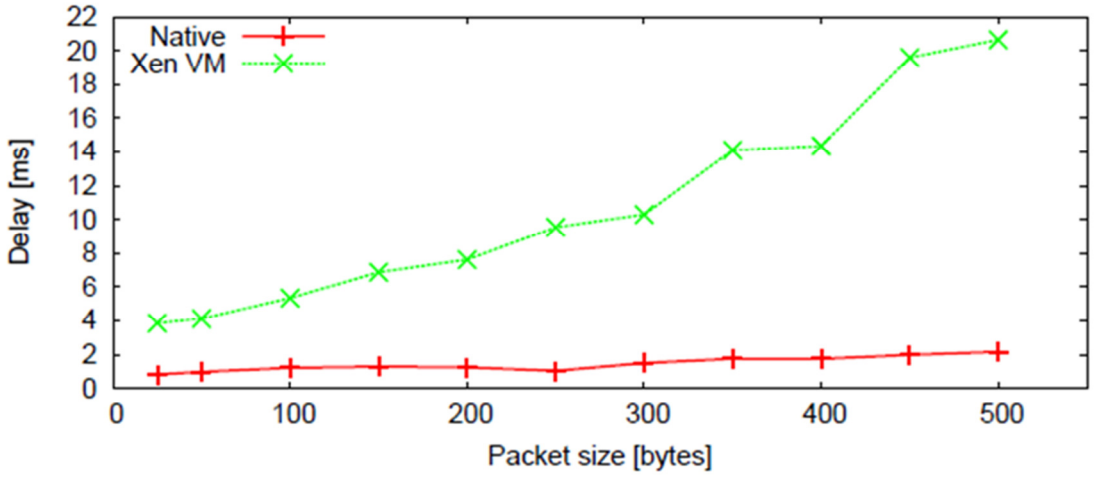


Figure 2.4 Overhead introduced by the Xen virtualization: packet transmission delay as a function of the packet size for the native system and for the virtualized environment

2.4.2 Sharing Overhead

On our virtualized platform, the Xen guests share the same physical resources. It is then important to understand what is the maximum network load at which the sharing of the same resource causes losses or excessive delays. The Xen host connects the Xen guests to the physical network interface through a linux network bridge. The operative system provides a fair share of CPU time to the virtual machines, and thus to the read and write locks on the network bridge. The bridge queue policy is First In First Out (FIFO) that coupled with the fair share of the locks avoids starvation and provides fair sharing of the resources. In this section we plan to assess what is the ”safe zone” to run our experiments so that the sharing overhead does not affect the validity of our results. We setup one UDP traffic generator on two Xen guests destined to another wireless node. One Xen guest is running AODV and the other is running OLSR as routing. In order to force the maximum sharing overhead, both Xen guests generate data traffic at the same time. This is possible as the guests are synchronized through NTP. We compare the packet loss and the end-to-end delay experienced by our system to the one of a similar setup on a native linux system. Figure 2.5 shows the average end-to-end delay

for increasing packet sizes. We can observe that the delay remains negligible up to packet sizes of 400 bytes for the virtualized environment, for larger packet sizes the delay increases considerably.

We can conclude that for the data load used in our experiments (50 bytes every 150 ms) the overhead introduced by the virtualization system is negligible.

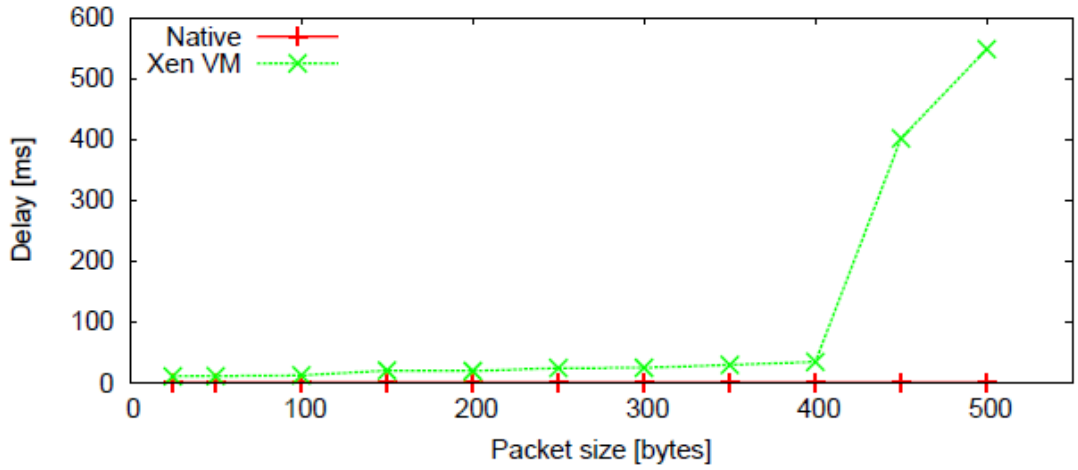


Figure 2.5 Overhead introduced by the Xen virtualization: packet transmission delay as a function of packet size for two UDP flows generated on two virtual machines and on a native linux system

2.5 Field Experiments

We performed a set of eight rounds of experiments. In these eight rounds we varied the use of AODV and OLSR (singularly or in parallel), the use of the interference nodes and the number of mobile nodes. Table 2.3 reports the setup we used in each experiment. In this session we initially present the details of the setup for each experiment round. We then show that running subsequent experiments leads to different environmental conditions and therefore to incomparable performance results. Finally, we show that, using our platform, the two routing protocols experience the same environmental conditions over time and therefore their performance can be compared without any ambiguity.

2.5.1 Experimental Setup

Several software components were run on the Xen host and guest machines throughout all of our experiments.

1) Host Setup:

Mactrace: Each Xen host runs a mactrace tool that periodically (every 200 ms) broadcasts hello messages, which contain position and timestamp information. These hello messages are generated directly at layer 2 using linux raw sockets. This solution avoids the intrinsic delays introduce by higher network layers (e.g. IP and UDP). Each node keeps a neighbor table storing its current neighbors together with the time the last packet was received from each particular neighbor. Upon receiving a mactrace hello message, each node will update its neighbor table (i.e. add the message sender to the table if it was not present before or update the time this neighbor was last seen). The neighbor table is refreshed every second, deleting the neighbors from which no packet was received since the last refresh. Gathering the mactrace logs from all the nodes, we are able to construct the network connectivity matrix over time for further investigations.

Positiontrack: Each Xen host queries the gpsd daemon every second to obtain the GPS position. The GPS position is then stored together with the GPS timestamp and the system clock timestamp.

Tshark: Each Xen host runs tshark. Tshark is an open source packet analyzer that allows us to save information relative to each single packet sent or received through the network interface relative to wireless card. In particular we can record the exact time each packet was either sent or received.

Synchronization: Before performing our experiments, all the hosts are synchronized to the same time using NTP. This allows temporal correlation among the trace logs.

2) Guest Setup:

Routing Protocols: On each node, we run two Xen guests in addition to the Xen host. Each guest runs Gentoo linux distribution (kernel version 2.6.18 with Xen support). One Xen guest runs AODV-UU [46], an implementation of the Ad-hoc on demand distance vector routing (AODV) [41] from Uppsala University, and the other Xen guest uses the Optimized Link State Routing (OLSR) [42] implementation from olsr.org [47]. AODV is a reactive Ad-hoc routing protocol that uses route requests and route replies to build a multi-hop route on demand. OLSR is a proactive link-state routing protocol, which periodically sends Hello and Topology Control (TC) messages to obtain link state information throughout the whole network. These protocols are the most commonly used reactive and proactive routing protocols in research testbeds, and are often compared as earlier shown in related work. In addition, reliable linux implementations exist for both protocols. In contrast, the geo-routing scheme, Greedy Perimeter Stateless Routing (GPSR), often discussed in urban VANET studies, still lacks reliable implementations. Thus, we select AODV and OSLR to be the benchmarks for comparison. Table 2.1 and Table 2.2 show the routing parameter settings for AODV and OLSR. These variables are based on previous work [21] and account for the vehicular scenario used in our experiments. Each Xen guest also runs the tshark tool to capture all traffic through its network interface. In addition, the kernel routing table is logged to record route changes at any given time.

Network Traffic: We developed a simple application to generate a constant UDP stream of 50 bytes data segments. Each packet will store in its data segment its packet sequence number and the timestamp when the packet was created. Each traffic flow transmits 800 packets at 150 millisecond intervals.

Table 2.1 AODV PARAMETERS

Hello message interval	500ms
Allow hello loss	2
Delete period	1s
Active route timeout	2s

Table 2.2 OLSR PARAMETERS

Hello message interval	500ms
Hello message validity time	1s
Topology Control (TC) message interval	1s
Topology Control (TC) message validity	2s

3) Physical Setup:

Figure 2.6 displays a top view of the physical topology setup. All the experiments were performed around the Engineering IV building at UCLA. We split our nodes into two types: Fixed nodes and mobile nodes. Fixed nodes do not move and their antenna is placed on top of a 1.5 meter stand to ensure good signal propagation. Mobile nodes are placed on vehicles and their antenna is placed on the top of the car to avoid interference due to the car body shell. Four fixed nodes, represented by the pin icon in Figure 2.6, were placed at the four corners of the building. The resulting rectangle is approximately 60 meters wide (East-West) and approximately 50 high (North-South). Each fixed node is in line of sight with the nodes placed at neighboring corners. Therefore, neighbor nodes are within the transmission range of each other. The building blocks diagonal connections, and thus nodes placed at opposite corners cannot directly communicate with each other. In addition to the four nodes in the corners, we placed two interference nodes indicated by the wave icon on Figure 2.6. These interference nodes generate layer 2 broadcast packet bursts of random length uniformly distributed between 0 and 100 packets of random size uniformly distributed between 50 and 1000 bytes. After each burst they idle for a random period of time uniformly distributed in the interval [0,30] seconds.

Figure 2.6 Top view of topology setting for all experiment rounds

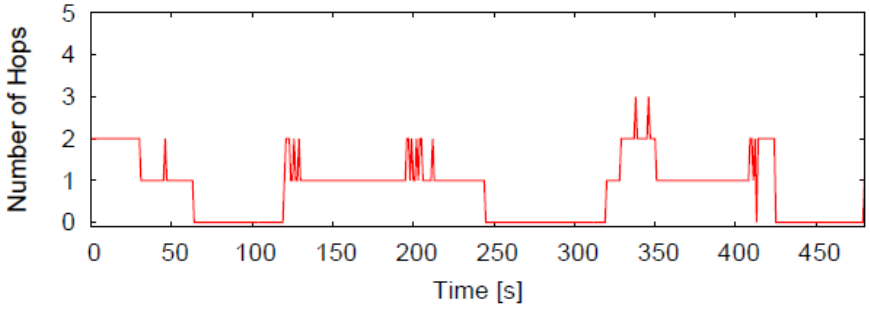
In each experiment round we change the routing protocols that are running: AODV alone in rounds 1 and 2; OLSR alone in rounds 3 and 4; both AODV and OLSR in parallel in rounds 5 through 8. In order to emulate a change in the environmental condition we run each single experiment first with the interference nodes shut down (rounds 1, 3, 5 and 7) and then with the interference nodes transmitting (rounds 2, 4, 6, 8). We also explore different mobility patterns: one set of experiments involves one single mobile node placed on a car revolving clockwise around the building. With this setup each Xen guest on the mobile node sends the UDP stream to one of the 4 peer Xen guests on the fixed nodes at a time, in a roundrobin manner. Another set of experiments involves two mobile nodes placed on two cars both revolving clockwise around the building. A summary of the 8 rounds of experiments is reported in Table 2.3.

Table 2.3 EXPERIMENT ROUNDS SUMMARY

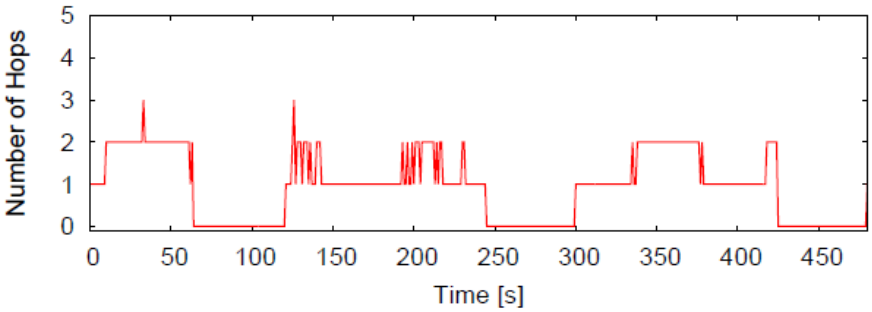
Experiment Name	AODV	OLSR	Interface	Mobile Number
Round 1	X			1
Round 2		X		1
Round 3	X		X	1
Round 4		X	X	1
Round 5	X	X		1
Round 6	X	X	X	1
Round 7	X	X		2
Round 8	X	X	X	2

2.5.2 Repeatability of Experiments

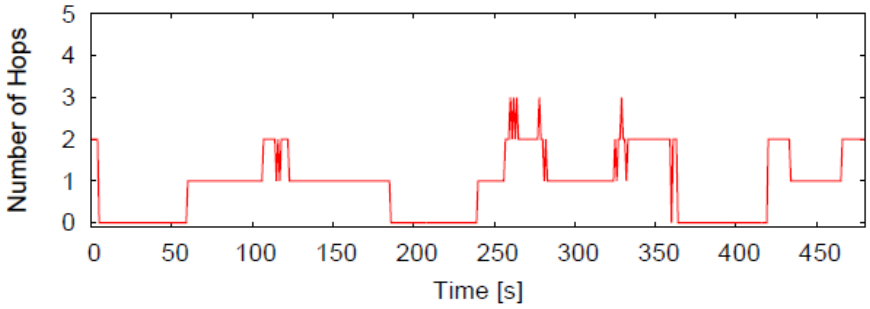
Using the logs from our mactrace tool, we constructed the network connectivity matrix. With this matrix, we can know, at any given time, if any two nodes are connected. By running the Dijkstra shortest path algorithm, we can obtain the optimal hop count which is defined as the shortest hop count from the UDP sender to the receiver. Figure 2.7 displays the optimal hop count as a function of time in experiment rounds 1 through 4. Hop counts equal to zero represent idling periods of the sender. In the considered four rounds we are using a very simple mobility pattern, a single car revolving around the building. With such mobility pattern, we expect to observe periods of time when the sender is 1 hop away from the receiver alternated to periods of time when the sender is 2 hops away from the receiver. In fact, it is what we can observe in Figure 2.7, exception made for few points in which the sender is reported 3 hops away, due to loss of some mactrace hello packets. However, each experiment peculiarity is represented by the relationship between 1-hop and 2 hops periods. In fact, we can observe in each round different period durations and also different patterns of alternation. All this reflects on the performance of the routing algorithms. In Figure 2.8 we show the packet hop count distribution for experiment rounds 1 through 4. Packet hop count is the number of hops a successfully received packet took to reach the destination. Comparing the results from AODV round 1 and AODV round 2, we note that the hop count distributions are not the same in different runs. Same thing happens for OLSR round 3 and OLSR round 4. Even with such a controlled mobility, the environmental changes between experiments cause even the same protocol to perform differently. Therefore, no meaningful conclusions can be drawn from the comparison of two different protocols running at different times.



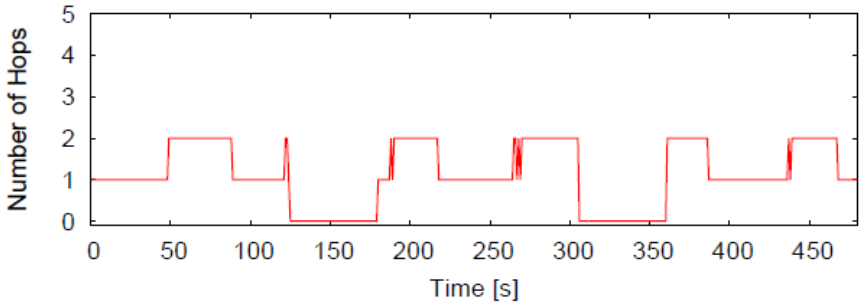
(a) Round 1



(b) Round 2



(c) Round 3



(d) Round 4

Figure 2.7 Network connectivity: optimal hop count between sender and receiver over time for rounds 1 through 4

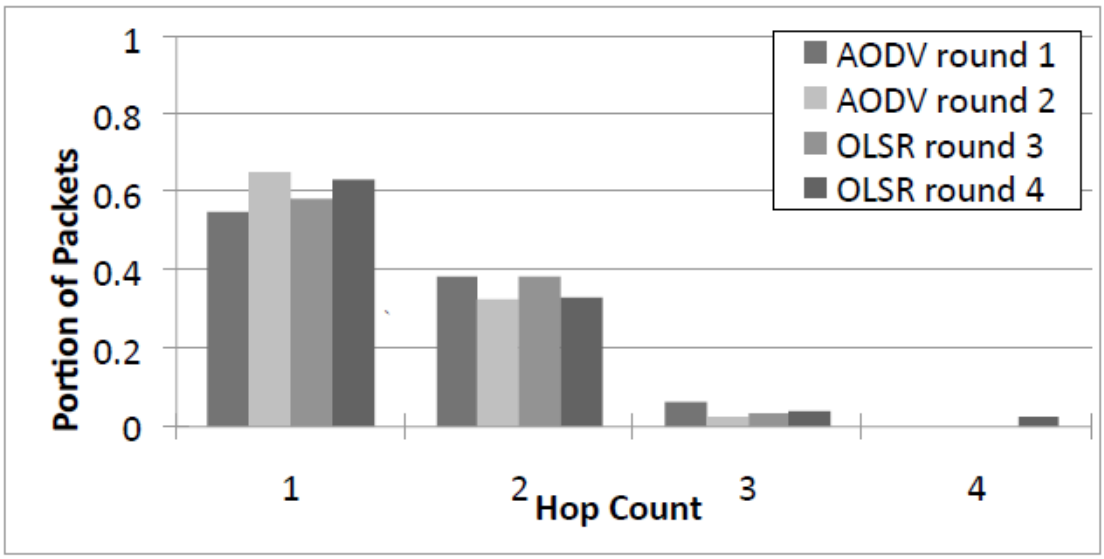


Figure 2.8 The packet hop count distribution for experiment rounds 1 through 4

2.5.3 Parallel Evaluation

Thus far, we have shown the drawbacks of running experiments at different times. In this section we present the advantages of running experiments in parallel. Figure 2.9 presents the instantaneous optimal hop count together with the actual packet hop count of AODV and OLSR obtained in experiment round 8 from one mobile node to the other. Optimal hop counts equal to zero represent periods of inactivity of the sender, and packet hop counts of zero represent situations in which no packet was delivered to the destination. We can observe that both routing protocols consistently react to changes in the underlying connectivity. This consistency in the reaction to changes in the physical network topology is only achievable if the two protocols are experiencing the same environmental conditions.

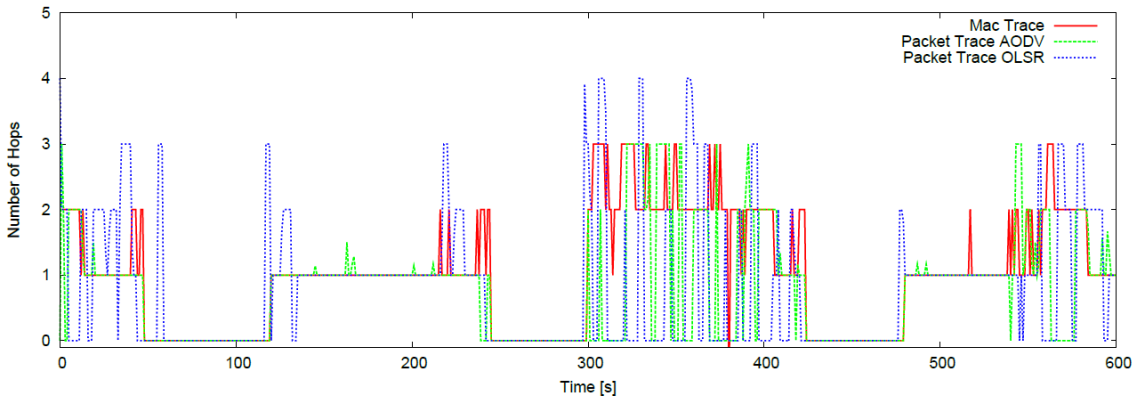
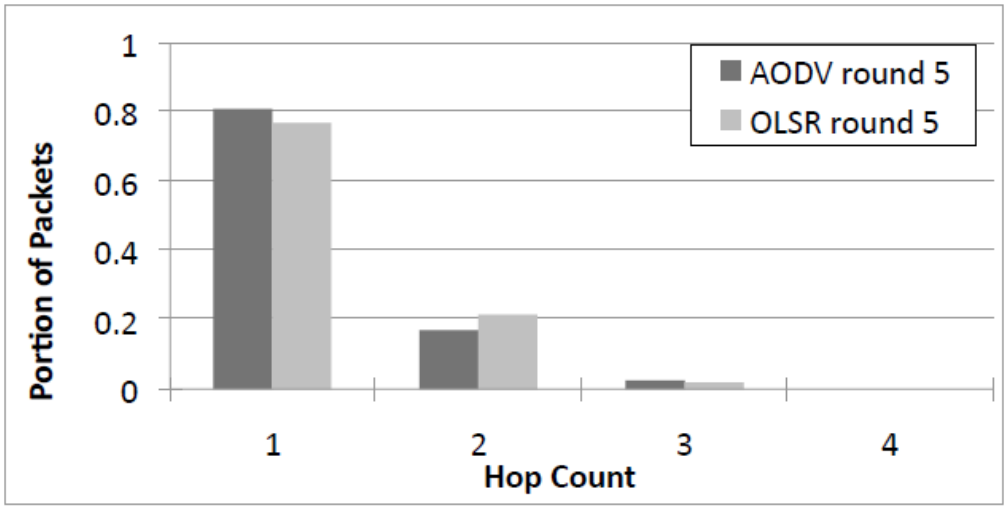
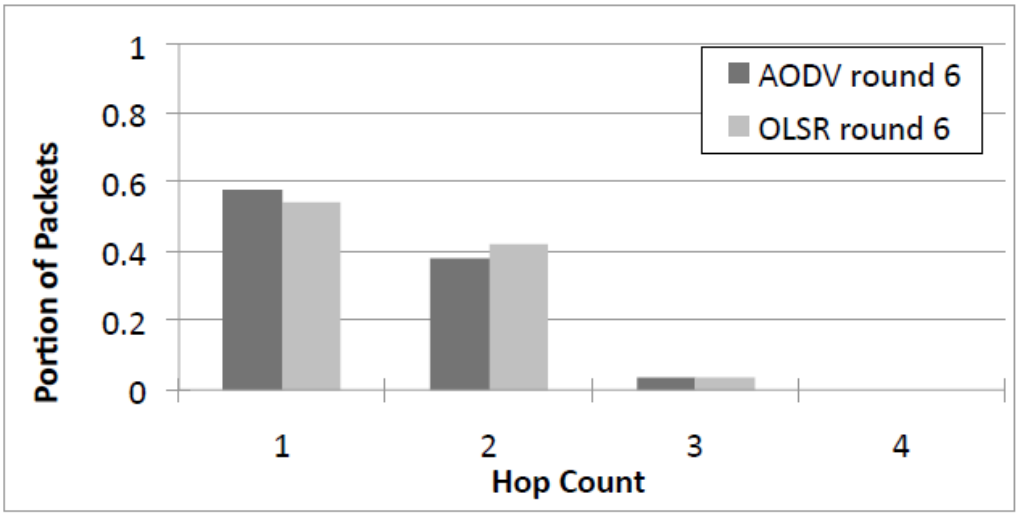


Figure 2.9 Parallel evaluation: optimal hop count and packet hop count for AODV and OLSR over time in round 8

In the following paragraph, we discuss the results obtained during the parallel experiments, namely rounds 5 through 8. Figure 2.10 present the packet hop count distribution for both AODV and OLSR in experiments rounds 5 and 6. If we were to compare the packet hop count distribution of OLSR in round 5 and the one of AODV in round 6, we would observe that OLSR provides a lower hop count than AODV. Vice versa, if we were to compare the packet hop count distribution of AODV in round 5 and the one of OLSR in round 6, we would observe that AODV provides a lower hop count than OLSR. In this case different experiments provide opposite results. Instead, if we consider the parallel case, we can observe that the relationship between the performance of AODV and the performance of OLSR is invariant throughout the two rounds. In fact, AODV has generally a lower hop count that OLSR. The same conclusions can be drawn for the packet deliver ratio reported in Figure 2.11: AODV consistently outperforms OLSR in both rounds.

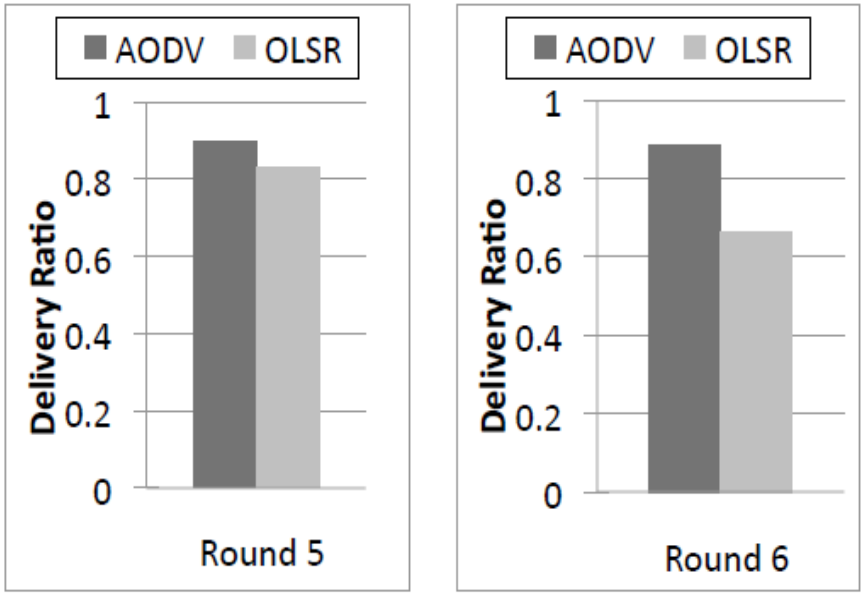


(a) Round 5



(b) Round 6

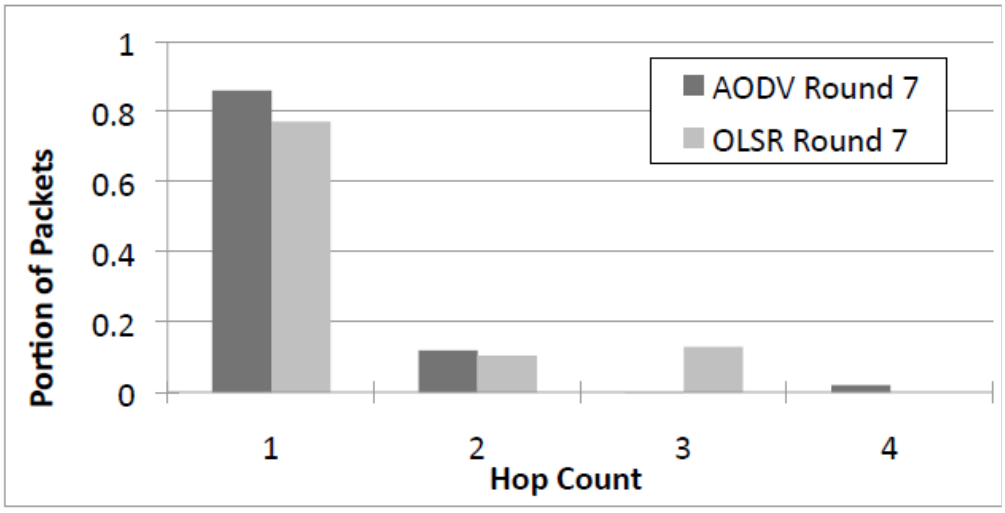
Figure 2.10 Parallel evaluation: packet hop count distribution for AODV and OLSR in rounds 5 and 6



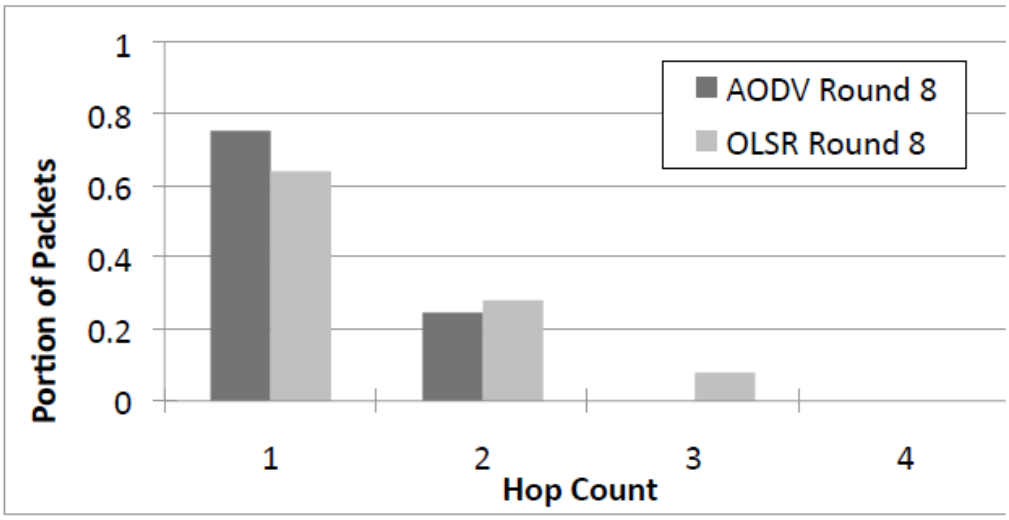
(a) Round 5 (b) Round 6

Figure 2.11 Parallel evaluation: packet delivery ratio for AODV and OLSR in rounds 5 and 6

In the following we prove that even increasing the complexity of the mobility our proposed platform can still provide meaningful results. Indeed, in rounds 7 and 8 we send our UDP stream from one mobile node to another. In this case the network connectivity is too complicated to be consistently reproduced in two subsequent experiments. Figure 2.12 shows the packet hop count distribution for AODV and OLSR in rounds 7 and 8. The same conclusions drawn for rounds 5 and 6 are valid in this case. Indeed AODV consistently outperforms OLSR both in terms of hop count and delivery ratio, shown in Figure 2.13.

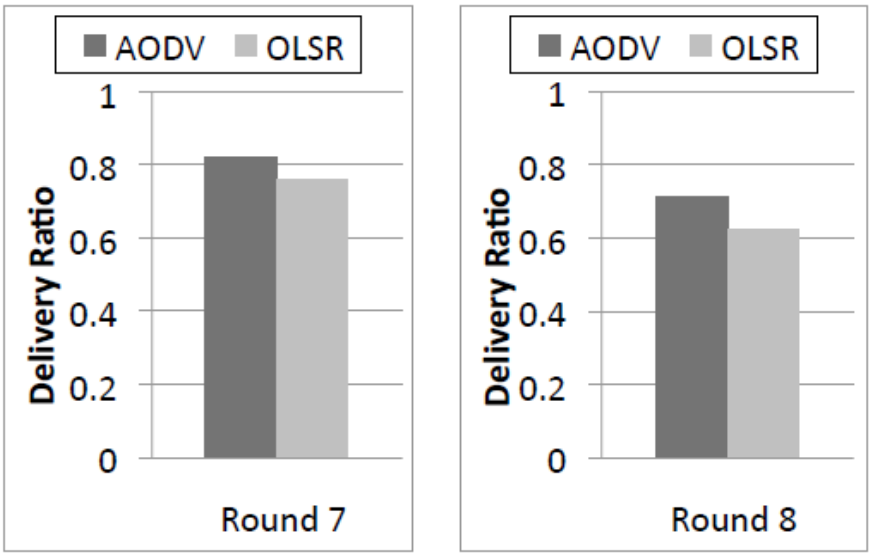


(a) Round 7



(b) Round 8

Figure 2.12 Parallel evaluation: packet hop count distribution for AODV and OLSR in rounds 7 and 8



(a) Round 7 (b) Round 8

Figure 2.13 Parallel evaluation: packet delivery ratio for AODV and OLSR in rounds 7 and 8

2.6 Summary

Running experiments in a testbed is the most reliable way to compare the performance of different protocols in realistic scenarios. However, each scenario in a VANET is characterized by its own mobility, propagation obstructions and environmental interference. These environmental variations make it difficult to faithfully replicate experiments. The virtual machine, parallelized testbed we proposed can perform parallel runs under the same environment conditions. Therefore it provides more consistent and reliable results. In addition, the vehicle positions and network connectivity can also be logged for future use as traces in simulation studies. Though our experiments, we learned that if the environment has uncontrolled interference (like the two sideline bursty interference generators), it is not possible to compare different protocols at different times, even if the mobility pattern is simple.

Also, we observed through our MAC-layer traces that simple repeatable mobility patterns do not imply repeatable network connectivity.

In this Mobile Cloud case study, we have shown that by introducing software virtualization into mobile nodes, new services can be support, such as our parallel experiment platform. However, issues still remain where the system is not flexible enough. In specific, once the system is deployed, dynamic reconfiguration is still difficult.

To address these issues, we look at the programmability and control introduced by Software-Defined Networking. In the next chapter, we provide background on SDN and the first step in wireless and mobile SDN: Radio Access Network SDN.

Chapter 3. Wireless and Mobile Software-Defined Networking

3.1 Introduction

The growth in mobile data, the inherent need to simultaneously operate over multiple wireless technologies, and the rapidly evolving mobile services market impose significant challenges for wireless and mobile networks. SDN, with its separation of the control and data planes, can be used to bring benefits into wireless and mobile networks space. In this chapter, we first provide background on the SDN technology. Then, we describe UCLA's SDN campus deployments. Followed by proposed solutions for Radio Access Network (RAN) SDN, including SDN for the mobile packet core and SDN for WiFi access networks. We then present two prototype SDN testbeds that we built to show how a wireless and mobile SDN can perform in real world situations.

3.2 Background on SDN and OpenFlow

The core concept of SDN [2] is the separation between the control plane and the data plane. The latter is used for the data forwarding while the other is exploited for the network traffic control. This separation enables faster configuration and provisioning of network connections. Instead of individually accessing and configuring each network devices, a network administrator can program the behavior of the network in a centralized way. In addition to simplifying the deployment of new protocols and applications, SDN brings programmability that can enable a more efficient and efficient network.

Figure 3.1 shows a high level concept of SDN. Network intelligence is centralized in software-based SDN controllers, which has global view of the network and are capable of controlling, through standard protocols, the underlying network devices. In SDN, network devices are no longer required to be fully distributed systems that implement and understand all the different network protocol standards. Instead, network devices can accept input from SDN controllers in making network decisions. This can save a lot of resources, as network devices can be less sophisticated architectures, and not requiring custom configurations. In addition, SDN controllers, with a better view of the overall network, can program network behavior that was difficult in a distributed matter.

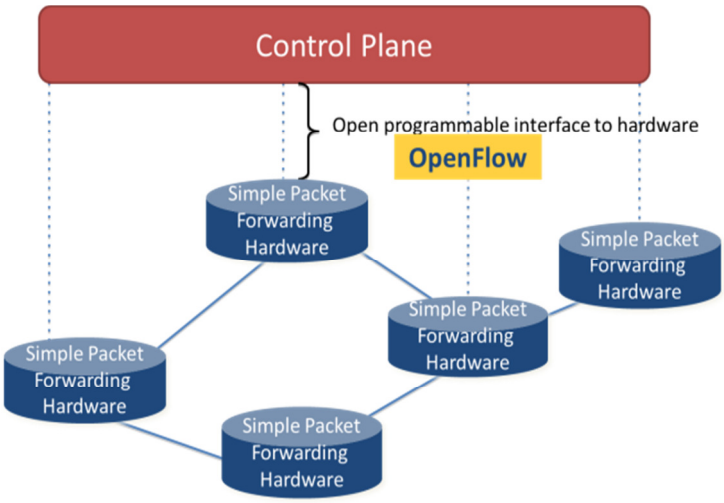


Figure 3.1 Software Defined Networking Concept

OpenFlow (OF) [3][4] is the most commonly used standard protocol for communication between the SDN control plane and data plane. Figure 3.2 shows an OpenFlow network comprising the two main components in SDN architecture: the OpenFlow controller and several OpenFlow-enabled switches that communicate using a secure channel.

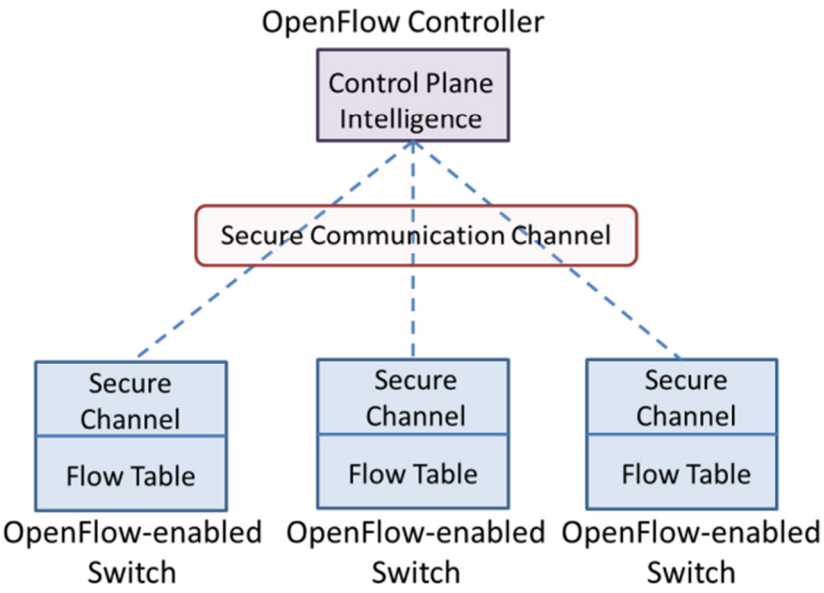


Figure 3.2 OpenFlow components

The controller is a software program element that is used to modify the content of flow tables, which are located in each OpenFlow-enabled switch. Each flow table contains a list of flow entries, which consist of Match Fields, Priority, Counters, Instructions, Timeouts, Cookie, and actions associated. When a packet comes to the switch there is a lookup into the matching field of each entry. If there is a match, the packet will be processed according to the actions of the matching flow entry. On the other hand, if a packet does not match, a table-miss occurs. In this case, different actions can be taken as specified in the table-miss flow entry, for instance, the OpenFlow-enabled switch can encapsulate the packet and send it to the controller through the secure channel or directly drop the packet.

OpenFlow controller controls the behavior of the network by sending flowmods packets to modify the content of flow tables. The controller has two ways to add rules in the switch: (I) proactively, where the controller takes the initiative and adds rules before packet arrival into the network; or (II) reactively, where the controller reacts because of an event in the network, such as a previously unrecognized packet. An example of such an operation would be an Access Point (AP)/switch that sends a data packet to the controller (by encapsulating the

packet in an OpenFlow control packet called packet-in) because it does not know how to deal with it. Then, the controller sends the flowmods packet to the APs/switches with instructions.

One notable feature in an OpenFlow network is that once a specific traffic flow matches the flow table, the OpenFlow-enabled switch "knows" how to treat this flow and does not need further interactions with the OpenFlow controller. While this allows switches to forward traffic efficiently, issues arise when the flow table rules are no longer consistent with the network condition. In other words, if network conditions such as topology have changed, until the controller inserts/updates the flow table entry, an OpenFlow-enabled switch will use the old (and potentially incorrect) rule. We show how this is an issue later in chapter 4 where node mobility is common in mobile networks.

3.3 UCLA SDN Campus Deployment

ULCA, under the NSF EAGER Project, is establishing a live OpenFlow test bed on the UCLA campus, with the goal of allowing researchers in different disciplines to run experiments requiring high volume data exchange, intense processing, and/or seamless mobility as a production level service on the campus network. UCLA links the campus OF network to national OF fabrics via the regional CENIC OF network. This deployment looks to support four key applications from the domains of eScience, GENI and smart transportation, health care, and manufacturing. The campus implementation support the above applications consists of a 10Gbps OF-enabled hub switch connected to CENIC and multiple second-tier OF switches associated with the four projects.

Although SDN technologies are currently being widely discussed and are key elements in the GENI architecture, there is little operational or campus-level architectural experience with

using them. The project explores the issues associated with deploying SDNs and connecting them to real-time information sources. One primary objective of the project is to rework current UCLA network engineering practice where extensive hardware reconfiguration is required to support the sophisticated processing and efficient storage of data by several of the research centers. Flexible reconfiguration is a new paradigm for campus networks.

In addition to the UCLA campus deployment, the Network Research Lab (NRL) in the Computer Science Department also has several SDN deployments. Figure 3.3 shows our SDN/OpenFlow testbed that consist of an OpenFlow controller; two OpenFlow enabled WiFi routers, and several client devices and switches. We use the Floodlight controller as the OpenFlow controller.

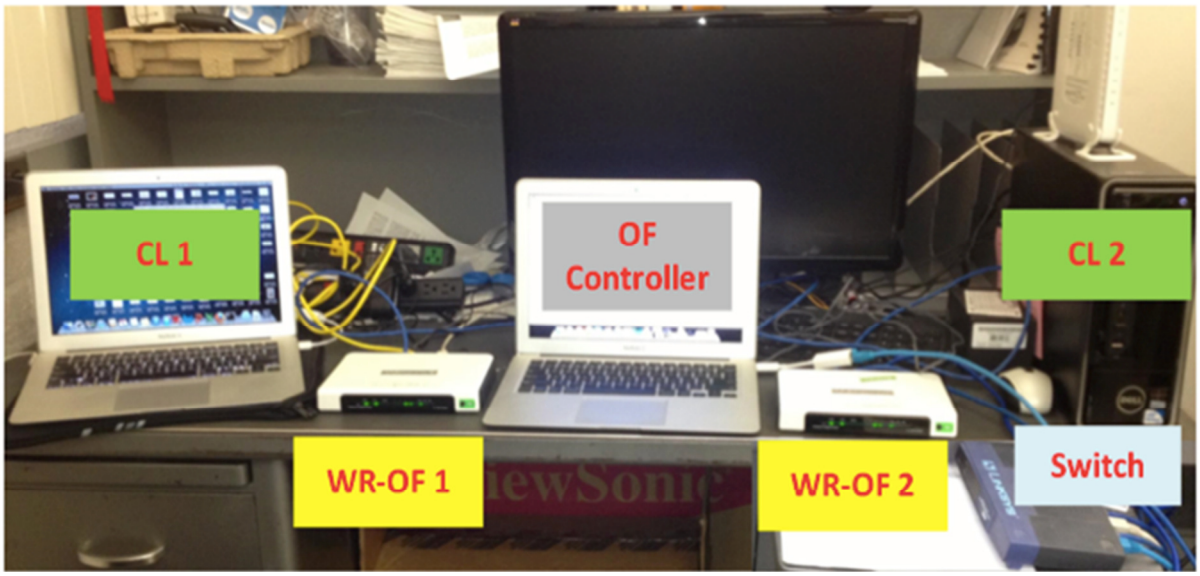


Figure 3.3 NRL Wireless Access Testbed

In addition to the physical testbed setup, we also have a hybrid physical/emulated environment that utilizes Mininet [54]. This is shown in Figure 3.4. Mininet is a network emulator that is able to emulate complete networks in a single system. The OpenFlow controller that controls this architecture is also the Floodlight controller. One of the deployments built upon this emulated testbed is an QoS SDN solution using Mininet/WiFi access [55].

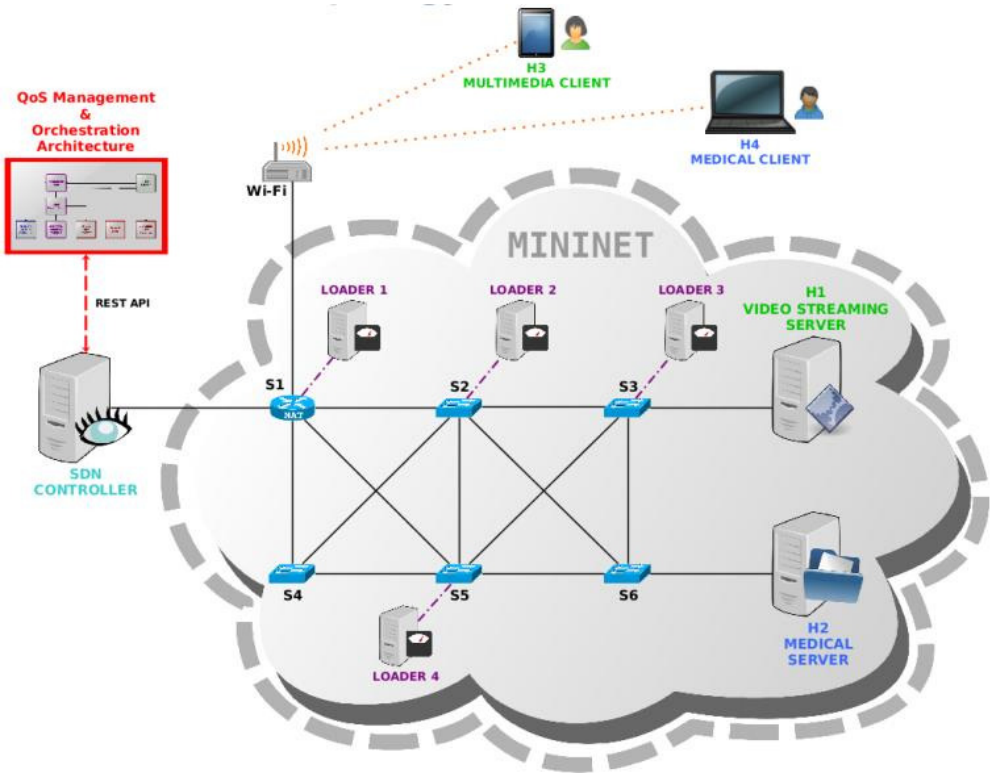


Figure 3.4 NRL OpenFlow deployment using Mininet

Also, another deployment on NRL is architecture for dynamic adjustment of virtual network (VN) resources under SDN environments (edge networks) [56]. The key idea is to identify the available resources, and to increase or decrease the resources allocated in the VN based on the client's usage and current network resources, aiming to maintain the quality required by the client.

3.4 Proposed Radio Access Network SDN Solutions

SDN architectures are believed to be able to bring flexibility, scalability and distribution into mobile networks, especially for the packet core. Efforts, primarily led by the Open Networking Foundation's Wireless and Mobile Discussion Group [8], has been made to define and prioritize use cases of 3GPP Evolved Packet Core (EPC) based networks that could benefit from the use of SDN/OpenFlow protocols. This involves using SDN/OpenFlow

protocols in the core network to compliment the EPC connected by wired physical links but supporting wireless users.

In addition to the mobile packet core, SDN also bring benefits for WiFi access. Examples include using SDN to provide more personalized and reliable services in dense WiFi access network. Also, SDN can be used for Unified Access Network for Enterprise and Large Campus, where a unified access network uses the same controller to manage both wired switches and wireless access points with users in standardized way.

In this section we describes how SDN can be applied to enhance the mobile packet core, in specific, how SDN/OpenFlow can be used for a flexible and scalable packet core, and SDN and SDN/OpenFlow for Service Chaining. We also describe efforts incorporating SDN to enhance WiFi access.

3.4.1 Flexible and Scalable Packet Core

The need for service providers to support video streaming and other data intensive applications has greatly increased traffic in mobile networks. Current deployments of LTE and 3G networks rely on centralized Evolved Packet Core (EPC) to handle end user sessions before the traffic is routed to their specific destinations, such as the Internet and corporate networks. The EPC provides for Quality-of-service (QoS) levels, per flow monitoring, taffic steering/engineering, and controls how traffic reaches networks. As more and more mobile users create more and more traffic, scalability is now becoming a concern for EPC network.

Introducing SDN, and separating traffic switching from control, can be used to help solve the scalability issues. For example, if there is an increase in demand or threshold congestion, additional switches may be provisioned to handle the load. Also, if the operator wants to take down switches for maintenance or upgrade, other switches can be configured easily through

SDN to handle the load level. Another advantage of the separation of control and data is that they have different approaches to prevent failures, so the separation makes failure recovery easier to accomplish with less effort. For the data plane, redundant switches and paths are configured in case of switch failure. While for the control plane, redundant servers or virtual machine instances are provide as backups to the primary servers/VMs. This provides flexibility as the control and data plane elements are independently scaled to meet demand.

Figure 3.5 shows an EPC network that is realized using SDN/OpenFlow and control/data plane separation and can scale control and data paths independently.

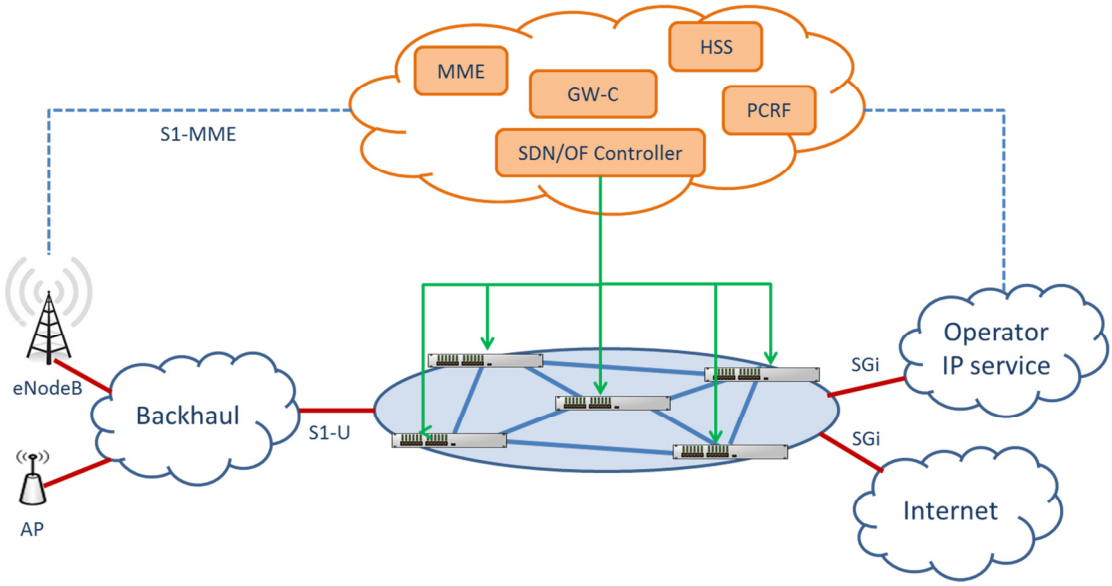


Figure 3.5 Mobile Packet Core with SDN/OF

The OF controller and OF enabled switches switch architecture is in the EPC network and is transparent to the connecting networks.

As stated earlier, separating control and data would allow each of the entities to be scaled independently. If there is an increase in demand or threshold congestion, additional switches may be provisioned or configured to handle the extra load. Similarly, control plane functions can be independently scaled. The separation can also be expected to provide more flexibility for fault tolerance, by providing better redundancy and resilience for the system.

3.4.2 Service Chaining with SDN

As traffic from users pass into the access network, it usually passes runs through multiple services deployed by mobile operators to provide enhancement of user experience. These service act on the user traffic to provide different functions, examples include caching, traffic optimization, Firewall, access control, traffic analytic, etc.

Services are organized into multiple service chains for different set of requirements, based on requirements from either the user or mobile operator. As shown in Figure 3.6, user's flows may go through different service chains based on user identity, flows' characteristics, or policy preconfigured.

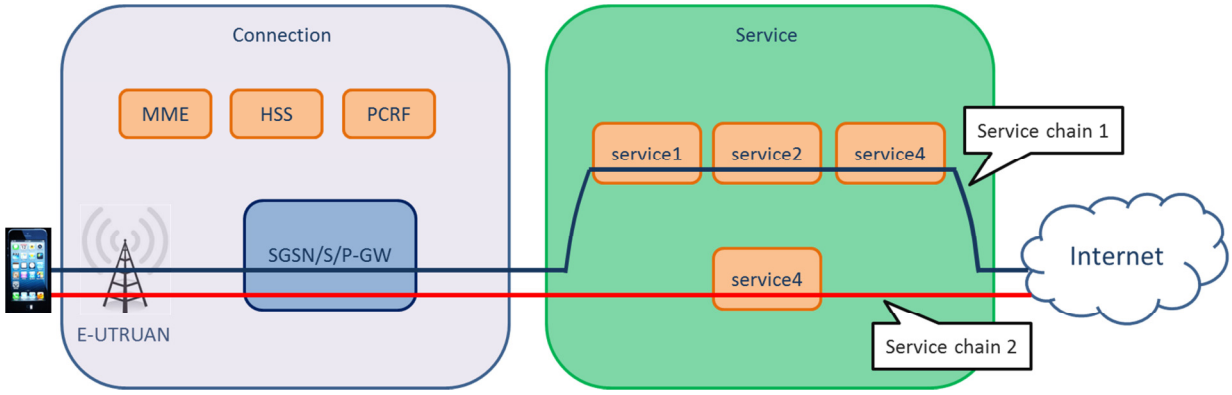


Figure 3.6 Current Service Chaining in Mobile Service Domain

We can see that typical service chains are simply serial chains, preconfigured as shown in Figure 3.6. The serial nature of the chains results in several challenges. (I) It is difficult and requires considerable effort to insert new service or to upgrade existing ones. (II) Modifying, such as adding, removing, or changing, a single service might lead to a reconfiguration of all services within the same chain. (III) A failure of a single service might disable service of the entire chain. (IV) Each service has to process and forward all the traffic flows, which increases the overall processing, the traffic delay and also service costs.

SDN/OpenFlow address these challenges by adding programmability and flexibility to chaining services. Using SDN/OpenFlow, mobile operators can selectively steers traffic to

the desired service in any order to provide flexibility and agility. Figure 3.7 shows how service chaining operates with the introduction of SDN/OpenFlow.

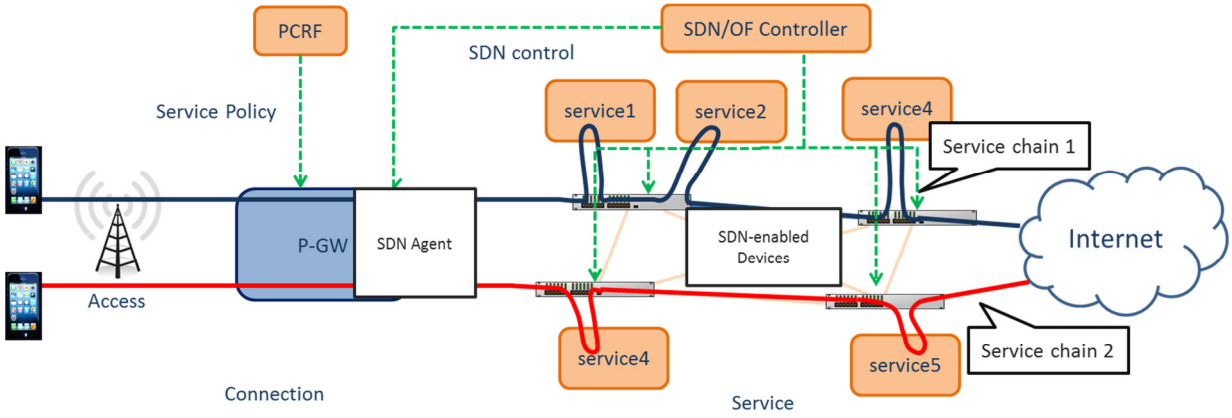


Figure 3.7 Service chaining with SDN/OF

As user traffic enters from the access, the SDN agent classifies user traffic based on user profile, radio access and application type. This traffic is then mapped to associated service chains for processing according to the policy from the Policy and charging rules function (PCRF). The SDN agent coordinates with the SDN/OF controller to translate policy and service requirements into rules for the underlying SDN-enabled devices. The rules build the path to specific services as required. Using SDN provides the agility to direct traffic flows to any services in a flexible and programmatic way.

3.4.3 SDN for Dense WiFi Access

In the world today, large part of the population lives in high-density areas. Many people live in multi-resident building that share networking resources. This results in high density and can have impact to network performance. High interference and congested channels is commonplace, while misconfiguration often leads to poor channel and power allocation. Even though multiple access points are available, users can access only theirs, occasionally leading to poor coverage which in turn degrades the channel for everyone. A number of

factors contribute to this: lack of coordination between individual homes, no expertise from users, and poor manageability of WiFi itself.

One of the efforts of SDN is to enhance performance for dense WiFi access [52]. An SDN framework is presented for designing a dense WiFi network which aims to provide users with a personalized, fast and reliable network service. Figure 3.8 shows a virtualized WiFi architecture. Each user configures and accesses their own personal network. The network controller configures the APs, and based on its strategy maps personal APs to the physical infrastructure to optimize performance.

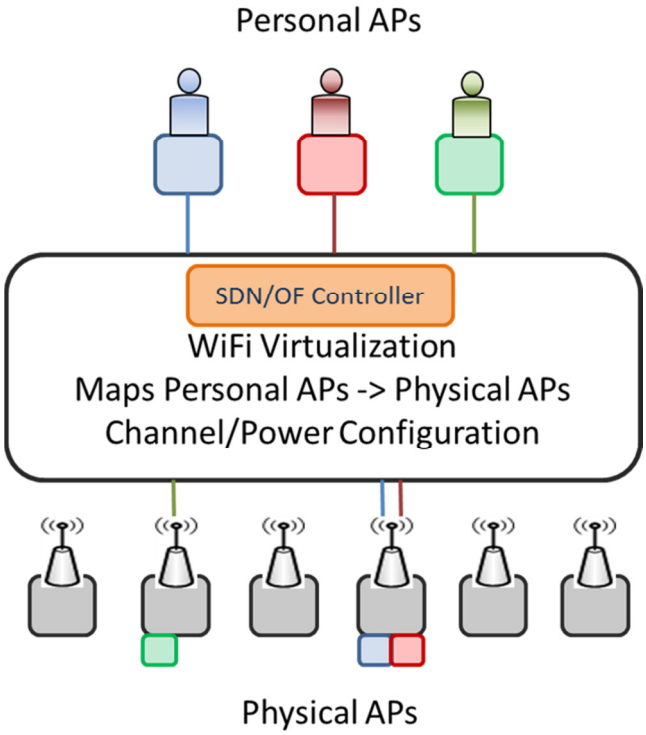


Figure 3.8 Virtualized WiFi architecture

In this architecture, the personal AP will follow the users everywhere. As users move around, they feel like they are always connected to their personal AP, which are in fact multiple physical APs that are mapped by the SDN/OF controller. By providing the mapping between personal APs and physical APs, the SDN/OF controller can also turn off APs that are not needed, which can save power and create less interference environment.

3.4.4 SDN for Unified Access in Enterprise and Large Campus

In the previous scenario, the user moves around with their device and stays connected to the same personal AP. In this section we describe another case where the user moves around and accesses different devices. In the modern enterprise and large campus, users want to access the network resource from anywhere (Wi-Fi, remote or wired ports), using any device including BYOD (Bring Your Own Device). SDN can be used allow and control user access to network resources based on user identity, the device in use, the location of access and which applications are desired, no matter which medium the user connected to. Figure 3.9 shows the architecture overview.

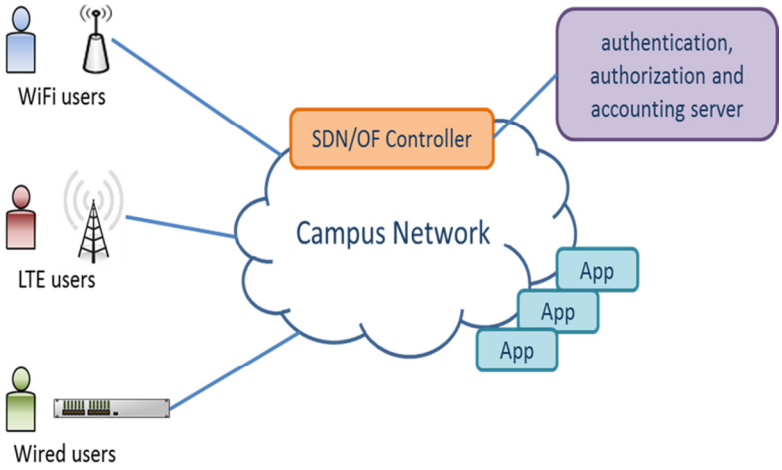


Figure 3.9 Unified Access in Enterprise and Large Campus

User access, regardless of the medium used to connect, is controlled by the SDN controller as traffic enters the network. The SDN controller works with the authentication, authorization, and accounting server to authenticate users to identify them. The SDN controller then evaluates the user's access policy to make decisions whether to add/delete network pathways, which controls the user's access to the network resources.

3.5 Prototype SDN Testbeds

In this section, we describe two SDN prototypes that we built to show how such systems can operate in real-world situations. The first is SmartFlow, a prototype that shows how SDN can be used for mobile traffic offloading in a programmatic way. The second is Remote Control Virtualcast, where we show how SDN can be used to share resources on virtual networks and manage and control network flows.

3.5.1 SmartFlow-Intelligent mobile offloading

The “SmartFlow” demo is about the ability to not only engineer but also move traffic, if and as needed, in a network is critical to cope with unexpected situations such as sudden traffic peaks, network disruptions and ensure business continuity. OpenFlow and SDN-based approaches can be instrumental in making the network and its functions programmatic. Traffic and data offloading is one example of immense importance and applicability in mobile and wireless networks. Figure 3.10 shows our demo setup. Mobile user is connected to two access networks, and the choice of which access network to use to provide traffic to the mobile user is done by a SDN controller.

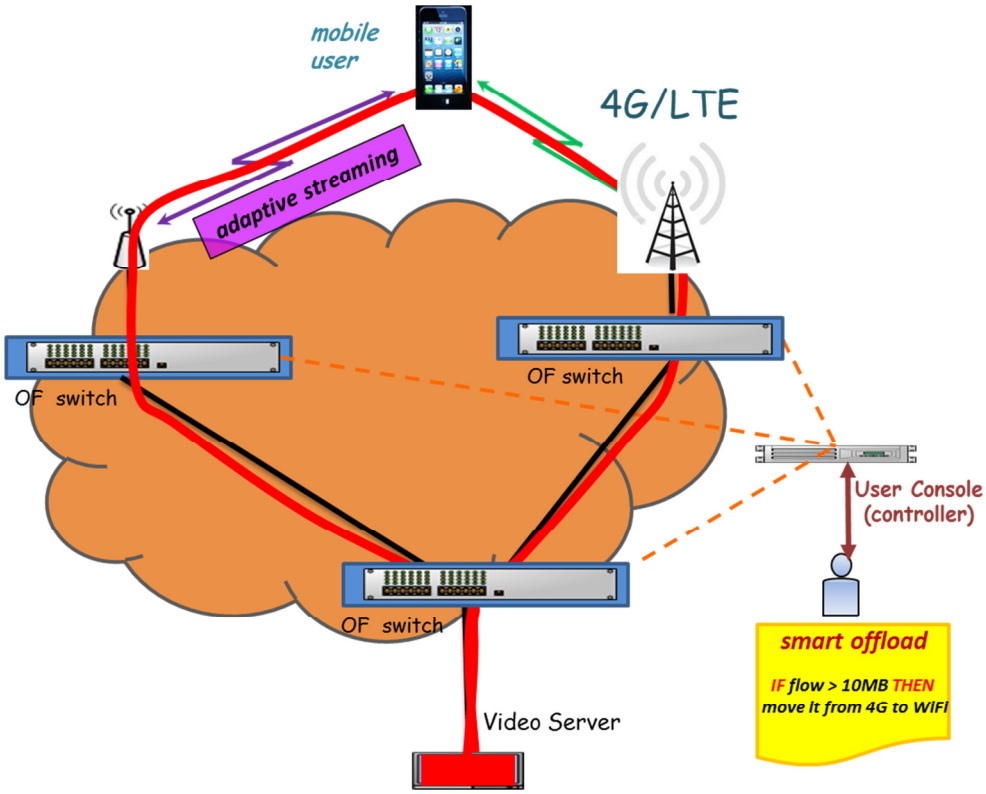


Figure 3.10 SmartFlow-Intelligent mobile offloading

In our demo, we program the SDN controller to monitor traffic flow on the network, and move traffic from one access network to another based on aggregated traffic flow. We use this to showcase the ability of SDN to deal with unexpected conditions (e.g. traffic on one network exceeding capacity). Being able to react to urgent situations or prevent network anomalies is of paramount significance for a service provider. A better groomed network (e.g., load-balanced, QoS- observant) means SLA (Service Level Agreement) compliance, OpEx savings, less outages and dropped calls, and eventually better quality of experience for the users or customers whatever the application is.

We show in this demo how SDN presents a great innovation allowing for dynamic, on-demand provisioning, controlling and management of networks in a programmatic fashion. This comprises functions such as bandwidth management (link capacity, radio frequency or wavelength programmable assignment.

3.5.2 Remote Control Virtualcast

The “Remote Control Virtualcast” demo is about sharing resources on virtual networks and managing, controlling network flows, in particular video flows (although the concept applies to any type of flows), through WiFi access points. We chose video traffic flows for visibility and demonstrability purposes but also because video traffic, from the server all the way to the client, is one of the most common traffic types in mobile networks today.

In this demo, we program our SDN testbed and its OpenFlow-enabled devices so that several video servers can play their content (different video streams) to a number of mobile clients, all servers using the same multicast address. Without using SDN and being able to program the devices, in this case our NEC switches and Wireless access points, clients would concurrently receive all video streams thus a mixed signal that would not be viewable, as shown in Figure 3.11.

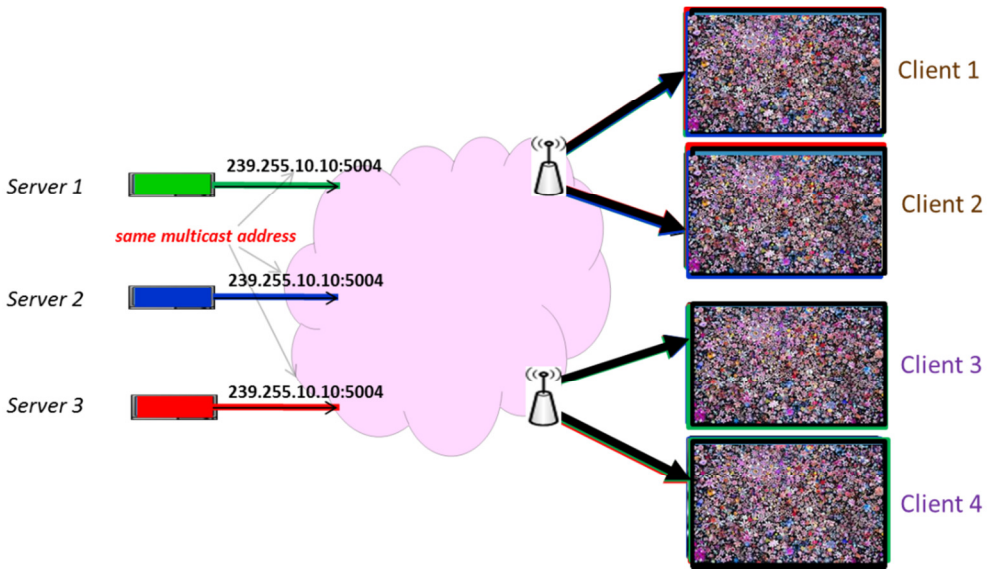


Figure 3.11 Servers using same Multicast Address Resulting in Mixed Signal

By introducing SDN, including a specialized controller called the Flowvisor [53] that creates network slices and allows multiple tenants to share the same network infrastructure. Our prototype can program the devices so that the different streams play on different clients without mixing up the content, as shown in Figure 3.12.

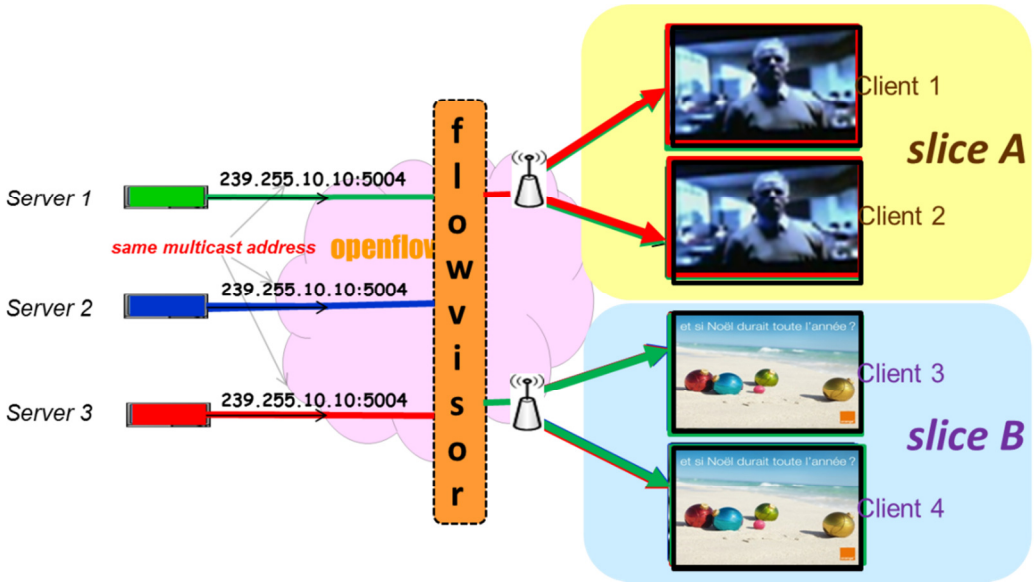


Figure 3.12 Remote Control Virtualcast

As an additional feature we programmed the devices so that the video streams would rotate amongst the clients on preset time intervals, hence the name “Remote Control” for our demo. For example given three servers S1,S2 and S3 and two clients C1 and C2, S1 is streaming to C1, S2 to C2, while S3 stream plays nowhere (i.e., it is blocked) then after 30 secs, S3 is streaming to C1, S1 to C2 while S2 is on mute. For simplicity, if we assume m video servers and n video clients we considered, without loss to generality, m>=n. The “Remote Control” app is completely transparent to both the servers and the clients, i.e., they are totally unaware of what is transpiring. This in a way promotes and exhibits the independence and the isolation of the video streams and the server-client relationship.

3.6 Summary

In this chapter, we provided background on SDN, UCLA SDN deployments, and how RAN SDN can be used to provide benefits in mobile access networks, including the mobile packet core and WiFi access. We also shown, through our prototype SDN testbed deployments, on services provided though SDN on those area.

However, while RAN SDN provides benefits to mobile networks, it mainly addresses the wired side of the cloud. In specific, it deals with traffic that has passed the access radio, and does not provide capabilities within mobile node and networks.

In the next chapter, we introduce Software-Defined Vehicle Networks, which we use SDN to enhance the Mobile Cloud in peer-to-peer space. Using SDN in MANETs/VANETs is one of the potential wireless infrastructures for Software-Defined Mobile Networks, and we show how it can be used toward a Software-Defined Mobile Cloud.

Chapter 4. Software-Defined Vehicle Network

4.1 Introduction

We have shown how VANETs can be used as one of the potential underlying network architectures to provide services in the Mobile Cloud. However, there are still challenges in the deployment of VANETs' applications, such as changing and reconfiguring existing network behavior, lack of flexibility to apply network policies, and applying automation and traffic differentiation. Therefore, open and flexible vehicular architectures are key requirements to allow experimenters to test their solutions in productive environments, as well as to improve the management of network resources, applications, and users.

To address these challenges, we look at SDN [2]. SDN brings a flexible way to control the network in a systematic way, with OpenFlow [3][4] as the most commonly used SDN protocol. The flexibility of SDN makes it an attractive approach that can be used to satisfy the requirements of VANET scenarios. Applying SDN principles to VANETs will bring the programmability and flexibility that is lacking in today's distributed wireless substrate, while simplifying network management and enabling new V2V and V2I services.

In this chapter, we focus on applying SDN into VANETs. In specific, we look at the architecture, operations, and benefits of Software-Defined Vehicle Network (SDVN) services and new functionalities to support them. By decoupling the control and data planes in VANETs, network intelligence and state can be logically centralized and the underlying network infrastructure is abstracted from the applications. Thus, it will be possible to have highly adaptive, flexible, programmable, and scalable VANETs environments. A use-case on

routing is presented to demonstrate the benefits of integrated SDN VANETs architectures in forwarding data in multipath scenarios.

The reminder of the chapter is structured as follows. Section 4.2 provides background information on VANET in relation with SDN/OpenFlow. Section 4.3 describes the architecture and operations of SDVN. Benefits and services for SDVNs are presented in Section 4.4. Section 4.5 presents simulation evaluation and Section 4.6 presents the conclusion and future work.

4.2 Background

In this section we describe some background information on VANET and its relationship with SDN/OpenFlow used through the chapter. In our SDVN architecture, OpenFlow is used as base and is integrated to VANET wireless environments.

In a typical VANET, Vehicles communicate with each other through V2V communication in Ad-hoc fashion, and V2I communication through road-side-units (RSU) and mobile broadband (e.g. 4G/LTE). Figure 4.1 shows the components and communications with a typical VANET.

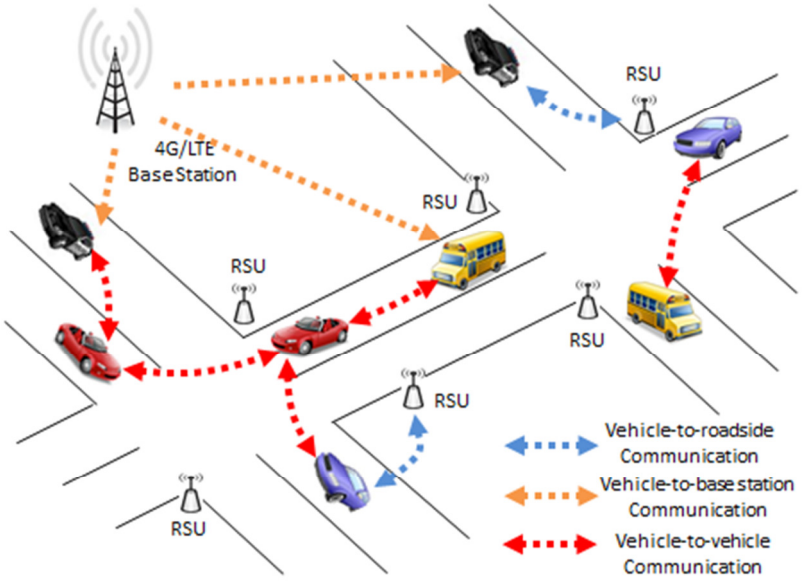


Figure 4.1 VANET Component and Communications

Traditional VANET services include vehicle and road safety services, traffic efficiency and management services, and infotainment services. Vehicle and road safety services are those that target the decrease of traffic accidents and loss of life to vehicle occupants. Traffic efficiency and management services aim to improve traffic flow, traffic coordination, and to provide local and map information. Infotainment services aims to provide information and entertainment such as multimedia data transfer and global Internet access.

In mobile networks, the introduction of SDN and OpenFlow will enable the programing of base stations' wireless data plane and enhance the functionalities of the core networks. Thus, it will improve the management of resources and mobile devices and create a great opportunity for new services and control functions. In dynamic wireless mobile environments, such as VANETs, the use of SDN can reduce interference; improve the usage of channels and wireless resources, as well as the routing of data in multi-hop and multi-path scenarios. We describe the benefits of SDVN s later in this chapter.

4.3 SDVN Architecture

In this section, we describe the architecture of SDVN and its operations. The goal is to describe how VANETs take advantage of SDN concepts and functionalities to improve resource utilization, select best routes, and facilitate network programmability.

4.3.1 Architecture Overview

To enable SDVN, our architecture incorporates the following SDN components:

- **SDVN controller**: The logical central intelligence of the SDVN system. The SDVN controller controls the network behavior of the entire system.
- **SDVN wireless node**: The data plane elements that are controllable by the SDVN controller. They are the vehicles that receive control message from the SDVN controller to perform actions.
- **SDVN RSU**: Stationary data plane elements that are controllable by the SDVN controller. They are the infrastructure RSUs that are deployed along road segments.

Our proposed architecture extends SDN to operate in mobile wireless VANET scenarios. In our architecture, we choose to use different wireless technologies for controlling and forwarding planes as expected in future VANET systems: Long range wireless connection (i.e., LTE/Wimax) for control plane, and high bandwidth wireless connection i.e., Wi-Fi for data plane. The practical reason is that in VANETs not all nodes are easily reachable from the Infrastructure via RSUs. Figure 4.2 shows the communication between components in our SDVN.

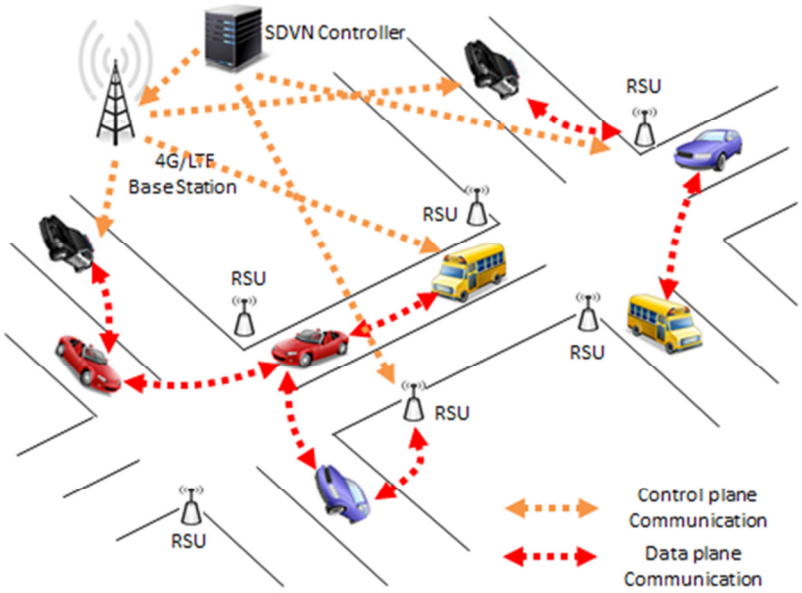


Figure 4.2 Software-Defined Vehicle Network Communications

4.3.2 SDVN Wireless Node

Figure 4.3 shows the internal components of a SDVN wireless node. It contains all the functionality of an OpenFlow-enabled switch in traditional OpenFlow networks, plus additional intelligence to enable different modes of operation in VANET environments. The number of WiFi and LTE interfaces used is based on configuration and the service that the SDVN wireless node is required to support. The SDVN module is the combination of packet processing and the interface that accepts input from a separated control plane.

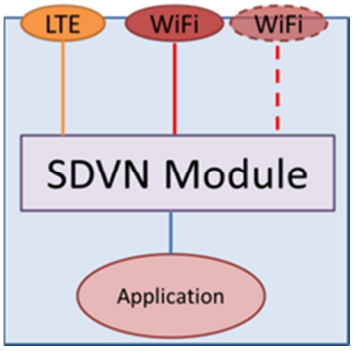


Figure 4.3 SDVN Wireless Node Internals

One distinct characteristic of Ad-hoc networks is that the nodes act both as Hosts (sending/receiving traffic) and Routers (forwarding traffic on behalf of other nodes). An SDVN wireless node is therefore both an SDN data plane forwarding element and an end-point for data. Traffic from any wireless node (e.g. application traffic) will run through its own SDVN module before being sent, which allows the SDVN controller to determine the access of user traffic into the network.

4.3.3 SDVN Controller

Different from the device SDVN agent, which is used more as a backup in case of loss of communication, the SDVN controller is the primary intelligence in SDVN. The SDVN controller is responsible for populating the flow tables of the SDVN wireless nodes with flow rules to control how traffic is moving in the network. There are two instances of the architecture depending on the connection conditions between this SDVN controller and SDVN wireless nodes:

- **Constantly Connected SDVN Controller**: a stable connection is maintained between SDVN controller and the SDVN wireless node. An example would be the LTE control channel in an urban environment. This controller-node communication is similar to that of a wired SDN system, where flow rules are inserted reactively or proactively based on policy.

- **Intermittently Connected SDVN Controller**: the connection between the SDVN controller and SDVN wireless nodes is intermittent. It is assumed that a SDVN wireless node will not always be able to establish a connection to the global controller. Flow rules are pushed by the SDVN controller during periods where connectivity is established; they are enforced by the local SDVN agent on the wireless node with knowledge on how to treat traffic based on policy, or; they are created by a combination of the two above methods.

4.3.4 Operations Overview

Figure 4.4 shows the System operations overview on how an SDVN operates.

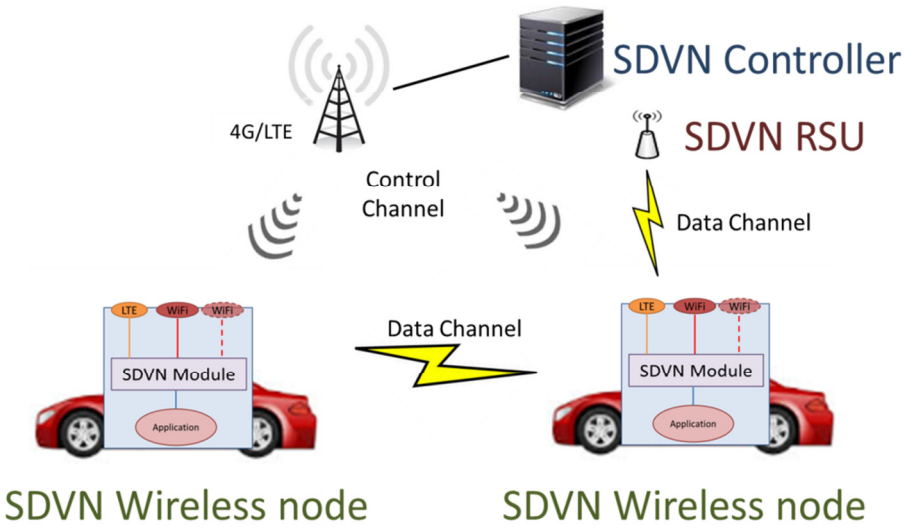


Figure 4.4 System Operations Overview

We use a modified OpenFlow protocol with some additional components added to make the design viable under Ad-hoc network environments. However, The basic operation is the same, each SDVN wireless node, when deciding how to handle traffic, will base the decision on flow rules inserted by the SDVN controller. If a packet arrives that is not matched in the flow table, an OpenFlow-like behavior is invoked, and the SDVN controller can insert new

flow rules in corresponding flow table. The SDVN controller maintains a path database to record path and flow information.

4.3.5 Control Modes

While the concept of SDN is the separation of control and data plane, there are differences in how SDVN can operate based on the degree of control of the SDVN controller. We classify our architecture into three operational modes:

- **Central Control Mode:** This is the mode where the SDVN controller controls all the actions of underlying SDVN wireless nodes and RSUs. In specific, all the actions that the SDVN data element performs are explicitly defined by the SDVN controller. As shown in Figure 4.5, the SDVN controller will push down all the flow rules on how to treat traffic.

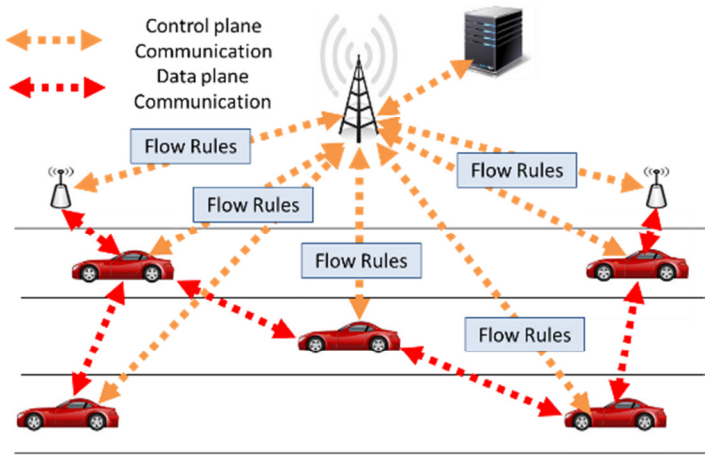


Figure 4.5 Central Control Mode

- **Distributed Control Mode:** This is a mode where underlying SDVN wireless nodes and RSUs do not operate under any guidance from the SDVN controller during data packet delivery. This control mode in essence is very similar to the original self-organizing distributed network without any SDVN features, except that the local agent on each SDVN wireless node controls the behavior of each

individual node (e.g., run GPSR routing), as shown in Figure 4.6.

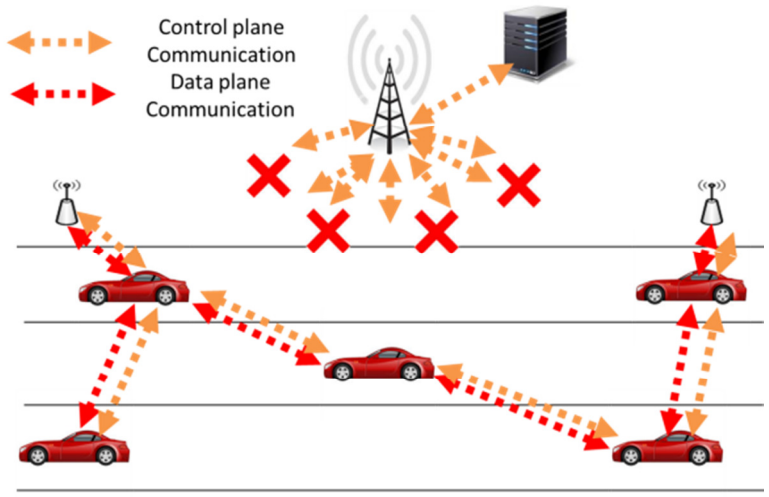


Figure 4.6 Distributed Control Mode

- **Hybrid Control Mode:** This mode includes all the operational modes of a system where the SDVN controller exerts control anywhere between full and zero. Figure 4.7 shows an example, where the SDVN controller does not hold complete control, but instead can delegate control of packet processing details to local agents. Therefore control traffic is exchanged between all SDVN elements. One example would be that instead of sending complete flow rules, the SDVN controller instead sends out policy rules which define general behavior, while the SDVN wireless nodes and SDN RSUs use local intelligence for packet forwarding and flow level processing. In specific, the SDVN controller instructs SDVN wireless nodes and RSUs to run a specific routing protocol with certain parameters.

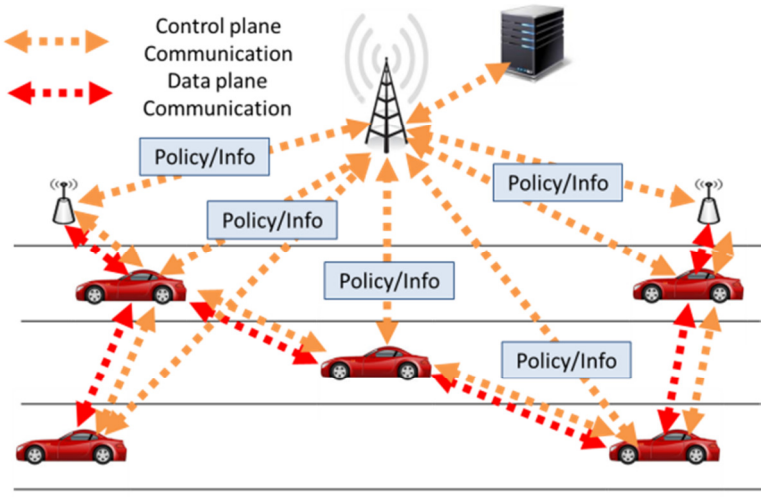


Figure 4.7 Hybrid Control Mode

The central control mode behaves similar to that of wired SDN architectures, where the SDN controller will insert all the rules. However, since the inherent problem of wireless channel is its reliability/availability, there are always potential communication losses between mobile nodes with the controller. This is the reason why SDVN must have failure recovery mechanisms that guarantee that the system can still function, even if at a reduced level, when SDVN controller communication is lost or disrupted. The local agent located on each SDVN wireless node has the intelligence to deal with such disruptions. For example, when communication to the SDVN controller is lost, the system can revert back to running a traditional routing protocol, such as GPSR. Later in the chapter, we demonstrate in simulation how this fallback mechanism can maintain good packet delivery even during SDVN controller disruption.

4.4 Multi-Frequency SDVN Architecture

In this section we describe an instance of the SDVN architecture where the SDVN controller selects frequencies for wireless node transmissions. In wired SDN/OpenFlow, the action field of a flow table rule is usually the output port. When moving SDN to MANET, the

output port becomes the specific wireless interface and the specific frequency within that interface. A wireless node typically has multiple wireless interfaces; moreover the channel frequency of each interface (say, WiFi or LTE) is software reconfigurable.

We propose two alternative architectures using SDVN-based frequency selection, based on the number of wireless interfaces for each wireless node, and on their capability.

4.4.1 Multiple Wireless Interface Wireless Nodes

Figure 4.8(a) shows the operation on a wireless node that has multiple wireless interfaces that can be used for data plane.

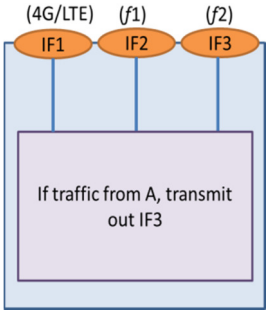
In this setup, wireless interface on nodes are preconfigured to specific frequencies. The SDVN controller does not directly control frequency, but instead chooses appropriate interface for data to transmit. The SDVN controller knows which radio is using which frequency, and chooses the appropriate one for each flow according to its policy.

4.4.2 Configurable Wireless Interface Wireless Node

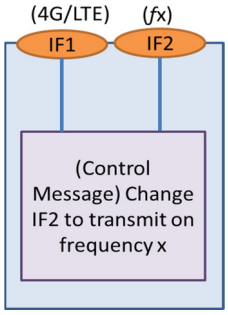
Figure 4.8(b) shows the operation on a wireless node where wireless interfaces can be configured to transmit on a different frequency.

In this setup, the SDVN controller can directly control the transmission frequency of a wireless interface. This flexibility is provided by a more advanced architecture which requires radio frequency to be part of the SDVN control message component, as shown in Figure 4.8(b). The radios themselves must also be able to accept external commands to change the frequency, such as cognitive radios [57][58].

If advanced cognitive radios are used as the wireless interface to transmit data, the SDVN controller then must gather information from the radios to coordinate spectrum management in order to make decisions.



(a) Multi-Interface Wireless Node



(b) Configurable Wireless Interface Wireless Node

Figure 4.8 Frequency Selection-based Architectures

4.5 SDVN Benefits and Services

As a separated control plane, SDN brings flexibility and programmability into the network. This brings awareness into the system so that it can adapt to changing conditions and requirements. In specific, this awareness allows SDVN to make better decisions based on the combined information from multiple sources, not just individual perception from each node. Also, dynamic and flexibility can react to sudden events, suitable for reacting to emergencies

and changing requirements. In this section, we describe the benefits of SDVNs, and describe several services that can be enhanced by utilizing these benefits.

4.5.1 SDVN Benefits

We classify benefits of a SDVN into three individual areas:

- **Path Selection:** The awareness of SDN allows the system to make more informed routing decisions. For example, in a VANET scenario, data traffic can become unbalanced, either because the shortest path routing results in traffic focusing on some selected nodes, or because the application is video dominant which occupy big bandwidth on the path. When this situation is discovered by the SDVN controller, it can start a reroute traffic process to improve network utility and reduce congestion.
- **Frequency/Channel selection:** When a SDVN wireless node has multiple available wireless interfaces or configurable radios such as cognitive radios [58], SDVN can allow better coordination of channel/frequency used. For example, the SDVN controller can dynamically decide at which time what type of traffic will use which radio interface/frequency. This can be used to reserve channels for emergency traffic for VANET emergency services.
- **Power selection**: Because of the awareness, SDVN will have the information to decide whether changing the power of wireless interfaces, and therefore its transmission range, is a logical choice. For example, the SDVN controller gathers neighbor information from SDVN wireless nodes and determines that node density is too sparse and commands all nodes to increase power to achieve more reasonable packet delivery and reduce interference.

4.5.2 SDVN Services

Based on the benefits that we described earlier, we present services that can be enhanced using SDVN.

- **SDVN Assisted VANET Safety Service:** Improving road safety through the use of V2V communications is one of the primary use cases of VANETs. We show how a Software-Defined VANET can improve the services when compared to traditional methods. SDVN can be used to reserve or limit specific frequencies so that emergency traffic (or otherwise privileged traffic, such as security) uses this reserved path. The difference between this and traditional emergency channels is that reservation in our architecture is configurable dynamically. The SDVN controller can assign flows to these channels or remove them based on current traffic conditions and application requirements. This can also be used to offer different level of services based on policies. The way this can be done is by changing rules during an emergency period. Emergency traffic gets priority over the remaining traffic.
- **SDVN On Demand VANET Surveillance Service:** Surveillance service for emergency/authority vehicles is another area in which SDVN can be deployed. In traditional architectures, a requester (e.g. police car) must send out a request for the surveillance data (or even a broadcast for the data if the holder of the data is unknown to the requester). In a SDN-based system, this request is done by the SDVN controller. The SDVN controller simply inserts flow rules for the surveillance data to reach the requesting nodes. Also, when there are several requests for the same surveillance data, such as when multiple police request for video surveillance feed, the SDVN controller inserts rules so that the same copy is sent to multiple destinations.

- **Wireless Network Virtualization Service:** Network virtualization services aims to provide abstract logical networks over shared physical network resources. SDN has already been used in data centers to provide network virtualization services, and we can apply the same idea for SDVNs. The idea is to let different flows choose different radios/interfaces using different frequencies. If the radio frequencies used by each individual network is different, individual network's traffic are isolated from each other and we have thus effectively sliced the networks and created virtual wireless networks. One method would be the grouping of wireless nodes and RSUs, so each RSU only forwards traffic from a selected group of wireless nodes. Another more advance method would be to incorporate time slicing. The control of which network uses which radio interface/frequency for which time period is done by the SDVN controller, which makes the allocation of network traffic a programmable fashion. Time slicing for efficient OFDM spectrum allocation used for LTE networks can be applied in the SDVN to support one virtual wireless network per time slot. If multiple radio interfaces are available, multiple virtual networks can be supported in the same time slot. For example, ITS traffic is exchanged on frequency channel f1; MPEG DASH video is transmitted on frequency channel f2. Note that while the video packet broadcast on channel f2 is picked up by all neighbors tuned on f2, the nodes that will receive and forward the video packet is determined by SDVN controller intelligence. Additionally, the SDVN controller can set filters on node inputs so that some nodes, say, may reject certain traffic classes. This could be used, for example, to restrict the propagation of video surveillance traffic to law enforcement vehicles. This input filtering is an SDVN feature unique of wireless networks where broadcast is used, and can be used in combination with VANET surveillance services.

4.6 SDVN Service and Features

In this section we describe multiple SDVN services and features. We describe the scenarios, configuration, and evaluation, including simulation setups and results. All evaluations are modeled using the NS-3 simulator [59]. The goal of the evaluations is to demonstrate the services enabled by using SDVN.

4.6.1 SDVN Routing vs Traditional Ad-hoc Routing

In this evaluation we compare SDVN routing with traditional Ad-hoc routing. Figure 4.9 shows the overview for SDVN routing operation.

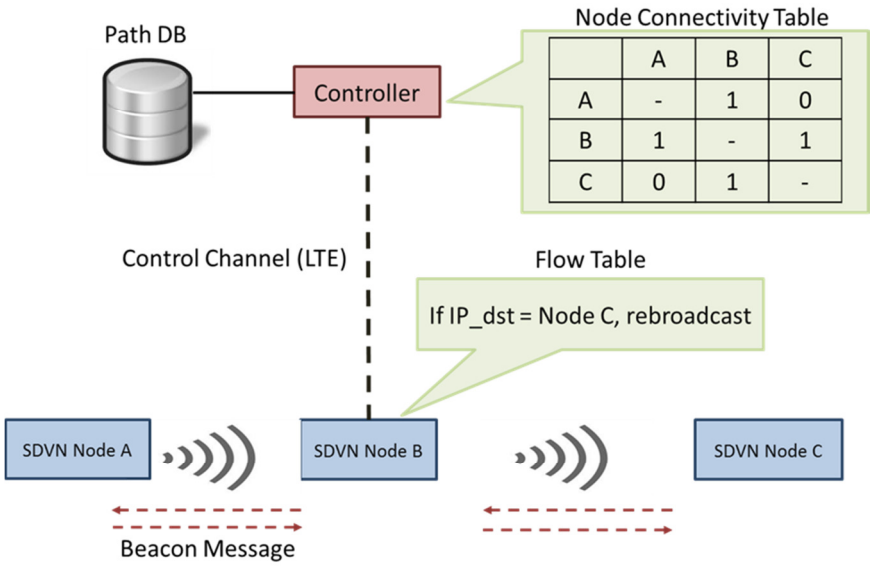


Figure 4.9 SDVN Routing Operation Overview

SDVN controller needs to learn the network topology to make intelligent decisions. For our SDVN routing we utilize beacon messages, a common feature in VANET systems. SDVN wireless nodes exchange beacon messages to learn information about immediate neighbors. This neighbor information is periodically updated to the SDVN controller, which uses this information to build a node connectivity graph. Exploiting this feature can provide a

lot of advantages for the management of mobility in a VANET scenario. We choose to use this beacon method over the Link Layer Discovery Protocol (LLDP) actually used in wired SDN systems. Using the information in the connectivity table, the SDVN controller will construct the path from sender to receiver, insert flow rules into the SDVN wireless nodes' flow tables, and keep established paths in a local path database for future references.

We simulate SDVN routing over a SUMO [60] generated road network shown in Figure 4.10, the road network is a grid type network that spans an area of 1000 x 1000 m, with each road segment = 200m. Node density varies is 50 nodes in the simulation. The SDVN controller LTE access is placed in the center of the simulation area where it is in wireless range of all SDN wireless nodes. Each SDVN wireless node has multiple wireless interfaces; short range using 802.11 with the propagation loss model to limit the transmission range to 250m, and long range using LTE. Each simulation run features a pair of random nodes in the topology running a NS3 echo client-server streaming session, with a packet generation rate of 4 packets/s and packet size of 1024 byte. Beacon message interval is 500ms. SDN wireless nodes will update neighbor information to the SDVN controller at intervals of 1s Simulation parameters were chosen based on MANET comparison studies [61]. Each set of simulations is averaged over 10 runs each running for 5 minutes. Table 4.1 lists all the parameters

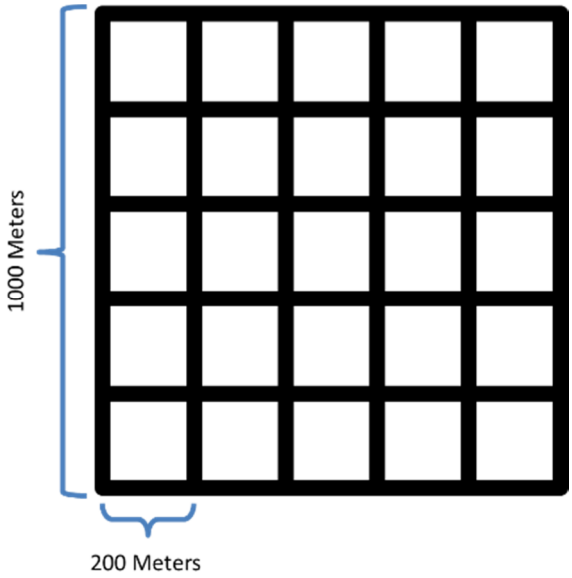


Figure 4.10 Grid Road Network

Table 4.1 SIMULATION PARAMETERS

Parameter	Value
Simulated Area	1000 m × 1000 m
Layout	Grid layout
Road Segment Length	200 meters
Road Structure	Two way two lanes
Speed Limit	5~20 m/sec
Node count	30~50 vehicles
Transmission Range	250m
Simulation Time	200 seconds
SDVN Beacon Transmission Interval	500 msec
SDVN Neighbor Update Transmission Interval	1 seconds
App traffic data rate	4 packets/sec
App traffic packet size	1024 Bytes

We first evaluate our system under different node density and mobility scenarios. Figure 4.11 shows the packet delivery ratio under different node speeds.

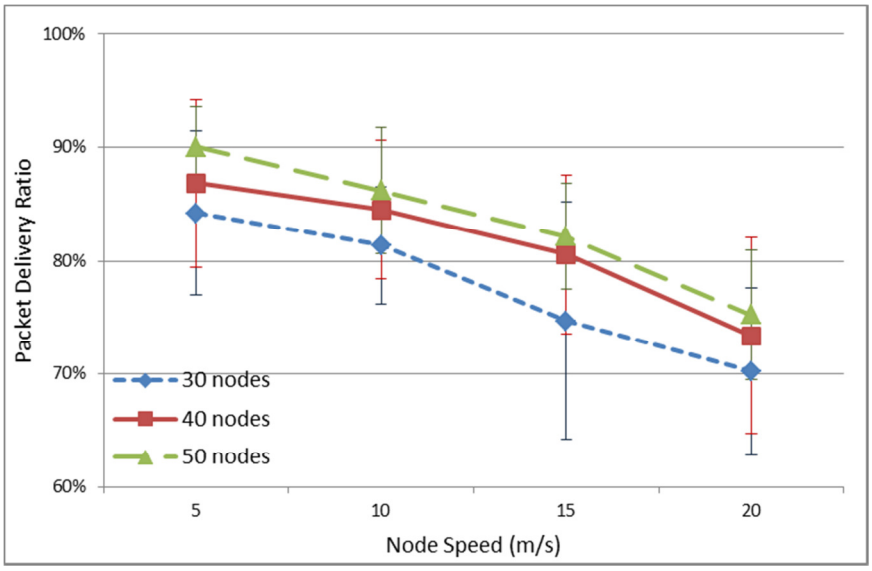


Figure 4.11 PDR under Different Node Speeds

We can see that packet delivery ratio drops from both the increase in node mobility and decrease in total node count. This is expected as routing will fail when there is no path between sender and receiver, and both of the factors will increase the chance of not finding a valid path.

Figure 4.12 shows the comparison of SDVN routing to other traditional MANET/VANET routing protocols, including GPSR, OLSR, AODV, and DSDV. We use this evaluation to demonstrate the feasibility of SDVN.

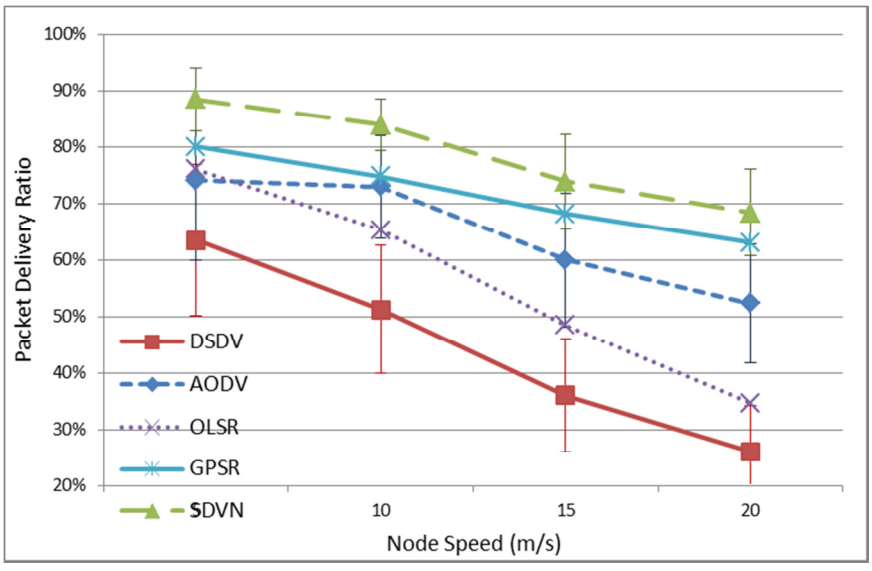
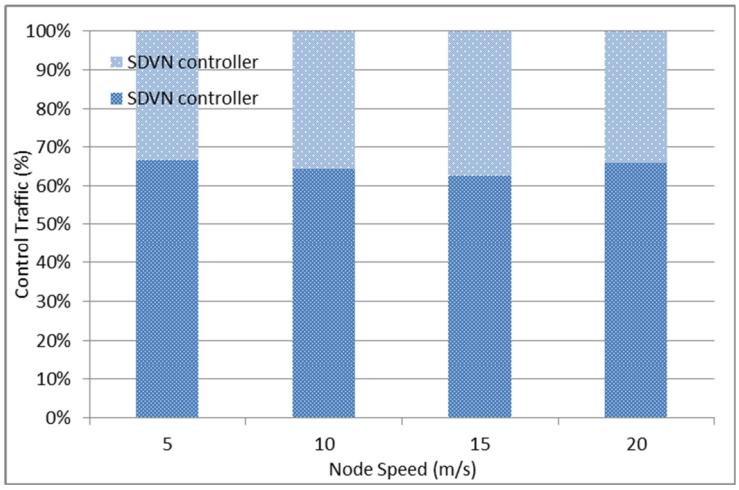


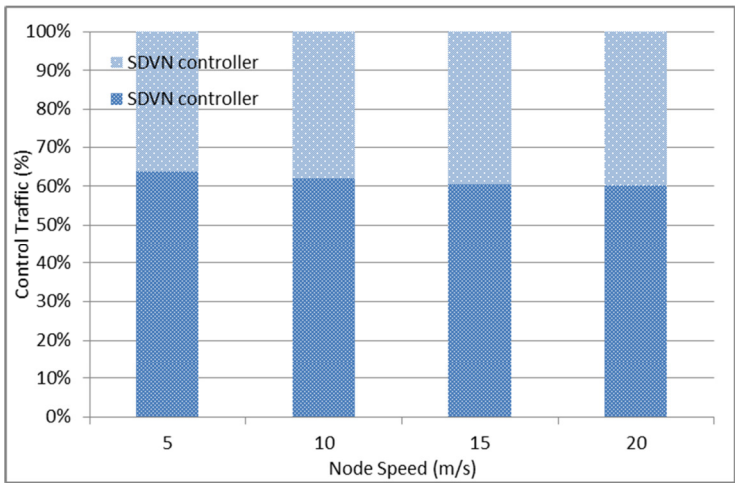
Figure 4.12 PDR comparison: SDVN vs Traditional Ad-hoc Routing

We can see that our SDVN routing outperforms the other traditional Ad-hoc routing protocols. The aggregated knowledge that the SDVN controller has is the major reason. As SDVN wireless nodes update the SDVN controller about neighbor information, the SDVN controller immediately detects that there is topology change and sends out control messages as needed. Therefore our SDVN system responds much faster to topology change.

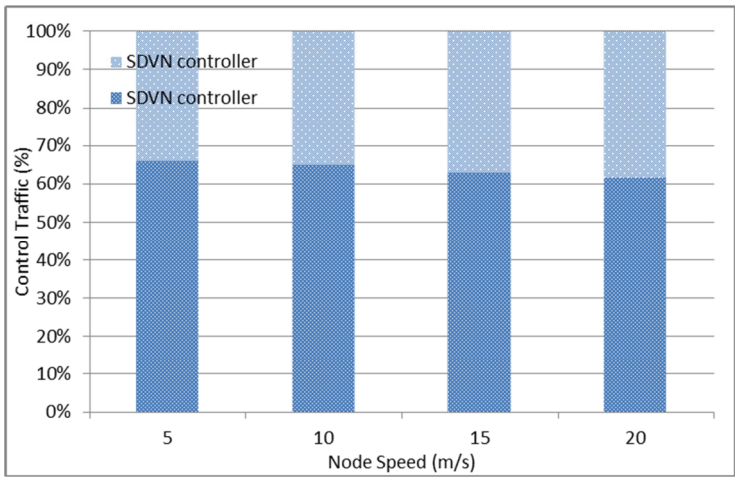
The fast response to topology change requires control messages between SDVN controller and SDVN wireless nodes. We evaluated the amount of overhead caused by this exchange. More precisely, we account for the SDVN wireless nodes sending neighbor information to the SDVN controller, and SDVN controller sending rules to SDVN wireless nodes. We measure the amount of control traffic from both and compare in Figure 4.13.



(a) 30 Nodes



(b) 40 Nodes



(c) 50 Nodes

Figure 4.13 Control traffic breakdown

Figure 4.14 shows the control traffic rate generated by the SDVN controller to modify flow rules. First, we can see that as node density increases, the SDVN controller needs to handle more SDVN wireless nodes and must send out more control messages. Also, we see that the control rate increases with speed because the SDVN controller must keep up with topology change. The exception is in the 30 node and 20 m/s case where the number of control sent by the SDVN controller decreases. This is happening because with only 30 nodes it is more likely that the path does not exist. No rules can be inserted, and the total control traffic actually decreases. Overall, since control traffic does not carry large payloads, at 30 packets/s the overhead traffic is manageable. Nevertheless, as with all centralized control systems, scalability is a concern that should be addressed, the amount of control message will only increase as more SDVN wireless nodes are under the SDVN controller's control. Also, topology change, which is common in VANETs, will also increase control messages. Methods to reduce these messages for scalability and freshness should therefore be investigated, such as delegated some functionality to local SDVN controllers on each individual node.

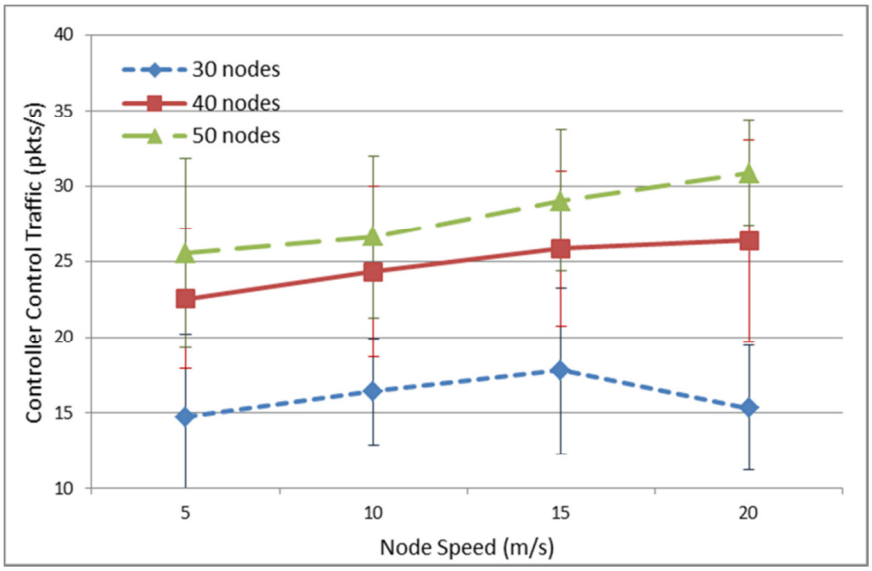


Figure 4.14 SDVN Controller Control Traffic

4.6.2 SDVN Failure Recovery

In this evaluation we demonstrate how fallback mechanisms utilized by SDVN wireless nodes can still provide good packet delivery even when communication to SDVN controller is lost.

To enable this feature, each SDVN wireless node now has a local SDVN agent, as shown in Figure 4.15. Traditional Ad-hoc routing protocols (e.g., GPSR, AODV, DSDV, and OLSR) are supported in agents as fallback mechanisms, to allow the SDVN network to revert back to Ad-hoc network operation even in the case where SDVN controller communication is unavailable.

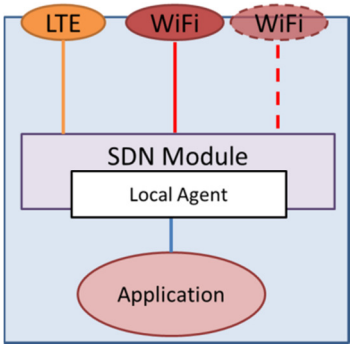


Figure 4.15 SDVN Wireless Node with Local Agent

One thing to note that the functionality of the agent depends on what features are enabled on the SDVN wireless node. The local SDVN agent can either be the backup controller when connection to the SDVN controller is lost or the primary SDVN intelligence while receiving input from the SDVN controller. In scenarios where the connection to a SDVN controller is stable and has full control, this SDVN agent has minimal intelligence.

Once again, the simulation is performed over the SUMO generated grid road network using the same experiment parameters. To demonstrate the need for fallback mechanisms, we first show a scenario where no fallback mechanism is used. Figure 4.16 shows this scenario where there is a controller failure for 100 seconds, as shown by the dash lines.

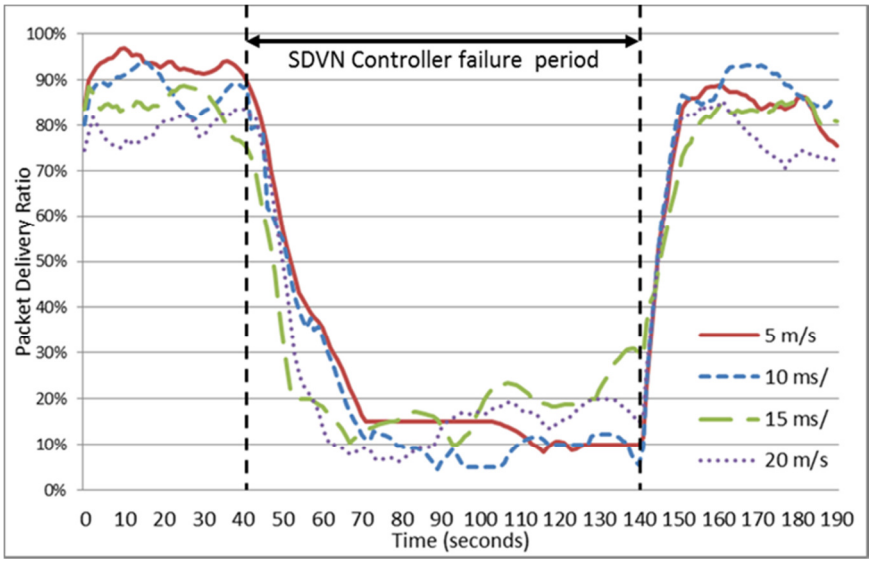


Figure 4.16 SDVN Controller Failure without Fallback Mechanism

We can see that Packet delivery ratio immediately starts to drop as SDVN controller no longer insert fresh rules for SDVN wireless nodes. This demonstrates that operating SDVN in central control mode is dangerous if SDVN controller communication is not reliable enough. The nature of a VANET is that nodes will move around quickly, and stale rules will become obsolete much more quickly compared to a scenario where node mobility is low.

We then show the same scenario, except that this time the fallback mechanism, running GPSR routing, is triggered when SDVN controller communication is lost. Figure 4.17 shows the packet delivery ratio of this scenario.

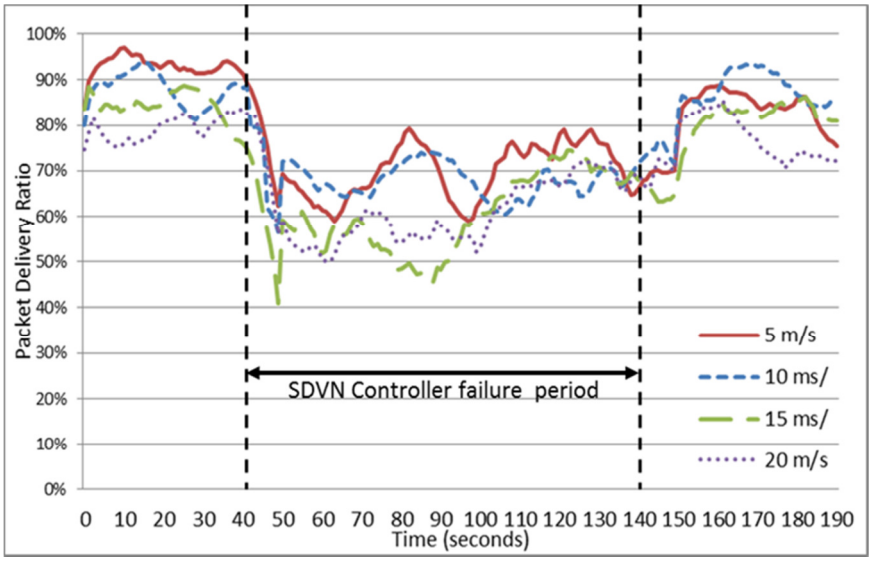


Figure 4.17 SDVN Controller Failure with GPSR as Fallback Mechanism

We can see that after an initial drop (after the loss of communication with SDVN controller and before GPSR is activated), good packet delivery ratio is restored. Once communication with SDVN controller is restored, the system once again reverts back to SDVN routing.

4.6.3 SDVN Transmission Power and Range

In this evaluation, we demonstrate how allowing the SDVN controller to dynamically control the transmission power of SDVN wireless nodes can improve packet delivery. The simulation again is performed over the SUMO generated grid road network; however the node density is 30 nodes. Figure 4.18 shows the result, with the dash line marking the time when the SDVN controller raises the transmission power so that transmission range is now approximately 400 meters (up from the previous 250 meters).

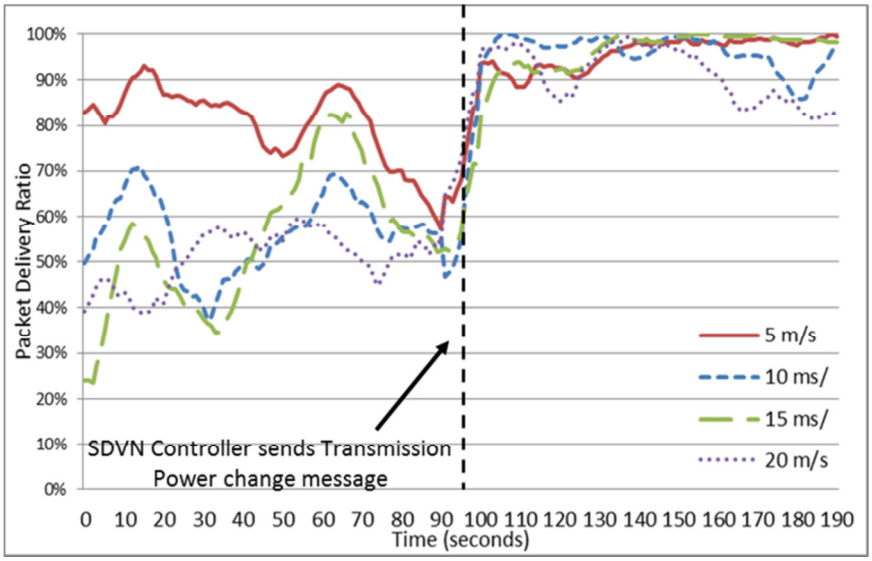


Figure 4.18 SDVN controlling Transmission Power and Range

We can see how immediately how packet delivery ratio becomes higher, as the new transmission range provides much better connectivity between nodes. The SDVN controller can make the judgment to adjust transmission power because it has global information on the entire VANET scenario. In specific, in our evaluation the SDVN controller increases transmission power because based on the information gathered from the SDVN wireless nodes, it determines that the connectivity between SDVN wireless nodes is too low.

4.6.4 SDVN Alternate Path Selection

In this evaluation, we demonstrate how using shortest path for transmission is not always the best option, and how our architecture can overcome this by allowing the SDVN controller to force traffic to use alternate paths. Figure 4.19 shows the scenario, there are multiple paths from source to destination, various levels of interference is on the shortest path, and SDVN can choose the alternate longer path with less interference.

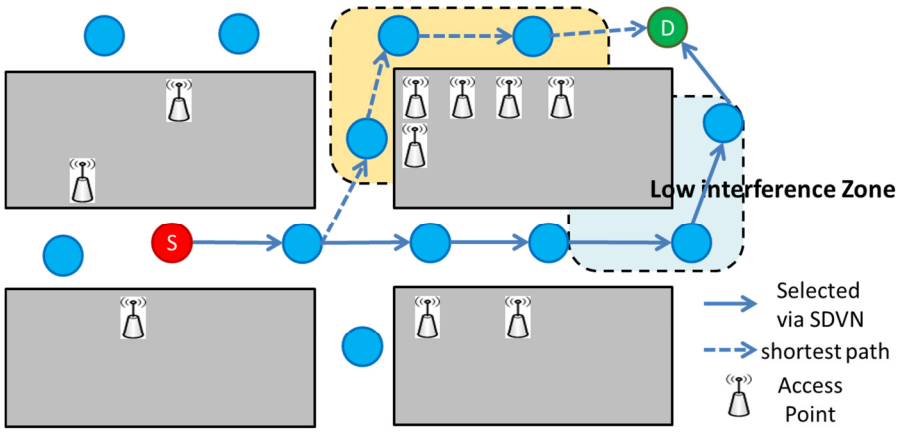


Figure 4.19 SDVN Alternate Path Selection Scenario

The SDVN can make the decision of using alternate paths based on information provided from underlying SDVN wireless nodes. In our specific example we estimate path quality by calculating neighbor beacon packet losses. Based on node position and past history, the SDVN controller can make an intelligent decision on whether a path should be considered to route traffic. Figure 4.20 shows the results where SDVN controller makes the decision of switching paths when there is 10% and 20% beacon losses.

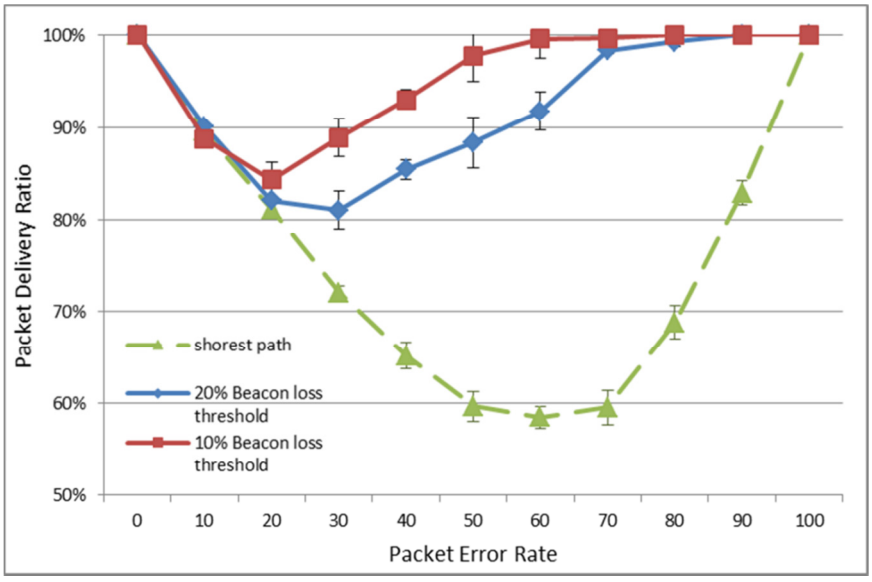


Figure 4.20 SDVN Alternate Path Selection Results

We can see that shortest path approaches performs poorly when interference causes packet drop. One thing to note that shortest path approaches has a lower bound threshold for packet drop, since interference not only affects application traffic, but also control traffic. When enough control packets are lost, the path is not considered to be valid anymore and the alternate path is naturally chosen.

Using beacon packet loss to estimate channel quality does not require any additional messaging from SDVN wireless node to SDVN controller, since all information can be calculated from periodic neighbor update. However, if SDVN wireless nodes are equipped with equipment that can measure channel quality (e.g. spectrum analyzer), this information can be easily incorporated into the SDVN wireless node-controller message exchange and be used to improve overall performance.

4.6.5 Multi-channel SDVN

We then demonstrate how allowing SDVN architectures perform in multi-interface multi-channel scenarios. The SDVN controller, based on measured channel conditions, is used to choose the better channel. For this evaluation, all traffic passes through the interference zone, as shown in Figure 4.21.

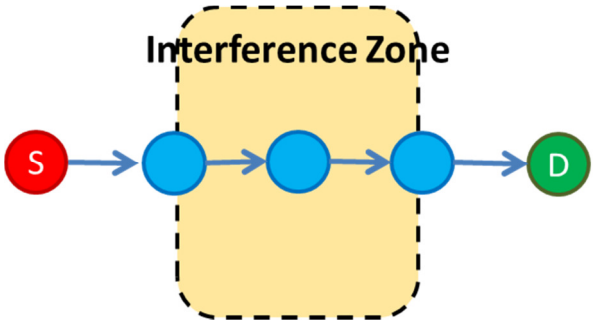
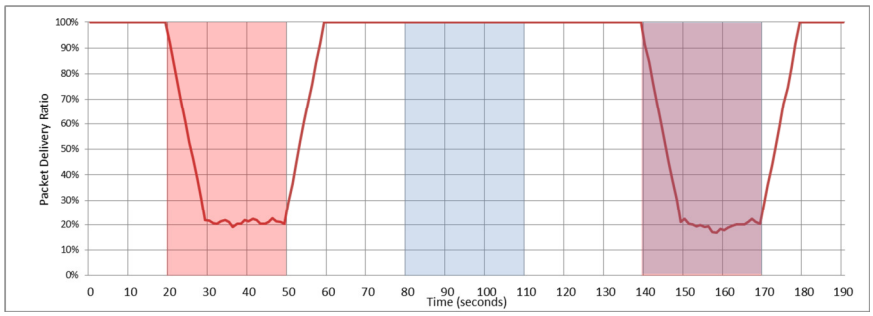
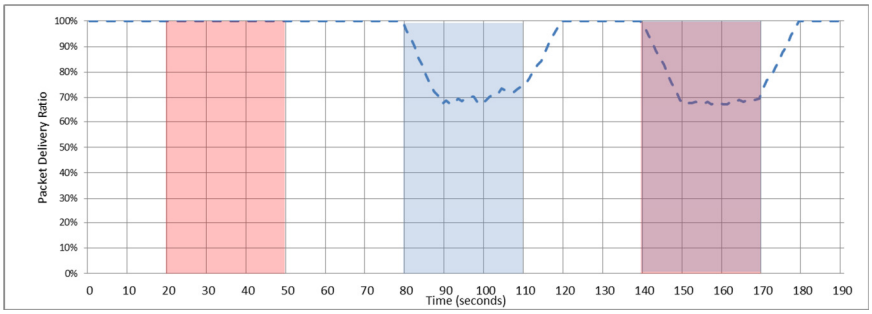


Figure 4.21 Multi-channel SDVN Scenario

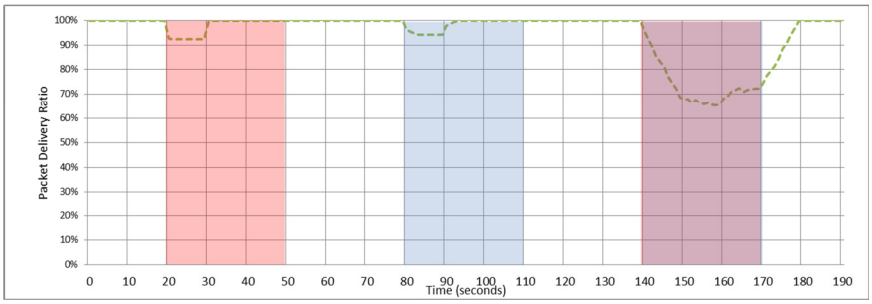
SDVN wireless nodes have two channels to transmit traffic, however not all channels experience interference for all time. We introduced three interference windows on the two channels. The first window is between 20~50 seconds and affects only channel1; The second windows is between 80~110 seconds and affects only channel2; The third window is between 140~170 seconds and affects both channels. The interference on channel1 is stronger. **Error! Reference source not found.** shows the results.



(a) Always transmit on channel 1



(b) Always transmit on channel 2



(c) Dynamic channel hop between channel 1 and 2

Figure 4.22 Multi-channel SDVN Results

We can see that packet delivery rate suffers if the system always chooses the same channel for data transmission. In our approach, the SDVN controller will instruct SDVN wireless nodes to use the channel that it calculates to be better. Once packet loss is recorded by the SDVN controller, it will hop the application traffic to use the better channel, resulting to only initial loss of packets. In the case where all channels are affected by interference, the SDVN controller chooses the channel witch it perceived to have less interference.

4.7 Summary

In this chapter, we introduced Software-Defined Networking into the Mobile Cloud, and proposed the architecture and services toward a Software-Defined Vehicle Network (SDVN). The architecture captures the components and requirements needed to deploy SDN in VANET, and we described several different operational modes and the services that can be provided. We demonstrate and evaluated several services: (I) SDVN routing and compared it with traditional MANET/VANET routing protocols, (II) how SDVN fallback mechanism is a key feature that must be provided to apply the SDN concept into mobile wireless scenarios, (III) transmission power adjustment as one of the possible services that can be provided by SDVN. (IV) Using SDVN to choose alternate paths when shortest path is not necessary the best option, and (V) Utilizing multi-channel in SDVN.

For future work, there are several potential directions. First, although we demonstrated how a fallback mechanism can maintain good packet delivery in the case of SDVN controller failure, there are more advanced scenarios that should be considered. For example, there are cases of partial SDVN controller connectivity loss, where only a subset of mobile nodes loses communication to the controller. In this case, the isolated nodes might need to form their own SDVN cluster, or nodes with connectivity to the SDVN controller can act as relay nodes to

relay rules. The best action to take will depend on several factors such as node density, mobility pattern, or others.

Second, although we demonstrated the feasibility of SDVN, our current architecture still requires infrastructure support (e.g. LTE). Therefore, there are possibilities of alternate SDVN architectures such as where SDVN controller transmits control traffic in P2P mode using WiFi channels. While this allows to build a wireless SDVN system that is completely distributed and thus does not need infrastructure support, the communication with the SDVN controller can be delayed and even interrupted causing new complications.

Chapter 5. Conclusion

Virtualization technologies are becoming mainstream and the importance of software agility is increasing. Programmable and flexible architectures will bring about changes in network architecture that will support the flexible use of network resources. It is a critical technical element of fully realizing the benefits of cloud computing.

In this thesis, we showed how the use of flexible network platforms can be used to bring new services and applications into the Mobile Cloud. First, we demonstrate that the Mobile Cloud can enable new services with virtualization and resource sharing. We show this by our case study, where we used software virtualization to build a parallel experiment platform that made protocol performance evaluation easier based on side by side comparison.

We then took the Mobile Cloud one step further with Software-Defined networking by adding network programmability, flexibility, control, and introduced Software-Defined Mobile Networks. We begin by showing how SDN can be used in mobile access, and then proposed Software-Defined Vehicle Network (SDVN) architectures. The architectures capture the components and strategies needed to complement the Mobile Cloud with SDN. We evaluated several services built upon our SDVN including (I) SDVN routing, where we showed that SDVN routing having good performance comparing with traditional MANET/VANET routing protocols, (II) Failure recovery, where we showed how SDVN fallback mechanism can be used to compliment SDN based wireless systems to address concerns on the centralized nature of SDN. (III) Transmission power adjustment as one of the possible services that can be provided by SDVN. (IV) Using SDVN to choose alternate paths when shortest path is not necessary the best option, and (V) Utilizing multi-channel in SDVN.

Software-Defined systems are providing us with new ways of controlling our networks. The programmability and flexibility provide by these systems allow the deployment of services and applications that was difficult or cumbersome to accomplish before. By combining the capabilities of the Mobile Cloud and Software-Defined Mobile Networks, we created the Software-Defined Mobile Cloud, which provide us with the tools to adapt to the ever-evolving world of data.

References

[1] C. Harsch, A. Festag, and P. Papadimitratos. “Secure position-based routing for VANETs.” In Proceedings of IEEE 66th vehicular technology conference (VTC-2007), Fall 2007 (pp. 26–30), September 2007

[2] N. Mckeown, “Software-defined networking.” INFOCOM keynote talk, Apr, 2009

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, 2008.

[4] OpenFlow Switch Specification, Version 1.4.0 (Wire Protocol 0x05). [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

[5] M. Gerla. “Vehicular Cloud Computing”, VCA 2012 Proceedings, Cyprus, June 2012.

[6] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, “OpenRoads: empowering research in mobile networks,” SIGCOMM Comput. Commun. Rev., vol. 40, no. 1, pp. 125–126, January 2010.

[7] J. Vestin, P. Dely, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo, “CloudMAC: towards software defined WLANs,” ACM SIGMOBILE Mobile Computing and Communications Review, vol. 16, no. 4, pp. 42–45, 2013.

[8] Wireless & Mobile Working Group (WMWG). [Online]. Available: https://www.opennetworking.org/images/stories/downloads/working-groups/charter-wireless-mobile.pdf

[9] P. Dely, A. Kassler, and N. Bayer. “Openflow for wireless mesh networks.” In Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN), pages 1{6. IEEE, 2011.

[10] P. Baskett, Y. Shang, W. Zeng, and B. Guttersohn. “SDNAN: Software-Defined Networking in Ad hoc Networks of Smartphones”, Consumer Communications and Networking Conference (CCNC), 2013 IEEE

[11] T. Luo, H. Tan, and T. Quek, “Sensor openflow: Enabling sofware-defined wireless sensor networks,” Communications Letters, IEEE, Vol. 16 , Issue: 11, 2012.

[12] M. Mendonca, K. Obraczka, and T. Turletti, “The Case for Software–Defined Networking in Heterogeneous Networked Environments”, ACM converence on CoNEXT student workshop, 2012.

[13] C. Guimarães, D. Corujo, R. L. Aguiar, F. Silva, and P. Rosa, "Empowering Software Defined Wireless Networks Through Media Independent Handover Management", Proc. 2013 IEEE Globecom, Atlanta, USA, Dec 2013.

[14] J. Broch, D. Maltz, D. Johnson, Y. Hu, J. Jetcheva, ”A performance comparison of multi-hop wireless ad hoc network routing protocols”, Mobicom 1998.

[15] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch and D. Vollmer, ”Position-aware Ad Hoc Wireless Networks for Inter-vehicle Communications”, Mobihoc 2001.

[16] DSRC, ”http://grouper.ieee.org/groups/scc32/dsrc/”.

[17] eSafty, ”http://europa.eu.int/information society/programmes/esafety”.

[18] CarTALK2000, ”http://www.cartalk2000.net”.

[19] Valery Naumov, Rainer Baumann and Thomas Gross, ”An Evaluation of Inter-Vehicle Ad Hoc Networks Based on Realistic Vehicular traces”, MobiHoc ’06, May 22-25, 2006.

[20] ORBIT: Open-Access Research Testbed for Next-Generation Wireless Networks, ”http://www.orbit-lab.org/”.

[21] Devashish Rastogi, Sachin Ganu, Yanyong Zhang, Wade Trappe, and Charles Graff, ”A comparative study of AODV and OLSR on the ORBIT testbed”, Milcom 2007.

[22] APE testbed ”http://apetestbed.sourceforge.net/”.

[23] R. S. Gray and D. Kotz, ”Outdoor experimental comparison of four ad hoc routing algorithms”, in Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems. ACM Press, 2004.

[24] A. Festag, H. Fler, H. Hartenstein, A. Sarma and R. Schmitz, ”Fleet-Net: Bringing carto-car communication into real world”, in Proc. 11th ITS World Congress, October 2004.

[25] M. Jerbi, S.-M. Senouci, M. Al Haj, ”Extensive experimental characterization of communications in vehicular ad hoc networks within different environments”, in Proc. IEEE VTC 2007.

[26] R. Mangharam, J.J. Meyers, R. Rajkumar, D.D. Stancil, J.S. Parikh, H. Krishnan and C. Kellum, ”A multi-hop mobile networking test-bed for telematics”, in Proc. SAE World Congress, 2005.

[27] MyCarEvent Project ”http://www.mycarevent.com/”.

[28] CVeT Testbed ”http://www.vehicularlab.org/testbed.php”.

[29] C. Pinart, P. Sanz, I. Lequerica, D. Garc´ıa, I. Barona, and D. S´anchez-Aparisi, ”DRIVE: a reconfigurable testbed for advanced vehicular services and communications”, in Proceedings of the 4th international Conference on Testbeds and Research infrastructures For the Development of Networks & Communities, March 18-20, 2008, Innsbruck, Austria .

[30] J. Haerri, F. Filali, and C. Bonnet, ”Performance Comparison of AODV and OLSR in VANETs Urban Environments under Realistic Mobility Patterns”, Proc. of Med-Hoc-Net 2006, 5th IFIP Mediterranean Ad-Hoc Networking Workshop, June 14-17, 2006, Lipari, Italy.

[31] J. Bernsen and D. Manivannan, ”Unicast routing protocols for vehicular adhoc networks: A critical comparison and classification”, Pervasive and Mobile Computing 5 (2009) 1-18.

[32] C. Lochert, B. Scheuermann and M. Mauve, ”Probabilistic Aggregation for Data Dissemination in VANETs”, VANET 2007:Proceedings of the 4th ACM International Workship on Vehicular Ad Hoc Networks.

[33] A. Bachir and A. Benslimane, ”A Multicast Protocol in Ad hoc Networks Inter-Vehicle Geocast”, in Proc. 58th IEEE Vehicular Technology Conference, Orlando, USA, October 2003.

[34] Y.C. Tseng, S.Y. Ni, Y.S. Chen, and J.P. Sheu, ”The broadcast storm problem in a mobile ad hoc network”, Wirel. Netw., vol. 8, no. 2/3, pp. 153-167, Mar.-May 2002.

[35] E. Giordano, R. Frank, G. Pau, and M. Gerla, ”Corner: a realistic urban propagation model for VANET”, in The Seventh International Conference on Wireless On-demand Network Systems and Services (WONS), 2010.

[36] K.C. Lee , S. Lee , R. Cheung , U. Lee and M. Gerla, ”First Experience with CarTorrent in a Real

Vehicular Ad Hoc Network Testbed", in Proc. VANET MOVE, May 2007.

[37] MIT ROOFNET "http://pdos.csail.mit.edu/roofnet/doku.php?id= roofnet".

[38] Berlin Roof Net (BRN) "http://sar.informatik.hu-berlin.de/research/projects/2005-BerlinRoofNet/berlin roof net.htm".

[39] Microsoft Research: Mesh Networking "http://research.microsoft.com/en-us/projects/mesh/".

[40] CMU wireless emulator "http://www.cs.cmu.edu/emulator/".

[41] RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing.

[42] RFC 3626: Optimized Link State Routing Protocol (OLSR).

[43] Xen "http://www.xen.org/".

[44] Gentoo "http://www.gentoo.org/".

[45] MadWifi project "http://madwifi-project.org/".

[46] AODV-UU "http://core.it.uu.se/core/index.php/AODV-UU".

[47] OLSRD "http://www.olsr.org/".

[48] Wireshark "http://www.wireshark.org/".

[49] GPSD "http://gpsd.berlios.de/".

[50] P. Apparao, S. Makineni, and D. Newell, "Characterization of network processing overheads in Xen", In Proceedings of the 2nd international Workshop on Virtualization Technology in Distributed Computing, November 17, 2006

[51] ONF Solution Brief, "OpenFlow-Enabled Mobile and Wireless Networks", September 2013

[52] Y. Yiakoumis, M. Bansal, S. Katti, and N. McKeown. "SDN for Dense Home Networks", Open Networking Summit, 2014.

[53] Flowvisor "http://onlab.us/flowvisor.html"

[54] Mininet "http://www.mininet.org/".

[55] F. Ongaro, A. Corradi, M. Gerla, E. Cerqueira, and L. Foschini, "Enhancing Quality of Service in Software-Defined Networks." Master Thesis, UNIBO, Bologna, Italy, July. 2014.

[56] R. Gomes, L.F. Bittencourt, E. Madeira, E. Cerqueira, and M. Gerla. "An Architecture for Dynamic Resource Adjustment in VSDNs based on Traffic Demand", IEEE Global Communications Conference (IEEE Globecom 2014), Austio, TX, December. 2014.

[57] P. Pawelczak, R. Venkatesha Prasad, L. Xia, and I. Niemegeers. "Cognitive radio emergency networks-requirements and design." In the First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN), pages 601{606. IEEE, 2005.

[58] I. Akyildiz, W. Lee, and K. Chowdhury. "Crahns: Cognitive radio ad hoc networks." Ad Hoc Networks, 7(5):810{836, 2009.

[59] NS-3, http://www.nsnam.org/

[60] Simulation of Urban Mobility (SUMO), http://sumo-sim.org/

[61] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols." In Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98), pages 85–97, Dallas, TX,

October 1998. ACM