Wireless Backhaul Software Defined Networks: Routing Strategy and Resource Allocation Implementation

Muhammad Anis[1], Muhammad Zubair[2]
engineeranis696@gmail.com[1], m.zubair.1@research.gla.ac.uk[2]

Abstract

The necessity for software defined networks (SDN) to boost the degree of programmability of the cellular wireless backhauls is dependent on greater complexity of these backhauls. One significant challenge with SDN networks relates to provisioning of a consistent resource consumption of network resources and is also called resource allocation. How to properly load balance network resources is the main concern in the context of a SDN wireless backhaul. In order to manage these network and IT resources, SDN wireless backhaul can combine distributed backpressure policies. These can also be used to manage computing resources by allocating the processing load brought on by OpenFlow (OF) switch requests among the available distributed SDN controllers. The routing of level granularity in the SDN application greatly influences the data plane performance, indicating a trade-off between control plane and data plane performance overfill. Additionally, it demonstrates how a distributed, dynamic backpressure strategy takes care of load across SDN controllers, outperforming static mapping strategies by a wide margin.

Introduction

- **Limitations of Current Networks**

The current networks have the following limitations [1];

- Management issues
- Control requirements i.e.
    - Greater scale
    - Migration of Virtual Memory System (VMS) [2]
    - Configuring huge networks.

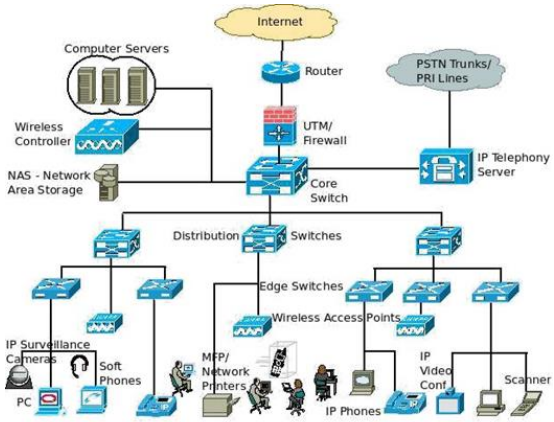

Figure 1.1: Current Network Architecture [3]
(Source: https://slideplayer.com/slide/16977834/)

Figure 1.1 shows the present or the Current Network Architecture being used. The networks had to be configured in a decentralized fashion, as indicated below, by utilizing numerous resources and efforts. Figure 1.2 illustrates the Current Networks Limitations whereas each machine has to be configured separately.

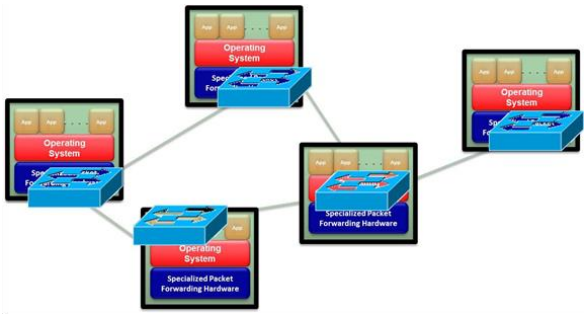

Figure 1.2: Current Networks Limitations
(Source: https://slideplayer.com/slide/16821370/)

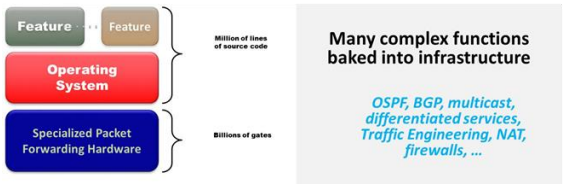

Figure 1.3: Complex Functionalities

Figure 1.3 depicts the Complex Functionalities baked into the infrastructure showing the operating system with specialized packets and the forwarding hardware. Similar issues would arise if we were unable to adapt flexibly to changing network conditions. As a result, software-defined networks were created (SDNs).

- **Software Defined Networking**

SDN is defined differently by several people. Most likely because SDN is changing as technology develops and new solutions are offered. SDN generally refers to the control of networks through software applications and SDN controllers as opposed to the conventional network administration consoles and commands, which might be laborious to maintain on a large scale and required a lot of administrative overhead. Since software-based control was considerably more flexible than the previous, rigid management consoles and command line interfaces, there was initially a tremendous lot of enthusiasm for SDN (CLI).

The awareness that many complicated information technology procedures that had to be handled via cumbersome management tools could now be automated and done considerably more efficiently was brought on by the capacity to operate networks through software swiftly. Emerging cloud and multitenant networks that demand more capacity and can't be slowed down by laborious administration operations have a strong preference for speed and automation. In reality, cloud automation (in all of its guises) quickly became a key use for SDN technology. Many SDN systems available today are actually hosting platforms for cloud automation solutions. There were more rigid views about how SDN architectures should be created and what constituted an SDN solution when SDN initially arrived on the technological scene. Customers now consider a wider range of SDN solutions when deciding which is best for them. Customers take into account what they're looking for in a policy-based automation solution rather than just the intricacies of the underlying SDN technology as the major use case for SDN has changed toward cloud automation.

Another completely open technology is SDN. This results in improved interoperability, increased innovation, and more adaptable, efficient solutions. A network may be managed by numerous SDN controller applications if it complies with the appropriate SDN standards.

This is preferable to having separate management consoles and commands for each network platform, which would promote vendor lock-in and complicate network administration. Multiple SDN standards are now being developed in various fields, and effective SDN strategies will always be built on open, interoperable multivendor ecosystems with essential open source technologies or standardized protocols. Along with the transition to SDN, a number of technological developments that have an impact on the architecture and design of contemporary data centers and enterprise networks must be taken into account. Most businesses are moving away from traditional client-server architectures and toward models where much more data is transmitted between servers within the data center (often referred to as east-west traffic). More advanced resource allocation policies and network scalability are needed for this.

Additionally, a lot of departments are very interested in switching to private, public, or hybrid cloud environments. Corporate departments now have access to public cloud services from organizations like Amazon, Microsoft, and Google, which show how flexible applications and services can be. The same service levels are increasingly expected from organizations own technical departments.

In fact, SDN is being considered as a significant factor in boosting IT agility and enhancing self-service services. Businesses are also spending money on big data applications to help them make better business decisions. Numerous servers must be used in enormous parallel to process these kinds of applications. The need for more capacity and automation is being driven by the requirement to manage enormous data quantities, which is putting more strain on the network. The need for more effective, flexible, and high-performing corporate network systems is influenced by all of these factors. SDN is designed to satisfy such requirements. An SDN strategy has overall advantages that benefit the entire enterprise. Because of the increased functionality of your infrastructure, you will have a competitive advantage. Because of the increased security, your company will operate more quickly, at a lower total cost of ownership, and with fewer hazards.

All the above can be summarized as;

- Defining SDN
  - It is System-layered
  - Programmable
  - Flexible
  - Extensible

- Future of SDN
  - It will Enable innovation
- There will be no need to design distributed control protocols
- It will have east to write, verify and maintain the interface for the programming.
- Network Operating System acts as control block

Basically, it is a concept to take care of the network that benefits from Open Flow protocols [5].

- Network administrators can manage network services through abstraction of lower level functionality.

- Decoupling the system which make decisions about where traffic is to be sent (the control plane) from the underlying systems which forward traffic to the selected destination (the data plane).

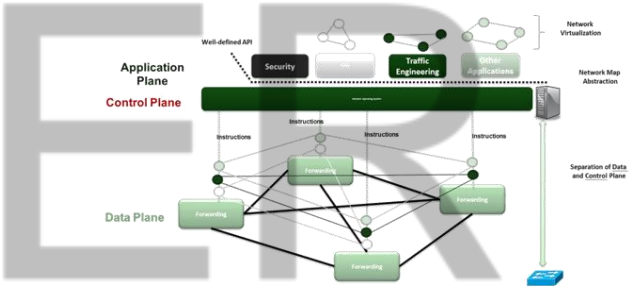

Figure 1.4: SDN with key Abstractions

Figure 1.4 above shows the software designed network (SDN) with key Abstractions showing application plane, control plane and the data plane.

SDN Basic Concept includes;

- Separate Control plane and Data plane entities.
- Running Control plane software on general purpose hardware.
- Programmable data planes.
- An innovative architecture controls not only networking devices but the entire network is managed.

Similarly, we have OpenFlow features which include;

- Enabling Innovation in Networks
- Interface between switches and Network OS
- Standard way to control flow-tables in commercial switches and routers just needing to update firmware.

Figure 1.5 below shows as to how OpenFlow software related to SDN with Management, Control and Data Plane with programmability, centralized intelligence and abstraction features allowed by OpenFlow.

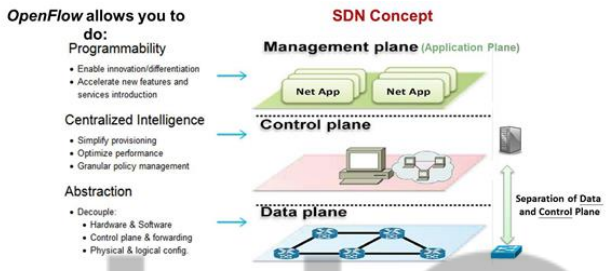

Figure 1.5: How OpenFlow is related to SDN

- **Backhaul in Networking**

Backhaul is frequently mentioned as being essential to telecom networks. Beginning at the cell site and ending in the mobile core is the backhaul network. The backhaul network is comprised of everything in between. The backhaul network can be further divided into the lower end, which is nearest to the cell site, and the upper end, which could be made up of aggregation sites [6].

Imagine backhaul network as a bridge between the cell site and the network's intelligence that can take care of different traffic types including data, and video. The links from base station to core network are called backhaul. In reality, it is backhauling traffic to the network from the base station.

Backhaul is traditionally viewed from a very transport-oriented perspective. You carry all of the signals from the base station or cell tower back to a location in the network where intelligence may be applied, such as service provisioning and quality of service (QoS) [7]. Applying that knowledge, you bring all of the traffic back to the cell tower. There are a lot of inefficiencies; you're not using the backhaul intelligently; you're just using it to transport all that traffic. Because you're merely utilizing a lot of equipment to carry traffic back and forth, that raises the cost of the network as a whole.

The degree of integration between radio and backhaul equipment is anticipated to be much higher in heterogeneous networks with substantial small cell deployments – perhaps with readily swappable backhaul interface modules enabling radio units to be upgraded to the desired backhaul choice for the site.

Only by using a well-defined and carefully chosen toolbox of backhaul technologies to manage both outdoor and indoor deployments on a wide scale will network scalability be possible.

The majority of the time, back haul networks are utilized to transport data from far POPs (Point of Presence) or concentration locations to a centralized location. The network's backhauling, which spans the entire area, has the highest capillarity. commonly employed with a three- or hub-and-spoke design, direct connectivity between locations, and no other possible traffic routes It calls for products with high durability, great deployment flexibility, and scalability in capacity.

In these circumstances, the equipment must support both full IP and hybrid configurations. Must be small in size and simple to install and activate. Even though split mount installations make up the great majority, full outdoor solutions are constantly expanding and introducing a new degree of freedom.
Let's now examine the wireless side. Wireless networks are undergoing an evolution. The demand for extra bandwidth

at cell sites will rise as this transformation takes place. Knowing these requirements is important in order to select appropriate technology and the network type for any application. Wireless service providers have extremely strict transport needs and standards for their services. Three main elements characterize the backhaul transport requirements for the wireless industry:

1) 2G to 5G standards [8]
2) Capacity requirement of the cell sites.
3) Performance in terms of latency, jitter and availability.

Operators may design a successful wireless backhaul business plan by selecting the appropriate technology, network, and architecture. Figure 1.6 below shows the Typical Generic Backhaul Network structure.

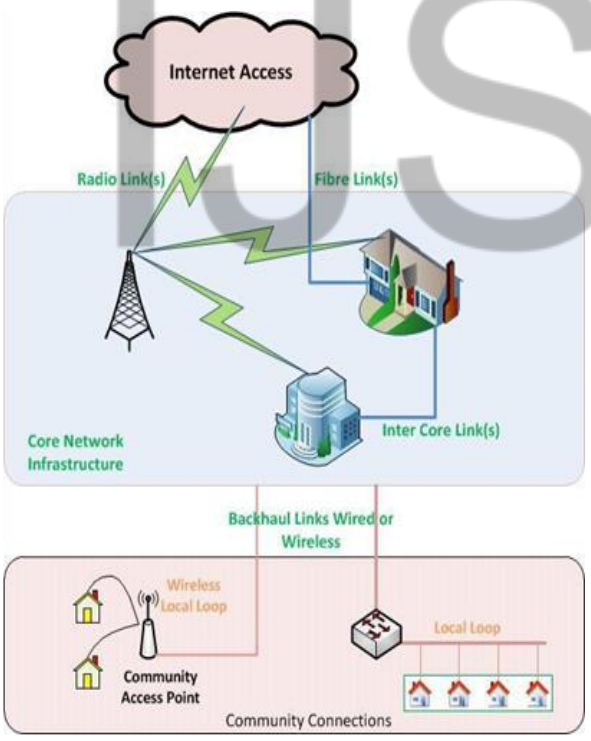

Figure 1.6: Typical Generic Backhaul Network

(Source: https://www.researchgate.net/figure/Backhaul-network-architecture-3_fig1_281091618)

- **Wireless Backhaul Present & Future**

Future 5G mobile networks are anticipated to further complicate the management of wireless backhaul resources. The following elements, among others, can be used to explain this tendency.

On the one hand, the anticipated higher density of small cells (SC), also known as low power base stations, which are a useful technique to boost network capacity due to smaller cell radii [9]. Conversely, increased network dynamicity brought on by user mobility, a decline in per-device reliability.

The concept of programmable SDN networks is to separate control plane from other network devices. It is hence transferred to the SDN controller, centralized software component in SDN. SDN applications [10] direct particular functions through controller, that communicates with network devices, using northbound interface (NBI). The data plane is composed of SCs that integrate backhaul network components, and it is configured using a public interface known as the southbound interface (SBI). The OpenFlow (OF) protocol has become the de facto protocol for this interface.

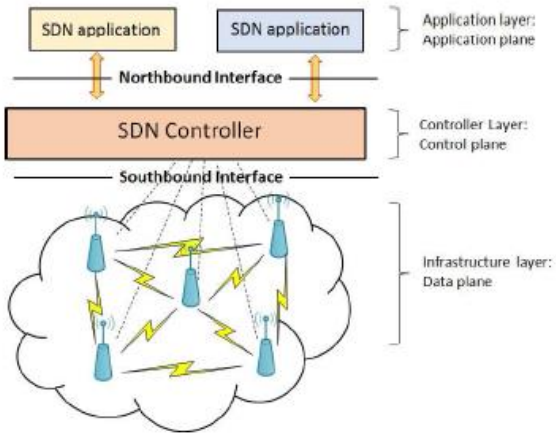

Figure 1.7: Software-defined wireless backhaul (centralized controller)

In Figure 1.7, the Software-defined wireless backhaul (centralized controller) is shown.

An important consideration when using SDN wireless backhaul is how to properly load balance network and IT resources.

Literature Review

- **Research Work**

Future network technologies such software defined networking, automatic management [13] and cloud networking have been the subject of ongoing research and development activities in recent years. ITU-T began working on Future networking standards late 2009, and it has created some first suggestions that lay out the fundamental guidelines for further, in-depth work (D. Matsubara, 2013).

Early attempts towards SDN, like Ethane (Casado et al. 2007), RCP (Caesar et al. 2005), and 4D (Greenberg et al. 2005), were a reaction to the networks' ossification. Researchers were becoming more aware of the need to simplify the functions built into network infrastructure.

A network with data and centralized control planes was introduced to a flow-based policy language by ethane. It proved that running a network under central management was feasible.

Some of the concepts that define modern SDN trace back before the word was even coined [TR10 2009]. At first, the Stanford OpenFlow project was referred to as SDN. The term has expanded to include any network architecture with the two characteristics listed below [Feamster et al. 2013]. Operations that implement judgments of how to handle packet data are separated from the decisions. Control and data plane separation is the term used to describe this characteristic. Using a clearly defined API, such as OpenFlow [14], the control and data planes communicate with one another.

Control plane enables the use of a variety of devices from a single point of view. The network's devices must expose their capabilities through a standard API in order for this property to be realized [15]. The public switched telephone networks (PSTNs) that implement SS7 protocol are distinguished by their separation of voice and the signaling necessary to provide call services [16].

PSTNs, on the other hand, are circuit switched networks. Through subscriber lines, telephones are connected to signaling switching points [17] that are linked together by voice trunks. Separate signaling links are used to carry the signaling necessary for message exchange between SSPs and other telephone network components. SDN incorporates concepts like flow-based routing into packet switched networks.

Active networking concentrated on enhancing the network's data plane performance. New data plane capability had to be deployed on network devices by code sent in data packets under several active networking configurations, most notably the capsule model. In initiatives like PlanetLab [Chun et al. 2003], traffic is divided into various execution environments based on packet size. SDN efforts were distinct from active networking since they concentrated on issues that network administrators faced right away. Additionally, the control plane's increased programmability has received a lot more focus.

These initiatives, together with business triumphs like Nicira's Network Virtualization Platform [Nicira 2012], have caused the industry to pay substantial attention to SDN. There are numerous difficulties in applying these ideas to wireless networks.

This is demonstrated by commercial products from a number of firms. According to Yap et al. [2010], the end user won't have

to worry about the specifics of whatever wireless network they are using to access services in the future. It presents an appealing scenario in which a user freely switches between WiFi and cellular networks while benefiting from flawless handovers.

There have primarily been two initiatives to use SDN principles to enhance cellular networks. The goal of SoftRAN [Gudipati et al. 2013] is to enhance the Radio Access Network's design (RAN). The RAN is the section of the cellular network architecture tasked with giving mobile devices wide-area connection. Due to the limited availability of spectrum resources, cellular carriers are keen to limit spectrum utilisation while preserving customer connectivity. To do this, SoftRan proposes better radio resource management [18].

I have determined the importance of the study with clearly defined objectives, which are discussed in the preceding sections, taking into account the literature review mentioned above.

Significance of the Study

In my research, I want to look into how backpressure policies can be used to control networking resources in SDN wireless backhaul. In order to balance network resources in wireless backhaul, centralized backpressure policy must be created. Additionally, this will involve examining the routing choices made by the SDN application and how much they influence the data plane performance. Similar to the static mapping policies, the research will focus on developing a distributed backpressure policy.

Research Objectives

Following objectives will be achieved through this research work.

- Proposing an effective policy routing framework for SDNs
- Balancing network resources (wireless backhaul), centralized backpressure policy will be used.
- Effect The impact of routing choices on control plane overload and data plane performance in the context of SDN applications.
- Building a simple platform of SDN which may be composed of one SDN controller managing disjoint sets of ten OF switches.

Research Methodology

In order to control network resources, my research will analyze varied scenarios/cases where backpressure policies might be incorporated into SDN wireless backhaul. The required analysis will be performed using NS-3/MATLAB.

- The first study analysis proposes an SDN routing policy architecture based on a centralized backpressure policy. The performance of the data plane in the proposed routing policy framework will be strongly impacted by how finely-grained the routing decisions are made in the SDN application and coming up with balance between performance data plane and control plane overload. The resource allocation strategy between network slices is then developed for backhaul/core networks.

- The second research analysis will suggest backpressure distributed policy to communicate with the computing resources of management by distributing available distributed SDN controllers' processing load generated through OpenFlow (OF) switch requests.

- The final solution will demonstrate how a distributed, dynamic backpressure method outperforms static mapping strategies by a significant margin and balances the processing load across several SDN controllers.

- In this thesis, we also discuss implementation methods for such applications in actual network settings

Simulation Tool

MININET simulation tools will be used for analysis.

- **MININET Overview**

A virtual network of hosts, switches, controllers, and links is created by the network emulator known as Mininet.

Standard Linux network software is used by mininet hosts, and its switches support OpenFlow for Software-Defined Networking and very flexible custom routing.

- **How it Works**

It runs numerous hosts and switches on a single OS kernel, Mininet leverages process-based virtualization. Mininet does not require the additional capabilities that the full Linux container design adds to allow full OS-level virtualization, such as process and user namespaces, CPU and memory restrictions.

- **Advantages**

- The greatest elements of emulators, hardware testbeds, and simulators are combined in Mininet.
- Boots in seconds as opposed to minutes.
- Scales better than single digits: hundreds of hosts and switches.
- Offers higher bandwidth and preconfigured virtual machine (VM) running on different operating systems.
- Offers performances that are interactive.

Figure 2.2 below shows a Snapshot of a Generic MININET Simulator.

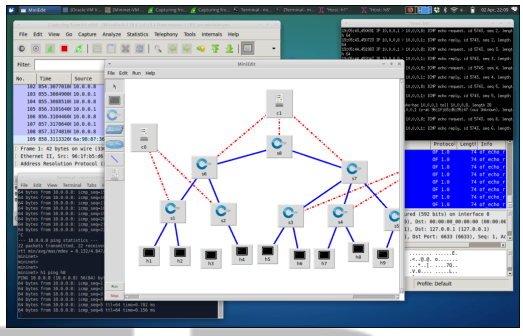

Figure 2.2: Snapshot of a Generic MININET Simulator

Developing a Conceptual Model

The essential concepts and ideas behind the backpressure algorithm used in legacy dispersed wireless networks will be described in this chapter. Based on this, a thorough analysis of the study problems/scenarios the researchers had to deal with will be done in order to implement the backpressure policy.

- **Backpressure Concept**

The study article by Tassiulas and Ephremides served as the foundation for the development of the backpressure concept. The strategy sought to reduce the network's total queue backlogs. It is accomplished by sending packets between network devices at each time slot to reduce queue backlog differentials. Later, Neely and Modiano [21] expanded the backpressure method by suggesting combining the original plan to

enhance the performance of wireless multi-hop networks. Researchers' interest in real-world applications for wireless multi-hop backhauls has lately increased due to the theoretical concepts generated by this study work.

- **SDN Paradigm for applying backpressure**

Following are some of the SDN paradigms where the backpressure can be applied.

Traffic Load Balancing

The development and implementation of more complicated network applications, like those that reduce power usage, is made easier by the SDN. However, in highly dynamic networks built on the SDN paradigm, effective load balancing solutions are required. The majority of the analysis for load-balancing SDN applications is based on the Equal Cost Multipath protocol.

The researcher in this study has put up a centralized approach to resource allocation employing shortest-path routing that ignores the path redundancy prevalent in backhaul networks.

Balancing load across multiple SDN Controllers

SDN controller is justified because of the following reasons;

- Administrative issues
- Scalability
- Fault Tolerance
- Controller latency reduction.

Distributed system implementation, however, presents a number of well-known difficulties. As a result, methods and algorithms are needed for a widespread deployment of SDN controllers in order to dynamically allocate switches to controllers. Switches (and the load they generate) can be distributed evenly among the SDN controller pool's resources in this fashion. 3.2 Load Balancing of Traffic Flows

Now, as shown below, I'll talk about how backpressure is used as a centralized SDN application to efficiently balance traffic flows across numerous OpenFlow switches integrated into small cell (SC) devices. Figure 3.1 shows the Software-defined wireless backhaul with SDN controller and applications.

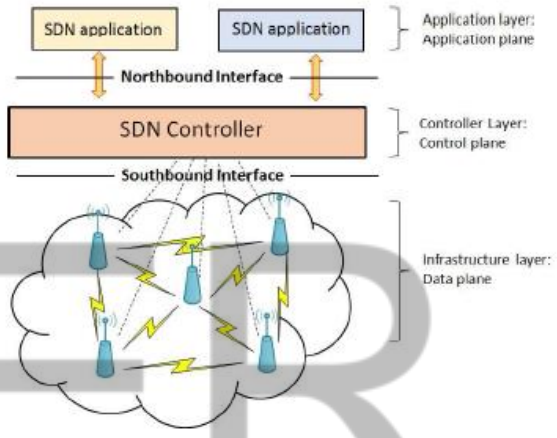

Figure 3.1: Software-defined wireless backhaul

SDN controller has two basic modules i.e.

(1) Link Discovery
(2) Topology manager.

Nodes are located via the link discovery module, which also keeps track of the network's physical links' current conditions. So, when an OpenFlow switch gets a packet, it checks its flow table for a matching forwarding rule. If there is a match the packet is transmitted. The If there is no match, the packet is encased in an OF request and forwarded to the SDN controller, who is responsible for issuing the rule to send this packet.

The SDN controller's built-in topology manager utilizes an algorithm to calculate the shortest path between two network objects that is akin to how routes are

calculated. However, depending on the levels of network congestion, this approach does not guarantee efficient use of the network resources. Instead the researcher came up with an improved routing by using of SDN application which applies backpressure policy to load balance and provides routing for traffic moving across a SDN wireless backhaul.

A database with information on the network topology and link status is kept up to date by the link discovery and topology manager modules. The information in this database was created by employing a discovery technique (using the Link Layer Discovery Protocol). This information technique enables the SDN application to retrieve the various pathways between any two network devices encapsulating a protocol. The SDN application additionally needs to poll the queue sizes of the ports in the OF switches housed in the SCs on a regular basis in order to execute a backpressure technique.

3.3 Research Analysis

Researcher suggest creating a model using Mininet to simulate the behavior of SDN networks based on our theoretical review. We'll deploy the SDN controller after using the Mininet in a virtual machine environment. Scalability will be examined. For the objectives of study, a framework will be developed to build and simulate SDN networks.

Simulation & Data Analysis

- **Model development**

Researcher looked at the SDN's behavioral traits using the network modelling tool Mininet. With the help of the well-known open source network emulator Mininet, virtual hosts, switches, and SDN controllers can be built. Before explaining how to instantiate SDN capabilities, Mininet must first be installed in a virtual machine (VM). Figure 4.1 shows the network architecture.

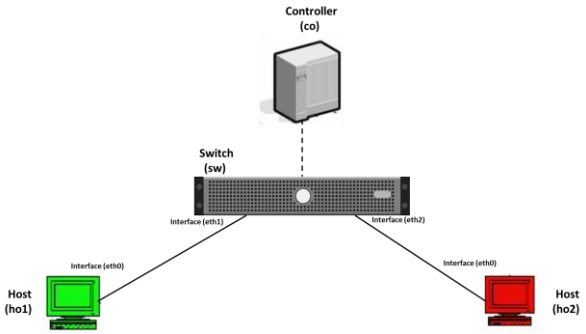

Figure 4.1: Network Architecture

It comprises of a controller, a switch and two hosts. The SDN architecture is further elaborated in Figure 4.2 below;

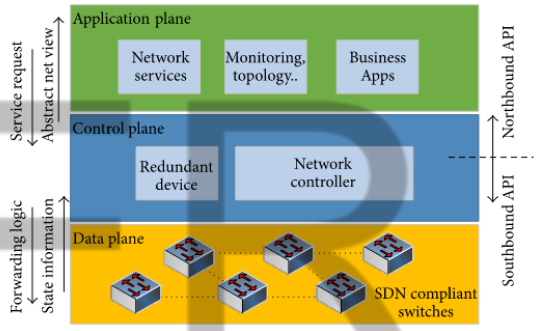

Figure 4.2: SDN Architecture Application, Control and Data Planes

The SDN controller Schematic is shown below;

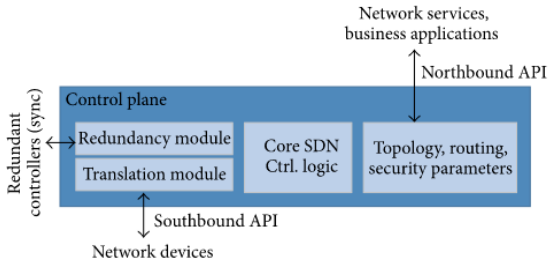

Figure 4.3: SDN controller Schematic

(Source: https://www.researchgate.net/figure/An-Overview-of-SDN-architecture-with-its-main-planes-data-control-and-application-plane_fig1_323563300)

Figure 4.3 shows the SDN controller Schematic. The switch capabilities are summarized in the below Table 4.1;

Platform Configurations	
Forwarding Capacity [1]	100G
1/10G SFP+	32
100G QSFP28	-
Stacking Ports	-
[1] Full line-rate forwarding on all ports	
Traffic Classification & Management	
IPv4 and IPv6 support	Yes
Packet Buffer Memory	6GB
OpenFlow Version	1.3+
Active Flow Entries	One million+
Flow Tables	10+
Flow Modification Rate	15,000+ mods/sec
Meter Table	Two-Rate, Three-Color - RFC 4115
Quality of Service (QoS)	8 queues per port WFQ and strict priority scheduling Metering on physical ports or logical interfaces
Traffic Shaping	On egress of physical ports or logical interfaces
Traffic Statistics	Packet and byte counters: Per flow, per physical port, per logical interface
Virtualization	
Virtual Forwarding Contexts (VFC)	Up to 256
Internal Control Plane	
CPU	Intel® Core™ Processor
Memory	16GB DDR3, 120GB HD
Management Interface	1 x 10/100/1000Base-T RJ-45 1 x Serial Console RJ-45 - 1 x USB 3.0
Software	Full Linux Server based
OpenFlow Control Module	Open vSwitch 2.3.1+
Configuration Management Protocol	REST API
Guest Virtual Machine	1 core, 2GB RAM, 5GB disk space
Management	
Configuration Interfaces	Enhanced CLI, REST API
Monitoring	SNMPv2c, sFlow, syslog
Physical	
Chassis Rack Height	1 RU
Typical Power	375W (DP2100), 450W (DP2200/DP2400)
Power Supplies	2 x AC or 2 x DC Redundant
Ventilation	Front-to-back or back-to-front
Regulatory Compliance	FCC, CE, NRTL, VCCI, BSMI, RoHS, WEEE, IEC, CSA, RCM

Table 4.1: SDN Switch

- **Simulation Setup**

We experimented with Mininet using the command;

$ git clone git://github.com/mininet/mininet

The basic Mininet install can then be completed using the command;

$ mininet/util/install.sh

A simple default virtual network consisting of two hosts, represented as h1 and h2, a switch, s1 and a controller, c0 that can be created using the command;

$ sudo mn

Mininet instance can be created in a VM. Once the installation is completed, the ping command can be applied to verify that connectivity exists within hosts.

$ sudo mn --test pingall

- **Creating Virtual Machine using MININET.**

Below text was generated through the tool:

```
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's1' )

        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( rightHost, leftSwitch )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Figure 4.4: Text File Creation for Virtual Network

Components are then added using the script:
leftHost = self.addHost('h1')
leftSwitch = self.addSwitch('s1')
And connected to each other by a link using the script:
self.addLink(leftHost, leftSwitch)

This text file is then saved and runs using the command:

$ sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo my topo --test pingall

In order to achieve scalability, the researcher modified Mininet network in order to create 64 hosts and 9 switches.

Hosts	Switches	Links	Ping Test (in Seconds)
2	1	2	5.133
10	3	30	18.326
64	9	72	86.347

Table 4.2: Performance (Ping Test)

Table 4.2 shows the Performance (Ping Test) result. As can be seen from the above table, as time to ping resources increases, network grows larger.

Results Analysis

Ping response times between the switch and controller are significant as enabled switch needs to communicate with a centralized SDN controller (the controller holds the routing tables, hence, whenever a packet with a new IP address arrives, the switch must ask the controller how to proceed). The system should have 2 CPUs and about 2 GB of RAM. Devices will stay linked even after the VM restarts, hence we can update the "network interfaces" file. The host-only network will receive an IP address from the server. Researcher installed and configured runtime environment using following commands;

```
$ sudo apt-get update
$ sudo apt-get install default-jre-headless
$ cd distribution-karaf-0.4.0-Beryllium
$ ./bin/karaf
$ opendaylight-user@root> feature:list
```

By accessing the User Interface under default port in a browser on the host system, one can view the User Interface, which is active on the VM and so has the same IP address as this VM. The default settings for the username and password are admin. The GUI will display the topology of the Mininet network and provide detailed information about the nodes and host connections. Researcher set up a network topology on a Mininet VM using 4 switches in a linear topology, each connected to a single host, and the controller VM with an IP address on the port. The following Mininet command was used:

```
$ sudo mn --topo linear,4 --mac --
controller=remote,ip=192.168.50.1,port=6033 --switch
ovs,protocols=OpenFlow13
```

We can then test if the network is working by pinging all nodes and verifying that every host is reachable.

Modeling the behavior of SDN network configurations is becoming more and more prevalent within cloud computing settings and related applications. We talked about installing Mininet and the controller on virtual machines, offered a list of frequently used commands, and covered how to record network message exchanges. Below is a bellman-Ford algorithm comparison of throughput and delay. Figure 4.5 shows the Throughput and Delay Comparison.

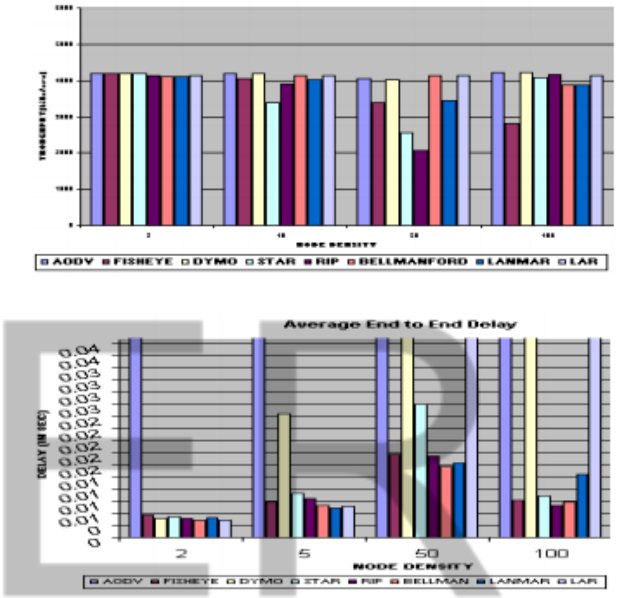

Figure 4.5: Throughput and Delay Comparison

Bellman Ford algorithm

A weighted digraph's shortest paths from a single source vertex to each of the other vertices are calculated using the Bellman-Ford algorithm. The Bellman-Ford-Moore algorithm is another name for the 1957 algorithm that Edward F. Moore and Bellman both published.

In a weighted graph, the Bellman Ford algorithm aids in determining the shortest route between any two vertices. It is capable of handling graphs with edges that can have negative weights. The Bellman Ford algorithm functions by overestimating the distance between the beginning vertex and each subsequent vertex. Then, by uncovering

fresh paths that are shorter than the outdated, overstated paths, it iteratively reduces those estimates. Figure 4.6 shows the Bellman Ford algorithm implementations.

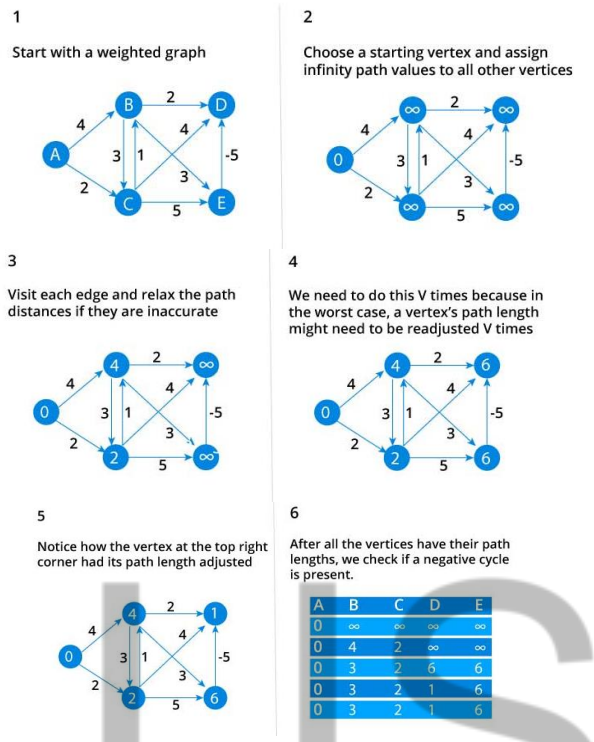

Figure 4.6: Bellman Ford algorithm implementation

Resource allocation and routing in SDN

A new breed of compute-intensive operations-using control plane architecture is being developed thanks to the processing capacity of SDN controllers. One central controller was initially intended to be used in SDN designs. However, SDN control plane needs to be dispersed in big network environments for obvious scalability and resilience reasons. As a result, despite being logically centralized, the control plane may actually consist of a number of controllers, each in charge of a specific SDN network domain and working together in a flat or hierarchical design.

A client usually correlates to flows transiting in the network. Each controller in distributed SDN architectures has complete knowledge of its own domain. Additionally, it has the ability to communicate with a central, upper-layer controller entity as well as with nearby peer controllers. However, the cost of communication latency and overhead is incurred during controller swaps.

Responsiveness is a second essential characteristic for any distributed algorithm used in SDN. In reality, sudden changes, such as variations in flow size, flow arrival/departure, and link/node congestion, can have an impact on the network's health. When a change in the system occurs in this scenario, convergence for the prior network state might not even be obtained. For this reason, having quick access to a high-quality solution is frequently preferable to a solution that is provably asymptotically optimal but has a low convergence rate. Therefore, it is essential that the resource allocation determined by a distributed algorithm is practical and, consequently, implementable at any iteration. Design of network resource allocation included:

- Traffic classification
- Queue mechanism
- Status collection
- Route calculation

Conclusion

- **Summary**

In my extensive investigation, I have found two instances that are pertinent for SDN wireless backhaul and where backpressure-based regulations are used, namely

- Balanced traffic flow loads
- Distribution of the processing load among the SDN controllers that are available.

Regarding the first SDN application that was developed, which was based on a centralized backpressure policy, an analysis is presented

that provides recommendations on how to implement such an application in a real-world OpenFlow environment. The simulation's findings demonstrate that a centralized backpressure technique along with suitable periodic route recalculations can greatly enhance data plane performance.

Regarding the latter, a distributed backpressure solution is suggested/provided along with some preliminary simulation results in order to balance the processing load among a physically distributed SDN controller architecture. A few pointers on how to put this dispersed broadcast of load information amongst nearby SDN controllers into practice are also given.

The following goals were attained as a result of this research endeavor.

- offering a centralized backpressure policy-based efficient architecture for policy routing for SDNs to balance the network's resources
- SDN application routing decisions affect data plane performance (overload and data plane performance in the control plane).
- Constructing a basic SDN platform that consists of one SDN controller handling a disjointed collection of ten OF switches

- **Future Recommendation**

It is anticipated that in future, this concept will serve as a springboard for the adoption of such a technology and, more generally, stochastic network optimization techniques.

References

[1] 5G Wireless Backhaul Networks: Challenges and Research Advances Xiaohu Ge1, Senior Member, IEEE, Hui Cheng1 , Mohsen Guizani2 , Fellow, IEEE, Tao Han1 , Member, IEEE.

[2] Efficient Virtual Memory for Big Memory Servers. Arkaprava Basu, Jayneel Gandhi, Jichuan Chang, Mark D. Hill, Michael M. Swift.

[3] A. Yahaya et al. iREX: Efficient automation architecture for the deployment of Inter-domain QoS policy. IEEE TNSM, 5(1):50–64, March 2008.

[4] Next Generation Mobile Networks Alliance, “Small cell backhaul requirements,” NGMN White paper, 2012.

[5] C.J. Bernardos, A. de la Oliva, P. Serrano, A. Banchs, L.M. Contreras, Hao Jin, and J.C. Zu˜niga, “An architecture for software defined wireless networking,” Wireless Communications, IEEE, vol. 21, no. 3, pp. 52–61, June 2014.

[6] N. McKeown, “Software-Defined Networking,” INFOCOM Keynote talk, April 2009.

[7] Open Networking Foundation, “OpenFlow Switch Specification (Version 1.3.0),” June 2012.

[8] Open Networking Foundation, “Available at: https://www.opennetworking.org”.

[9] C.A.B. Macapuna, C.E. Rothenberg, and M.F. Magalha?es, “In-packet bloom filter based data center networking with distributed openflow controllers,” in GLOBECOM Workshops (GC Wkshps), 2010 IEEE, Dec 2010, pp. 584–588.

[10] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in Proc. of the 7th USENIX Conf. on Networked Systems Design and Implementation, Berkeley, CA, USA, 2010, NSDI'10, pp. 19–19, USENIX Association.

[11] Advait D., F. Hao, S. Mujherjee, T.V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," in Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN 2013). ACM, 2013, pp. 7–12.

[12] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," IEEE Trans. on Automatic Control, vol. 37, no. 12, pp. 1936–1948, Nov. 1992.

[13] J. N´u˜nez-Mart´ınez, J. Baranda, and J. Mangues-Bafalluy, "Experimental evaluation of self-organized backpressure routing in a wireless mesh backhaul of small cells," Ad Hoc Networks, Elsevier, 2015.

[14] J. N´u˜nez-Mart´ınez, J. Baranda, and J. Mangues-Bafalluy, "A selforganized backpressure routing scheme for dynamic small cell deployments," Ad Hoc Networks, Elsevier, 2015.

[15] "The ns-3 network simulator, Avalaible at: http://www.nsam.org".

[16] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," Selected Areas in Communications, IEEE Journal on, vol. 23, no. 1, pp. 89–103, 2005.

[17] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks," in Procs of the 7th USENIX Conf on Networked Systems Design and Implementation, Berkeley, CA, USA, 2010, NSDI'10, pp. 17–17, USENIX Association.

[18] C. Hopps, "Analysis of an equal-cost multi-path algorithm," 2000.

[19] Aricent, "Demystifying routing services in software-defined networking," White Paper, 2013.

[20] Advanced Message Queuing Protocol, "Available at: http://www.amqp.org".

[21] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," in Network Operations and Management Symposium (NOMS), 2014 IEEE, May 2014, pp. 1–4.