MINT CAPSTONE PROJECT

Design and implementation of QoS aware Software Defined Cognitive Network

March 26, 2017

William Nderba
nderba@ualberta.ca

Table of Contents

Abstract

The growth of internet traffic, which is an indication of increased dependency of users on internet for service, business and Internet of things, brings challenges of cost, scale, control and management. Software Defined Networking with Network Function Virtualization (SDN/NFV) provides a cost effective solution to network management as opposed to the use of classical, monolithic network elements. Software Defined Cognitive Network (SDCoN) is a self-aware network design model that adapts to the changing network environment and present proactive solution to network challenges. To provide a proactive solution, the Software Defined Network controller is modeled with an intelligence abstraction layer. Machine learning and external deep learning libraries are used to analyze network statistics. The intelligence plane of the intelligence abstraction layer houses the services and application for data analytics and deep learning. Key consideration for design of SDCoN, besides the intelligent layer is the Southbound protocol for collecting network status stimuli. OpenFlow offers multi part messaging for network statistics collection from OpenFlow tables, Group tables and Meter tables. Cloud Radio Access Network (RAN) has the potential of virtualizing Baseband Pool resources for Remote Radio Units with SDN/NFV. Enabling end to end flows from the Radio Unit to vEPC provides the opportunity of implementing centralized spectrum sensing for cluster of base stations. A demonstration of bandwidth improvement with network path manipulation is demonstrated. Although the demonstration does not take advantage of the opportunities of the SDCoN model, it does demonstrate the feasibility in autonomic learned decision making with SDN. An SDN controller with integrated intelligent Plane Service and APP, synonymous with the MILA framework, provides answers to network management and operation challenges.

INTRODUCTION

The telecommunication network has made technological advance from 2G/GSM EDGE through 3G/WCDMA HSPA to 4G/ LTE service. In each evolution, the constant change is the increase in data throughput for users. Ericsson mobility report in 2016 mentions an increase of 45% in all mobile data with smartphones accounting for 50% of the projected growth [5]. By the year 2020, Cisco projects the global internet traffic would reach 2.3 Zettabyte [6].

The growth is driven by the upsurge of mobile internet users coupled with the ever increasing dependency on internet applications for daily activities. An office in New York is able to access project progress report in real time from a village in Kenya with a simple web database application. Business processes have become more dependent on the internet than ever. Emerging economies are harnessing the potential of internet technology with homegrown solutions for basic needs. In developed economies, Internet of Things (IoT), Automations of Systems and the ever evolving business on the go contributes to the continuous demand for internet traffic. Cisco projects the number of internet connected objects to reach 28.4billion in 2017 and 50bn by 2020 [8]. Wireless data service, cellular data and Wi-Fi, offers the flexibility that suites varying applications and services of internet users. People have become used to instant access to information with increasing complexity in variety of service quality and service management so much that slow internet is unacceptable. The increasing demand for video streams and application services would continue to drive data demands.

According to Naudts et al [7], data traffic volumes increase has not resulted in automatic increase in revenue. In most cases, revenue has stayed constant and it's declining in some cases. At the same time, cost of equipment has only dropped between 10% to 20% [7]. Telecom network operators are experiencing decline in profit making it difficult to reinvest in network infrastructure expansion to accommodate the increasing traffic trend. Monolithic and function specific network devices are cited by [7] as major reason for market demand and architectural design mismatch. Rethinking of network architecture and design that match network capability and market demand is required.

Resolving internet evolving demand requires that network devices and future generation wireless networks have extra capacity to accommodate the growing trend. The service demand complexity also requires inbuilt quality of service that match user service demand without architectural limitations. This would mean enabling quality of service per data flow with efficient bandwidth and spectrum utilization. A major consideration in matching market requirement with architectural design is ensuring a seamless change process. The new design should not disrupt the existing operational service flows. In addition, it should be cost effective. Reusing existing protocols and devices will reduce complexity and cost of deployment [7].

Advance in intelligent networking offers the opportunity to meet diverse and changing network demands that comes with internet traffic growth. Cognitive networks with Cognitive radio, coupled with Network Function Virtualization and Software Defined Networking, has the unique advantage of adaptability, reusability and self-learning in real-time network optimization [2].

Problem Statement

As explained, the internet data growth is putting telecom operators under pressure to increase capacity of bandwidth and to optimize radio spectrum usage. However, these resources are limited. Operators are also faced with the challenge of meeting diverse market Quality of Service (QoS) demands.
This leads to the following problem statement:
How to design an SDN Network that automatically and intelligently adjust to network service and policies requirement?
What are the key consideration in designing such an intelligent SDN Network?

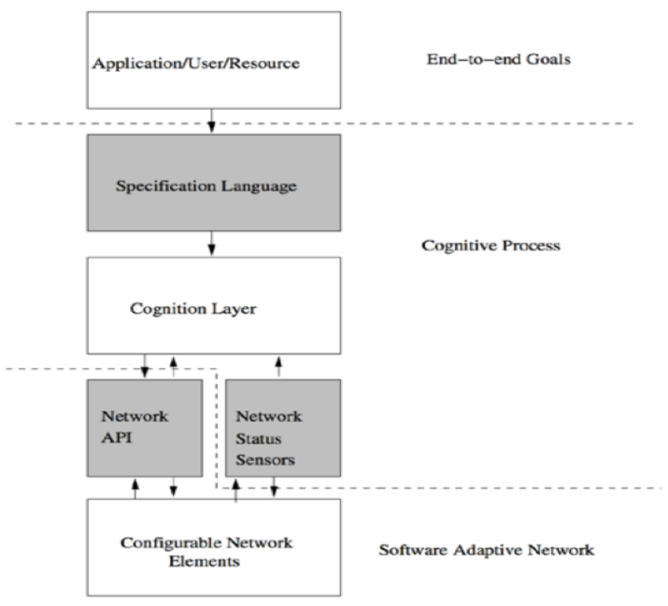
1.0 COGNITIVE NETWORK AND SDN

The nodes in a network, protocol layers and defined policies does not adapt reactively to changes in network. Also, network status communication, response mechanism and scope of network elements are limited by the layered protocol architecture and range of node response to changes. This is the reason network elements offers reactive response in adaptation to changes in network. The idea of cognitive networks is to solve the status and response mechanism limitation, allowing networks to optimize their network performance through observation and learning of network pattern and metrics [2]. Ryan W. Thomas et al in [2] explains that cognitive networks should be forward-looking and adapt to network changes before they occur. Meaning for a network to learn, predict and adapt, it has to be flexible and support integration of new network elements and network service improvement applications. Ryan W. Thomas et al defines cognitive network as the learning of current networking conditions, planning, deciding and taking action based on the learned condition with the network using the learned adaptation to make future decisions while maintaining and end-to-end performance goals [2].

Besides the adaptability to learned network patterns and metrics, cognitive network must have end-to-end network scope. Thus each element of the network is involved in this end-to end scope. To achieve performance optimization goals would require cross-layer optimization. Implying cognitive communication system has distinguishing scope from single element or single layer cognitive systems. To achieve this end-to-end cross-layer performance, network elements should support abstraction of the network functionalities through Software Adaptable Network (SAN) [2]. This offers abstraction, adaptability and cross-layer management.

In 2005, [2] designed a cognitive network layered abstraction model. The layered abstractions make the Application layer independent of the Cognitive process layer and Software Adaptive Network layer. In achieving end-to-end traffic goals, the requirements of the application layer are passed to the independent underlying layers. Software Adaptive Network layer interface with the network element hardware while cognitive process adapts to the goals of traffic flow.

Cognitive Network Layered abstraction Model



1.1 Cognitive Network Process

It is understood from [2] that the process by which cognitive networks learn network pattern, metrics and policies could be centralized or decentralized. Centralized benevolent control means all network elements operate with the permission and resource allocation of a central controller. As multiple streams of data flow through a network element, how each flow is treated does not depend on the selfish goals of the local network element. In achieving end-to-end cross layer, cross network elements data flow objective, an SDN controller makes this possible.

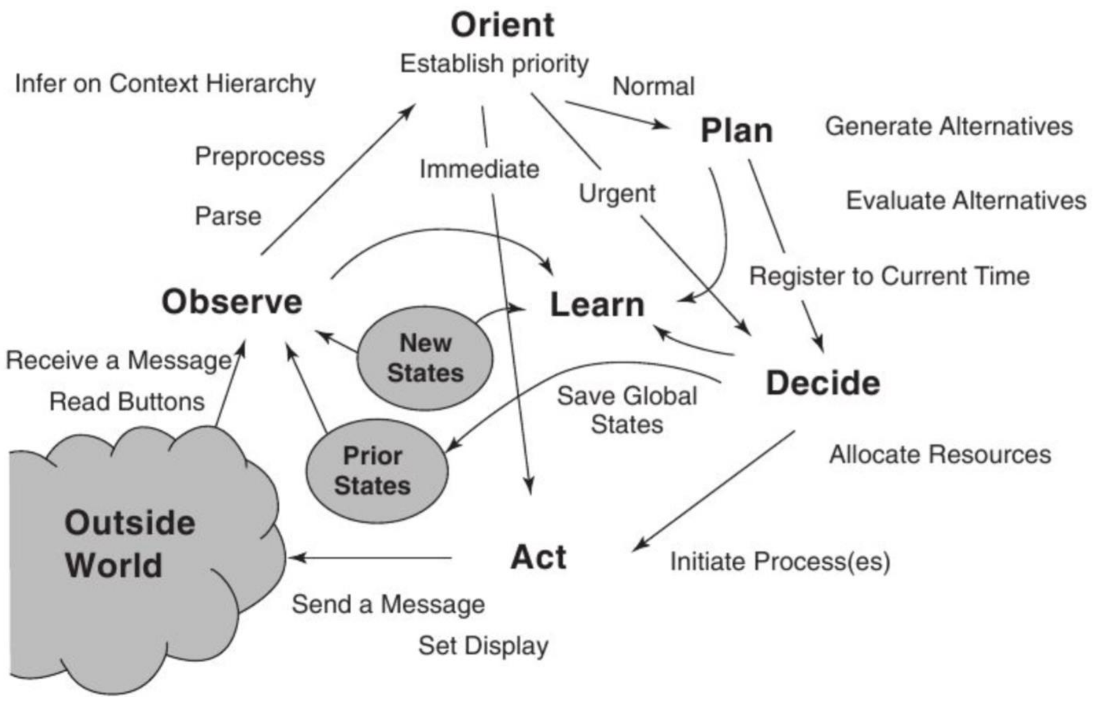
The scope of the network is not limited by the network elements rather, the instance of data flow within the network. The preferential treatment of data flow from one end to another may transcends a couple of network elements. Adopting a scope defined by the data flow introduces a question of the price of anarchy [10], thus if self-governed network element which undertakes selfish local flow goals is preferable to the centralized controller which harmonizes the overall network resources and data flow goals.

The state of each network element must be known to permit end-to-end flow of data. Data loss in not knowing the state of a network element as traffic flows from one network element to another directly affects revenue and should be avoided. Network elements know the status of the next switch, device or hop. Centralized or decentralized learning of network devices state should weigh the cost of packet overheads with single point failure.

Response-mechanism or feedback to change in the network environment defines the foundation of cognitive networking. Thomas et al [2] illustrates cognitive process with the feedback loop first deigned by Col. John Boyd. The loop has four stages: Observe, Orient, Decide and Act (OODA). That is - Observing stimuli from the environment, Orient state, Decide and take Action. Mitola [9] provides a more elaborate explanation. The limitation of creating a software adaptive network in the loop is overcome with innovations in SDN.

According to Mitola [9], streams of stimuli from the environment are **observed** and passed. In the observation, inference is made and stimuli is bound to prior observations from the environment to stablish a pattern. The **orient** phase binds observations to recent known set of stimuli to establish stimulus recognition. This is achieved when current stimulus is a match with prior experience. With nearly exact match between currently observed stimuli and prior experience, binding is done for easy inference. A **plan** is generated to deal with incoming message(Stimuli). This suggest that response to observation from the environment are not reactive. The plan phase deals with incoming stimuli deliberately and it does so by including reasoning over time. Deliberate response to a stimuli may not always be available in a situation where there is no planned response making reactive response suitable approach in such instances. Among the available plans, in the **decide** phase, a decision is made. The decision is based on policies, metrics or data flow goals. The selected decision is acted upon in the **Act** phase. This could mean undertaking an action within the network element or sending a signal for action to be taken on an external environment. The **learn** phase is the phase of perception. Thus perception through observation, decision and action. The inference hierarchy facilitates learning.

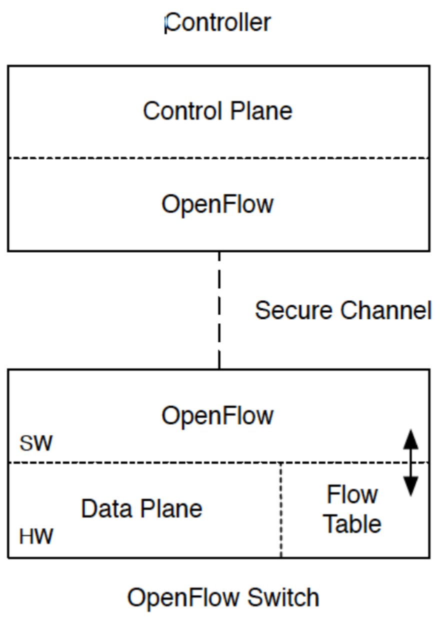
Cognition cycle implemented in Java [Mitola, Joseph. 2006]



1.3 SDN and OpenFlow for Networking

Software-Defined network is a layered architecture, separates management and control plane from data plane. Meaning the data or forwarding plane can be an independent unit/device interfaced with the control and management plane via standard intercommunication protocols. OpenFlow is a common protocol for southbound communication between SDN controller and forwarding switch or a base station as the case may be. [1] enables OpenFlow at the base station interface with forwarding switch which are controlled by a central SDN controller. Software Defined Cognitive Network (SDCoN), a Cognitive Network (CN) over SDN.

OpenFlow enabled Switch (reference [2])



1.4 SDN and Vendor Dependency

SDN model of separating the control from the forwarding creates flexibility in the programming of the forwarding plane [7]. The intelligence of the network is programmed in a central controller and path to devices are discovered through databases, thus the controller operates like an operating system program [2]. The forwarding plane which may be switch or routing devices are manipulated by the central intelligence via interfacing OpenFlow protocol. The route to a destination is determined by the Controller through the sharing of flow tables with OpenFLow enabled switches.

Switch configurations and requirements are modified using the OpenFLow protocol [2]. Thus for a switches to support OpenFLow, it must therefore be designed to conform to the OpenFLow standard. The separation of control and management from the forwarding plane and the use of OpenFLow protocol creates an open communication between platforms. The controller and Switch interface becomes less vendor dependent [7]. SDN with OpenFLow breaks the vendor dependency barrier in interconnecting different vendor network resources. An OpenFLow enabled switch can interconnect with an SDN controller via secured TCP/TLS channel [21].

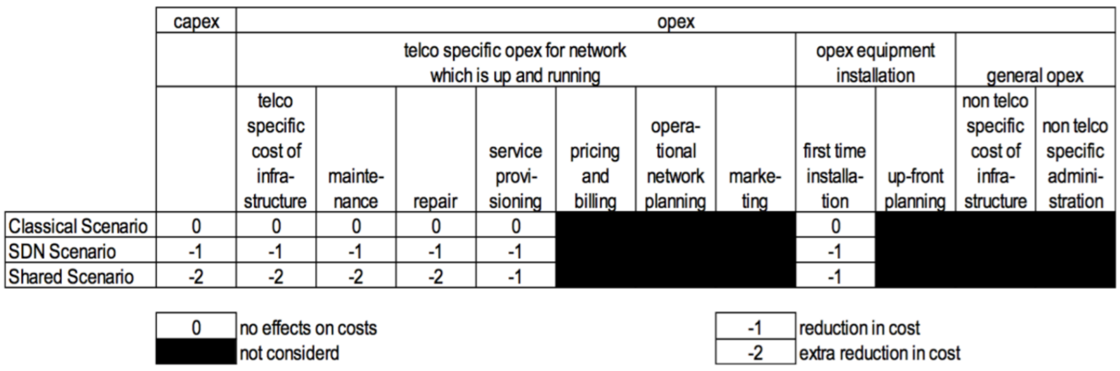
Network virtualization, resource is cut into slices such that different channels can access different slices of the resources without an overlap. All virtual slices are managed at the same time. The network resources abstraction into slices can be assigned to network devices in real time [7]. That is the essence of network virtualization. For example, function virtualization of radio network, Enhanced-UTRAN or Radio Base station resources is possible with FlowVisor [7] as an OpenFlow supported Controller. SDN with network function virtualization permits isolated networks with different addressing and forwarding schemes to access the same physical infrastructure [7]. OpenFlow supported SDN with network function virtualization further erodes vendor dependency.

1.5 Economic benefits of SDN

The economic benefit of SDN is higher compared with the existing or classical deployment. The existing distributed network, which is classified as the Classical Scenario, is compared alongside the SDN scenario, which has a centralized controller managing and controlling switches through Southbound protocol (OpenFlow). The third scenario, named the sharing scenario, is an SDN OpenFlow network with functional virtualization allowing the sharing of network resources among multiple operators with FlowVisor controller. [7]

The capital and Operational expenditure in each of classical Scenario, SDN scenario and Shared scenario are analyzed. The Capital expenditure (CAPEX), OpenFlow controllers, line cards and interface adapters constitute the major cost for SDN deployment [7]. Unlike the classical approach that deploys complicated logic on multiple device, SDN scenario requires a Central Controller to manage multiple devices at the same time. The shared scenario aligns network capacity with actual demand. This reduces the waste of having redundant or marginally used resources on one hand and an over utilization on the other. Shared scenario incorporated in radio network design with, FlowVisor controller, will achieve the possibility of resources demand alignment.

Overview of capital expenditures and operational expenditures and potential savings with the classical scenario as reference point
(Naudts et al [7])



	capex	opex										
		telco specific opex for network which is up and running							opex equipment installation		general opex	
		telco specific cost of infra-structure	mainte-nance	repair	service provi-sioning	pricing and billing	opera-tional network planning	marke-ting	first time installa-tion	up-front planning	non telco specific cost of infra-structure	non telco specific admini-stration
Classical Scenario	0	0	0	0	0	■	■	■	0	■	■	■
SDN Scenario	-1	-1	-1	-1	-1	■	■	■	-1	■	■	■
Shared Scenario	-2	-2	-2	-2	-1	■	■	■	-1	■	■	■

0	no effects on costs		-1	reduction in cost
■	not considerd		-2	extra reduction in cost

Operational expenditure (OPEX) considerations by [7] are power usage, maintenance and repair cost, cost of service provisioning and finally the cost of installing network equipment for the first time. The separation of management, control from forwarding devices reduces processing activity and subsequent power reduction in the forwarding plane for SDN network. Maintenance cost is less in SDN scenario with the number of software reduced and has fewer autonomous network devices. The shared scenario has different operators sharing the cost of replacement. Different operators can be responsible for different sections of the network reducing the cost of repair in Sharing scenario. Provisioning of service in SDN or shared scenario is automated – saves cost of manual operations and human errors. Installing equipment for the first time requires testing, and extensive checks. SDN offers the simulation platform for rigorous testing and doubles as learning opportunity for operators. However, Controller suffers the disadvantage of a single point of failure (depending on deployment topology).

2.0 CONSIDERTIONS FOR SDCoN

The idea of policy and service aware Software Defined Cognitive Network (SDCoN) is to design a software defined network structure that can learn from the network's structured-system data and do so automatically. This implies two major considerations. The first is the design or construction of the network system for learning to take place. The second consideration is enabling the learning to happen. That is, to make possible the learning of the network data. The important aspect of automatically learning about the network structured-system is to have real time understanding of the network and to make changes or adapt to network changes. Network automation exist in current SDN networks for Load balancing, QoS and Boarder Gateway Protocol. The key consideration for SDCoN is to explore existing automation options and project a design which is self-aware, self-correcting and always learning. In existing automation system, response to change is a reactive process. The computing program is unable to predict the network behavior and modify program ahead of time. Over the years, computing algorithms have improved. Reactive response time to network failures are faster with quicker resolution time. With increasing internet traffic growth, varying user service categories and radio spectrum constraints, it is important to predict accurately the network behavior and make modifications to accommodate the predicted change.

Deploying resource based on need is a good solution to resource constraints. To determine bandwidth, queues, spectrum etc for services based on need/requirement can be deployed in time. Bandwidth as a service(BaaS) [13] illustrates deploying bandwidth based on client Service Level Agreement[SLA]. This implies the allowable bandwidth is an external data input. All the same, [13] provides a strong case of managing resources based on need. The opposite would be deploying fixed bandwidth for client wither in use or not. Internet growth and network demands require resource on demand solution. SDCoN, unlike [13], learns from each action and predict future behavior.

So far, an explanation is provided for the distinction of SDCoN from existing automation computing and for instance [13]. The best model for SDCoN is a model and SDN as a network brain with Machine Learning (ML) or Artificial Intelligence(AI) providing cognitive network and service functions.

2.1 Considerations for Intelligent SDN Controller

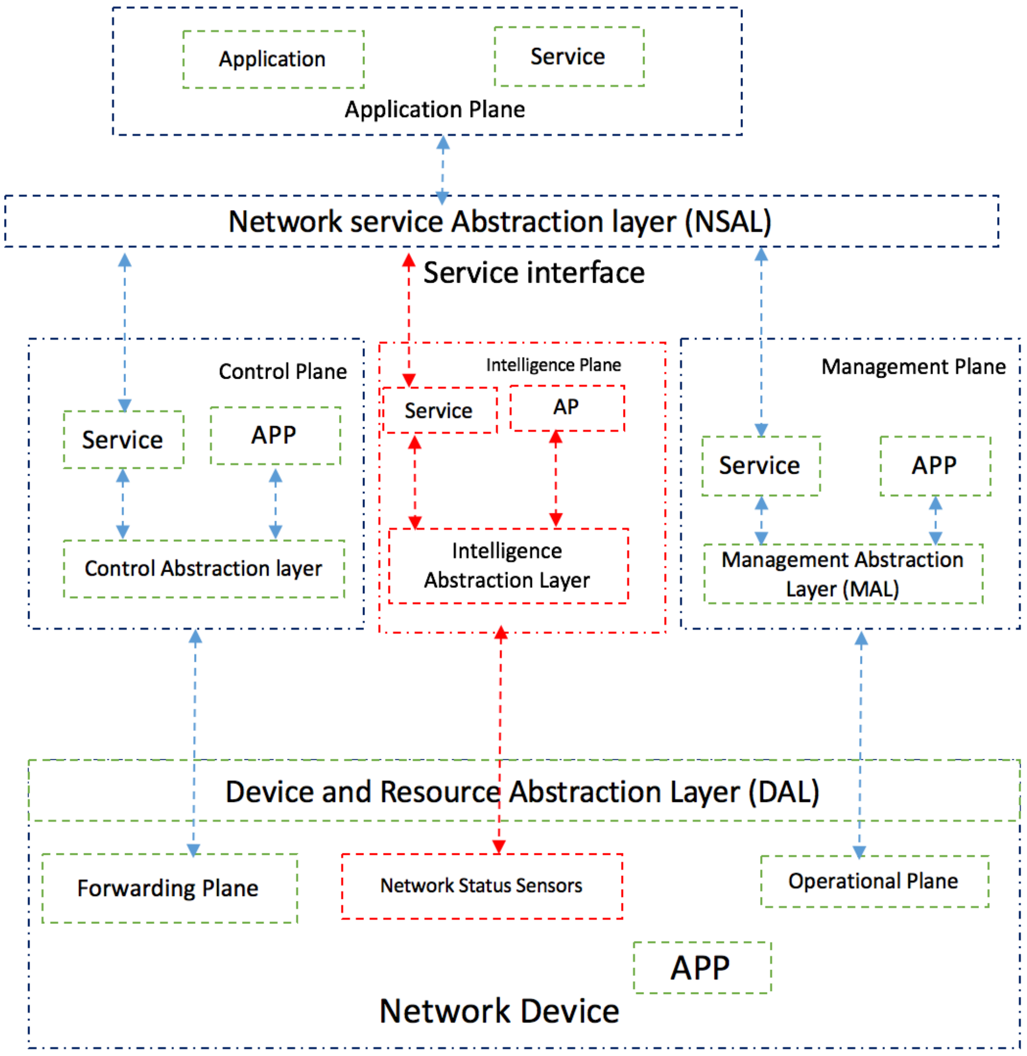
There are multiple controllers in the market. In this model, OpenDaylight (ODL) as the primary focus. There would be occasional mention and use of other controllers. The existing SDN layered model provides the flexibility of programmability. The existing model of an SDN network has Data (forwarding) plane, Control plane and Management plane.

In a layered view of an SDN network, as illustrated by Thomas et al [2], the Software Adaptable Network (SAN) layer offers abstraction, adaptability and cross-layer management. Autonomic functions of existing SDN networks are implemented through an application layer over Northbound protocol. For instance, implementation and validation of an SDN Controller for Transport Network Virtualization and Autonomic Operation [14] used Netconf – ODL northbound API – for control applications to interact with the network. In [1] the cognitive engine interacts with the network device abstraction layer and the control plane.

The SDN reference model (RFC 7426), there are two abstraction layers and an application plane. These are: Device and resource abstraction layer (DAL), Network Service Abstraction layer(NSAL) and an Application plane. In comparison, Thomas et al [2] cognitive abstraction model is a simplified version. The cognition layer is not explicitly defined in the reference model. It is considered as an additional intelligence layer, required for SDCoN. This would imply that designing an intelligent SDN, the functionality of ML/AI should be housed within the intelligent layer. The intelligent layer should be within the controller, interact with the control and management plane and have access to the service plane. It should also have services and applications. The RFC7426 defines an Application (APP) as standalone piece of software which does not offer interface API connection to any applications or services. Cognitive process in Thomas et al [2] has Network API and status sensor connections. Service defined by the RFC7426 performs single or multiple functions, provides one or more APIs connections to applications and services either on the same layer or across layers. Cognitive Process requires more than an APP, it is a combinations of service(s) and application(s).

The cognition process is combination an APP and Service. Service has inter-process communication to other services with cross layer functionality. Cross layer functionality implies the Application Plane can interact with cognition service process. The paper, Towards Software Defined Cognitive Networking [1] illustrates a model with the cognitive engine that interacts with OpenFlow enabled switch and the Controller.

The aim of this project is to propose a design that clearly identifies the intelligent layer and its functionality, and does not seek to reinvent the layers. In this project, the cognition layer is designed as an autonomous/ independent Intelligence Abstraction Layer (IAL) which can interact with other services and applications within or cross layers.



SDN Layer Architecture with Intelligence Abstraction layer

2.1.1 Intelligence Abstraction Layer (IAL)

The Intelligence abstraction layer is responsible executing the cognition cycle. The explanation of the cognition cycle in [9] as applied to the IAL:

Observation – Observe the network device environment with Network Status Sensors. Streams of stimuli are passed on to the IAL.
Orient – Observed data is matched with existing database to detect similarity with previous experience.
Plan- To make sure the network response is not reactive; a planned action is set in motion based on consistency of observed stimuli with previous experience.
Decide – A decision is made based on the available plan option. Decision maybe policy, service or any other applicable decision.
Act – Action phase could be on DAL or sending signal to NSAL for specific action to be taken. The Control Plane executes the final action. Control Plane configures Forwarding-Plane through Control-Plane Southbound Interface (CPSI) [15].
Learn – at this stage, the hierarchy of observation, decision and action trigger learning via perception. Analyzing statistical parameters builds experience. IETF-86 proceedings defined learning as a ‘procedure that estimates model parameters so that the learned model (algorithm) can perform a specific task’ [16]
Retrospection – This is an added state which is applicable in ML/AI. Machine learning introduces ‘sleep’ and ‘prayer’ epoch [9] for network retrospection - cognitive radio. Implying the ML/AI service for the controller would have an Online/real-time process and an Offline ML process.

The Intelligence and Abstraction layer functionality is to process information which may or may not be predictable. The information pattern may be so unpredictable that it cannot be modeled into a function. The network status stimuli passed on to the IAL may be of different; either be higher-dimensional or lower-dimensional data. These characteristics makes the use of ML/AI suitable. Also there are available open source code for ML. ML is suitable for recognition of signal patterns. Stimuli information from DAL. The Orient phase requires understanding relationship of various data, data optimization to avoid the challenge of hidden data and other ML functionalities like data pattern generation, anomaly detection and prediction of network events. With ML, the function is predicted from available data. This must include development of models that fit both the known and unknown and well as predict the future. Predicting the future implies learning. The challenge then is the complexity of dimensionality. ML; whether supervised, semi-supervised, unsupervised, or reinforcement; is to estimate model parameters such that the learned algorithm can perform desired task. [16]

2.1.2 Intelligent Controller Framework with ML/AI

With an intelligent Controller, or controller with an intelligent layer, the ML/AI can be implemented for traffic and routing management, resource optimization, security and network troubleshooting. The intelligence plane does the complex analytics of ML.

Traffic Control and Routing Optimization –The possibility of predicting the future, can be used to predict network traffic pattern. The fore knowledge of traffic pattern is helpful in optimized network. In such situations, congestion is controlled.
Resource optimization – The network resource is allocated on demand. Underutilization of resource is avoidable. There is optimization in management of Cloud resource.
Security – Enhanced security and anomaly detection including DDoS attack detection. [17]

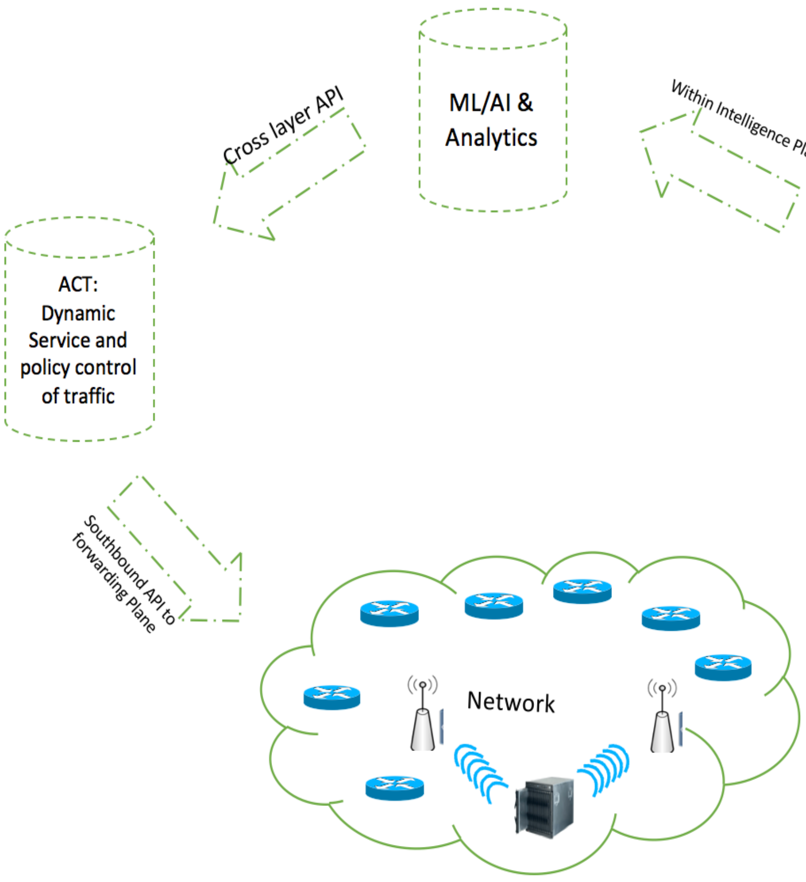
OpenDaylight Machine Learning & Artificial Intelligence (MILA) Framework [17], aptly captures the major considerations for an intelligent controller as: adapting the Controller for ML requirements and facilitate ML services development.

Application (APP) – The Intelligent Plane provides framework for ML applications. That's a standalone piece of software that uses underlining services to pass arguments at a function call time and does not offer interfaces to other applications or services [15].

Services - service software performs multiple functions, provides API connection. Service software API connects to other services either on the same layer or across different layers. Services can combine with other services to form a new service [15].

Intelligence Abstraction Layer – Provides access to the Intelligence Plane southbound interface. The network status sensor data is exchanges through the Southbound API which integrates existing or native network data collection and controls.

Policy and Service Awareness (ie QoS Aware Controller) - Network status sensor collects data from network. Machine learning and analytics is used to determine traffic flow policy and service requirement. Robotically change policy and service for traffic.

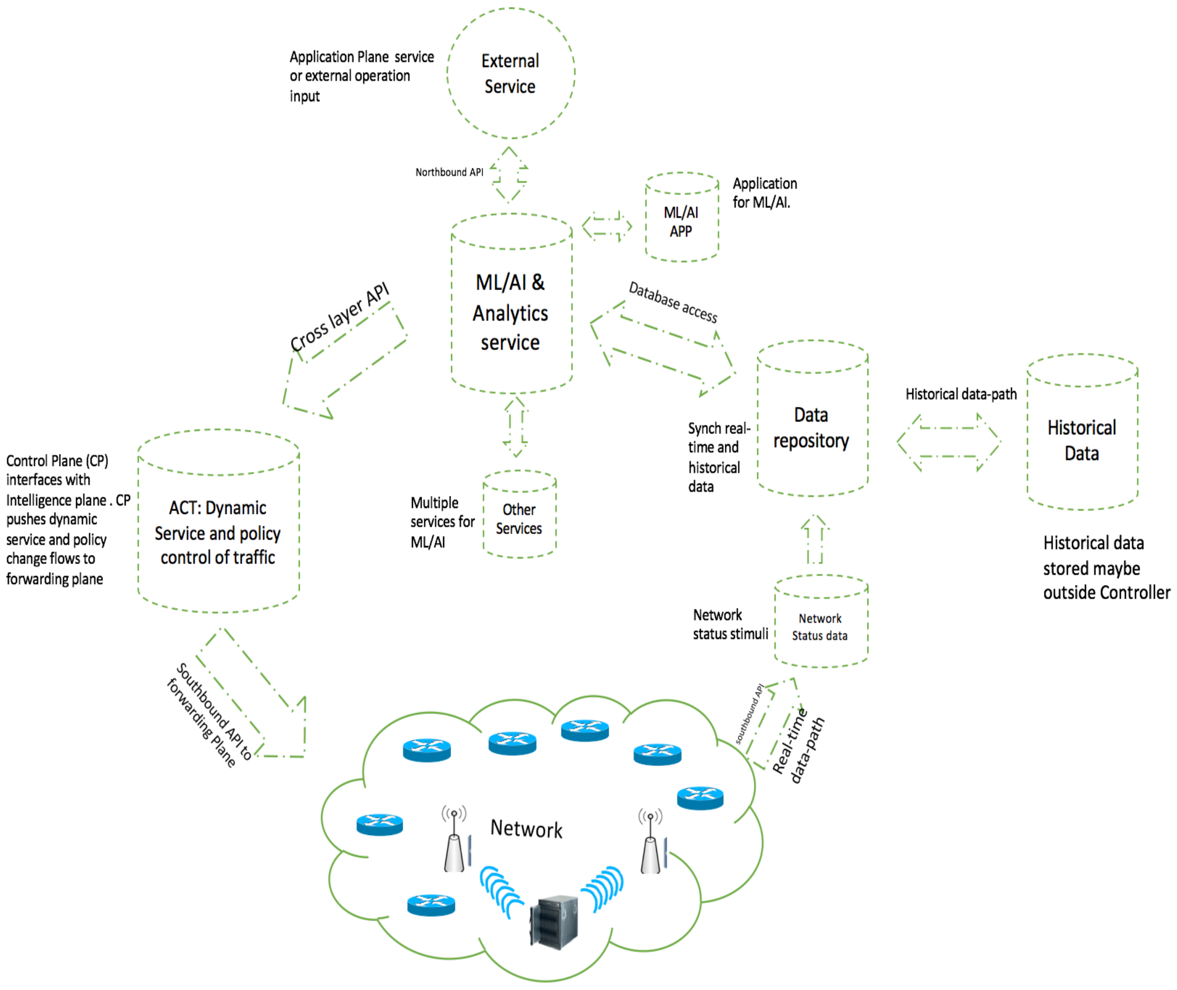

ML/AI & Analytics
Cross layer API
Within Intelligence Plane
ACT: Dynamic Service and policy control of traffic
Network Status Data collection
Southbound API to forwarding Plane
Southbound API
Network

Network Data collection to IAL: Data from the network: statistics, environmental stimuli and network device status; are collected and housed in a data repository. This means there must be a Data Collection Service (DCS) that routinely collects data from the DAL. The data collected, which are time stamped, are stored in with the Data Storage Service (DSS). In ODL, the data storage is TSDR. The data storage has Data Query Service (DQS), Data Purging Service (DPS) and Data Aggregation Services (DAS). Data Query Service provides means to access the stored data. Data collected cannot be stored forever in the controller database. To avoid losing information, the Data Purging Service should have backup/recovery system, which maybe, outside of the controller – example Operations

Support Subsystem (OSS) Database. This means the ML/AI should have both real-time and historical data path. Data Aggregation Service processes the combination of streams of data into datasets. Real-time data-path and historical data-path connection to the ML/AI data repository allows ML to access both existing data in the controller and historical data stored on an external server.

Data Analytics & ML: In the POC of OpenDaylight MILA framework, ODL Time Series Data Repository(TSDR) in collaboration with Waikato Environment for Knowledge Analysis (Weka) are used for advanced analytics and ML. Weka leverages the data collected in TSDR to make network predictions with analytic/cognitive engine [17]. The prediction, for instance could be traffic congestion on a particular network path in the next operation cycle (eg 24hrs). Possible best path options and decision is made. A cross layer service interaction with the control plane sends the actionable path selection flow to the forwarding plane.

Interface Communication: Interaction between two services, elements or entities could be implemented using Application Programming Interface (API), Inter-Process Communication (IPC) or Network Protocol [15]. OpenFlow is a well-known and well documented southbound API for SDN. Other southbound protocols can be used in place or in cooperation with OpenFlow. The ML/AI analytic module has data-paths to the data repository. The real-time network information is accessed through the real-time data path with historical data-path available to access existing data. Application Plane service(s) or an external operator may interact with the ML/AI module.



Application Plane service or external operation input
External Service
Northbound API
ML/AI APP
Application for ML/AI.
ML/AI & Analytics service
Cross layer API
Database access
Data repository
Synch real-time and historical data
Historical data-path
Historical Data
Historical data stored maybe outside Controller
Control Plane (CP) interfaces with Intelligence plane . CP pushes dynamic service and policy change flows to forwarding plane
ACT: Dynamic Service and policy control of traffic
Multiple services for ML/AI
Other Services
Network status stimuli
Network Status data
Southbound API to forwarding Plane
southbound API
Real-time data-path
Network

2.2 Considerations for Cognitive Radio over SDN

Radio networks in Cellular Network has base station and controller of the base station that is in-charge of allocation of radio resources. For example, in a GSM network, a Base Station Subsystem (BSS) has a base station controller responsible for multiple Base Stations. The situation no different with Radio Network Controller (RNC) that manages multiple Node B's in Wide-band Code-Division Multiple Access (WCDMA) network of a Radio Access Network (RAN). Long-Term evolution (LTE) presents a different topology. Each E-Node B, manages its own radio resources and does not have a central radio controller like GSM or WCDMA. Progress into Cloud Radio Access Network (CRAN) aims to centralized the Baseband Pool(BBP) resource of the individual base stations. Thus the radio resource management, which is done within the BBP, would be controlled from a cloud. This CRAN topology is similar to both BSS and RAN in that the radio resource is allocated by a centralized controller. Again, in all these three technologies, base stations and mobile users stay within allocated frequency spectrum of the network provider. In a populated area, the spectrum and frequency allocations for Base Stations are not enough at service peak hours. This leads to congestion. The national agencies, communication authorities, responsible for spectrum allocations, assign network providers with static spectrums. Operators are expected to operate within this spectrum. In large populated regions, static allocation of spectrums no longer serves varying demands.

Tao Jiang et al [11] stated the November 2002, U.S. Federal Communications Commission (FCC) Spectrum Policy Task Force report that there are more available radio spectra which are less utilized and at the same time there are other spectra experiencing congestion. The report concludes that spectrum scarcity is not the challenge rather the static allocation of spectra, which does not permit non-assigned users access, is the most significant cause of the spectrum problem. This means white spaces may not be adequate for future radio network technologies. The FCC task force report gives the indication that certain spectra are more utilized whilst others are barely used. This is not an efficient use of existing spectra. A congested spectrum provides less reliable communication. Thus, the current static spectrum allocation results in an inefficient use of spectrum and poor user communication experience [11].

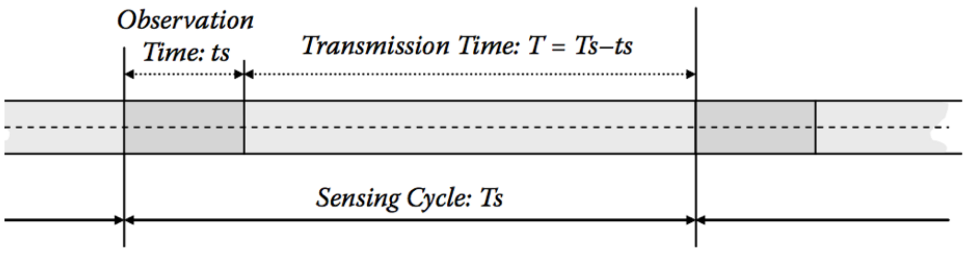
The spectrum inefficiency challenge can be resolved if users can share their assigned static spectrum with other users that are not licensed to operate within that spectrum. In the spectrum assignment, it is possible for Primary User (PU), thus user with license to operate within an underutilized spectrum, to allow usage of its spectrum by a Secondary User (SU), which has license to operate in a different and possibly congested spectrum. A primary user (PU) assigned a frequency spectrum band, can allow a secondary user (SU) to transmit within it, this is Sharing-Based Spectrum Allocation. Also, the SU may be allowed to operate in the band allocated to the PU in the absence of the PU, this scenario is Opportunistic Spectrum Allocation (OSA). Implying that in sharing-based spectrum allocation the SU is permitted to transmit as long as SU interference does not degrade the quality of service of the PU. Equally, in OSA, the SU can operate in the frequency band allocated to the PU as long as the PU is absent. This opens avenue for primary legacy frequency band allocations, like TV/broadcasting, navigation, GSM, WCDM etc, to be used by SU when the PU is absent [4].

Spectrum inefficiency will be reduced with sharing-based spectrum allocation and the Opportunistic Spectrum Allocation. The radio network should be intelligent to observe the presence or absence of a PU. To know a primary user, exist, it should be aware of its surrounding environment and be able to read external stimuli. Secondly, it should be able to learn from observed stimuli and adapt itself in accordance with the changes observed from environment (external radio parameters). Since the radio environment varies, it is expected to adapt to observed variations. Some of the variations maybe parameters like transmit-power, carrier-frequency, and modulation strategy. It uses the methodology of understanding-by-building to learn from the environment and adapt. So, in either sharing-based spectrum allocation and Opportunistic Spectrum Allocation, the radio network is expected to exhibit awareness, intelligence, learning, adaptivity, reliability and effectiveness. A radio that show these characteristics is a Cognitive Radio (CR). A Cognitive Radio Network (CRN) is a solution to spectral inefficiency. Sensing in OSA CRN involves both physical and MAC layer. Physical layer sensing is observing the environment to detect PU signals to know if a PU is present or not. MAC layer sensing tells which channels to sense. CRN OSA aims to identify and use a channel with no operating PU at a specific place (geographic location) and time [11].

2.2.1 Cognitive Radio Sensing Model

The aim of cognitive radio is to provide more spectrum access to users. In sensing, the SU should not introduce interference in the PU band. The effectiveness in providing more spectrum access depends on the accuracy of sensing with no interference.

Sensing of the PU band by SU is done periodically. There is the need to determine the time in which it is considered sufficient and optimal for sensing to stop and transmission by SU to begin. At the period of sensing (which is equal to t_s), the SU does not transmit. This is to avoid the confusion between PU and SU signal as the RF front-end cannot differentiate between the two. Hence, there is a period of sensing, at which if no interference is detected, is preceded by transmission of SU. The sensing and transmission period of the SU forms a cycle of time = T_s. Hence the duration which SU transmits is = $T_s - t_s$. The primary objective is to find the right balance between observation time (t_s) and transmission time (T_s). In a network, to have a model that prevents interference of the PU band, sensing of the SU is synchronized to occur at the same time. Again SU sensing information is shared with the base station which shares with other SUs which form cooperative sensing. Cooperative sensing among SUs prevents fading and shadowing [18].



The cognitive sensing Cycle. Reference [18]

There are different SU sensing methods. Primary transmitter detection, which includes matched filter detection, energy detection and feature detection, the SU is doing detection from the primary transmitter. Cooperative transmitter detection depends on shared SU detection information. The third method is primary receiver detection. The surest way of know the PU is active is if it receiver local oscillator leakage power. This is can be measured by SU in the primary receiver detection to check the presence of a PU. The fourth is interference temperature management [18].

The PU activity is modeled to determine the period PU band is used and the time it is not used. The model depicts PU activity as exponentially distributed inter-arrivals. Meaning the PU has two states, either active (birth) or inactive(Death). So the PU has a ON (busy) state at which the it uses the spectrum and an OFF (idle) period when the spectrum is not used. The arrival of PU in the spectrum are independent of each other and follows the Poisson process with the length of the ON, OFF periods exponentially distribution [18].

Cognitive Radio Optimal Sensing Framework – Adaptive Cooperative Sensing:
Below is a scenario for the CR radio sensing framework from [18]

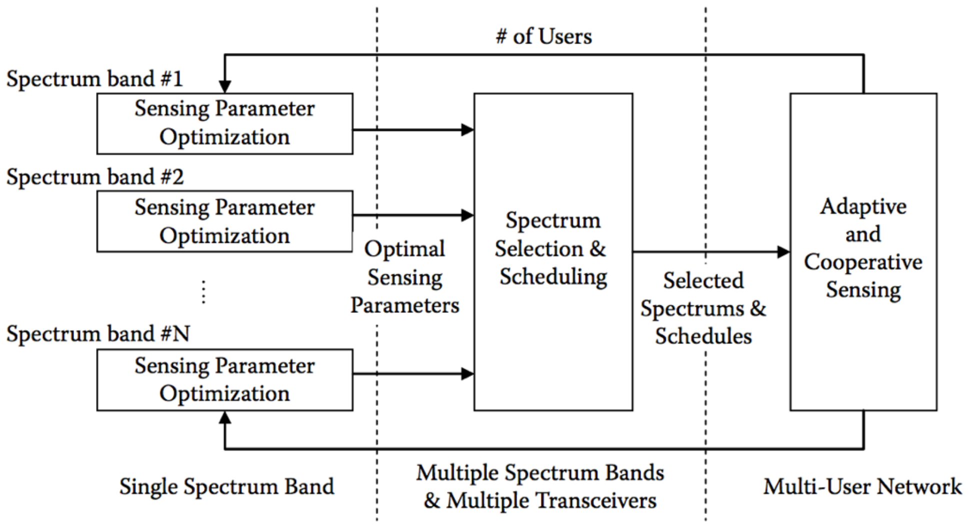
1. **Sensing parameter optimization phase**: The base station determines the parameters required for optimal sensing of the allowable spectrum bands. The required parameters for optimal sensing of each spectrum band is determined.

2. **Spectrum selection and scheduling**: When SU join the CR networks, the most suitable spectrum band is selected by the base station for the SU through spectrum selection and scheduling methods. The optimal sensing parameters are set for the SU with consideration for the sensing schedules (to avoid introduction of interference through sensing) and number of its transceivers.

3. **Spectrum Monitoring**: The SU monitors spectrum bands. SU does the monitoring continuously with the assigned optimized sensing parameters and predetermined schedule. The monitoring results are sent to the base station.

4. **Determination of Spectrum Availability**: The base station, based on the results from spectrum monitoring, determine availability, or otherwise, of the spectrum.

5. **Adaptive and cooperative sensing phase**: If the base station detects changes in the sensing environment that affects sensing results, the sensing parameters are re-optimized. The new optimized parameters are announced to the SUs. The cooperative sensing of SUs help in detection changes that affect the sensing performance.



# of Users
Spectrum band #1
Sensing Parameter Optimization
Spectrum band #2
Sensing Parameter Optimization
Spectrum band #N
Sensing Parameter Optimization
Optimal Sensing Parameters
Spectrum Selection & Scheduling
Selected Spectrums & Schedules
Adaptive and Cooperative Sensing
Single Spectrum Band
Multiple Spectrum Bands & Multiple Transceivers
Multi-User Network

Optimal Sensing Framework – Adaptive Cooperative Sensing
Reference [18]

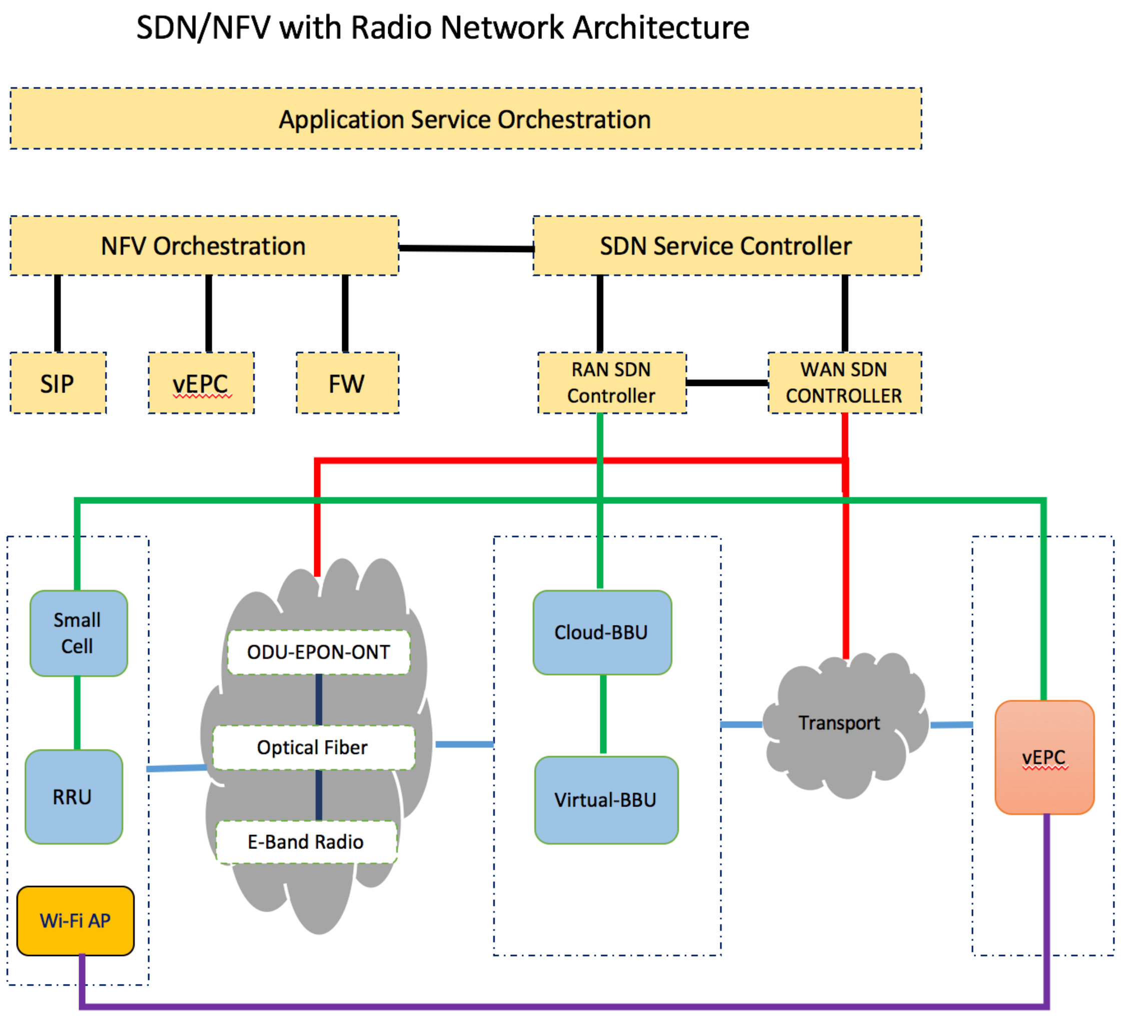
2.3 SDN with Cognitive Radio Framework

The cognition process, observe, orient, plan, decide, act and learn, does applies to radio cognitive networks. The decision on observation from the radio network environment - which includes spectrum bands sensing by cognitive radio user and base station – is made by radio base station controller.

In recent radio architecture (eg. 3G/4G-LTE), the Baseband Unit (BBU), connects to Remote Radio Units (RRU) for radio signal processing. The optical cable linking the RRU to the BBU can be extended to a long distance over access backbone network. This makes it possible to house the radio control functionality in a cloud. Thus, creates a pool of baseband resources which can be allocated on demand. Thus creating BBUs in a Cloud connecting RRUs, over Gigabit optical link - identical to Cloud RAN architecture.

With SDN and Network Function Virtualization (NFV), the virtual BBU resource is deployed on demand for different RRUs. The BBU connects to the Evolved Packet Core network nodes (MME, SGW, PGW, ANDSF, HSS, etc) [19] which may also be housed in the cloud. This makes it possible to use SDN/NFV to manage the radio access network elements.

SDN/NFV in the cloud, with the intelligent controller module, would have learning of results from radio environment sensing with ML/AI and a subsequent prediction of radio resource management decision. It is also possible to combine 3GPP access technology with non-3GPP (eg. Wifi) technologies [19].



(SDN/NFV architecture for Radio Network. Reference [20] [19])

WAN SDN Controller - SDN/NFV with Transport Network virtualization and autonomic operation [14] [20].

Wi-Fi AP - the functionality in the vEPC: authentication, Access Network Discovery and Selection Function (ANDSF) and Packet Data Network Gateway (PGW), which links users to internet [19].

3.0 Cognitive Controller Service and Application Model Design

The Intelligent plane has two streams of information flow: The real time and historical data. ML has both real-time, online ML and a historical offline ML operations [17]. The real-time online ML data is collected via pipe with OpenFlow protocol statistics from the Network whereas the offline historical ML obtains ML data via the controller data repository.

3.1 Processing Data and statistics from the network

The collection of data: information about network environment stimuli, network status, fault information, interface tx/rx data rate, policies, services and link states.

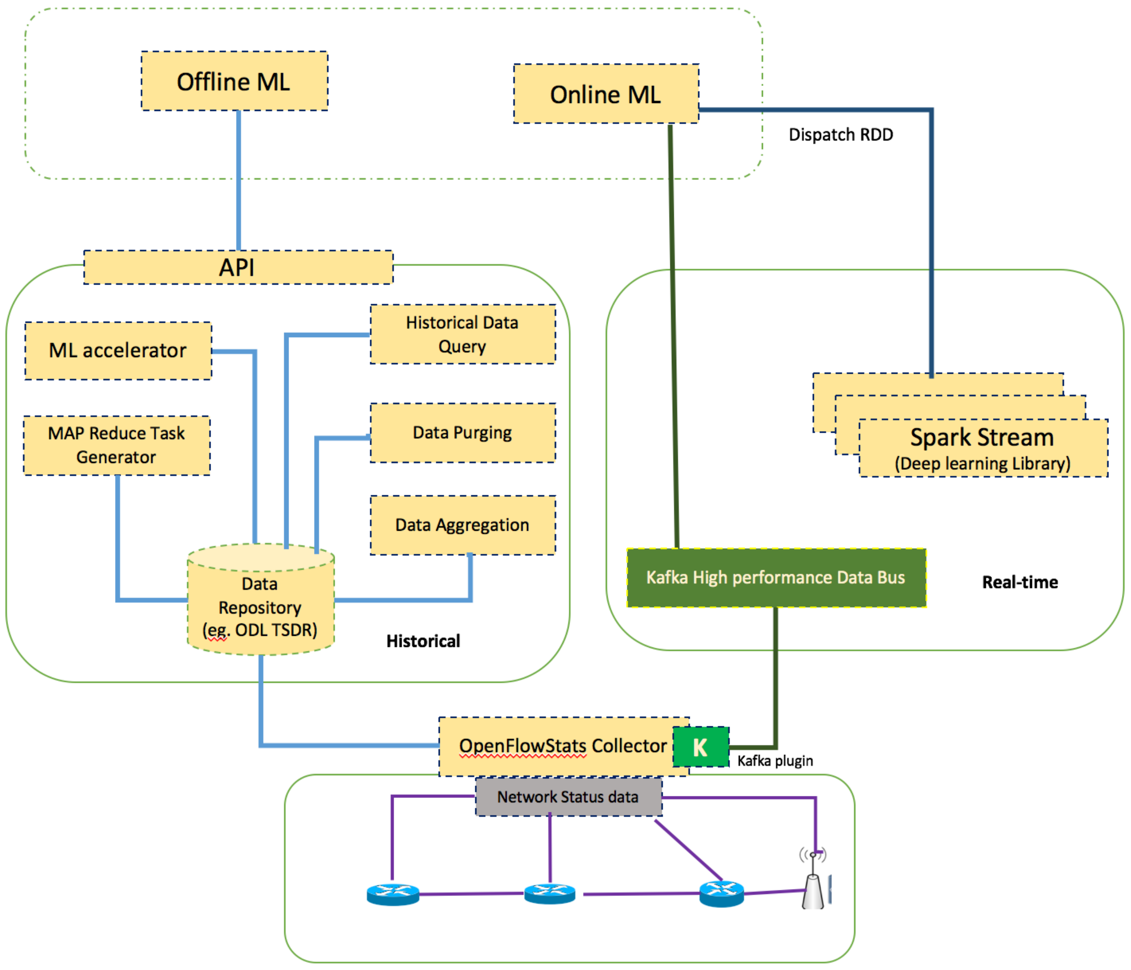
Online ML Service: the implementation requires collection of network status information directly through the OpenFlow statistics collector. This can be achieved by creating a stream of real-time data pipeline with Kafka plugin [17]. Kafka provides the opportunity to reliably transfer streams of data between the network and the controller in real-time. The online ML is able to generate response to streams of data that comes via Kafka.

Offline ML Service: Offline ML uses historical data of the network. That is the network statistics collected via OpenFlow. The OpenFLow statistics collected from the network at periodic data collection cycle is fetched via API or IPC for the offline ML.

The collected data and statistics is analyzed with ML / deep learning algorithm.

External ML Libraries: The ML service with the intelligence plane can use external deep learning libraries. Sample deep learning libraries include: Spark Mlib, Theano, Torch, Caffe and DeepLearning4J [17]. In the illustration below, Spark Mlib is used. The Spark Mllib package use Resilient Distributed Datasets (RDD) based API.

Machine learning Algorithm: The statistics obtained through OpenFlow protocol is used to generate ML algorithm. There algorithm could be supervised or unsupervised learning. For instance, the use of unsupervised K-means clustering is simple concept. Again, unsupervised learning does not require teaching the algorithm something about the OpenFlow data first. Like all ML algorithms, the acquired data, Network information, is cleaned before running the algorithm. The acquired data can be visualized with a graphical plot.

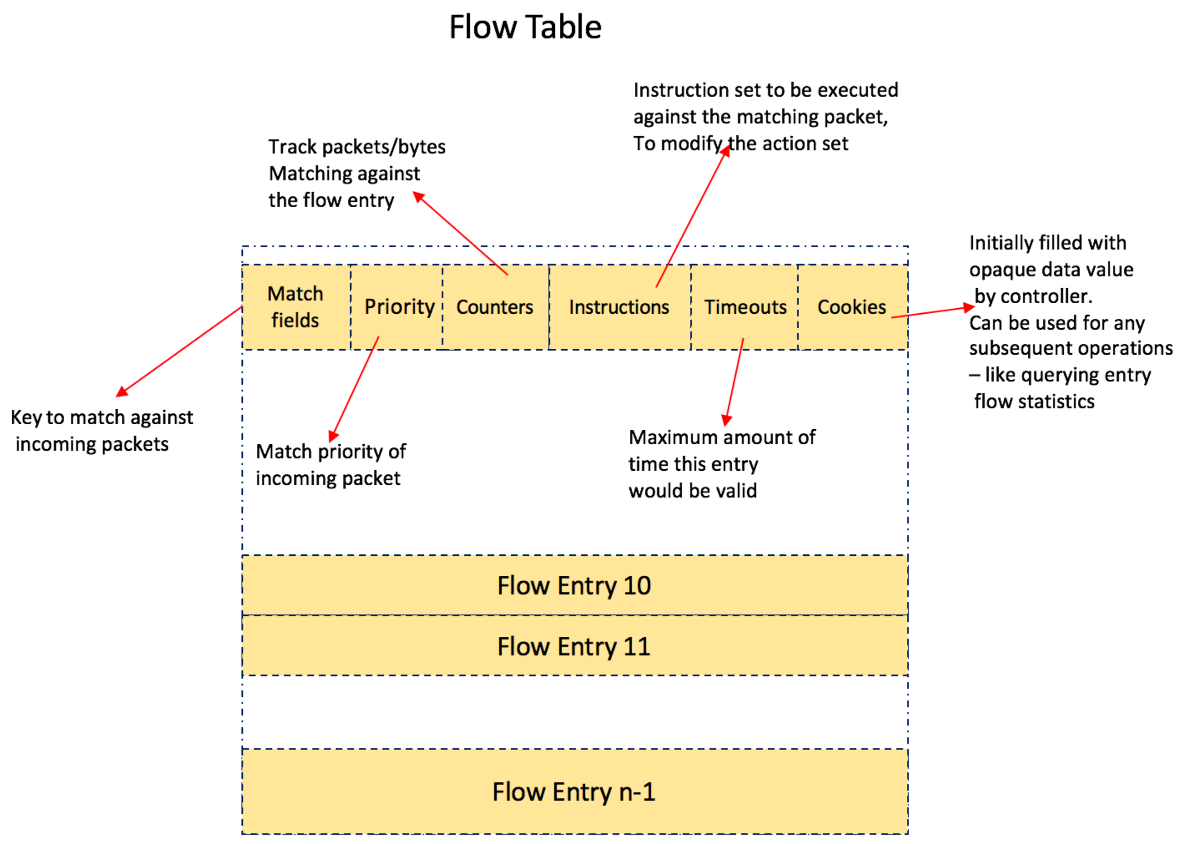


(Intelligent SDN with Machine Learning Module. Reference [17])

3.2 Sending and Receiving data through Southbound Protocol - OpenFlow

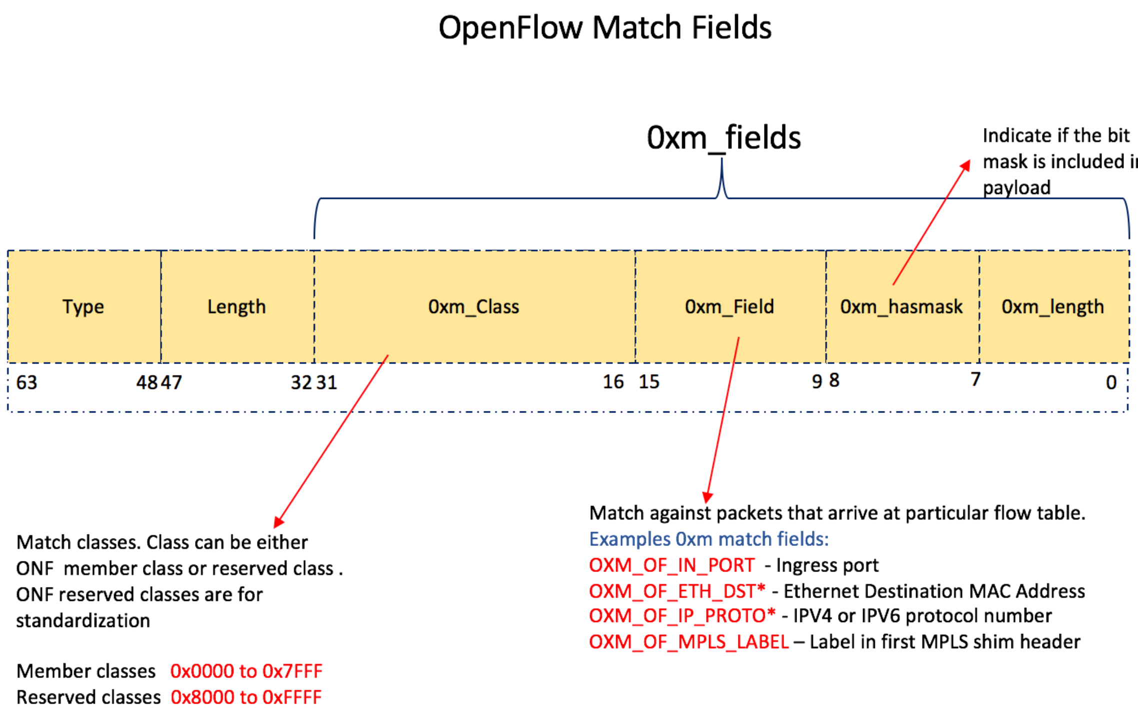
The Southbound protocol, OpenFlow, is used to forward actionable instructions to the network. For instance, if it is learned that the end-to-end path of a packet would be congested in the next 24hrs, the action to change the path has to be forwarded to the forwarding plane. This means making OpenFlow table modifications. The OpenFlow pipeline processing on the OpenFlow switch can either be implemented in hardware or software. Essentially, three main components of OpenFlow are: Flow table, Group table and Meter table.

Flow table: is an essential part of an OpenFlow switch. The Flow table as the datapath equivalent in traditional switches. OpenFlow pipeline processing consist of multiple flow tables (possible to have a single table). These tables contain entries that determine how packets should be processed. The Flow tables are numbered sequentially – starting from zero (0) to n-1. The switch capacity determines the number of flow tables. The controller is informed of the capability of the switch *Handshake message* during OpenFLow connection establishment. [21]

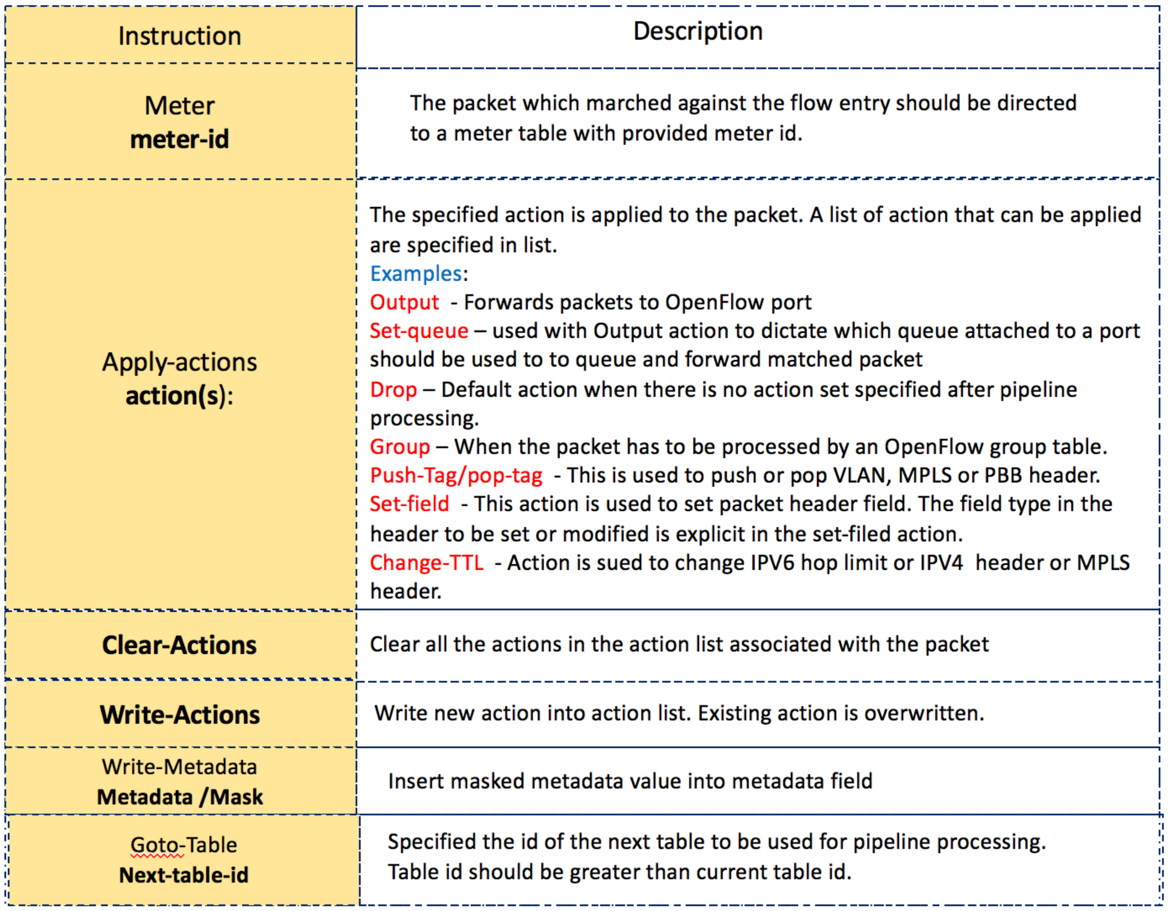

Flow Table
Instruction set to be executed against the matching packet, To modify the action set
Track packets/bytes Matching against the flow entry
Initially filled with opaque data value by controller. Can be used for any subsequent operations – like querying entry flow statistics
Match fields
Priority
Counters
Instructions
Timeouts
Cookies
Key to match against incoming packets
Match priority of incoming packet
Maximum amount of time this entry would be valid
Flow Entry 10
Flow Entry 11
Flow Entry n-1

OpenFlow **Match fields** is used to match against incoming traffic. The flow match fields have OpenFLow Extensible Match (0xM) format. The 0xM is a Type-Length-Value (TLV) format of size of varying size from 5 to 259 bytes. The Match fields has Type (set with

OFPMT_OXM) and length (actual length match fields) and Payload field (which is a set of 0xM fields). [21]

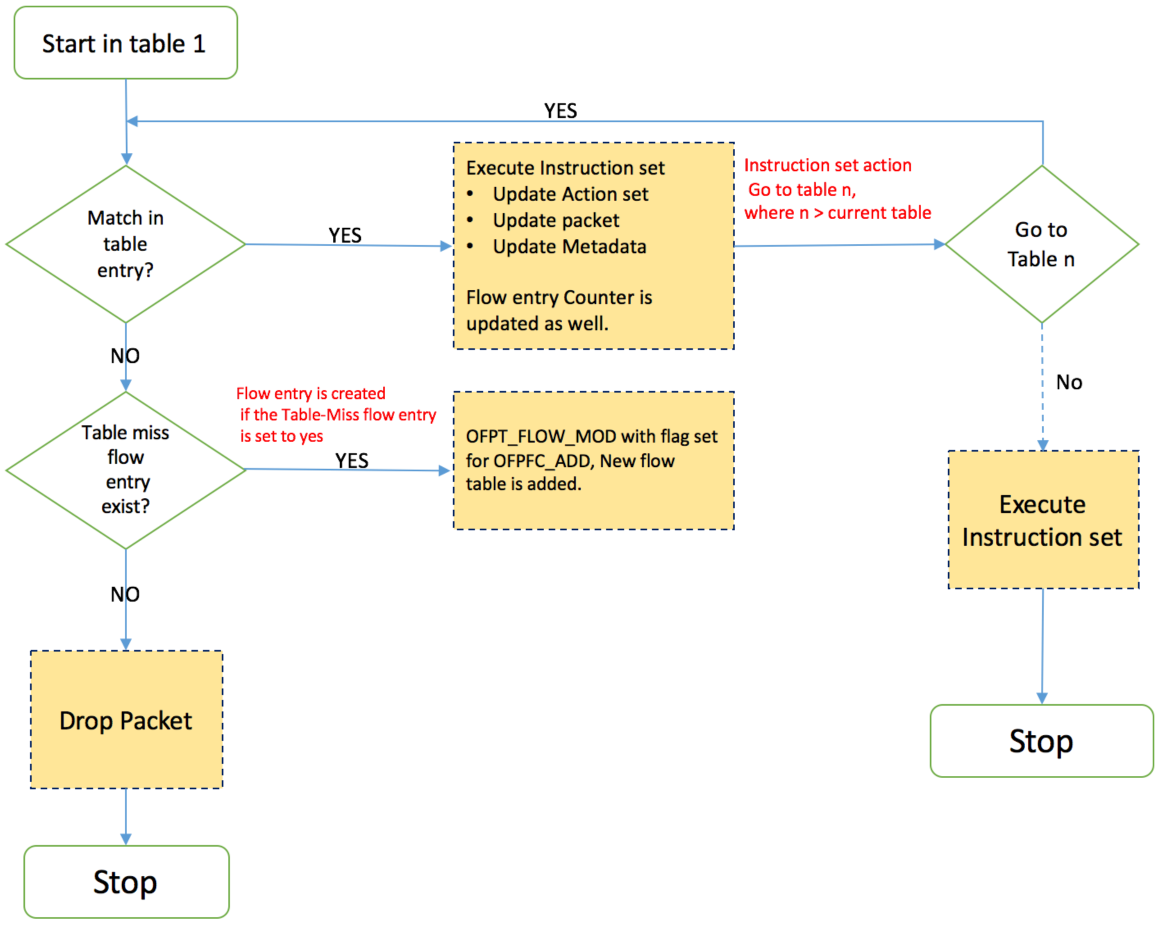

OpenFlow Match Fields
0xm_fields
Indicate if the bit mask is included in payload
Type
Length
0xm_Class
0xm_Field
0xm_hasmask
0xm_length
63
48
47
32
31
16
15
9
8
7
0
Match classes. Class can be either ONF member class or reserved class . ONF reserved classes are for standardization
Member classes 0x0000 to 0x7FFF
Reserved classes 0x8000 to 0xFFFF
Match against packets that arrive at particular flow table.
Examples 0xm match fields:
OXM_OF_IN_PORT - Ingress port
OXM_OF_ETH_DST* - Ethernet Destination MAC Address
OXM_OF_IP_PROTO* - IPV4 or IPV6 protocol number
OXM_OF_MPLS_LABEL – Label in first MPLS shim header

The **Priority** is used when an incoming packet matches against two or more flow entries in the flow table. Priority field is more of a tie breaker in deciding which flow entry in the flow table should be best match an incoming packet. The **Counter** field is sued for statistics. The counter is updated anytime a packet is matched against an entry in a flow table. This gives information per flow table (Reference count of active entries) and per flow entry (received packets, received bytes, duration). The **instruction** set has six (6) set of instructions which is executed (in the order presented below) when the packet matches against a flow entry. In general, the actions inside an action set are executed in the following order, no matter the order in which they are added in the action set: Copy TTL inwards, Pop, Push-MPLS, Push-PBB, Push-VLAN, Copy-TTL outwards, Decrement TTL, Set, QoS, Group, Output - [21



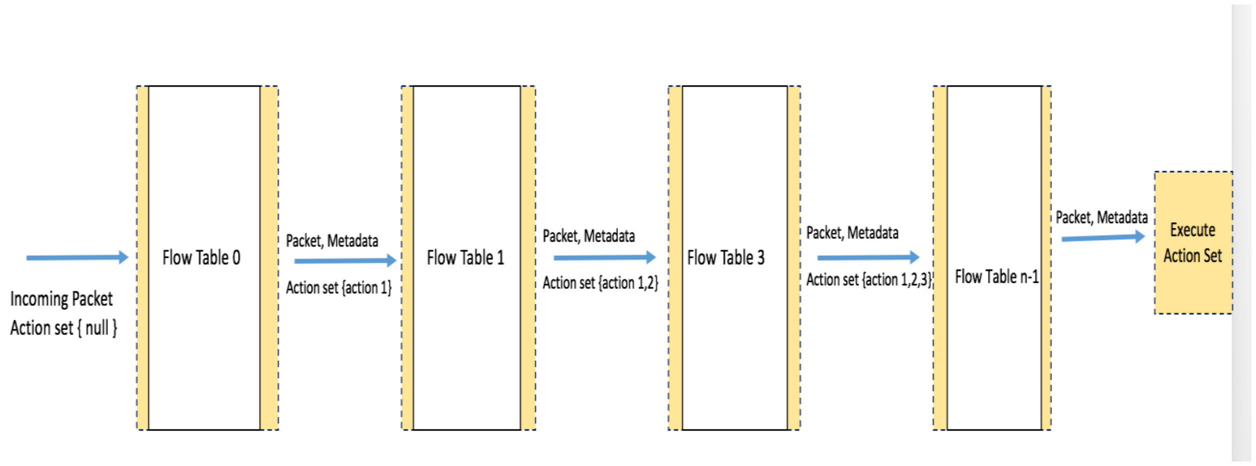
Instruction	Description
Meter **meter-id**	The packet which marched against the flow entry should be directed to a meter table with provided meter id.
Apply-actions **action(s)**:	The specified action is applied to the packet. A list of action that can be applied are specified in list. Examples: Output - Forwards packets to OpenFlow port Set-queue – used with Output action to dictate which queue attached to a port should be used to to queue and forward matched packet Drop – Default action when there is no action set specified after pipeline processing. Group – When the packet has to be processed by an OpenFlow group table. Push-Tag/pop-tag - This is used to push or pop VLAN, MPLS or PBB header. Set-field - This action is used to set packet header field. The field type in the header to be set or modified is explicit in the set-filed action. Change-TTL - Action is sued to change IPV6 hop limit or IPV4 header or MPLS header.
Clear-Actions	Clear all the actions in the action list associated with the packet
Write-Actions	Write new action into action list. Existing action is overwritten.
Write-Metadata **Metadata /Mask**	Insert masked metadata value into metadata field
Goto-Table **Next-table-id**	Specified the id of the next table to be used for pipeline processing. Table id should be greater than current table id.

An OpenFlow entry has two **Timeout** specifications which are set by the controller. They are *idle_timeout* and *hard_timeout*. When a flow entry counter, has no packet or byte count for a period equal to the *idle-timeout,* the switch deletes the flow entry. A flow entry has a time-span from the time it is added to the flow table. When the time-span is up, the switch deletes the flow entry. When statistics is required and there is the need to query the flow entry, **cookie** field is used. In OpenFlow table, a new entry can be added to the flow table, modified, removed and multiple tables can be synchronized (thus synchronizing one flow table with another). [21]


Start in table 1
YES
Match in table entry?
YES
Execute Instruction set
• Update Action set
• Update packet
• Update Metadata
Flow entry Counter is updated as well.
Instruction set action Go to table n, where n > current table
Go to Table n
NO
Flow entry is created if the Table-Miss flow entry is set to yes
Table miss flow entry exist?
YES
OFPT_FLOW_MOD with flag set for OFPFC_ADD, New flow table is added.
No
Execute Instruction set
NO
Drop Packet
Stop
Stop

The incoming packet is first matched with against the first entry of the first flow table. The matching flow entry which has highest priority selected for further processing of the packet. The instruction set in the flow matched flow entry is executed. The instruction may require further forwarding of the packet Meter table or Group table. [21]

OpenFlow Processing pipeline:



Group table: The group table has the following keys: Group ID, Group type, Counters and Action Bucket. The group table is used for operations such as Multicast forwarding, Multipath forwarding, switch port abstraction and fast failover support (etc). [22]

Group table provides abstraction for the following: Multipath forwarding and multicast Equal Cost Multi Path(ECMP) forwarding, Interface ports as single virtual port for forwarding and Fast failover. Group table is referred to by the flow table when an action set of the packet contains a group action with specified group table id. The group table action bucket contains a set of actions to be executed [22].
A group id uniquely identifies a group. There are 4 Group types: All (broadcast and multicast forwarding), Select (used for multipath/ECMP forwarding), Fast failover (providing fast failover support for flow entries) and Indirect (provide and indirect object for multiple entries use the same output action) [22]

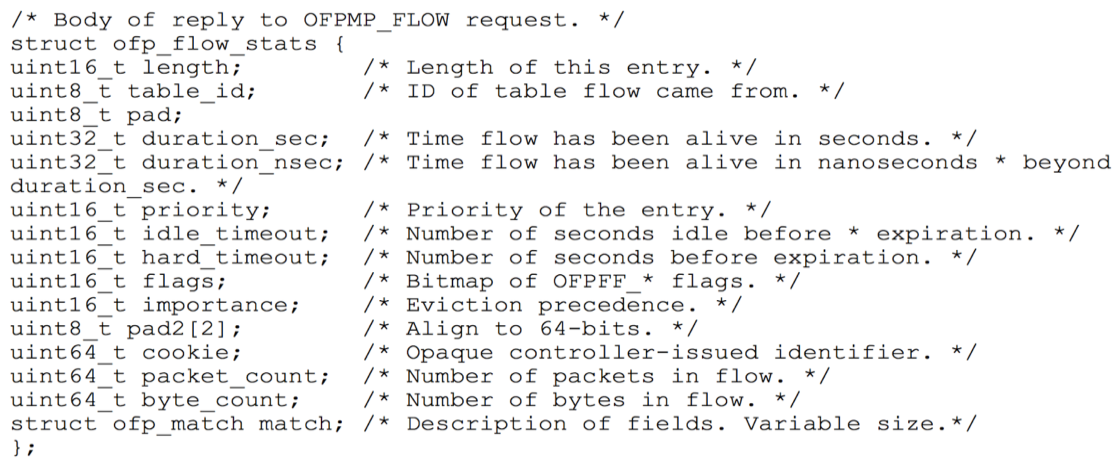
Meter Table: Meter table is used for QoS operations. This ranges from simple QoS operations like rate-limiting to more complex forms as Diffserv. When a packet arriving in a switch is matched with the flow table entries and the instruction indicates Meter with specified *meter-id*, the switch forwards the packet to the meter table entry with the specified meter-id [22].

A meter table has the following keys: meter id, meter bands and counters. The meter band defines how the packet should be forwarded. A meter entry may have more than a single meter band. Each meter band has a band type. The band type can be any of the following: Drop packet, DSCP remark, Counter for specific band and an optional argument (type specific) [22].

Fetching Statistics from OpenFlow switch with Multipart Messaging
OpenFLow messages that are too large (size is bigger than 64KB) to be encoded in a single message can be broken into parts and forwarded. Multipart messaging is used to encode OpenFLow request and reply messages bigger than 64KB. This is typically suitable for collection of statistics. Multipart messages may be interleaved with other messages; the unique transaction id makes reassembling of messages possible. [23]

Fetching Statistics from individual Flow Entry: Assuming we want to collect statistics from a particular / individual flow entry. OFPMP_FLOW message is sent to the switch with the flag set to OFPMP_REQ_. The OFPMP_FLOW message request has the following: flow table id (besides the flow table id, OFPTT_ALL can be set to indicate all tables), match field, output port (which can be set to OFPP_ANY) and output group (which can also be set to OFPG_ANY). The request also has cookie value and cookie mask. When the switch receives this message, identifies the entry in the flow table and constructs a reply message. The reply of the OFPMP_FLOW message has cookie value filled with the value of the multipart request message. [23]

Body of reply to OFPMP_FLOW message request - message fields



```
/* Body of reply to OFPMP_FLOW request. */
struct ofp_flow_stats {
uint16_t length;        /* Length of this entry. */
uint8_t table_id;       /* ID of table flow came from. */
uint8_t pad;
uint32_t duration_sec;  /* Time flow has been alive in seconds. */
uint32_t duration_nsec; /* Time flow has been alive in nanoseconds * beyond
duration_sec. */
uint16_t priority;      /* Priority of the entry. */
uint16_t idle_timeout;  /* Number of seconds idle before * expiration. */
uint16_t hard_timeout;  /* Number of seconds before expiration. */
uint16_t flags;         /* Bitmap of OFPFF_* flags. */
uint16_t importance;    /* Eviction precedence. */
uint8_t pad2[2];        /* Align to 64-bits. */
uint64_t cookie;        /* Opaque controller-issued identifier. */
uint64_t packet_count;  /* Number of packets in flow. */
uint64_t byte_count;    /* Number of bytes in flow. */
struct ofp_match match; /* Description of fields. Variable size.*/
};
```

OpenFlow Multipart Reply Message body fields. Source of C code: [23]

Fetching Statistics from aggregate of flow tables: The OFPMP_AGGREGATE multipart message sent to the switch for aggregate flow table statistics. There is still table id, match field, output group, output port, cookie and cookie mask [23].

Fetching Statistics from Table: When statistics of the entire flow table, OFPMP_TABLE message is sent to the switch with empty body field [23]

Fetching Statistics from a Port: The message sent to the switch is OFPMP_PORT_STATS. The body has only the port number. The port number field in the body can be set to OFPP_ANY for all ports [23].

Fetching Port Queue Statistics: The message sent to the switch is OFMP_QUEUE_STATS. The body fields are port number and queue id. All ports is OFPP_ANY and all queue is OFPQ_ALL [23].

Fetching Meter Statistics: The message sent to the switch is OFPMP_METER. The body requires the meter id field and it could be set to OFPM_ALL for all [23].

Fetching Group Statistics: The message to the switch is OFPMP_GROUP and the body field has the group id required [23].

Besides statistics, information about the switch, flow tables, Group and Meter tables features can be can be fetched from the switch using multipart messaging.

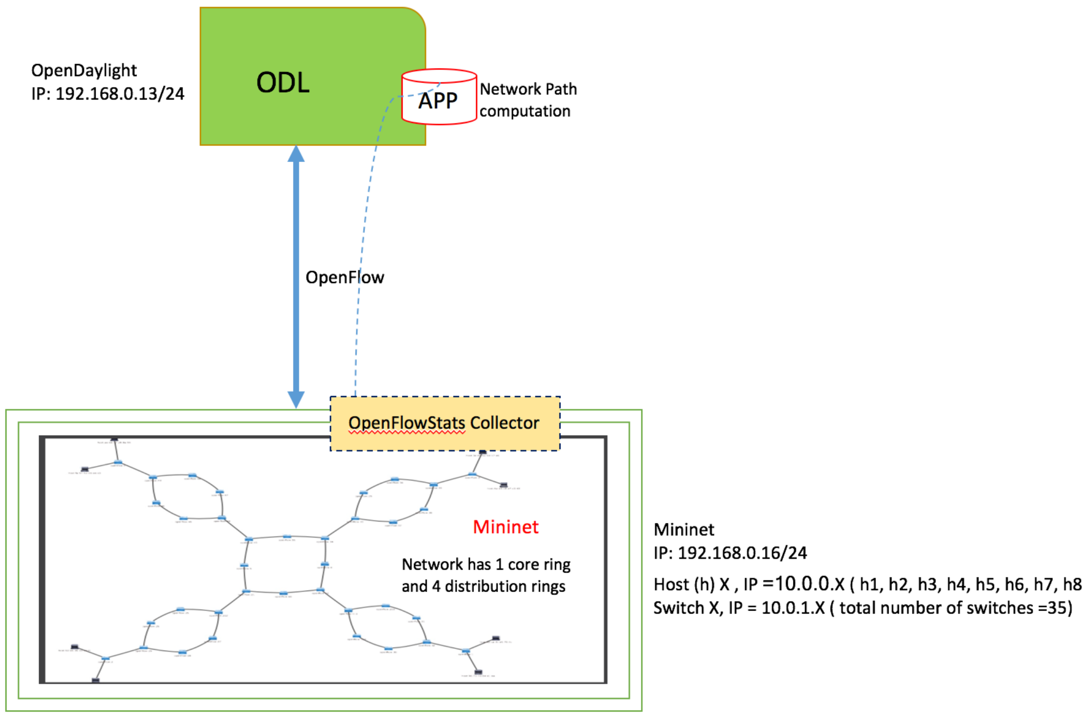
4. Scenario Implementations

The implementation considers different scenario of services over SDN and the results is analyzed.

4.1 CASE I: Improving Bandwidth Utilization

Description: The flow of packets through a network are assigned the same path end-to-end. To balance the load on the network, the path is changed to a newer link of a lower cost. This is primarily balancing the load on the network based on the cost of the path observed from the forwarding plane.

4.1.1 CASE I: Network Topology:

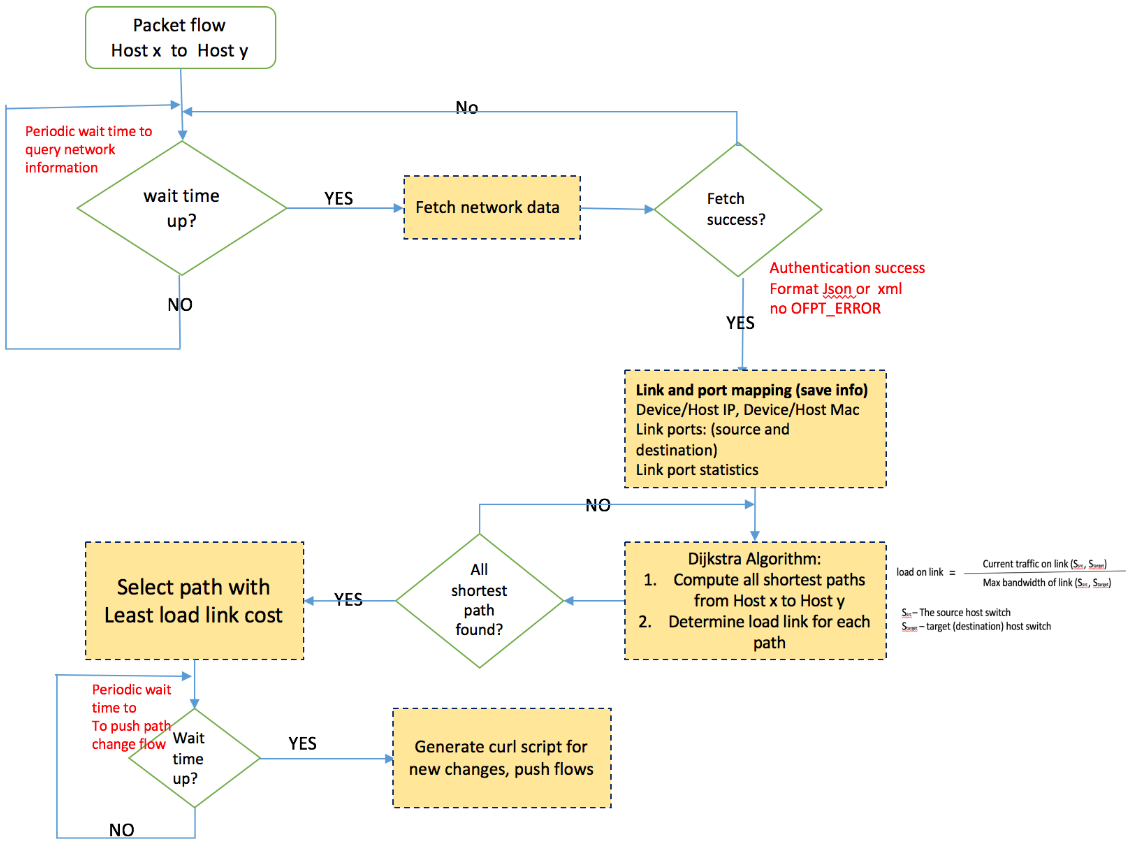

OpenDaylight
IP: 192.168.0.13/24
ODL
APP
Network Path computation
OpenFlow
OpenFlowStats Collector
Mininet
Network has 1 core ring and 4 distribution rings
Mininet
IP: 192.168.0.16/24
Host (h) X , IP =10.0.0.X (h1, h2, h3, h4, h5, h6, h7, h8
Switch X, IP = 10.0.1.X (total number of switches =35)

Mininet and ODL are run on different virtual machines. The network topology consists of 8 network Host connected to 4 different distribution rings which are interconnected with a core ring. The host network and Switch nodes are in different networks segments. To avoid looping, spanning-tree protocol is enabled on each bridge.

The network topology data is fetched and the cost of each path is analyzed. The interface and port of each node is considered. The cost takes into consideration how busy, or how much flow passing through a port interface. A new path is computed for the host-to-host traffic. The flow for the new path is pushed to the network. The impact on the available bandwidth before and after the flow is demonstrated.

4.1.2 Algorithm

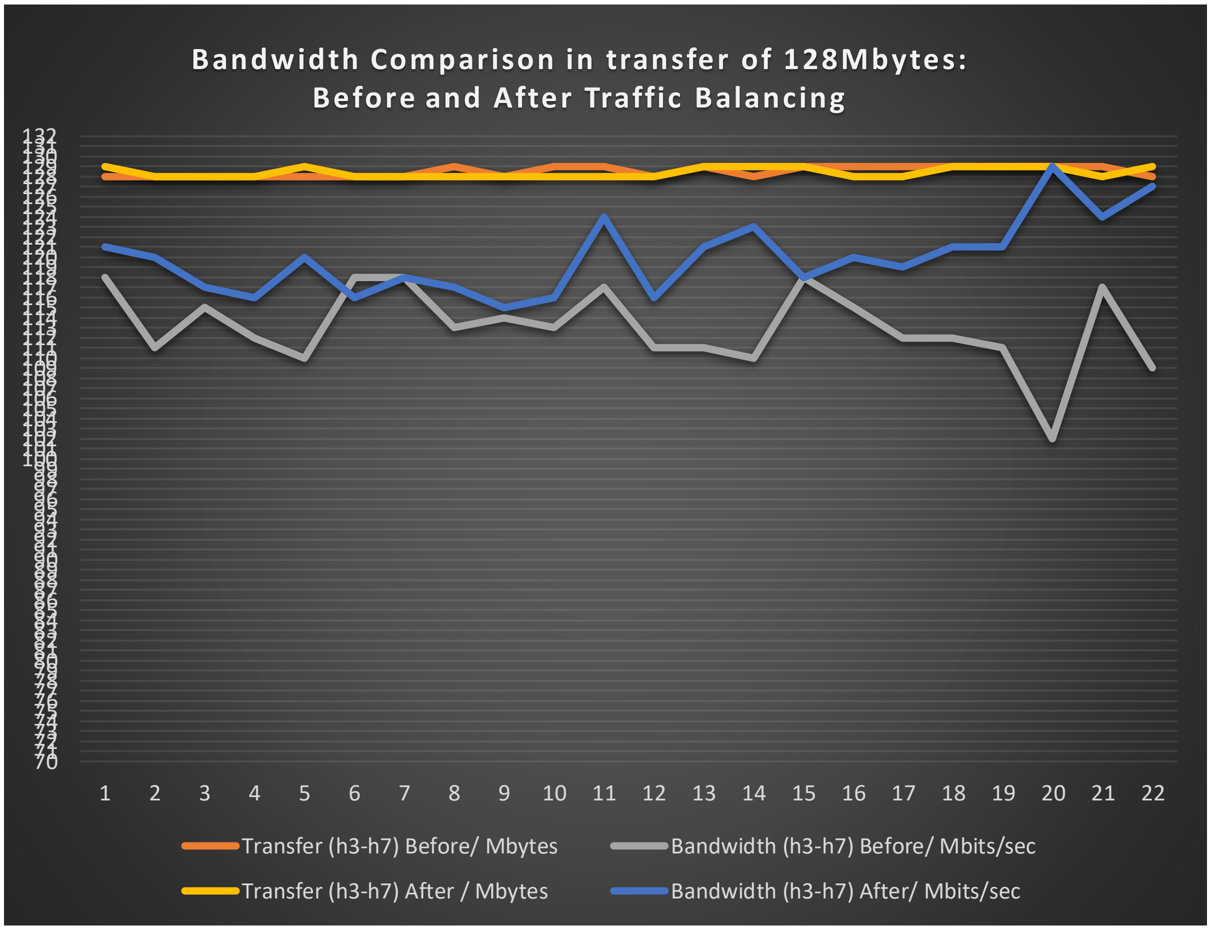
The paths from source to destination is determined with Dijkstra Algorithm. The utilization of the link as a ratio of the allowable bandwidth on the link is computed for each path. The path with least cost end-to-end is selected.


Packet flow
Host x to Host y
No
Periodic wait time to query network information
wait time up?
YES
Fetch network data
Fetch success?
Authentication success
Format Json or xml
no OFPT_ERROR
NO
YES
Link and port mapping (save info)
Device/Host IP, Device/Host Mac
Link ports: (source and destination)
Link port statistics
NO
Select path with Least load link cost
YES
All shortest path found?
Dijkstra Algorithm:
1. Compute all shortest paths from Host x to Host y
2. Determine load link for each path
load on link = Current traffic on link (Ssrc, Starget) / Max bandwidth of link (Ssrc, Starget)
Ssrc – The source host switch
Starget – target (destination) host switch
Periodic wait time to To push path change flow
Wait time up?
YES
Generate curl script for new changes, push flows
NO

Network device information and port statistics is fetched from the data plane. The format is either json or xml. The link port mapping of connected host, device, mac, IP address and link port statistics are saved to be compared with the next json or xml file fetched in the periodic network query. Extended Dijkstra algorithm is used to determine the available short paths from source host to destination host. The cost associated with each path is determined. The least cost path is selected and script generated to change OpenFLow tables.

4.1.3 Results

The end-to-end bandwidth is measured at an even interval with iperf from source host to destination host. A transfer of 128Mbytes over the link from source to destination is measured before and after running the algorithm to balance load path.



It is observed that, with transfer of 128Mbytes from host 3 to host 7, the bandwidth after traffic path balancing (load balancing) improved. Bandwidth utilization is improved with observation of live network data, observing, analyzing and modifying the path for traffic.

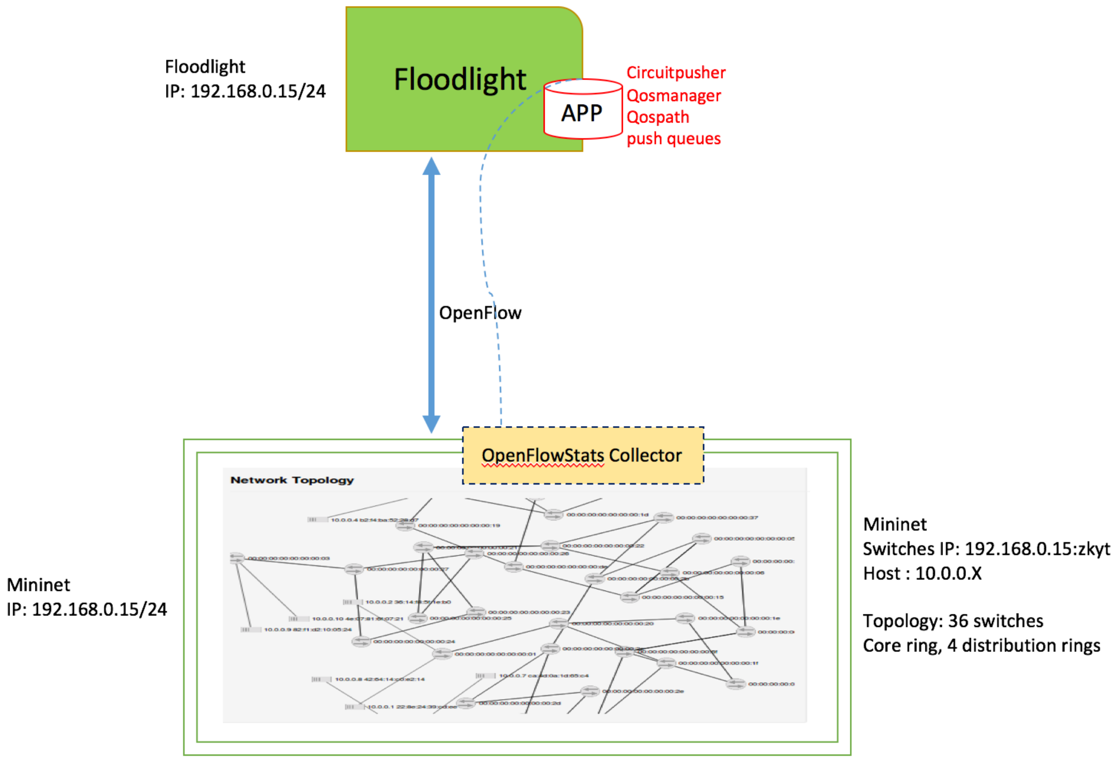
4.1.4 Improvement of CASE I

Case I implementation did not use deep learning algorithm. The prediction of best path can be further improved with sufficient network data that make use deep learning algorithm.

4.2 CASE II: Modifying Network Traffic Service & Policy

Description: The objective is to demonstrate dynamically changing traffic service and policy over an SDN network based on observed network information. In Case II demonstration, the controller is Floodlight-qos-beta. This is a fork() of floodlight (by Ryan Wallner) and has integrated Quality of Service and web user interface. The modification of the network QoS is done through modification of QoS applications programs which are part of Flodlight-qos-beta.

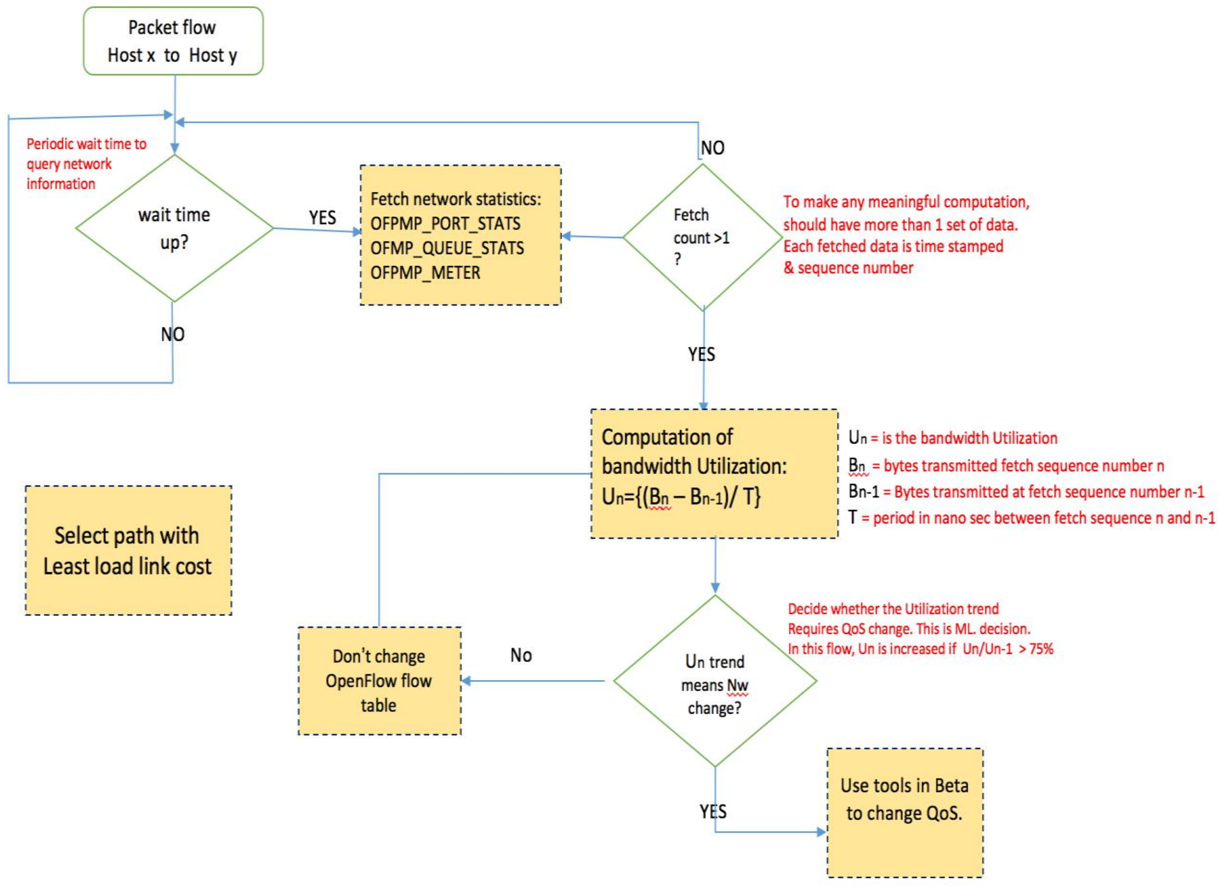
4.2.1 CASE I: Network Topology:



Mininet and Floodlight are run on the same virtual machine, 192.168.0.15/24.

4.2.2 Algorithm

OpenFlow statistics is fetched from the network (port, queue, meter, table or Group statistics) using multi part OpenFlow messaging. The statistics table fetched is dependent on the QoS parameter of interest. In this scenario, the aim is to improve link utilization by modifying the QoS settings on the flow path.

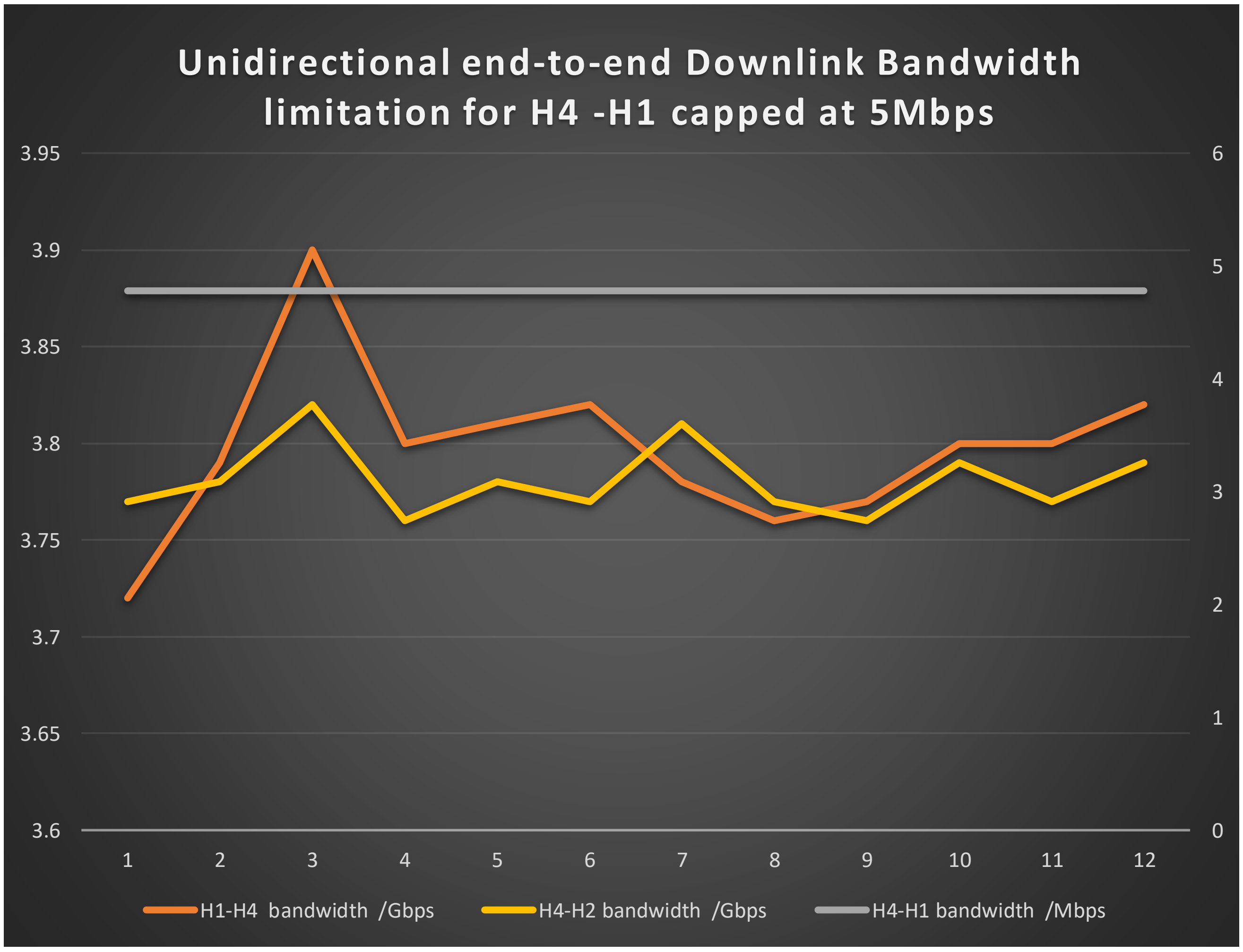


Packet flow Host x to Host y
Periodic wait time to query network information
wait time up?
YES
NO
Fetch network statistics: OFPMP_PORT_STATS OFMP_QUEUE_STATS OFPMP_METER
Fetch count >1 ?
NO
YES
To make any meaningful computation, should have more than 1 set of data. Each fetched data is time stamped & sequence number
Computation of bandwidth Utilization: Un={(Bn – Bn-1)/ T}
Un = is the bandwidth Utilization
Bn = bytes transmitted fetch sequence number n
Bn-1 = Bytes transmitted at fetch sequence number n-1
T = period in nano sec between fetch sequence n and n-1
Select path with Least load link cost
Decide whether the Utilization trend Requires QoS change. This is ML decision. In this flow, Un is increased if Un/Un-1 > 75%
Un trend means Nw change?
No
Don't change OpenFlow flow table
YES
Use tools in Beta to change QoS.

The multi part messaging reply body field has *duration_nsec* and *duration_sec* for time flow has been alive in the table. There is also *packet_count* and *byte_count* for number of packets in flow and number of bytes in flow respectively. The link utilization is computed with the byte count per port. If the successive utilization is increasing (say U_n/U_{n-1} is increasing at rate >=35%) or decreasing, QoS can be modified to accommodate this change. The bandwidth can also be limited for a an Host-to-Host flow in the downlink as demonstrated below.

4.2.3 Results

In this scenario, traffic policy is modified with flow of traffic from Host 4 to Host 1. Host 1 is server and uplink from H4 is capped at 5Mbps. However, the downlink H1-H4 is uncapped. The QoS policy affects only H4 to H1 as the bandwidth from H4 to H2 is unaffected.



4.2.4 Improvement of CASE II

Case II implementation did not use deep learning algorithm. Data was few and analyzed without an ML tool. The result can be improved with bandwidth, policy and service prediction based on sufficient data for deep learning.

Conclusion

The implementation key considerations for the design of and intelligent controller is the intelligent plane which, in this model, is suggested should be implemented with a Machine learning. Without the use of deep learning tool, simple programs aimed at adjusting service policy and traffic path is observed to give more control on the bandwidth utilization to the operator. This project is however, limited on a full implementation to demonstrate intelligent decision making by a controller. Equally, an attempt to use Mininet Wi-Fi (fork() of Mininet) was unsuccessful in simulating automatic radio parameter changes because it provides no API or IPC yet. A continuation of this project with scenario implementation of deep learning is recommended.

References

[1] I. Ahmad, S. Namal, M. Ylianttila and A. Gurtov, "Towards software defined cognitive networking," 2015 7th International Conference on New Technologies, Mobility and Security (NTMS), Paris, 2015, pp. 1-5. doi: 10.1109/NTMS.2015.7266528

[2] R. W. Thomas, D. H. Friend, L. A. Dasilva and A. B. Mackenzie, "Cognitive networks: adaptation and learning to achieve end-to-end performance objectives," in *IEEE Communications Magazine*, vol. 44, no. 12, pp. 51-57, Dec. 2006. doi: 10.1109/MCOM.2006.273099

[3] S. Namal I. Ahmad S. Saud M. Jokinen A. Gurtov "Implementation of OpenFlow based cognitive radio network architecture: SDN&R" in Wireless Networks pp. 1-15 2015 Springer.

[4] L. Zhang, T. Jiang, J. Shen, Z. Wang and Y. Cao, "Proportional fair scheduling based On primary user traffic patterns for spectrum sensing in cognitive radio networks," *2012 1st IEEE International Conference on Communications in China (ICCC)*, Beijing, 2012, pp. 302-306. doi: 10.1109/ICCChina.2012.6356897

[5] Ericsson Mobility Report (2016. June 3). Available: https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf June,

[6] Cisco VNI (2016, June 2). White paper: The Zettabyte Era—Trends and Analysis (Online). Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

[7] B. Naudts, M. Kind, F. J. Westphal, S. Verbrugge, D. Colle and M. Pickavet, "Techno-economic Analysis of Software Defined Networking as Architecture for the Virtualization of a Mobile Network," *2012 European Workshop on Software Defined Networking*, Darmstadt, 2012, pp. 67-72. doi: 10.1109/EWSDN.2012.27

[8] Cisco (2013). Connections counter: The internet of everything in motion. News blog: http://newsroom.cisco.com/feature-content?type=webcontent&articleId=1208342. Accessed on 22. Mar 2014.

[9] III, Joseph Mitola. "Chapter 5 - Cognitive Radio Architecture". Cognitive Radio Architecture: The Engineering Foundations of Radio XML. John Wiley & Sons. © 2006.Books24x7.<http://common.books24x7.com.login.ezproxy.library.ualberta.ca/toc.aspx?bookid=44319> (accessed Dec 16, 2017)

[10] Papadimitriou C. Algorithms, games, and the internet. In Proceedings of the thirty-third annual ACM symposium on Theory of computing 2001 Jul 6 (pp. 749-753). ACM.

[11] Jiang T. Cognitive Radio Networks: Efficient Resource Allocation in Cooperative Sensing, Cellular Communications, High-Speed Vehicles, and Smart Grid. [serial online]. 2015;Available from: Books24x7, Ipswich, MA. Accessed November 6, 2016. (pp. 1-4)

[12] Y. Jararweh, M. A. Ayyoub, A. Doulat, A. A. A. A. Aziz, H. A. B. Salameh and A. A. Khreishah, "SD-CRN: Software Defined Cognitive Radio Network Framework," 2014 IEEE International Conference on Cloud Engineering, Boston, MA, 2014, pp. 592-597.
doi: 10.1109/IC2E.2014.88

[13] A. Amaro, W. Peng and S. McClellan, "Bandwidth as a service: Transport layer bandwidth guarantees for cloud environments," 2016 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2016, pp. 487-492.
doi: 10.1109/ICTC.2016.7763519

[14] M. Siqueira et al., "An optical SDN Controller for Transport Network virtualization and autonomic operation," 2013 IEEE Globecom Workshops (GC Wkshps), Atlanta, GA, 2013, pp. 1198-1203. doi: 10.1109/GLOCOMW.2013.6825156

[15] Haleplidis, Evangelos, et al. *Software-defined networking (SDN): Layers and architecture terminology*. No. RFC 7426. 2015.

[16] IETF-86 Proceedings (2014, Nov 11) Applicability of Machine Learning to SDN[online]. Available: https://www.ietf.org/proceedings/86/slides/slides-86-sdnrg-7.pptx

[17] YuLing Chen, Prem Sankar, et al. (2016, September 28). OpenDaylight Machine Learning & Artificial Intelligence (MILA) Framework [Online]. Available: http://schd.ws/hosted_files/opendaylightsummit2016/9d/ODL%20MILA%20Framework.ppt

[18] Kaushik R. Chowdhury, Won-Yeol Lee, and Mehmet C. Vuran. Cognitive Radio Networks. Dec 2008, Chapter 1, pp 3 -35.

[19] Frédéric Firmin. (2012, March 6). The Evolved Packet Core (Online). Available: http://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core

[20] Fujitsu Network Communications Inc. (2015, January 30). The Benefits of Cloud-RAN Architecture in Mobile Network Expansion[Online]. Available: http://www.fujitsu.com/downloads/TEL/fnc/whitepapers/CloudRANwp.pdf

[21] Smiler. S, K. OpenFlow cookbook : over 110 recipes to design and develop your own OpenFlow switch and OpenFlow controller. pp 77-99

[22] Smiler. S, K. OpenFlow cookbook : over 110 recipes to design and develop your own OpenFlow switch and OpenFlow controller. pp 115-127

[23] Smiler. S, K. OpenFlow cookbook : over 110 recipes to design and develop your own OpenFlow switch and OpenFlow controller. pp 167-179

[24] Fontes R. Rothenberg C. (2015, May 28) Emulator for Software-Defined Wireless Networks (Online). Available: https://github.com/intrig-unicamp/mininet-wifi

[25] Ryan Wallner. (2012 Dec 13). Floodlight with QoS module and tools to manage QoS state in an OF network (Online). Available: https://github.com/wallnerryan/floodlight-qos-beta

[26] Nayan Seth. (2016, May 1) SDN Load Balancing (Online). Available: https://github.com/nayanseth/sdn-loadbalancing

APPENDIX

Code reference files for codes available on this shared drive:

https://drive.google.com/open?id=0Bw4lSwoV0yedS2JDLUptNEV5ZlU