UNIVERSITY OF ALBERTA

MINT 709
CAPSTONE
PROJECT REPORT

M-CORD:

MOBILE CENTRAL OFFICE RE-ARCHITECTED DATA CENTER

INSTRUCTOR: JUNED NOONARI

PRESENTED BY: RISHABH CHUGH

Table of Contents

ACKNOWLEDGEMENT 5

ABSTRACT 6

CHAPTER – 1 7

2ND GENERATION WIRELESS TECHNOLOGY 7

1.1 GSM (Global System for Mobile Communication) 7

1.1.1 INTRODUCTION 7

1.1.2 RADIO INTERFACE 7

1.1.3 NETWORK STRUCTURE 8

1.2 CDMAOne (IS-95) 9

1.3 D-AMPS (IS-136) 9

1.4 PERSONAL DIGITAL CELLULAR (PDC) 9

2.5G WIRELESS TECHNOLOGY 10

1.5 HSCSD 10

1.6 iDEN 10

1.7 GPRS 10

1.8 EDGE 10

CHAPTER - 2 11

3rd GENERATION WIRELESS TECHNOLOGY 11

2.1 CDMA 2000 and EV-DO 11

2.2 UMTS WCDMA 12

2.3 HSPA 12

CHAPTER - 3 14

4th GENERATION – LTE (LONG TERM EVOLUTION) 14

3.1 Key Requirements of LTE Design 14

3.2 Key Enabling Technologies and Features of LTE 16

3.2.1 Orthogonal Frequency Division Multiplexing (OFDM) 16

3.2.2 SC-FDE and SC-FDMA 18

3.2.3 Channel-Dependent Multi-user Resource Scheduling 18

3.2.4 Multi-antenna Techniques 19

3.2.5 IP-Based Flat Network Architecture 20

3.3 LTE Network Architecture 20

CHAPTER - 4 22

SOFTWARE DEFINED NETWORK (SDN) 22

4.1 INTRODUCTION 22

4.2 STATUS QUO IN NETWORKING 24

4.3 WHAT IS SOFTWARE - DEFINED NETWORKING? 25

4.3.1 Terminology 28

4.3.2 Alternative and Broadening Definitions 29
4.3.3 Standardization Activities 30
4.3.4 History of SDN 30
4.4 SOFTWARE - DEFINED NETWORKS: BOTTOM - UP 32
4.4.1 Layer I: Infrastructure 33
4.4.2 Layer II: Southbound Interfaces 35
4.4.3 Layer III: Network Hypervisors 37
4.4.4 Layer IV: Network Operating Systems/Controllers 41
4.4.5 Layer V: Northbound Interfaces 46
4.4.6 Layer VI: Language-Based Virtualization 48
4.4.7 Layer VII: Programming Languages 50
4.4.8 Layer VIII: Network Applications 54
CHAPTER – 5 62
CENTRAL OFFICE RE-ARCHITECTED 62
AS A DATACENTER (CORD) 62
5.1 INTRODUCTION 62
5.1.1 Commodity Hardware 63
5.1.2 Software Building Blocks 65
5.1.3 Transformation Process 66
5.2 Virtualizing Legacy Devices 66
5.2.1 Benefits and Challenges 67
5.2.2 Virtualizing the OLT 67
5.2.3 Virtualizing the CPE 68
5.2.4 Virtualizing the BNG 68
5.2.5 End-to-End Packet Flow 69
5.3 Service Framework 69
5.3.1 Benefits and Challenges 69
5.3.2 Scalable Services, Not Virtual Devices 70
5.3.3 Layers of Abstraction 71
5.4 Future Plans 72
CHAPTER – 6 74
M-CORD: MOBILE-CENTRAL OFFICE 74
RE-ARCHITECTED AS A DATA CENTER 74
6.1 INTRODUCTION 74
6.2 Overview of the CORD and M-CORD Architecture 74
6.3 USES 77
6.4 CONCLUSION 78
CHAPTER – 7 80

M-CORD Architecture for Traffic Offloading 80
CHAPTER – 8 84
NETWORK SLICE MANAGEMENT INSIDE M-CORD BASED 5G FRAMEWORK 84
8.1 SYSTEM COMPONENTS 85
8.1.1 Virtualized access network 85
8.1.2 Virtualized core network 85
8.1.3 Transport network slice management 86
CHAPTER – 9 87
NETWORK FUNCTION VIRTUALIZATION FOR HIGH-PERFORMANCE 5G 87
9.1 INTRODUCTION 87
9.2 RELATED WORK 88
9.2.1 Mobile Service Chaining preliminaries 88
9.2.2 Other related work 89
9.3 PROTOTYPING ENVIRONMENT 89
9.4 THE 5G CORE NETWORK PROTOTYPE 90
9.4.1 Implementation details 91
9.4.2 Evaluation of the prototype 91
9.5 HANDOVER 93
9.6 CONCLUSION 94
CONCLUSION 96
REFERENCES 97

ACKNOWLEDGEMENT

I owe many thanks to my mentor who helped and supported me during this project. Deepest thanks to Mr. Juned Noonari for his guidance, monitoring, constant encouragement and correcting various documents of mine with attention and care. He has taken the pain to go through the project and make necessary corrections as when needed, and I am very grateful for that.

Through this opportunity, I would like to express a deep sense of gratitude to Dr. Mike Macgregor and Mr. Shahnawaz Mir for their cordial support, valuable information and guidance, which helped me in completing this task through various stages.

I would also thank the University of Alberta and MINT Department without which this project and my master's degree would have been a distant reality.

ABSTRACT

Mobile is one of the astonishing technical achievement for human communication and information exchange in terms of performance, multimedia, speed, and connectivity it delivers. Connectivity is the essential establishment that provides a great mobile experience, which is powered by evolving mobile technologies such as 1G, 2G, 3G, 4G and now 5G. The 1G mobile technology merely used for analog voice communication; then with 2G, voice communication advanced to digital along with meager data rate for message and e-mail services. Mobile broadband, 3G was the drastic shift in the mobile industry with high data rates to provide multimedia experience to users. And to deliver faster and better real-time connectivity 4G LTE and LTE advanced was introduced.

But as the networking industry advances and the number of physical devices increases there is a need for mobile technology that can support IOT with regular high rates with increase bandwidth spectrum, 5G mobile technology provides a solution to the problem since there is no clear definition and way for the mobile carriers to deploy 5G services in the market.

M-CORD provides an open source reference solution to deploy 5G mobile networks, a cloud-native solution built on SDN, NFV and other cloud technologies. Developed on the CORD infrastructure platform lays the foundation for 5G networks and services.

CHAPTER – 1

2ND GENERATION WIRELESS TECHNOLOGY

The second generation is the first wireless networking technology which is digital, circuit-based and has a narrowband carrier but was only preferable for voice and fixed data communications. GSM, IS-95(CDMA One), IS-136(D-AMPS) and PDC were the technologies in 2G communication.

1.1 GSM (Global System for Mobile Communication)

1.1.1 INTRODUCTION

The world's most conventional 2G technology, applied in most of Europe and Asia founded in 1982, initially was GSM group (Groupe Special Mobile), later the acronym was only adopted from its French version. The technical rudiments defined in 1987, and it became marketable in 1991 with Radiolinja in Finland. 3GPP (3rd Generation Partnership Project) which primarily formed for 3G, but it took over maintenance and development of GSM.

1.1.2 RADIO INTERFACE

GSM employs TDMA (Time Division Multiple Access which allows several users to use the same frequency channel by dividing the signal into many different time slots.) with slow frequency hopping between duplex pair of channels. SDMA (Space Division Multiple Access methods which use the same frequency at the same time in different cells (spaces).) another channel access method used in GSM where cells using same frequency needs to be typically separated by two cells. FDMA (Frequency Division Multiple Access which allocates users one or several frequency bands) allows the receiver to discriminate among different frequency by tuning to a desiderated channel. The gaussian shift-keying modulation scheme is also employed to increase the battery life of the mobile station as it encodes data by modifying the frequency of the signal.

GSM network consists of four different cell size according to coverage area each cell operated according to a different environment:

Macro Cells are the ones where the base station antenna installed on the average rooftop above the building.

In **Micro Cells** the height of the antenna is under average roof top level, and they deployed in urban areas.

Whereas, **Pico Cells** are installed indoors and have a diameter in a few meters.

However, **Umbrella Cells** cover shadow region of smaller cells and fill gaps between those cells and are built on top of tall buildings or in high places.

The height of the antenna, antenna gain, and propagation conditions determines the radius of the cells varying from 100 meters to tens of kilometers. When the mobile station is at a greater distance (practically more than 35 km) from the base station, timeslot overlaps. Therefore extended cell specification used in which cell radius doubled or more by utilizing 2 or more timeslots per users.

GSM also supports Indoor coverage by deploying power splitters to provide RF signal from antenna outdoors to the indoor antenna distribution system, which built inside the buildings like airport or shopping complexes. Whereas, in suburban areas, in-building penetration of RF signal is used rather than deploying a separate indoor antenna system. [2]

1.1.3 NETWORK STRUCTURE

The network behind the GSM/GPRS system is a high-level and complicated architecture responsible for providing services to the customers. 3G and LTE systems have evolved from this basic architecture. [2] The GSM architecture subdivided as:

- **BASIC STATION SUBSYSTEM:** It has Mobile Stations which connected over the air interface to Base-Station transceiver (BTS), the Base-Station Controller (BSC), handles the mobility across the directly connected BTSs, and also transport the aggregated traffic from BTSs to the switching core.
- **NETWORK SWITCHING SUBSYSTEM:** This system consists of Mobile Switching Center (MSC) and subscriber databases. MSC is interconnected to PSTN (Public Switch Telephone Network) and provides needed switching to connect the dialer to the call receiver. The mobile subscriber location for control purposes is determined using the Home Location Register (HLR) and Visitor Location Register by MSC.

The GPRS system comprises of SGSN (Serving GPRS Support Node) and GGSN (Gateway GPRS Support Node), upgrading the GSM system (as in Figure 1.1) and replacing the BTS with PCU (Packet Control Unit) for managing data. SGSN

functionalities are similar to that of MSC as it provides location and mobility management services. GGSN connects the GPRS network to the internet and other IP networks. [1]

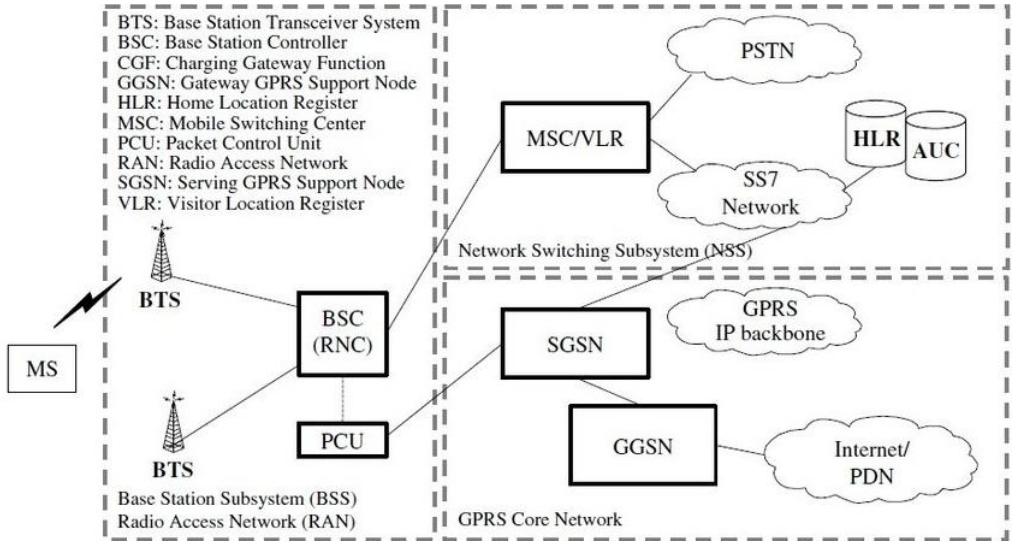


Figure 1.1 GSM Network Architecture

1.2 CDMAOne (IS-95)

The more efficient and high-quality 2G telecommunications standard is also known as TIA-EIA-95 developed by Qualcomm uses CDMA (Code Division Multiple Access) a channel access method where multiple users can simultaneously transmit information (stream of bits) over the single channel by employing spread spectrum technology and special coding scheme. Since it serves a large number of the user from a small number of sub-sites, so, CDMA-based technology has a significant economic advantage over TDMA or Frequency Division Multiplexing.

CDMA widely deployed in the USA, South Korea, Canada, Mexico, Israel, Australia, and China. [3] CDMA will be evolved to CDMA2000, WCDMA and I-CDMA in 2.5G and 3G standards.

1.3 D-AMPS (IS-136)

The technology primarily used by Cingular Wireless, AT&T Wireless and US Cellular was Digital extension of Advanced Mobile Phone System and superseded IS-54, by including features text messaging and data capabilities taken from GSM and CDMAOne. [2]

1.4 PERSONAL DIGITAL CELLULAR (PDC)

2G standard exclusively developed by NTT DoCoMo in Japan provided services like voice, supplementary services (call-waiting, voice mail, three-way calling, call forwarding), data services (up to 9.6 kbps CSD) and packet-switched services wireless data (up to 28.8 kbps PDC-P). [2]

2.5G WIRELESS TECHNOLOGY

It is a bridge between 2G and 3G wireless technologies. 2.5G uses 2G spectra and requires almost the same network infrastructure. [2] It has more bandwidth than 2G and less expensive than 3G. 2.5G wireless technology approaches are:

1.5 HSCSD

High-Speed Circuit Switched Data (HSCSD) is an upgrade of Circuit Switch Data over GSM (2G) networks. It provides data services at least 3 times (43.2 kbps on the fully deployed network) better than 2G. It uses multiple channels, allowing users to enjoy faster rates for internet, e-mail, calendar, and file transfer services.

1.6 iDEN

Integrated Digital Enhanced Network, developed by Motorola deploying enhanced compression and modulation technologies to deliver a data rate of 64kbps. It used in North America, South America, China, and Japan.

1.7 GPRS

General Packet Radio Service provides data speed of up to 171 kbps. It reallocates GSM timeslots from voice to data users, therefore decreasing voice rates but increasing data rates. Rolled out in China, Europe, and the US.

1.8 EDGE

Ericsson developed enhanced Data GSM Environment designed to provide data rates up to 384 kbps. It uses 3G transmission technology but 2G frequency range. [4]

CHAPTER - 2

3rd GENERATION WIRELESS TECHNOLOGY

Data applications began with the 2G systems because of a paradigm shift in circuit switching, it provided a significant increase in voice rates, but very ineffectual for data as could brace only tens of kbps. 3G cellular systems were a revelatory leap for 2G as not enhanced data rates with better voice capacity but also supported advanced services and applications like multimedia, and even envisioned for providing better Quality of Service (QoS). [1] Major 3G technologies discussed as follows:

2.1 CDMA 2000 and EV-DO

CDMA 2000 an evolution IS-95 standards was earlier worked upon by Qualcomm and CDMA group, the following official standardized process moved to 3GPP2 in 1999. The first evolution of IS-95 towards 3G was CDMA2000-1X, where 1X means it uses the same bandwidth as IS-95. Supplemented Channels (separate logical channels) enhanced data capabilities up to 307 kbps. It did not fulfill 3G requirement as referring to as a 2.5G system. Though CDMA2000-1X theoretically doubled the capacity of IS-95by adding 64 traffic channels to forward link orthogonal to the original set of 64. The uplink was enhanced through coherent modulation and downlink matched uplink by addition of (800Hz) power control. As CDMA2000 and IS-95A/B could be on the same carrier, so migration was facile.

CDMA2000-1X standard elaborated to CDMA2000-1X-EVDO (EVolution, Data only) to achieve ITU standardized data rates of up to 2Mbps, but it supported only data traffic. EV-DO designed as an HDR (High Data Rate) solution for nomadic application meeting the 2Mbps low mobility requirements of IMT-2000. It was then upgraded to full-mobility requirements and became the first system to provide real broadband-like speed to mobile subscribers. It was deployed in 2002, three years ahead of HSDPA, a similar system used by GSM operators.

EV-DO, an asymmetric system provided a downlink speed of 2.4 Mbps and an uplink rate of 153 kbps. The downlink was implemented using a TDMA link. The system also supported QPSK and 16QAM modulation and coding rates, which caused variation in data speed from 38.4kbps to 2457.6kbps. EV-DO Rev. A was the advancement of EV-DO which improved data rates to 3.07Mbps and 1.8Mbps in downlink and uplink, respectively with more of the symmetric link. [1]

2.2 UMTS WCDMA

Universal Mobile Telephone Service, a 3G system has ETSI developed an evolution of IMT-2000. 3GPP manages UMTS is a solution for the countries that deployed GSM, basically in Europe. [5] It was considered as an integrated solution for mobile data and voice in a wide area, by offering a data rate of up to 384 kbps in high mobility conditions and up to 2mbps in fixed environments.

The infrastructure for UMTS include:

(1) A core network, for switching, routing and subscriber management.
(2) The UTRAN (UMTS Terrestrial Radio Access Network)
(3) The UE (user equipment).

Though the architecture is similar to that of GSM/GPRS where BTS becomes Node-B, BSC is RNC (Radio Network Controller), the NSS is CN, and the MS is UE.

While the UMTS had a similar architecture as that of GSM/GPRS, the W-CDMA (Wide-band CDMA) is a radical departure of 2G air-interface, evolved from IS-95. Direct Sequence Spread Spectrum CDMA provided channelization, synchronization, and scrambling by multiplying user data with pseudo-random codes. W-CDMA employed by FDD and TDD techniques, however, FDD was more widely used. It operated on 5MHz bandwidth, which supported almost 100 simultaneous voice calls, with a peak data rate of varying from 384 kbps to 2048 kbps. When compared to CDMA2000, W-CDMA supported multi-code use by the single subscriber to enhance data speed, a wider choice of spreading factors and diversity in transmission by applying Altamonte space-time coding. [1] [6]

2.3 HSPA

High-Speed Packet Access is a software upgrade of UMTS-WCDMA by 3GPP is a combination of HSDPA (High-Speed Downlink Packet Access) and HSUPA (High-Speed Uplink Packet Access). HSDPA was initially deployed by AT&T in 2005 and later became famous worldwide [5].

Since the trend demanded higher throughput on download, so 3GPP UMTS improved the downlink (HSDPA) by defining new transport channel also known as High-Speed Downlink Shared Channel (HS-DSCH) that provided peak theoretical data speed of up to 14.4 Mbps. Unlike W-CDMA, HSDPA uses 16 Walsh codes, out of which 15 managed user traffic. A single subscriber could use 5, 10, or 15 codes to get higher throughputs, even though, practically HSDPA provided the user with throughputs varying in between 500 kbps to the 2Mbps range.

To achieve better throughput and capacity [7], [8], HSPA applied various new techniques:

- **Advanced Modulation and Coding (AMC):** Different modulation such as QPSK and 16QAM **and** rate ¼ through rate 1 coding. The modulation and coding schemes vary as per user and per frame depending on the channel quality of downlink.
  For each user link, highest modulation and coding technique are assigned to support under given signal to interference condition by maximizing throughput and capacity of the system. This selection is measured by Channel Quality Indicator (CQI) by the report presented by the HSDPA mobiles to the base station.

- **Fast Dynamic Scheduling:** HSDPA systems ensured that they are working at a highest possible rate, the scheduler exploits the assortment channel conditions of the distinct user at a discrete time by scheduling the delivery of packets to synchronize with fading peaks and avoid scheduling during troughs. The dynamic scheduler could also assign the whole cell capacity to a user when conditions are appropriate. Hence, it increases the system capacity and transcends the utilization of resources. In HSDPA, the scheduler located at Node-B instead of RNC as in W-CDMA.

- **Hybrid Automatic Repeat Request (H-ARQ):** A retransmission technique that needed as delays and inaccuracy in channel quality feedback could cause errors in link adaption and can correct through ARQ, but multiple retransmission could lead to more delays, so, this method softly combines multiple inaccurate transmissions to retrieve from errors quickly. It termed as chase combining. HSDPA mobile system also features incremental redundancy in which additional error-correction coding is encapsulated with subsequent retransmission to enhance error-free reception.

  HSUPA, an Enhanced Uplink, has a new uplink channel called the Enhanced Dedicated Channel (E-DCH) to UMTS-WCDMA that features multi-code transmission, H-ARQ, short transmission time interval, and fast scheduling to support uplink throughput of up to 5.8Mbps, with solid offering to the subscriber in range of 500kbps-1Mbps. Such rates enable application such as VoIP, uploading images and videos, and sending massive e-mails. [1]

CHAPTER - 3

4th GENERATION – LTE (LONG TERM EVOLUTION)

The rapid growth in the adoption of fixed-line broadband around the world, the mobile community, assimilated the need to build a mobile broadband system that is proportional with DSL and competent enough to support the rapid growth in IP traffic. During 2005, two groups within 3GPP started working on a standard to hold the expected massive increase in IP data traffic. The Radio Access Network (RAN) group began work on the Long Term Evolution (LTE) project, and the Systems Aspects group initiated work on the Systems Architecture Evolution (SAE) project. The LTE group developed a new radio access network called Enhanced UTRAN (E-UTRAN) as an evolution to the UMTS RAN. On the other hand, the SAE group came up with all new IP packet core network architecture called the Evolved Packet Core (EPC). Together, EUTRAN and EPC are formally called the Evolved Packet System (EPS) [1].

3.1 Key Requirements of LTE Design

LTE was designed with the following objectives in mind to effectively meet the growing demand [9].

- **Performance on Par with Wired Broadband**: The primary objective of LTE was to make mobile Internet experience as good as or better than that achieved by residential wired broadband access systems deployed today by important network performance parameters that enhance user experience are high throughput and low latency.

  To achieve high throughputs, the peak data rate targets were set to be at 100Mbps and 50Mbps for the downlink and uplink, respectively by 3GPP. It is greater order of magnitude better than 3G systems. In addition to peak data rates, the LTE design goal was to achieve an average downlink throughput at least 3–4 times enhanced than that of the original HSPA and an average uplink throughput that is 2–3 times better.

  It also stipulated that these higher data rates achieved by making a 2–4 times improvement in spectral efficiency. LTE requirements also raised the cell edge bit rate while maintaining the same site locations as deployed today.

  The network latency was kept very low to enable support for delay-sensitive applications like voice and interactive gaming. The target round-trip latency for the LTE radio network was to be less than 10ms, better than the 20–40ms delay observed in many DSL systems.

Also, LTE also aims to reduce latency associated with control plane functions like session setup. Improving QoS capabilities to support a variety of applications was also essential.

Along with that LTE aimed for performance parity with wired broadband systems, it did so while simultaneously elevating the requirements on mobility.

- **Flexible Spectrum Usage:** The frequency band and amount of spectrum owned by distinct mobile operators around the world are contracting. As many LTE deployments likely to be in the re-architected spectrum that currently used for 3G or 2G services, the amount of spectrum that could be made available for LTE would also depend on how rapidly individual operators wished to migrate to LTE. To be a global standard and to make it fit for deployment by a wide variety of operators, 3GPP mandated a high degree of spectrum flexibility.

  Operators could deploy LTE in 900MHz, 1800MHz, 700MHz, and 2.6GHz bands. LTE supported a variety of channel bandwidths: 1.4, 3, 5, 10, 15, and 20MHz. It also ordained that end-user devices able to operate at all the channel bandwidths lower than their maximum capability; for example, a 10MHz mobile device will support all bandwidths up to 10MHz. The smaller 1.4MHz and 5MHz channels optimized for GSM and CDMA reframing to hold deployments where operators were unable to free more substantial amounts of spectrum. Both frequency division duplexing (FDD) and time division duplexing (TDD) to accommodate paired as well as unpaired spectrum allocations supported by LTE.

- **Co-existence and Interworking with 3G Systems as well as Non-3GPP Systems**: A large number of existing mobile subscribers, it is an essential requirement that LTE networks interwork seamlessly with existing 2G and 3G systems. Many cellular companies were likely to phase in LTE over a while with initial deployments made in areas of high demand such as urban cores. Service continuity and mobility—handoff and roaming—between LTE and existing 2G/3G systems are critical to obtaining an absolute customer experience. As LTE aimed at high global standard attractive to a variety of operators, interworking requirement extended to non-3GPP systems such as the 3GPP2 CDMA and WiMAX networks. Moreover, to facilitate fixed-mobile convergence, interworking requirements applied to all IP networks including wired IP networks.

  - **Reducing Cost per Megabyte:** To reduce the growing gap between wireless data consumption and revenue, it was essential that substantial reductions be achieved in the total network cost to deliver data to end users. 3GPP realized this issue and made reducing the cost per megabyte of data a key design criterion for LTE. Some design criteria were tied directly to cost efficiency. These include: High-capacity, high-spectral air-inter-effective ace

- Capable enough to deploy in the subsisting spectrum and reuse cell sites and transmission equipment.
- Interworking with bequest systems for cost-effective migration
- Interworking with non-3GPP systems for one global standard to achieve higher economies of scale
- An architecture with fewer network components and protocols
- A unique IP packet core for voice and data
- IP architecture to capitalize colossal development community and boost economies of scale through convergence with wired communication systems
- Support for lower-cost Ethernet-based networks
- Provide support for Base stations with lower power and space requirements; could in many cases be put inside existing base station cabinets or mounted beside them

- Provide a platform for self-configuring and self-optimizing network and technologies to minimize installation and management cost

3.2 Key Enabling Technologies and Features of LTE

Service and performance requirements and essentials achieved, LTE design incorporates several important enabling radio and core network technologies [10–12]. A brief introduction to some of the essential enabling technologies applied in the LTE design.

3.2.1 Orthogonal Frequency Division Multiplexing (OFDM)

The critical factor that distinguishes 3G systems and LTE is the use of Orthogonal Frequency Division Multiplexing (OFDM) as the underlying modulation technology. 3G systems such as UMTS and CDMA2000 deployed worldwide are based on Code Division Multiple Access (CDMA) technology.

CDMA used a narrowband signal over a wider bandwidth to achieve interference resistance and performs outland shingly well for low data rate communications such as voice, where a large number of subscribers can be multiplexed to obtain high system capacity. However, for high-speed applications, CDMA becomes not defensible as large bandwidth is required to attain useful amounts of spreading.

OFDM emerged as a technology of choice for achieving high data rates. It is the core technology used by a variety of systems including Wi-Fi and WiMAX. The edge of OFDM over CDMA led to its selection for LTE:

- **An elegant solution to multipath interference:** The challenge to high bit-rate transmissions in a wireless channel is inter symbol interference (ISI) caused due to multipath. In a multipath environment, when the time delay between the various signal paths is an essential factor of the transmitted signal's symbol period, a transmitted symbol may reach at the receiver during the next symbol and lead to inter symbol interference (ISI). At higher data rates, the symbol time is less; hence, it only takes a short delay to cause ISI, which poses a challenge for broadband wireless. OFDM is a multicarrier modulation technique that overcomes the difficulty elegantly. The core idea behind multicarrier modulation is to divide high-bit-rate data stream into many parallel lower bit-rate streams and modulate each stream on separate carriers—often called subcarriers, or tones. Disassociating the data stream into many parallel streams boosts the symbol duration of each stream such that the multipath delay spread is only a small fraction of the symbol duration. OFDM is a spectrally effective method of multicarrier modulation, where the subcarriers selected in a way that they are all orthogonal to one another over the symbol duration, hence, to avoid non-overlapping subcarrier channels to eliminate inter-carrier interference. In OFDM, any residual inter symbol interference can also be removed by the use of guard intervals between OFDM symbols that are greater than the expected multipath delay. By making the guard interval more significant than the expected multipath delay spread, ISI can eliminate. Addition of a guard interval, however, causes power wastage and a decrease in bandwidth efficiency.

- **Reduced computational complexity:** Fast Fourier Transforms (FFT/IFFT) can be used to implement OFDM, and the computational requirements develop slightly greater than linearly with data rate or bandwidth. The computational complexity of OFDM represented by O($BlogBT_m$) where B is the bandwidth and T_m is the delay spread. The complexity due to this is quite lower than that of a time-domain equalizer-based system—the traditional

method for combating multipath interference—which has a complexity of $O(B^2T_m)$. Reduced complexity is especially alluring in the downlink as it simplifies receiver processing and thus decreases mobile device cost and power consumption. It is significant concerning wide transmission bandwidths of LTE coupled with multi-stream transmissions.

- **Graceful degradation of performance under excess delay:** As the delay spread exceeds the value it is designed for it causes the performance of an OFDM system to degrade gracefully. Substantial coding and low constellation sizes can be implied to provide fallback rates that are effectively more robust against delay spread. It stated that OFDM is well suited for adaptive modulation and coding, which contains the system to make the best use of the available channel conditions. The difference can be noticeable with the abrupt degradation which outstands error propagation that single-carrier systems experience because of the exceeding value of the delay spread.

- **Exploitation of frequency diversity:** OFDM eases coding and interleaving across subcarriers in the frequency domain, which increases robustness against burst errors caused due to portions of the transmitted spectrum undergoing deep fades. The channel bandwidth to be scalable with the use of OFDM without having any impact on the hardware design of the base station and the mobile station. It is a way that allows LTE to deploy in a variety of spectrum allocations and distinct channel bandwidths.

- **Enables efficient multi-access scheme:** OFDM can apply as a multi-access scheme by segmenting different subcarriers among various users, and the scheme is referred to as OFDMA and exploited in LTE. OFDMA tender the ability to get fine granularity in channel allocation, which can be utilized to achieve useful capacity enhancements, particularly in slow time-varying channels.

- **Robust against narrowband interference:** Only a few of the sub-carriers are affected by the narrowband interference because of the robustness of the OFDM

- **Suitable for coherent demodulation:** OFDM systems make it easy to do pilot-based channel estimation, which delivers them ideal for consistent demodulation schemes that are more power efficient.

- **Facilitates use of MIMO:** MIMO described as multiple input multiple outputs is a signal processing technique that use multiple antennas at both the transmitter and receiver to improve the performance of the system. The MIMO techniques could be effective, if the channel conditions are in a way that the multipath delays do not lead inter symbol interference or we can say that, the channel should be a flat fading channel and not a frequency selective one.

- At high data rates, this not implied, and thus MIMO techniques do not apply well in traditional broadband channels. So, we use OFDM, which converts a frequency selective broadband channel into numerous narrowband flat fading channels which makes the MIMO models and techniques work well. The effectiveness that MIMO techniques provide helps to improve system capacity which in a way gives OFDM ahead over other technologies and is the main reason for its choice. MIMO and OFDM have already been combined and applied in Wi-Fi and WiMAX systems.

- **Efficient support of broadcast services:** An OFDM network as a single frequency network (SFN) can operate by synchronizing base stations to timing errors well within the OFDM guard interval. It in a way allows broadcast signals from distinct cells to combine over the air to effectively enhance the received signal power, leading to higher data rate broadcast transmissions for a given transmit power. LTE design thus implies the OFDM capability to improve efficiency in broadcast services.

Though all these advantages drove 3GPP to adopt OFDM as their modulation choice, it should consider that OFDM also has a few disadvantages. The highlighted among these is the OFDM signals having a high peak-to-average ratio (PAR), which is the reason for non-linear ties and clipping distortion when passed through an RF amplifier. To mitigate this problem requires the use of expensive and in-efficient power amplifiers with high requirements on linearity, which enhances the cost of the transmitter and power wastage.

While the increased amplifier costs and power in-efficiency of OFDM are tolerated in the downlink as part of the design, for the uplink LTE selected a variation of OFDM that has a lower peak-to-average ratio. The modulation of choice for the uplink is called Single Carrier Frequency Division Multiple Access (SC-FDMA).

3.2.2 SC-FDE and SC-FDMA

LTE incorporated a power efficient transmission scheme for the uplink in order to increase the battery life and reduce the cost. Single Carrier Frequency Domain Equalization (SC-FDE) which conceptually similar to OFDM but instead of transmitting actual data symbols using the Inverse Fast Fourier Transform (IFFT), the data symbols sent as a sequence of QAM symbols with a cyclic prefix added, and the IFFT added at the end of the receiver. SC-

FDE retains all the advantages of OFDM such as multipath resistance and low complexity while having a low peak-to-average ratio of 4-5dB. SC-FDMA the multiuser version of SC-FDE is implemented in the uplink of LTE, which allows multiple users to use parts of the frequency spectrum. SC-FDMA almost resembles OFDMA and can be thought of as "DFT pre-coded OFDMA." SC-FDMA also preserves the PAR properties of SC-FDE even though it increases the complexity of the transmitter and the receiver.

3.2.3 Channel-Dependent Multi-user Resource Scheduling

LTE gets enormous flexibility in how the use of the OFDMA scheme allocates channel resources. It is possible to design algorithms to allocate resources flexibly and dynamically to meet arbitrary throughput, delay, and other requirements as OFDMA permits allocation in both time and frequency. Channel-dependent scheduling that enhances overall system capacity is dynamically supported.

It assumed that each user would be experiencing uncorrelated fading channels; it is possible to assign subcarriers among users such that the overall capacity increased. This technique, called frequency selective multiuser scheduling, focuses on transmission power in each user's best channel portion, hence uplifting the total ability. Frequency selective scheduling needs channel tracking and is generally only viable in slow varying channels.

The potential capacity gains for the involved overheads negated for fast varying channels. OFDMA allows frequency selective scheduling that combined with multi-user time domain scheduling, which then schedules users during the crests of their fading channels. The

modulation and coding adapted for capacity gains for the instantaneous signal-to-noise ratio conditions for each user subcarrier. OFDMA is a way to achieve frequency diversity for high-mobility users. The signal can be made more robust against frequency selective fading or burst errors, by coding and interleaving across subcarriers in the frequency domain using a uniform random distribution of subcarriers over the whole spectrum. Control signaling and delay sensitive services best provided by frequency diverse scheduling.

3.2.4 Multi-antenna Techniques

The LTE standard provides extensive support for implementing advanced multi-antenna solutions to improve link robustness, system capacity, and spectral efficiency. Depending on the deployment cases, various techniques used. Multi-antenna techniques supported in LTE include:

- **Transmit diversity:** A technique applied to skirmish multipath fading in the wireless channel. It sends copies of the same signal, coded differently, over multiple transmit antennas. LTE transmit diversity applies Space-frequency block coding (SFBC) techniques complemented with frequency shift time diversity (FSTD) when four transmit antenna used. Transmit diversity contemplated for common downlink channels that cannot make use of channel-dependent scheduling. Its applications also help user transmissions such as during low data rate VoIP, where the additional overhead of channel-dependent scheduling may not excuse. Transmit diversity enhances system capacity and cell range.

- **Beamforming:** Multiple antennas in LTE can be used to transmit the same signal appropriately weighted for each antenna element to focus the transmitted beam in the direction of the receiver and away from interference, thus enriching the received signal-to-interference ratio. Beamforming provides eloquent improvements in coverage range, capacity, reliability, and battery life. It plays a significant role in delivering angular information for user tracking. LTE supports Beamforming provides support in LTE downlink.

- **Spatial multiplexing:** Spatial multiplexing in which multiple independent streams are transmitted in parallel over multiple antennas and separated at the receiver using multiple receive chains through the use of signal processing. It is done because the multipath channels as seen by the different antennas sufficiently decorrelated that would be the cause of scattering rich environment. Theoretically, spatial multiplexing provides data rate and capacity gains proportional to the number of antennas used.

  It is best suitable under good SNR, and light load conditions and hence tend to have a more pronounced effect on peak rates rather than overall system capacity. LTE standard supports spatial multiplexing with up to four transmit and receiver antennas respectively.

- **Multi-user MIMO:** The complexity and cost considerations of spatial multiplexing due to multiple transmit chains withdraws support for the uplink. Instead, multi-user MIMO (MU-MIMO) used, which allows multiple users in the uplink, each with a single antenna, to transmit using the same frequency and time resource.
  The signals received from the distinct MU-MIMO subscribers are separated at the base station receiver using accurate channel state information of each subscriber which obtained through uplink reference signals that are orthogonal between subscribers.

3.2.5 IP-Based Flat Network Architecture

Along with the air-interface, the other radical aspect of LTE is the flat radio and core network architecture [13]. Fewer nodes and a reduced hierarchical structure for the LTE network defines the word "Flat." A flat architecture means fewer nodes implies a lower infrastructure cost and lower latency requirements. Thus, leading to fewer interfaces and protocol-related processing, and reduced interoperability testing, which further brings the value of the development and deployment down. Fewer nodes also optimize of the radio interface, merging of some control plane protocols, and short session start-up time.

The 3GPP network architecture has evolved over a few releases. 3GPP Release 6 architecture, which is conceptually very similar to its predecessors, has four network elements in the data path: the base station or Node-B, radio network controller (RNC), serving GPRS service node (SGSN), and gateway GPRS service node (GGSN). With the Release 7, 3GPP introduced a direct tunnel option from the RNC to GGSN, which eliminated SGSN from the data path. LTE, on the other hand, will have only two network elements in the data path: the enhanced Node-B or eNode-B, and a System Architecture Evolution Gateway (SAE-GW). In this architecture, LTE merges the base station and radio network controller functionality into a single unit. A functional entity called the Mobility Management Entity (MME) which includes controls path provides control plane functions related to subscriber mobility and session management. The MME and SAE-GW could collocate in a single entity called the access gateway (a-GW).

The importance of LTE flat architecture is that all services, including voice, are supported using IP protocols that are being on the IP packet network. The previous generations of systems, which used separate circuit-switched sub-network for supporting voice with their own Mobile Switching Centers (MSC) and transport networks, LTE envisions to use only a single evolved packet-switched core, the EPC, over which all services are supported, thus in a way reduces huge operational and infrastructure cost. However, it should be noted that although LTE was designed for IP services with a flat architecture, there are still non-IP aspects of the 3GPP architecture, such as the GPRS tunneling protocol and the PDCP (packet data convergence protocol) within the LTE network architecture due to backward compatibility reasons.

3.3 LTE Network Architecture

3GPP Release 8 presented the core network design to support LTE known as Evolved Packet Core (EPC). EPC designed to provide a high-capacity, all IP, reduced latency, flat architecture that significantly reduces cost and hands advanced real-time and media-rich services with high quality of experience. It not only extended support to new radio access networks such as LTE but also provide interworking with legacy 2G GERAN and 3G UTRAN networks connected via SGSN. Functions of the EPC include access control, packet routing and transfer, mobility management, security, radio resource management, and network management.

The EPC includes four new elements:

(1) Serving Gateway (SGW), which is a terminal interface for the 3GPP radio access networks;

(2) Packet Data Network Gateway (PGW), which enables IP data services, perform routing, allocates IP addresses, enforces policy, and provides access for non-3GPP access networks;

(3) Mobility Management Entity (MME), which identifies as well as authenticates and authorizes users and provide support for user equipment context;

(4) QoS aspects were looked upon by Policy and Charging Rules Function (PCRF),.

The end-to-end architecture including how the EPC supports LTE as well as current and legacy radio access networks. Describing briefly each of the four new elements is provided here:

- **Serving Gateway (SGW):** The SGW is a circumscribe point between the RAN and core network and manages user plane mobility. When terminals move across areas served by different eNode-B elements in E-UTRAN, as well as across other 3GPP radio networks such as GERAN and UTRAN it performs as the mobility anchor. Initiation of network-triggered service request procedures and downlink packet buffering is done by SGW. Various other functions include lawful interception, packet routing, and forwarding, transport level packet marking in the uplink and the downlink, accounting support for per user, and inter-operator charging.

- **Packet Data Network Gateway (PGW):** The PGW is the termination point of the EPC toward other Packet Data Networks (PDN) such as the Internet, private IP network, or the IMS network providing end-user services. An anchor points for sessions toward external PDN and provides functions like user IP address allocation, policy enforcement, packet filtering, and charging support. Operator-defined rules for resource allocation to control data rate, QoS, and usage are defined in policy enforcement. Deep packet inspection for application detection is performed by packet filtering.

- **Mobility Management Entity (MME):** The signaling and control functions to manage the user terminal access to network connections, assignment of network resources, and mobility management function such as idle mode location tracking, paging, roaming, and handovers are performed by MME. It also controls all control plane functions related to subscriber and session management. The security functions which include, providing temporary identities for user terminals, interacting with Home Subscriber Server (HSS) for authentication, and negotiation of ciphering and integrity protection algorithms. Selection of the appropriate serving and PDN gateways and legacy gateways for handovers to other GERAN or UTRAN networks are the responsibilities of MME. Moreover, MME is the point where lawful interception of signaling is made. Along with that MME manages thousands of eNode-B elements, which is one of the key attributes that distinguish from 2G or 3G platforms using RNC and SGSN platforms.

- **Policy and Charging Rules Function (PCRF):** An interconnection of the Policy Decision Function (PDF) and Charging Rules Function (CRF). The PCRF interfaces with the PDN gateway and provides a base for service data flow detection, policy enforcement, and flow-based charging. The PCRF was actually defined in Release 7 of 3GPP ahead of LTE. Even though not much deployed with pre-LTE systems, it is mandatory for LTE. Release 8 further enhanced PCRF functionality to include support for non-3GPP access (e.g., Wi-Fi or fixed line access) to the network. [1]

CHAPTER - 4

SOFTWARE DEFINED NETWORK (SDN)

4.1 INTRODUCTION

Information, in the form of digital packets, travels around the world through technologies are performed by the distributed control and transport network protocols running inside the routers and switches. Even though they are worldwide adopted, traditional IP networks are complex and hard to manage [14]. Each individual network devices are configured separately using low-level and often vendor-specific commands in order to express the desired high-level network policies by the network operators. Along with the complexity in configuration, network environments have to endure the dynamics of faults and adapt to load changes. The current IP networks do not allow automatic reconfiguration and response mechanisms. Therefore, it is highly challenging to enforce the required policies in such a dynamic environment.

The problem worsens when current networks are also vertically integrated. The networking devices are stacked with the control plane (that makes a decision of traffic administration) and the data plane (that supervenes the forwarding directives of the control plane), which creates hindrance in flexibility, innovation, and evolution of the networking infrastructure. Ultimately, the capital and operational expenses of running an IP network have been inflated lately because of the inertia the traditional IP networks hold, which causes a delay in new routing protocol to be fully designed, evaluated, and deployed. which is the major factor for delay in the shift from IPv4 to IPv6, as this daunting task is simply not feasible in practice due to clean state approach of IP architecture [15], [16].

Software-defined networking (SDN) [17], an emerging paradigm shift in the networking industry, [18] gives hope to change the limitations of traditional network infrastructures by separating and fragmenting the vertical integration of network's control logic from the underlying routers and switches that forward the traffic.

Secondly, it separates the control and data planes, so that causes network switches to simply become forwarding devices and so the control logic is implemented in a logically

centralized controller (or network operating system[1]), thus in a way simplifying policy enforcement and network (re)configuration and evolution [19]. It is significant to highlight that a logically centralized programmatic model does not hypothesize a physically centralized system [20]. In fact, such a solution would be averted because of the need to guarantee adequate levels of performance, scalability, and reliability. Instead, physically distributed control planes [20], [21] resorts the production-level SDN network designs.

A programming interface that is well defined between the switches and the SDN controller separates the control plane and the data plane. The controller has a well-defined application programming interface (API) that exercises direct control over the state in the data plane elements. The profound example of such an API is OpenFlow [22], [23]. One or more tables that showcase packet- handling rules (flow table) are in OpenFlow switch. Each rule defines a subset of the traffic and certain actions (dropping, forwarding, modifying, etc.) are performed on the traffic, and it has certain rules that controller can follow to perform roles of a general middle-box like a switch, router, firewall, load balancer, traffic shaper.

SDN principles define network policies, their implementation in switching network and rules for forwarding traffic to address such concerns separately. Creation and introduction of new abstractions in networking, simplification of network management and facilitation of network evolution and innovation are made easier because of this separation, which has led to the achievement of desired flexibility, breaking the network control problem into tractable pieces.

It is known that SDN and OpenFlow started as academic experiments [22], lately, over the past few years, they gained important traction in the industry. Due to this most vendor of commercial switches has now started including support of the OpenFlow API in their equipment. Open Networking Foundation (ONF) [23] with the main goal of promotion and adoption of SDN through open standards development got funded by the big giants like Google, Facebook, Yahoo, Microsoft, Verizon, and Deutsche Telekom due to the momentum created by SDN. Google, for example, has interconnected its data centers across the globe by deployed an SDN. Deployment of this production network is for three years, thus helping the company to improve operational efficiency and significantly reduced cost.

VMware's Network virtualization platform, NSX [24], is another example. NSX is a commercial solution that is entirely based around SDN principles and is responsible for delivering a fully functional network in software, and underlying networking devices are provisioned independently. The world's largest IT companies have recently joined SDN fellowships such as the ONF and the Open Daylight initiative [25], that proves it's significance and another indicator that highlights the importance of SDN from an industrial perspective

Open-flow has simplified three-layer stack that is presented as High-level network services, controllers, and the controller/switch interface. The scope of this survey is a quite narrow and in-depth treatment of fundamental aspects of SDN are missing. The essential building

blocks of an SDN like the network operating systems (NOSs), programming languages, and interfaces are not thoroughly discussed in recent surveys. Scalability, security, and dependability are areas that still have cross-layer issues that need analysis.

Section 4.4 is the core of this survey, which presents an extensive and comprehensive analysis of the SDN infrastructure building blocks using a bottom-up, layered approach. The option for a layered approach is based on the fact that SDN allows the idea of networking based on two basic concepts common in other computer science disciplines: separation of concerns (leveraging the concept of abstraction) and recursion. Our layered, bottom-up approach divides the problem of networking into eight parts:

1) Hardware infrastructure;

2) South-facing interfaces;

3) Network virtualization (hypervisor layer between transmission devices and NOSs);

4) NOSs (SDN and control platforms);

5) Northbound interfaces (to offer the upper layers, mainly network applications, a common programming abstraction);

6) Special purpose libraries or programming languages and compilers perform virtualization by slicing techniques;

7) Programming of network languages; and

8) Network applications. We also look at cross-cutting issues like debugging and troubleshooting mechanisms.

4.2 STATUS QUO IN NETWORKING

The functionality of computer networks can be seen as being divided among data, control, and management planes. The plane which is responsible for (efficiently) forwarding data to the networking devices is the data plane. Forwarding tables of the data plane elements are populated by the control plane. The software services are defined in the management plane, such as simple network management protocol (SNMP)-based tools [27], that remotely monitor and configure the control functionality. Management plane is where network policy is defined, the policy enforcement is the job of the control plane, and forwarding data accordingly is executed by data plane.

Due to tightly coupling of the control and data planes, and being embedded in the same networking devices, the whole structure was highly decentralized in traditional IP networks. As it was considered essential for the design of the Internet in the early days, was possibly the most appropriate way to guarantee network resilience, which was a crucial

design goal. Moreover, with a rapid increase of line rate and port densities, network performance is still better with the traditional approach.

As the outcome of this is quite complex and relatively static, thus, makes it the fundamental reason for rigidity, and complex to manage and control (e.g., [14]–[16], [19], and [28]). These two characteristics make innovation difficult because of its vertical integration.

The most common thing in today's networks is network mis-configurations and related errors. A very undesired network behavior may be the result (including, among others, packet losses, forwarding loops, setting up of unintended paths, or service contract violations) of a single misconfigured device, that could also lead to the compromise of the correct operation of the whole Internet for hours [29], [30].

Proprietary solutions of specialized hardware are offered by a small number of vendors, operating systems, and control programs (network applications) in order to provide support for network management. The innovation and addition of new features and services (for instance, access control, load balancing, energy efficiency, traffic engineering) is hampered due to the capital and operational cost of building and maintaining a networking infrastructure along with long return on investment cycles, moreover, network operators acquire and maintain specialized team for management solutions. Abundance of specialized components and middle-boxes, such as firewalls, intrusion detection systems, and deep packet inspection engines, proliferate in current networks are installed in a way to alleviate the lack of in-path functionalities within the network. Therefore, becomes the cause of increased complexity of network design and its operation.

4.3 WHAT IS SOFTWARE - DEFINED NETWORKING?

A network architecture where the remotely controlled plane de-coupled from the former is defined as SDN manages the forwarding state in the data plane is managed by a remotely controlled plane de-coupled from the former is defined as SDN. The SDN as a network architecture has four pillars:

1) Control and data planes are decoupled that causes the removal of Control functionality network devices to make it just simple (packet) forwarding elements.
   The SDN as a network architecture has four pillars:

2) Control and data planes are decoupled that causes the removal of Control functionality network devices to make it just simple (packet) forwarding elements.

3) Instead of being destination based, forwarding decisions are flow now based. A set of actions (instructions) and a set of packet field values acting as a match (filter) criterion is defined by the term flow. Basically, we define flow as a sequence of packets between a

source and a destination with respect to SDN/OpenFlow. Forwarding devices receive all packets of a flow that have identical service policies [31], [32]. The functions of varied types of network devices, including routers, switches, firewalls, and middle-boxes are unified through flow abstraction [33]. The implemented flow tables have the capabilities of enabled unprecedented flexibility [22].

4) SDN controller or NOS is where the control logic is moved as an external entity. Commodity server technology and abstractions facilitate the programming of forwarding devices based on a logically centralized, abstract network view, and other essential resources are used to run NOS on a software platform. Therefore, it is similar to a traditional operating system

5) On top of the NOS runs a network that is programmable through software applications and interacts with the underlying data plane devices. This is what defines the fundamental characteristic of SDN.
There are several additional benefits that the logical centralization of the control logic provides. First, modification of network policies with the use of high-level languages and software components is made simpler and less error prone in comparison with low-level device-specific configurations.

Secondly, maintain the high-level policies intact with a control program that can automatically react to spurious changes of the network state. Thirdly, the development of more sophisticated networking functions, services, and applications is simplified with the centralization of the control logic in a controller with global knowledge of the network state.

Ideally, any forwarding behavior desired by the network application (the control program) should be allowed according to the **Forwarding Abstraction** while hiding details of the underlying hardware. OpenFlow provides one such realization of abstraction, which is the equivalent to a ‘‘device driver’’ in an operating system.

SDN applications should be shielded from the vagaries of distributed state, making the distributed control problem a logically centralized one as defined in **Distribution Abstraction.** SDN resides in the NOS where its realization requires a common distribution layer. This layer has two essential functions. First, the control commands are installed on the forwarding devices. Secondly, status information is collected about the forwarding layer (network devices and links), so that network applications are offered a global network view.

Specification Abstraction, allow a network application to express the desired network behavior without being responsible for implementing that behavior itself. Network programming languages, as well as virtualization solutions, are achieved through this abstraction. SDN controller globally exposes a network that is a simplified, abstract model

of the network, into a physical configuration, is expressed by the abstract configurations approach. The figure shown below depicts the SDN architecture, concepts, and building blocks.

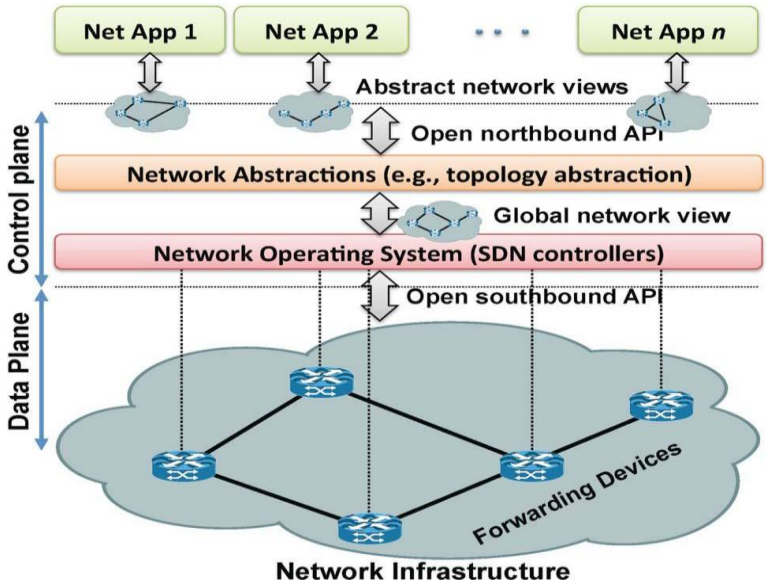


Fig. 4.1: SDN architecture and its fundamental abstractions

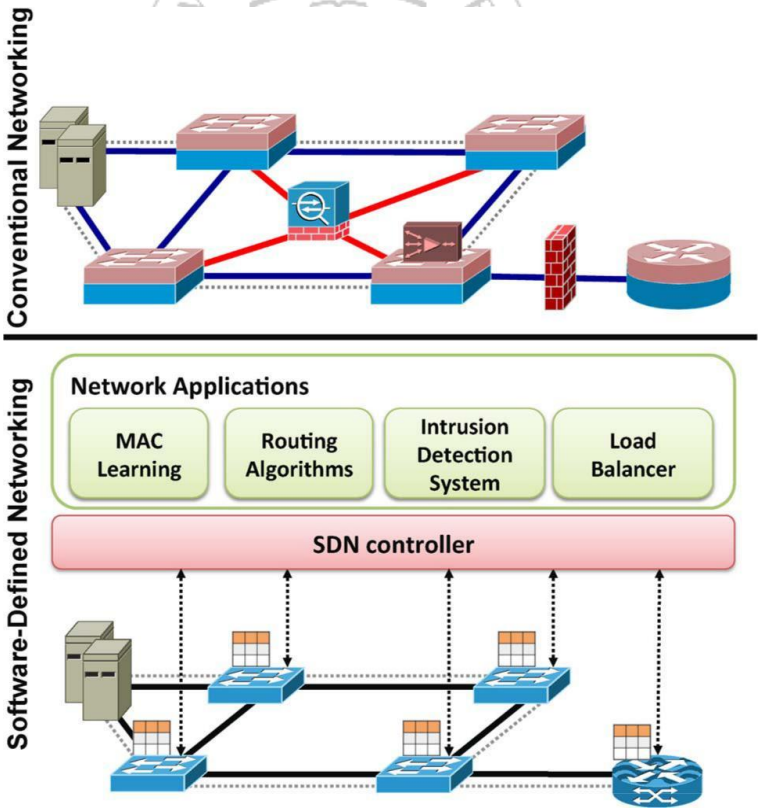


Fig. 4.2: Traditional networking vs. SDN. With SDN overview, management becomes simpler, and middleboxes services can be delivered as SDN controller applications

As previously mentioned, the addition of new functionality to traditional networks was difficult due to the strong coupling between control and data planes, a fact illustrated in Figure above. Developing and deploying of new networking features (e.g., routing algorithms) were very difficult because of the coupling of the control and data planes (and its physical embedding in the network elements), since it would require modification of the control plane of all network devices through the installation of new firmware and, along with hardware upgrades in some cases. Hence, expensive, specialized, and hard-to-configure equipment (also known as middle-boxes) such as load balancers, intrusion detection systems (IDSs), and firewalls, among others then becomes a compulsion for the new networking features, that are commonly introduced via middle-boxes that requires to be placed strategically in the network, thus making the existing even harder to later change the network topology, configuration, and functionality. So in SDN, this decoupled as the control plane from the network devices and it becomes an external entity: the NOS or SDN controller. There several advantages to this approach:

- Sharing of the control platform and/or the network programming languages because of the abstractions. Thus it becomes easier to program.

- Control plane software modules can be reused as all applications can take advantage of the same network information (the global network view), leading (arguably) to more consistent and effective policy decisions.

- Actions (i.e., reconfigure forwarding devices) can be taken from any part of the network by these applications. Therefore, the location of the new functionality can be devised without any prior precise strategy. Moreover, it makes the integration of different applications more straightforward [35]. For example, load balancing and routing applications can be combined sequentially, with load balancing decisions having precedence over routing policies.

4.3.1 Terminology

The different elements of an SDN are identified as:

1) **Forwarding Devices (FD):** A set of elementary operations are performed by this hardware- or software-based data plane devices. Actions on the incoming packets (like., directing to specific ports, dropping, forwarding to the controller, rewriting some header) are taken by the forwarding devices which have well-defined instruction sets (e.g., flow rules). These instructions are then determined by southbound interfaces (e.g., OpenFlow [22], ForCES [36], protocol-oblivious forwarding (POF) [37]) and they are then installed in the forwarding devices by the SDN controllers implementing the southbound protocols.

2) **Data Plane (DP):** Wireless radio channels or wired cables can be used to interconnect forwarding devices. The data plane represents the network infrastructure that comprises of the interconnected forwarding devices.

3) **Southbound Interface (SI):** The southbound API defines the instruction set of the forwarding devices, which is part of the southbound interface. Furthermore, the communication protocol between forwarding devices and control plane elements is defined by SI. The interaction between control and data plane elements is formalized by this protocol.

4) **Control Plane (CP):** Control plane elements program the forwarding device through well-defined SI embodiments. It is therefore known as the ‘‘network brain.’’ All control logic is in the applications and controllers, which is the base for the control plane.

5) **Northbound Interface (NI):** Application developers are offered an API known as Northbound Interface by the NOS, i.e., a common interface for developing applications. Typically, the low-level instruction sets are abstracted by a northbound interface which then is used by southbound interfaces to program forwarding devices.

6) **Management Plane (MP):** Implementation of network control and operation logic through NI is leveraged upon by the set of applications (such as routing, firewalls, load balancers, monitoring, and so forth) in the management plane. Essentially, the policies defined in a management application, which are ultimately translated to southbound-specific instructions which are used to program the behavior of the forwarding devices.

4.3.2 Alternative and Broadening Definitions

Since its inception in 2010 [38], the original Open seen it’s the scope of the original Open-Flow-centered SDN term has been broadened beyond architectures with a cleanly decoupled control plane interface. Business-oriented views drive SDN irrespective of the decoupling of the control plane could cause further broadening on the definitions of SDN. Alternative SDN definitions [39], as follows only for the sake of completeness and clarity:

1) **Control Plane/Broker SDN:** New APIs that allow applications to interact (bi-directionally) with the network is offered with a networking approach that retains existing distributed control planes. An SDN controller that acts as a broker between the applications and the network elements is often called orchestration platform. Control plane data is presented effectively to the application through this approach and a certain degree of network programmability by means of ‘‘plug-ins’’ is allowed between the orchestrator function and network protocols. A hybrid model of SDN is corresponded due to this API driven approach, as manipulation is enabled for the broker and devices such as routers and switches can directly interact with the control plane. Examples of this view on SDN include Internet Engineering Task Force (IETF) recent standardization’s efforts (see Section III-C), and the Open Daylight project design’s philosophy [25] that goes beyond the OpenFlow split control mode.

2) **Overlay SDN:** Introducing an overlay network where managing tunnels between hypervisors and/or network switches, at the (software- or hardware-based) network edge is dynamically programmed through this approach. The underlay remains untouched in the distributed control plane in this hybrid approach. Hence a logical overlay that utilizes the underlay as a transport network is provided by the centralized control plane. The overlay tunnels are installed through this flavor of SDN which follows a proactive model. In hypervisors, the overlay tunnels are usually terminated inside virtual switches or in physical devices acting as gateways to the existing network. Recent data center network virtualization [40] uses this approach, and variety of tunneling technologies (e.g., stateless transport tunneling [41], virtualized layer 2 networks (VXLAN) [42] are based on it, network virtualization using generic routing encapsulation (NVGRE) [43], locator/ID separation protocol (LISP) [44], [45], and generic network virtualization encapsulation (GENEVE) [46]) [47].[48]

Other attempts to define SDN includes one initiative from the IRTF Software-Defined Networking Research Group (SDNRG) which comes with a management plane in parallel

with the control plane. Solutions classified in two categories: control logic which has control plane southbound interfaces and management logic which has management plane southbound interfaces that implies the management plane is a control platform that accommodates traditional network management services and protocols such as SNMP [27], BGP [49], network configuration protocol (NETCONF) [51], and path computation element communication protocol (PCEP) [50]. Along with broadening definitions above, the term SDN is often defined as extensible network management planes (e.g., OpenStack [52]), white-box/bare-metal switches with open operating systems (e.g., Cumulus Linux), open-source data planes (e.g., Pica8 Xorplus [53], Quagga [54]), specialized programmable hardware devices (e.g., NetFPGA [55]), virtualized software-based appliances (e.g., an open platform for network functions virtualization (OPNFV) [55]), in spite the fact that it lacks a decoupled control and data plane or common interface along its API.

4.3.3 Standardization Activities

The Service Innovation & Market Requirements (SIMR), WG of SDN, is worked upon by the Broadband Forum (BBF), whose main objective in multiservice broadband networks is to delegate support in hybrid environments where only some of the network hardware is SDN enabled.

The service orchestration with APIs for existing networks from the approach of SDN is done by the Metro Ethernet Forum (MEF).

To embrace new control interfaces for both wired and wireless technologies, P802.1CF project which standardizes SDN capabilities on access networks based on IEEE 802 infrastructure is taken upon at the IEEE, the 802 LAN/MAN Standards Committee

A set of requirements for transport SDN was released by the Optical Internetworking Forum (OIF) Carrier WG. The initial activities included the description of the features and functionalities needed to support the deployment of SDN capabilities in carrier transport networks. Open Data Center Alliance (ODCA) an organization that is working on the unifying of the data center in the migration to cloud computing environment by application of interoperable solutions, and is also defining new requirements for cloud deployment. Operational issues and opportunities that are associated with the programmable capabilities of network infrastructure are analyzed by the Alliance for Telecommunication Industry Solutions (ATIS).

A newly defined Industry Specification Group (ISG) is devoting efforts to network function virtualization (NFV) at the European Telecommunication Standards Institute (ETSI). Accelerating innovation by allowing programmability inside the network is the main goal shared by NFV and SDN which are in way complement to each other, and altogether they aim to change the network operational model through automation and shifting to software-based platforms.

Finally, the study on the management of virtualized networks, an effort aligned with the ETSI NFV architecture is done by the mobile networking industry 3rd Generation Partnership Project consortium.

4.3.4 History of SDN

Even though it is a recent concept, SDN leverages on networking ideas with a long history [26]. Programmable networks are worked upon to build SDN, like active networks [58], programmable ATM networks [59], [60], and control and data plane separation was proposed in the network control point (NCP) [61] and routing control platform (BCP) [62].

A long history of Data plane programmability us represented by Active networks [58] is one of the early attempts on building new network architectures based on this concept. The focus by active networks is for each node to have the capability to perform computations on or modify the content of, packets. There are two distinct approaches proposed by active networks: programmable switches and capsules. The changes in the existing packet or cell format are not implied by the former because there is an assumption that processing of packets by the switching devices is supported by the downloading of programs with specific instructions. Whereas Capsule approach suggests that packets should be replaced by tiny programs, which has encapsulated transmission frames and are executed at each node along their path.

Current approaches for designing and deploying programmable data plane devices are represented by ForCES [36], OpenFlow [22], and POF [37]. Modification of forwarding devices to support flow tables, which can be dynamically configured by remote entities through simple operations such as adding, removing, or updating flow rules, i.e., entries on the flow tables is essentially relied upon by these new proposals.

It dates back to the 1980s and 1990s when the earliest initiatives on separating data and control signaling started. The NCP [61] made the first attempt to separate control and data plane signaling. AT&T introduced NCP to improve the management and control of its telephone network which yielded to a faster pace of innovation of the network and improved its efficiency through new means, by taking advantage of the global view of the network. Similarly, the separation of the control and data planes for improved management in ATM, Ethernet, BGP, and multiprotocol label switching (MPLS) networks was among the other initiatives which Tempest [63], ForCES [30], RCP [62], and PCE [50] proposed, respectively.

More recently, a proposal for the decoupling of the control and data planes for Ethernet networks was initiatives from the SANE [64], Ethane [65], OpenFlow [22], NOX [32], and POF [37]. Interestingly, significant modifications on the forwarding devices were not essential enough for implementation of these solutions, thus making them attractive not only for the networking research community but also to the networking industry. OpenFlow-based devices [22], for instance, can easily coexist with traditional Ethernet devices, enabling a progressive adoption (i.e., not requiring a disruptive change to existing networks).

Network virtualization from the 1990s had gained new traction with the advent of SDN which now had gained new traction with the advent of SDN. The first initiatives to introduce network virtualization was through the Tempest Project [63], by the introduction of the concept of switchlets in ATM networks. The core idea was to enable multiple independent ATM networks to share the same physical resources by allowing multiple switchlets on top of a single ATM switch. Early initiatives that targeted on the creation of virtual network topologies on top of legacy networks, or overlay networks as by MBone [66]. This was further followed up by several other projects such as Planet Lab [67], GENI [68], and VINI [69]. FlowVisor came up with network infrastructure for compute and storage that promoted a hypervisor-like virtualization architecture. Koponen et al. also proposed a network virtualization platform (NVP) [70] that multitenant data centers are using as SDN as a base technology.

OpenFlow-based NOSs introduced the concept of NOS, such as NOX [32], Onix [20], and ONOS [71]. Indeed, NOSs existence back for decades. Other NOSs are JUNOS [72], ExtremeXOS [73], and SR OS [74]. Despite being more specialized they target network devices like high-performance core routers, and also abstract the underlying hardware to the network operator, to make it easier for them to control the network infrastructure along

with simplifying the development and deployment of new protocols and management applications.

Finally, ‘‘technology pull’’ drivers that back to the 1990s when a movement towards open signaling [75] began to happen. The idea was to promote the wider adoption of projects such as NCP [61] and Tempest [63]. Proposing of open and programmable interfaces was done for the separation of the control and data signaling. Curiously, this can be seen recently with the advent of OpenFlow and SDN, with the lead of the ONF [23]. The movement becomes crucial to promote open technologies into the market, also hopes from the lead equipment manufacturers to support these open standards and thus fostering interoperability, competition, and innovation.

4.4 SOFTWARE - DEFINED NETWORKS: BOTTOM - UP

A composition of different layers, as shown below depicts an SDN architecture. Each layer has its own specific functions. Layers such as the southbound API, NOSs, northbound API, and network applications are always present in an SDN deployment while others like hypervisor- or language-based virtualization may be present only in particular deployments.

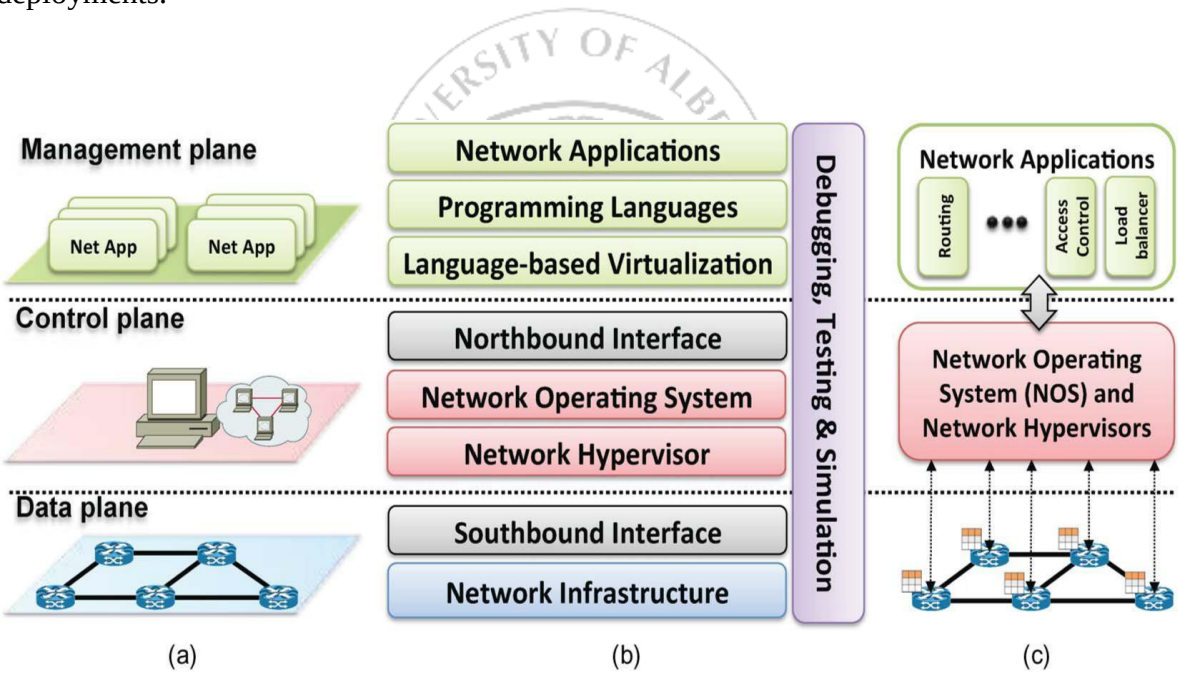


Fig. 4.3: Software-Defined Networks in (a) planes, (b) layers, and (c) system design architecture

A tri-fold perspective of SDNs is presented in the figure above. Fig. (b) represents the SDN layers, as explained above, while fig. (a) and (c) gives a plane-oriented view and a system design perspective, respectively.

Following a bottom-up approach, each layer will be introduced. The core properties and concepts explain different technologies and solutions for each layer respectively. Along with that some debugging and troubleshooting techniques and tools are discussed.

4.4.1 Layer I: Infrastructure

Similar to a traditional network an SDN infrastructure is composed of a set of networking equipment (switches, routers, and middlebox appliances). The fact that now differentiates those traditional physical devices from an SDN is that they are just simple forwarding elements without embedded control or software to take autonomous decisions. A rationally centralized control system, i.e., the NOS and application take up the network intelligence from the data plane devices.

Ensuring the configuration and communication compatibility and interoperability between discrete data and control plane devices is an essential approach, these new networks are built (conceptually) on top of open and standard interfaces (e.g., OpenFlow). We can say that these open interfaces enable controller entities to dynamically program heterogeneous forwarding devices, which something quite impossible in traditional networks, because of the large variety of proprietary and the distributed nature of the control plane and closed interfaces.

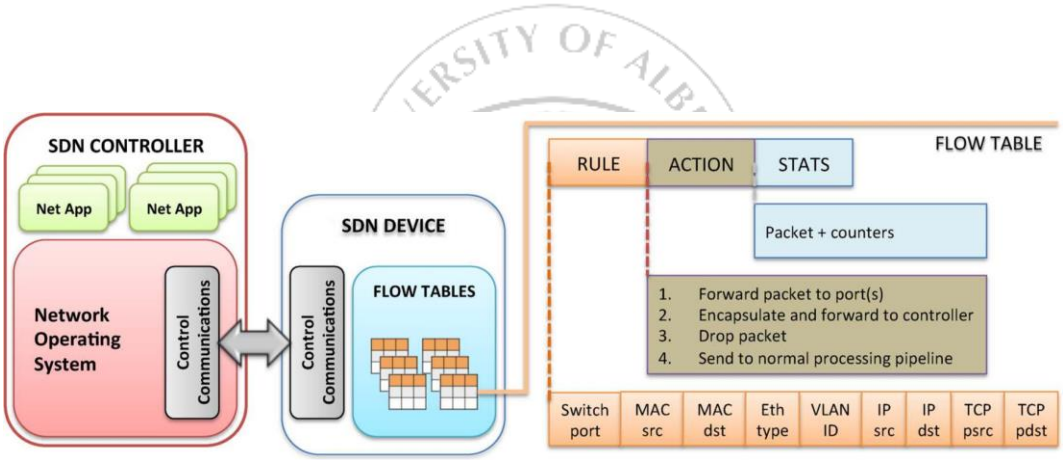


Fig. 4.4: OpenFlow-enabled SDN devices

Two crucial components, the controllers and the forwarding devices that form the basis in an SDN/OpenFlow architecture, as depicted in the figure above. Packet forwarding is done by a data plane device that could be a hardware or software element, while a controller is a software stack (the ‘‘network brain’’) running on a commodity hardware platform. Each entry of a flow table in an OpenFlow-enabled forwarding device is based on a pipeline of flow tables which can be defined in the three parts:

1) A matching rule;

2) Execution on matching packets that causes an action(s); and

3) Counters that keep statistics of matching packets.

OpenFlow, the most widespread design of SDN data plane devices that is a high-level and simplified model. Specifications including POF [37], [76] and the negotiable data path models (NDMs) from the ONF Forwarding Abstractions Working Group (FAWG) [77], are being worked upon in SDN-enabled forwarding devices.

Inside an OpenFlow device, a sequence of flow tables defines a path that shows how packets should be handled. With the arrival of a new packet, the lookup process starts in the first table, and it ends either with a match in one of the tables of the pipeline or with a miss (the case when no rule is found for that packet). By combining different matching fields, a flow rule can be defined. No default rule means the packet is discarded. Typically there is a default rule installed in the switch that directs the packet to the controller. The rules are prioritized as the natural sequence number of the tables and the row order in a flow table. Possible actions include:

1) The packet is forwarded to outgoing port(s);

2) Encapsulating it and forwarding it to the controller;

3) Dropping it;

4) Sending it to the normal processing pipeline; and

5) Sending it to the next flow table or to special tables, such as group or metering tables

OpenFlow with each version defines new specifications alongside new match fields that are Ethernet, IPv4/v6, MPLS, TCP/UDP, etc. Although, only a few those matching fields are conformable to a given protocol version along with many actions and port types being an optional feature. Flow match rules are based on the random combinations of bits of varied packet headers using bit masks for each field. OpenFlow version 1.2 introduces extensibility capabilities through an OpenFlow Extensible Match (OXM) that are based on type-length-value (TLV) structures which eases the process of adding new matching fields. While OpenFlow version 1.4 brings the improvement in the overall protocol extensibility with the TLV structures that have been also added to ports, tables, and queues in replacement of the hard-coded counterparts of earlier protocol versions.

There are several OpenFlow-enabled forwarding devices that are available on the market, both as commercial and open source products. Many of them are off-the-shelf, ready to deploy, OpenFlow switches and routers being the other appliances. Nonetheless, this is fast changing market as Some of the latest devices released in the market go far beyond that Gigabit Ethernet (GbE) switches that are deployed for business purposes and are already supporting up to 32000 Layer 2 (L2) + Layer 3 (L3) or 64 000 L2/L3 exact match flows [78]. More than 80000 layer 2 flow entries [123] are being delivered in Enterprise class 10GbE switches. Other switching devices (e.g., EZchip NP-4) provide optimized TCAM

memory that supports from 125 000 up to 1 000 000 flow table entries [124] that uses high-performance chips.

The needs of future SDN deployments are clearly depicted with a sign of growth in the size of flow tables. have Various kinds of OpenFlow-enabled devices have been produced by networking hardware manufacturers, and these devices range from equipment for small businesses (e.g., GbE switches) to high-class data center equipment (for instance high-density switch chassis that provide 100GbE connectivity for edge-to-core applications, with tens of terabits per second of switching capacity).

Covenanting solutions for data centers and virtualized network infrastructures [80]–[82] comes with the emergence of software switches. Instances of such software-based OpenFlow switch implementations include Switch Light [83], ofsoftswitch13 [84], Open vSwitch [85], OpenFlow Reference [86], Pica8 [150], Pantou [87], and XorPlus [53]. It is clearly shown in recent reports that the number of virtual access ports is already larger than physical access ports on data centers [82]. The drivers behind this trend are Network Virtualization Network functions moved to the edge (with the core performing traditional IP forwarding) software switches like Open vSwitch have been deployed, thus enabling network virtualization [70].

Small, startup enterprises such as Big Switch, Pica8, Cyan, Plexxi, and NoviFlow have devoted to SDN and are increasing in number. From this, it is clearly implied that SDN is springing a more competitive and open networking market which was one of its original goals. The emergence of so-called ‘‘bare metal switches’’ or ‘‘whitebox switches,’’ are the other effects of this openness triggered by SDN, where software and hardware are sold separately and which gives the end user the freedom to load an operating system of its choice [88].

4.4.2 Layer II: Southbound Interfaces

Control and forwarding elements connected through bridges are the Southbound Interfaces (or southbound APIs), thus are the crucial instruments that clearly separates the control and data plane functionality. However, the underlying physical or virtual infrastructure tightly ties these APIs to its forwarding elements.

To make it ready for commercialization if built from scratch a new switch can typically take two years, while the up-gradation cycles can take up to nine months. It can take from six months to one year [89] for developing software for a new product. The initial investment is high and risky. The southbound APIs being the central component of its design, represents as one of the major barriers for the introduction and acceptance of any new networking technology. In meanwhile, proposals for the emergence of SDN southbound API such as OpenFlow [22] is seen as quite welcoming by many in the industry. As standards promote interoperability, vendor-agnostic network devices deployments, and

has already been demonstrated by the interoperability between OpenFlow-enabled equipment from different vendors.

Broadly accepted and deployed open southbound standard for SDN is OpenFlow. As it provides with a common specification for implementing Open-Flow-enabled forwarding devices, and the communication channel between data and control plane devices (e.g., switches and controllers).

Three information sources for NOSs are provided by the OpenFlow protocol. First, when a link or port change is triggered event-based messages are sent by forwarding devices to the controller. Second, forwarding devices generate flow statistics and are collected by the controller. Third, forwarding devices sent packet-in messages to the controller when they do not know what to do with a new incoming flow or as there is an explicit action like ‘‘send to controller’’ in the matched entry of the flow table. Flow-level information to the NOS is provided by these essential information channels.

Although it is the most visible, OpenFlow is not the only available southbound interface for SDN. As there are other API proposals such as ForCES [36], Open vSwitch Database (OVSDB) [90], POF [37], [76], OpFlex [91], OpenState [92], revised open-flow library (ROFL) [93], hardware abstraction layer (HAL) [94], [95], and (PAD) [96] being the programmable abstraction of data path . The proposal by ForCES undermines a flexible approach to traditional network management without introducing any change in current network architecture, i.e., logically centralized external controller. It means that even though the control and data planes are separated, but can potentially be kept in the same network element. However, there is a need to upgrade the control part of the network element on-the-fly with third-party firmware.

Advanced management capabilities for Open vSwitches is provided by another southbound API called OVSDB [90]. Open vSwitch offers networking functions along with its capabilities to configure the behavior of flows in a forwarding device. As it allows creating multiple virtual switch instances for the control elements, setting quality of service (QoS) policies on interfaces, attaching interfaces to the switches, configuring tunnel interfaces on OpenFlow data paths, managing queues, and collecting statistics. Therefore, the OVSDB is considered as a complementary protocol to OpenFlow for Open vSwitch.

POF [37], [76], with the main goal to enhance the current SDN forwarding plane is the competitor to OpenFlow. With OpenFlow, in order to extract the required bits to be matched with the flow tables entries switches have to understand the protocol headers. This parsing causes a relative amount of burden for data plane devices, particularly when we consider that OpenFlow version 1.3 where it already contains more than 40 header fields. Along with this inherent complexity, every time new header fields are included in or removed from the protocol backward compatibility issues arises. To overcome this, in a way to make the forwarding plane protocol oblivious POF proposes a generic flow instruction set (FIS), where is not needed to know a forwarding element, by itself, or

anything related about the packet format in advance. It sees forwarding devices as the white boxes with only processing and forwarding capabilities. In POF, a sequence of generic keys and table lookup instructions are installed in the forwarding elements for a controller task to see packet parsing. This then causes the behavior of data plane devices to be completely under the control of the SDN controller. A POF switch is an application and protocol agnostic which is similar to CPU in a computer system.

OpFlex, a recent southbound interface [91] in comparison to OpenFlow (and similar to ForCES), the main purpose of OpFlex, is to distribute part of the complexity of managing the network back to the forwarding devices, in order to improve scalability. Similar to OpenFlow, here also the policies are logically centralized and are being abstracted from the underlying implementation. The differences between OpenFlow and OpFlex comes when devising a southbound interface that is where to place each piece of the overall functionality.

Unlike OpFlex and POF, OpenState [92] and ROFL [93] do not have any new set of instructions for programming data plane devices. Instead OpenState ex-tends finite machines (stateful programming abstractions) as an extension (superset) of the OpenFlow match/ action abstraction. Finite State Machines implements a variety of stateful tasks inside forwarding devices, i.e., without augmenting the complexity or overhead of the control plane. All tasks related only to local state, such as media access control (MAC) learning operations, port knocking, or stateful edge firewalls are directly performed by forwarding devices, without any jitter in control plane communication and processing.

Whereas, ROFL, comes with a proposal where an abstraction layer hides the details of the different OpenFlow versions, thus in a way providing a clean API for software developers and thus simplifying application development.

HAL [94], [95] is closely related to southbound API but not exactly it. It is rather a translator that allows a south-bound API like OpenFlow to have control on heterogeneous hardware devices. Therefore, it lies between the southbound API and the hardware device. It's the viability of SDN control in access networks such as Gigabit Ethernet passive optical networks (GEPONs) [97] and cable networks (DOCSISs) have been demonstrated in recent research experiments [98]. Similarly, to HAL is PAD [96], a proposal which goes a bit further as it is also working as a southbound API by itself. Particularly, PAD enables the control of data path behavior using generic byte operations, defining protocol headers and providing function definitions by allowing more generic programming of forwarding devices.

4.4.3 Layer III: Network Hypervisors

A consolidated technology in modern computers is virtualization. Virtualization of computing platforms has become mainstream due to the fast developments in the past

decade. The number of virtual servers has already out-passed the number of physical servers [99], [70].

Hypervisors allow different virtual machines to share the same hardware resources. In a cloud infrastructure-as-a-service (IaaS) where each user can have its own virtual resources, starting from basic computing to storage.

This has developed a new revenue and business model where users allocate resources on demand, at a relatively low cost, from shared physical infrastructure. Simultaneously, providers can make better utilization of the capacity of their installed physical infrastructures, which in a way helps them to create new revenue streams without any significant increase in their capital expenditure and operational expenditure (OPEX) costs. Migration of virtual machines along with creation and/or destruction on demand can easily be implemented by utilizing today's virtualization technology, which in a way allowing the provisioning of elastic services with flexible and easy management. But the thing is that virtualization has been only partially realized in practice. Even though it possesses great advances in virtualizing computing and storage elements, and sill the network is statically configured in a box-by-box manner [40].

Network topology and address space are two dimensions captured along main network requirements. Network topologies and services, like flat L2 or L3 services, or even more complex L4–L7 services for advanced functionality requires a different type of workload. Currently, in order to support the diverse demands of applications and services causes difficulty for a single physical topology. Along with that, it is hard to change address space in current networks. So, virtualized workloads have to operate in the same address of the physical infrastructure. Therefore, it makes it difficult to keep the original network configuration for a tenant, migration of virtual machines to arbitrary locations becomes an issue, and along with that, the addressing scheme is fixed and difficult to change. For instance, IPv6 is not supported by virtual machines (VMs) of a tenant if the underlying physical forwarding devices can handle only IPv4.

To complete virtualization is only feasible when the network should also support similar properties to the computing layer [40], that network infrastructure should provide arbitrary network topologies and addressing schemes. Simultaneously both the computing nodes and the network should be configured by each tenant. The migration of the corresponding virtual network ports should automatically be triggered by Host Migration. Although we have long-standing virtualization primitives such as VLANs (virtualized L2 domain), NAT (virtualized IP address space), and MPLS (virtualized path) to provide full and automated network virtualization. But the drawback is that these technologies are anchored on a box-by-box basis configuration, i.e., there is no single unifying way that we can leverage configuration (or reconfigure) of the network in a global manner. As a result of this, current network provisioning can take months, whereas computing provisioning takes only minutes [70], [100]–[102].

SDN hopes to change such situation by providing the availability of new tunneling techniques (e.g., VXLAN [42] and NVGRE [43]). For example, solutions such as FlowVisor [103], [104], [105], FlowN [106], NVP [70], OpenVirteX [107], [108], IBM SDN VE [109], [110], RadioVisor [111], AutoVFlow [112], eXtensible Datapath Daemon (xDPd) [113], [114], optical transport network virtualization [115], and version-agnostic OpenFlow slicing mechanisms [116], proposed, evaluated, and deployed recently in current scenarios for on-demand provisioning of virtual networks.

1) **Slicing the Network:** The earliest technology to virtualize an SDN is FlowVisor. The idea to allow multiple logical networks to share the same OpenFlow networking infrastructure. So it comes up with an abstraction layer that slices a data plane based on off-the-shelf OpenFlow-enabled switches, which allows multiple and diverse networks to coexist. There are five slicing dimensions in FlowVisor: bandwidth, topology, traffic, device CPU, and forwarding tables. Moreover, each network slice is supported by a controller, i.e., it allows multiple controllers to coexist on top of the same physical network infrastructure. Each controller can act only on its own network slice. We can say that a particular set of flows on the data plane defines a slice. With a viewpoint of the system design perspective, FlowVisor is a transparent proxy that intercepts OpenFlow messages between switches and controllers. The link bandwidth and flow tables of particularly each switch are portioned by it. A minimum data rate is received by each slice, and each guest controller gets its own virtual flow table in the switches.
Similarly, to FlowVisor, [107], [108] a proxy between the NOS and the forwarding devices is OpenVirteX. It aims to provide virtual SDNs through topology, address, and control function virtualization. These properties become a necessity in multitenant environments where virtual networks need to be managed and migrated according to the Virtual network topologies and have to be mapped onto the underlying forwarding devices, with completely managing their address space without depending on the underlying network elements addressing schemes with the help of virtual addresses.

Another SDN-based virtualization proposal is Auto-Sliced [117]. Which has minimal mediation or arbitration by the substrate network operator and focuses on the automation of the deployment and operation of virtual SDN (vSDN). Moreover, is targets scalability aspects of network hypervisors as it optimizes resource utilization and mitigates the flow-table limitations by precisely monitoring the flow traffic statistics. Similarly to Auto-Slice, we have AutoV-Flow [112] that allows multi-domain network virtualization. However, in this case, instead of having a single third party to control the mapping of vSDN topologies, AutoVFlow uses a multi-proxy architecture that gives the right to network owners to implement flow space virtualization in an autonomous way just by exchanging information among the different domains.

FlowN [106,118] is analogous to container-based virtualization, a slightly different concept, i.e., a lightweight virtualization approach. FlowN, in the context of cloud platforms was primarily conceived to address multitenancy and is designed to be scalable

and allowing a unique shared controller platform to be used for managing multiple domains in a cloud environment. A virtual network is under full control of each tenant, and they are free to deploy any network abstraction and application on top of the controller platform.

The composition of SDN hypervisor [119]. With the main objective to allow the cooperative (sequential or parallel) execution of applications that are developed with distinct programming languages or devised for diverse control platforms. Along with the typical functions of network hypervisors, it also offers interoperability and portability.

2) **Commercial Multitenant Network Hypervisors:** All challenges of multitenant data centers are still not addressed in the aforementioned approaches. For examples, users without any modification in the network configuration of their home network want to migrate their enterprise solutions to cloud providers. The requirements of both tenant and service provider have mostly been failed with the existing networking technologies and migration strategies. A network hypervisor is capable of anchoring multitenant environment by abstracting the underlying forwarding devices and physical network topology from the tenants. Moreover, access to control abstractions and management of virtual networks independently should be in the hands of each tenant and should be isolated from other tenants.

Various commercial virtualization platforms based on SDN concepts have started to appear due to greater market demand for network virtualization and the recent research on SDN showing promise as an enabling technology. Network virtualization platform (NVP) [70] has been proposed by VMWare that allow the creation of independent virtual networks for large-scale multitenant environments to provide the necessary abstractions. A complete network virtualization solution that has an independent service model, topologies, and addressing architectures over the same physical network allows the creation of virtual networks. With NVP, it gives tenants freedom from the underlying network topology, configuration, or other specific aspects of the forwarding devices. Tenant's configurations are translated by NVP's network hypervisor and requirements into low-level instruction sets to be installed on the forwarding devices. Manipulating the forwarding tables of the Open vSwitches in the host's hypervisor, there is a platform that uses a cluster of SDN controller. Thus, Forwarding decisions are exclusively made on the network edge. After the decision, host hypervisor receives the packet that is tunneled over the physical network (the physical network sees nothing but ordinary IP packets).

SDN VE [109], [110], another commercial and enterprise-class network virtualization platform proposed by IBM. SDN VE uses Open Daylight is one of the building blocks of the so-called software-defined environments (SDEs) that is used by SDN VE. It is a complete implementation framework for network virtualization. Like NVP, in order to achieve advanced network abstraction that enables application-level network services in large-scale multitenant environments, it uses a host-based overlay approach. Interestingly, one single instantiation of SDN VE 1.0 has the capacity to support up to 16 000 virtual networks and 128 000 virtual machines [109], [110].

Finally we realize that currently there are already a few network hypervisor proposals that leverage the advances of SDN and they still have several issues to be addressed which include, the improvement of virtual-to-physical mapping techniques [120], the definition of the level of detail that should be exposed at the logical level, and nested virtualization for the support [35]. This could be anticipated that this ecosystem is likely to expand in the near future as network virtualization will most likely play a key role in future virtualized environments, similar to the expansion we have witnessed in virtualized computing.

4.4.4 Layer IV: Network Operating Systems/Controllers

Accessing lower level devices, managing the concurrent access to the underlying resources (e.g., hard drive, network adapter, CPU, memory), and providing security protection mechanisms was done through abstraction (e.g., high-level programming APIs) in traditional operating systems. These functionalities increase productivity, make the life of system and application developers easier. Evolving of various ecosystems (e.g., programming languages) and the development of a myriad of applications have been due to their widespread use

Management and configuration of networks have so far been at a lower level, instruction sets have been device specific, and we have mostly closed proprietary NOSs (e.g., Cisco IOS and Juniper JUNOS). Moreover, the systems abstracting device-specific characteristics and providing common functionalities are still absent in networks. For example, in order to solve networking problems, designers of routing protocols have to deal with the complex distributed algorithm. Network practitioners have been doing this over and over again.

Logically centralized control offered by a NOS [32] in SDN promises to facilitate network management and ease the burden of solving networking problems. The basic function of a NOS is to provide abstractions, essential services, and common APIs to developers in a traditional operating system. NOS provides with generic functionality as network state and network topology information, device discovery, and distribution of network configuration. It also defines network policies a developer no longer needs to care about the low-level details like data distribution among routing elements. Thus, have the capacity of fostering innovation at a faster pace by reducing the inherent complexity for creating new network protocols and network applications.

SDN architecture has a critical element called NOS or controller and has the ability to support control logic (applications) in order to generate the network configuration based on the policies defined by the network operator. It abstracts the lower level details of connecting and interacting with forwarding devices (i.e., of materializing the network policies) as in traditional networks.

Architecture and Design Axes: There various types of controllers and control platforms with different design and architectural choices [20], [25], [121]–[124] and can be

categorized on a variety of aspects. Based on the architectural point of view, one of the most relevant is if they are centralized or distributed.

Centralized Versus Distributed: A single entity that manages all forwarding devices of the network is the controller. It is represented by a single point of failure and has scaling limitations. We need multiple controllers to manage a network with a large number of data plane elements. NOX–MT [125], Maestro [125], Beacon [124], and Floodlight [127] centralized controllers have been designed as highly concurrent systems, for achieving throughput required by enterprise-class networks and data centers. Multithreaded designs form the basis of this system in order to explore the parallelism of multicore computer architectures. For instance, Beacon can deal with more than 12 million flows per second by using large size computing nodes of cloud providers such as Amazon [124]. Trema [128], Ryu NOS [129], Meridian [130], and Programmable Flow [133], [131] centralized controllers target specific environments like data centers, cloud infrastructures, and carrier-grade networks. Rosemary [132] another type that offers specific functionality and guarantees, namely security and isolation of applications. A container-based architecture called micro-NOS uses SDN stack to achieve its primary goal of isolating applications and preventing the propagation of failures.

Scaling up a distributed NOS to meet the requirements of potentially any environment, from small- to large-scale networks. It is a centralized cluster of nodes which can offer high throughput for very dense data centers or a physically distributed set of elements that can be more resilient to different kinds of logical and physical failures. A hybrid approach is needed for a cloud provider that spans multiple data centers interconnected by a wide area network, with clusters of controllers inside each data center and distributed controller nodes in the different sites [21].

Some distributed controllers like Onix [20], HyperFlow [134], HP VAN SDN [122], ONOS [71], DISCO [123], yanc [196], PANE [197], SMaRt-Light [198], and Fleet [199]. They have weak consistency semantics, which means that data updates on distinct nodes will eventually be updated on all controller nodes. This implies that over a period of time each distinct nodes may read different values (old value or new value) for the same property. Even though it impacts the performance of the system, strong consistency offers a simpler interface to application developers.

Fault tolerance one of the characteristics of controllers which is implied in case one node fails, another neighbor node should take over the duties and devices of the failed node. T Some controllers have abilities for tolerating crash failures; they do not tolerate arbitrary failures, which means abnormal behavior of a node would not cause its replacement by a potentially well-behaved one.

Due to a single point of failure, a single controller cannot represent and manage a small network. So, we spread independent controllers across the network, for them manage a network segment, reduce the impact of a single controller failure. In case-control plane

availability is critical, a cluster of controllers becomes available and/or support more devices. It ultimately improves the control plane resilience and scalability and reduces the impact of problems caused by a network partition.

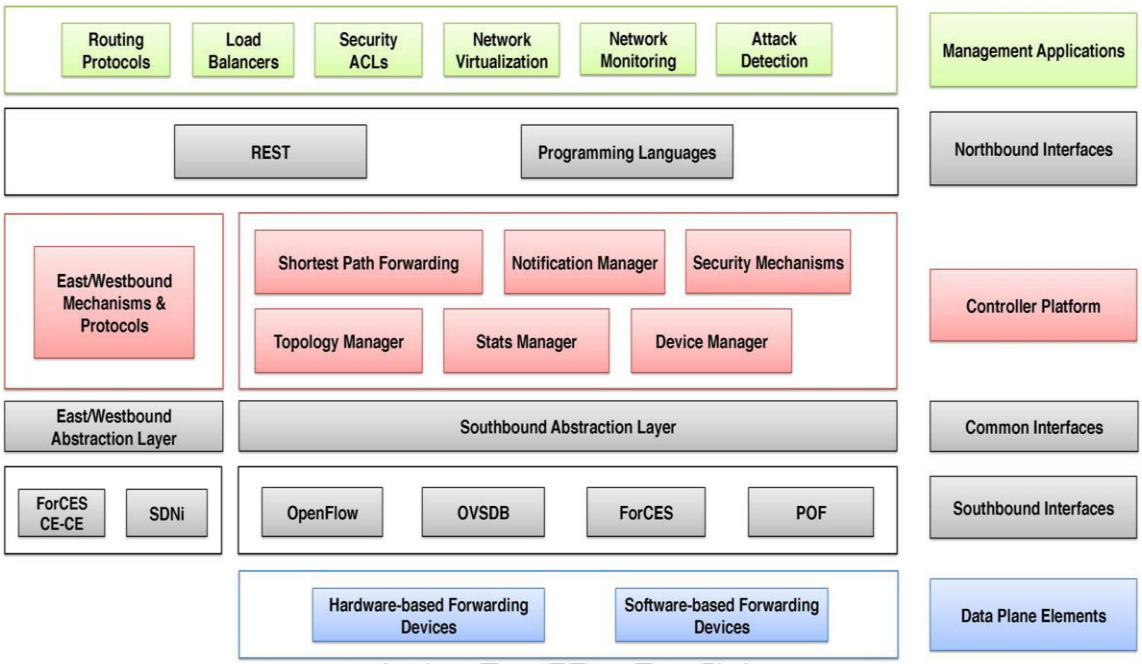


Fig. 4.5: SDN control platforms: elements, services, and interfaces

Dissecting SDN Controller Platforms: Based on the analysis of the different SDN controllers' figure below proposed to provide a first attempt to clearly and systematically dissect an SDN control platform. There are three well defined distinct layers:

1) The application, orchestration, and services;

2) The core controller functions; and

3) The elements for southbound communications. Northbound interfaces such as REST APIs [139] and programming languages such as FML [140], Frenetic [141], and NetCore [142] connects the upper-level layers. Southbound APIs and protocol plug-ins interface the forwarding elements on the lower level. A combination that base network service functions and the various interfaces form the core of a controller platform.

- **Core Controller Functions:** The basic functionality all controllers should provide is the base network service. Functions like base services of operating systems include program execution, input/output (I/O) operations control, communications, protection, and so on. Other OS level services and user applications use these services. Similarly, essential network control functionalities that network applications may use in building its logic. Like topology, statistics, notifications, and device management, together with shortest path forwarding and security mechanisms are supported. Like, the notification manager should

be able to receive, process, and forward events (e.g., alarm notifications, security alarms, state changes) [143]. Another instance could be to provide basic isolation and security enforcement between services and applications as a security mechanism like rules by application of low priority should not overwrite rules generated by high priority services

- **Southbound:** The lower level of control platforms, they are considered as a layer of device drivers. They are the common interface for the upper layers, they manage existing or new physical or virtual devices (e.g., SNMP, BGP, and NetConf) that allows a control platform to implement different southbound APIs (e.g., OpenFlow, OVSDB, and ForCES) and protocol plug-ins. They form the basis for backward compatibility and heterogeneity, i.e., to allow multiple protocols and device management connectors. Hence a mix of physical devices, virtual devices (e.g., Open vSwitch [144], [85], vRouter [145]) and a variety of device interfaces (e.g., OpenFlow, OVSDB, of-config [146], NetConf, and SNMP) can coexist on the data plane.

OpenFlow as a south-bound API is supported by most controllers. Open Daylight, Onix, and HP VAN SDN Controller are few that offer a wider range of southbound APIs and/or protocol plug-ins. OpenFlow and OVSDB protocols are supported by Onix. L2 and L3 agents are the HP VAN SDN Controller.

Along with service layer abstraction (SLA), Open Daylight allows several southbound APIs and protocols to coexist in the control platform. For example, originally it was architected to support at least seven different protocols and plug-ins: OpenFlow, OVSDB [90], NETCONF [51], PCEP [50], SNMP [147], BGP [49], and LISP Flow Mapping [25]. Hence, OpenDay-light in a single control platform is one of the few control platforms being conceived to support a broader integration of technologies

- **Eastbound and Westbound:** Figure below illustrates a special case of interfaces required by distributed controllers, i.e., Eastbound and Westbound. Initially, each controller has its own east/westbound API. Import/export data between controllers, data consistency model algorithms, and capabilities (e.g., check if a controller is up or notify a take over on a set of forwarding devices) of monitoring/notifications are the few functionalities of these interfaces.

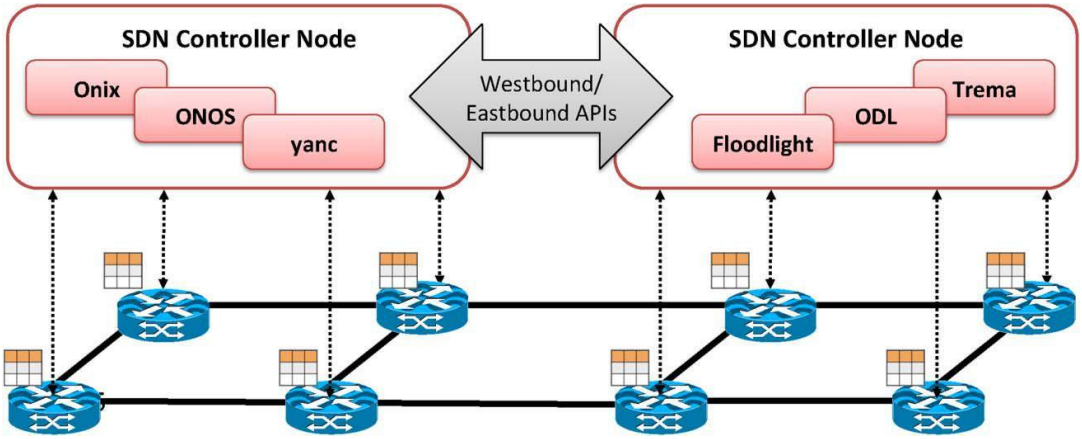


Fig. 4.6: Distributed controllers: east/westbound APIs

As southbound and northbound interfaces, east/westbound APIs are vital components of distributed controllers as standard east/westbound interfaces they identify and provide common compatibility and interoperability between different controllers. General requirements to coordinate flow setup and exchange reach-ability information across multiple domains is defined by SDNi [148]. In essence, such protocols can create more scalability and dependability on distributed control platforms which can be used in an orchestrated and interoperable way. The diversity of the control platform element can be enhanced by leveraging Interoperability. Indeed, diversity is one of the optimal ways to

increases the system robustness by reducing the probability of common faults, such as software faults [149].

Onix data import/export functions [20], ForCES CE–CE interface [36], [150], ForCES Intra-NE cold-standby mechanisms for high availability [151], and distributed data stores [152] define interfaces between controllers. An east/westbound API requires such as the Advanced message queuing protocol (AMQP) [153] an advanced data distribution mechanisms used by an east/westbound API uses DISCO [123], distributed concurrent and consistent policy composition [154] technique, transactional databases and DHTs(as used in Onix [20]), or strong consistency and fault tolerance [137], [152] through advanced algorithms.

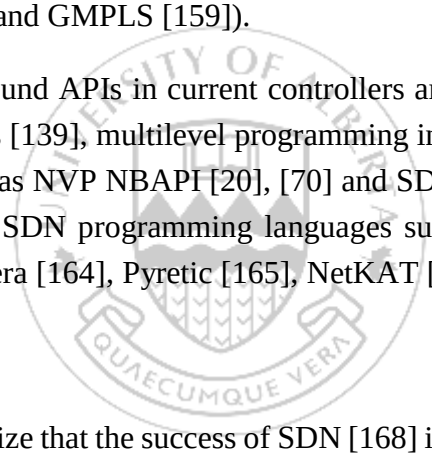
In a multi-domain setup, there is could be a requirement of east/westbound APIs that are more specific communication protocols between SDN domain controllers [155]. Essential functions of such protocols require applications to originate coordinate flow setup,

information to facilitate inter-SDN routing should be exchanged for reach-ability, a reach-ability update to keep the network state consistent, along with some others.

Heterogeneity another vital aspect for situations like besides communicating with peer SDN controllers, controllers also need to communicate with subordinate controllers (in a hierarchy of controllers) and non-SDN controllers [156], as in Closed-Flow [157]. East/westbound interfaces accommodate different controller interfaces to become interoperable, with a specific set of services, and the diverse characteristics of the underlying infrastructure which includes the diversity of technology, the geographical span, and scalability of the network, and the distinction between WAN and LAN across administrative boundaries. In such scenarios, controllers exchange different information which includes adjacency and capability discovery, topology information, billing information [156].

The fine distinction between eastbound and westbound horizontal interfaces[158], where SDN-to-SDN protocols and controller APIs are referred to as westbound interfaces while standard protocols used to communicate with legacy network control planes are eastbound interfaces (e.g., PCEP [50] and GMPLS [159]).

- **Northbound:** Northbound APIs in current controllers are of a wide variety, such as ad hoc APIs, RESTful APIs [139], multilevel programming interfaces, file systems, along with specialized APIs such as NVP NBAPI [20], [70] and SDMN API [160]. Northbound interfaces that stem out of SDN programming languages such as Frenetic [161], Nettle [162], NetCore [163], Procera [164], Pyretic [165], NetKAT [166], and other query-based languages [167].



Finally, it is essential to realize that the success of SDN [168] is due to the control platform. The issue that needs to be addressed is interoperability. It is intriguing that it was the first issue that south-bound APIs (such as OpenFlow) tried to solve. As in case of Wi-Fi and long-term evolution (LTE) networks [107] it needed specialized control platforms such as MobileFlow [160] or SoftRAN [170], data center networks have different requirements that require platforms such as Onix [20] or Open Daylight [25]. Therefore, diversity of networking infrastructures is a reality; there is essentiality for coordination and cooperation between different controllers. APIs that are standardized for multi-controller and multi-domain deployments are thus considered as a significant step.

4.4.5 Layer V: Northbound Interfaces

The two key abstractions of the SDN ecosystem are the northbound and southbound interfaces. Widely accepted proposal (OpenFlow) is already a representation of the southbound interface, but the issue lies with the common northbound interface. Anyway, SDN evolution would see the rise of the common northbound interface. To explore the full

potential of SDN, we need an abstraction that would allow network applications not to depend on specific implementations.

Software eco-system usually defines the northbound interface, not a hardware one, in contrast, south-bound APIs.

The forefront driver for such eco-systems is the implementations, which causes standards to emerge later leading to wide adoption [172]. Even though there is an initial and minimal standard for northbound interfaces but it can still be significant for the future of SDN. Discussions reveal its importance [171]–[178], and vote is positive for northbound APIs are indeed important but too early to define a single standard. Development of various controllers will certainly be the cause for coming up with a common application-level interface.

Application portability and interoperability among the different control platforms could only be possible by the promotion of open and standard northbound interface. POSIX standard [179] in operating systems is a similar northbound interface, an abstraction that guarantees programming language and controller independence. NOSIX [180] is another such instance of an effort in this direction. It defines portable low-level (e.g., flow model) application interfaces, makes southbound APIs such as Open-Flow look like ‘‘device drivers.’’ However, it would not be appropriate to claim that NOSIX is not a general purpose northbound interface, but regarded as a higher level abstraction for southbound interfaces. Indeed, it could be indeed part of the common abstraction layer in a control platform.

Floodlight, Trema, NOX, Onix, and Open Daylight, are some controllers that propose and define their own northbound APIs [173], [181]. Frenetic [161], Nettle [162], NetCore [163], Procera [164], Pyretic [182], and NetKAT [166] are the programming languages that abstract the inner details of the controller functions, and data plane behavior is extracted from the application developers. A wide range of powerful abstractions and application composition alongside fault tolerance in the data plane and numerous basic building blocks to ease software module and application development are some of the abilities of programming languages.

Another northbound interface is SFNet [183]. A high-level API that is used to translate application requirements into lower level service requests. However, due to limited scope, targeting queries to request the congestion state of the network and services like bandwidth reservation and multicast.

Different approaches form different proposals allow applications to interact with controllers. The idea of exploring a general control platform based on Linux and abstractions such as the virtual file system (VFS) comes from the Yanc control platform [135]. The development of SDN applications is simplified as programmers are able to use a traditional concept (files) to communicate with lower level devices and subsystems.

There is no single northbound interface that emerges as the winner, due to different requirements for different network applications. For instance, APIs for security applications would surely differ from those for routing or financial applications. Vertically oriented proposals are one such possibility for the evolution for the northbound APIs, so before any type of standardization occurs, the ONF undertook the challenge for making NBI WG in parallel to open-source SDN developments [57]. The architectural work [156] of ONF includes the possibility of north-bound APIs providing resources in order to enable dynamic and granular control of the network resources from customer applications, eventually across different business and organizational boundaries.

PANE controller [136] also provide with one kind of API. Module-specific quotas and access control policies that put on network resources are defined by a network administrator in PANE. Network resources can be requested by the end-host applications dynamically and autonomously by the API of PANE controller. For example, can easily be modified to use the PANE API can modify audio (e.g., VoIP) and video applications for reserving bandwidth for certain quality guarantees during the communication session. To make sure that bandwidth requests do not exceed the limits set by the administrator and to avoid starvation it has a compiler and verification engine, i.e., to make sure that other applications will not be impaired by new resource requests.

4.4.6 Layer VI: Language-Based Virtualization

The capability of expressing modularity and allowing different levels of abstractions while still guaranteeing de-sired properties like protection are the two essential characteristics of virtualization solutions. For instance, different views of a single physical infrastructure are allowed through virtualization techniques. As an example, a combination of several underlying forwarding devices could be represented by one virtual ‘‘big switch’’. The task of application developers is intrinsically simplified through this as they do not need to worry about the sequence of switches and where forwarding rules have to be installed, but rather consider the network as a simple ‘‘big switch.’’ Developing and deploying complex network applications are eased through such abstraction, such as advanced security-related services.

An intriguing instance of a programming language that offers this type of high-level abstraction of network topology is Pyretic [182]. Introducing network objects incorporates its concept of abstraction. An abstract network topology is inside, and the sets of policies applied to it. Information is hidden inside network objects, and required services are offered.

Static slicing is the language-based virtualization. Here, based on application layer definitions network is sliced by the compiler. A monolithic control program that already has slicing definitions and configuration commands for the network is the output of the compiler. In such situations, the hypervisor is not needed to dynamically manage the

network slices. Deployments with specific requirements would need static slicing, more in situations where higher performance and simple isolation guarantees are preferred as to dynamic slicing.

Splendid isolation is one such instance of static slicing [184]. In this, the network slices consist of three components:

1) Topology, which consists of switches, ports, and links;

2) Network infrastructure mapped with slice-level switches, ports, and links; and

3) Each port of the slice's edge switches has an associated predicate depending on the packets.

The topology has sliced nodes, ports, and links that represents a simple graph. Translation of the abstract topology elements into the corresponding physical ones will be done through mapping. To determine whether a packet is permitted to enter a specific slice is done by predicates. Each slice associates itself with a different application. A global configuration for the entire network is generated by the compiler that combines slices (topology, mapping, and predicates) and respective programs. It also ensures properties such as isolation are enforced among slices, i.e., there is traversing of the packet from slice A to slice B unless explicitly allowed.

libNetVirt [185], another solution for creating static net-work slices to integrates heterogeneous technologies. It is a flexible way to create and manage virtual networks in different computing environments through its designed library. OpenStack Quantum project [186] designed for OpenStack (cloud environments), is similar to libNetVirt but the latter is a more general purpose library which can be applied in different environments. Additionally, it goes one step ahead of OpenStack Quantum as it enables QoS capabilities in virtual networks [185].and with a generic network interface and technology-specific device drivers (e.g., VPN, MPLS, OpenFlow), the libNetVirt library consists of two layers: The network applications and virtual network descriptions are on the top layers. A NOX controller, an OpenFlow driver, manages the underlying infrastructure, to create isolated virtual networks by using OpenFlow rule-based flow tables. Using it as a bridging component in heterogeneous networks with the support of different technologies.

libNetVirt supports heterogeneous technologies, which does not restrict its application to OpenFlow-enabled networks. One per network slice by FlowVisor, AutoSlice, and OpenVirteX allow multiple controllers. A container-based approach is provided by FlowN where multiple applications from different users can coexist on a single controller. VLAN PCP bits for priority queues provisions QoS through FlowVisior. Methods for guaranteeing QoS are also provided by SDN VE and NVP.

4.4.7 Layer VII: Programming Languages

The significant shift on the computer industry [187], [188] is to drive more portable, and reusable code is because of the transition to high-level and powerful programming languages such as Java and Python from low-level hardware-specific machine languages, such as assembly for x86 architectures over the past 10 years

A similar trend if shift can be seen in network programming from low-level machine languages like OpenFlow (‘‘assembly’’) to high-level programming languages [70], [140], [141], [163], [162], [164], [165]. OpenFlow [22] and POF [37] the assembly-like machine language, [120], imitated the behavior of forwarding devices, that forced developers to spend too much time on low-level details rather than on problem-solving. Raw OpenFlow programs dealt with hardware behavior details such as over-lapping rules, the priority ordering of rules, and in-flight packets that caused inconsistency in the data plane, whose flow rules are under installation [161], [163], and [189]. It becomes difficult to reuse software, to create modular and extensive code, and leads to a more error-prone development process [165], [190], [191] because of the low-level language.

The high-level programming languages provide an abstraction that can be vital for addressing many of the issues of these lower level instruction sets [140], [161], [163], [162], [164], and [165]. In SDNs, high-level programming languages can:

- The task of programming forwarding devices can be simplified;
- Speeding up development and innovation and create a more productive and problem-focused environment for network programmers;
- Network control plane could have software modularization and code reusability;

Development of Network virtualization can be fostered. Programming languages in SDNs address several challenges much better. In the case of pure OpenFlow-based SDNs, there is a possibility that multiple tasks of a single application (e.g., routing, monitoring, access control) might interfere with each other. Like the functionality of one task [141], [189] should not be overridden by rules generated by another task. Another instance could of multiple applications running on a single controller [192], [165], [189], [193], and [194]. The rules generated by each application for its own needs and policies without further knowledge about the rules generated by other applications. As a result, rules that are generated and installed in forwarding devices could be conflicting, which is the cause of issue for network operation. Resolving this situation is by the help of programming languages and runtime systems.

Code modularity and reusability design techniques are very hard to achieve using low-level programming models [165]. Thus, the built of applications is monolithic and consist of building blocks that cannot be reused in other applications. Therefore, the development process is very time-consuming and error-prone. Another intriguing feature is the capability of creating and writing programs for virtual network topologies [182].

- Objects encapsulate both data and specific functions for application developers, easing focus on solving a particular problem without considering issue about data structures and their management, a concept derived from object-oriented programming. With context to SDN where instead of generating and installing rules in each forwarding device, simplified virtual network topologies that represent the entire network, or a subset of it could be created. Like an atomic big switch should be considered as the abstract network by the developer, instead of a combination of several underlying physical devices. The programming languages or runtime systems are for generating and installing the lower level instructions required at each forwarding device in a way to enforce the user policy across the network. Due to such abstractions, development of a routing application becomes a straight forward process. Similarly, a set of virtual switches could be represented a single switch, each of them representing a different virtual network. These two instances of abstract network topologies where the low-level instruction sets would be harder to implement. In contrast to a programming language or runtime system abstractions for virtual network topologies, as has already been demonstrated by languages like Pyretic [182].

- **High-Level SDN Programming Languages:** Implementation and abstractions for different important properties and functions of SDN like network-wide structures, distributed updates, modular composition, virtualization, and formal verification [35] are done through this powerful tool.

There are several problems that are found in the low-level instruction set. To address these issues, higher level programming languages proposes:

- Create a network which does not include low-level and device-specific configurations and dependencies, as in case of traditional network configuration approaches;
- Accomplishing different management tasks that are easy to understand and maintain network policies by means of abstractions;
- Multiple tasks (routing, access control, traffic engineering) decoupled;
- In order to avoid low-level instruction sets implementations of higher-level programming interface;
- Automation of solving forwarding rules problems, e.g., conflicting or incomplete rules that can prevent a switch event from being triggered;
- Different race condition issues inherent to distributed systems are addressed;
- Distributed decision makers should enhance conflict resolution techniques on environments;
- Data plane path setup is provided with native fault tolerance capabilities;
- Processing of new flows should reduce new latency;
- Stateful applications (e.g., stateful firewall) creation should be eased.

A way to cope with management requirements, such as monitoring [161], [164], [167], [195] programming language plays a vital role. Installation of rules, polling of the counters, receiving of the responses, combining the results that are needed, along with composing monitoring queries in conjunction with other policies is supposed to be done by runtime system of programming language. In order to easily implement monitoring modules or applications, application developers should utilize the simplicity and power of high-level query instruction.

The portability of the programming language which becomes a necessity for the developers as it does not require them to re-implement applications for different control platforms. It is considered a tool for significantly adding value to the control plane ecosystem. Decoupled back–ends mechanism provides a key architectural ingredient to enable platform portability. As in Java virtual machine, where a portable northbound interface which allows applications to run on different controllers without any kind of modification. Like, pyretic language requires only a standard socket interface and a simple OpenFlow client on the target controller platform [165].

SDNs have a proposal of several programming languages that proposes abstraction for OpenFlow-enabled networks. The declarative one has the predominant programming paradigm; the only exception is Pyretic, which is an imperative language. Most declarative languages have instances of logic and relative types, but they all are functional. As the end goal is almost always the same, but their expressiveness power varies from language to language and their intention to solve the problem of providing higher-level abstractions for the development of network control logic.

The functional and reactive programming languages are FML [140], Nettle [162], and Procera [164]. The reactive actions, i.e., triggered by events (e.g., a new host connected to the network, or the current network load) provides a base for writing policies and applications. So, different network configuration rules such as access control lists (ACLs), virtual LANs (VLANs), and many others could be expressed declaratively by the use of such languages. Rules like allow-or-deny policies are applied to ensure the desired network behavior of the forwarding elements.

Frenetic [161], hierarchical flow tables (HFTs) [189], NetCore [163], and Pyretic [165], are such SDN programming languages that are designed to provide a way of efficiently expressing packet-forwarding policies and deal with overlapping rules of different applications, offering advanced operators for parallel and sequential composition of software modules. Different integer priorities are assigned by Frenetic in order to avoid overlapping conflicts by using overlapping patterns while on the other hand, HFT makes use of hierarchical policies with enhanced conflict-resolution operators. It also has an intriguing feature of seeing-every-packet abstractions and race-free semantics. The former makes sure the availability of all control packets for analysis, and the later uses techniques to suppress unimportant packets. For instance, where packets arise from a network race

condition, in case of concurrent flow rule installation on switches, are simply discarded by the runtime system.

Programming languages such as Pyretic [165] helps to bind the key characteristics for the advanced operators for parallel and sequential composition. Operating multiple policies on the same set of packets is made feasible due to parallel composition, while sequential workflow of policies to be processed on a set of packets is facilitated by sequential composition. The later also allows multiple modules (e.g., access control and routing) to operate in a cooperative way. By applying the former, we can build complex applications out of a combination of different modules (in a similar way as pipes can be used to build sophisticated Unix applications).

FatTire [196] declarative programming language that heavily relies on regular expressions to allows programmers to describe network paths with fault-tolerance requirements and relies on a regular expression. As in the case of each flow can have its own alternative paths to deal with the failure of the primary paths. Interestingly, this feature is founded in a very programmer-friendly way, in a way that application programmer uses regular expressions with special characters, such as an asterisk. Particularly in FatTire, an asterisk will produce the same behavior as a traditional regular expression but will translate into alternative traversing paths.

FlowLog [190] and Flog [191] programming language brings the feature of model checking, dynamic verification, and stateful middleboxes. Like in Flog, where it is possible to build a stateful firewall application with only five lines of code [191].

A unified framework for controlling different network components, such as forwarding devices, middleboxes, and end-hosts is done by Merlin [198]. It also supports backward compatibility with existing systems. For this, it generates specific code for each type of component as it takes a policy definition as input, use compiler to determine forwarding paths, transformation placement, and bandwidth allocation. The compiled outputs basically consist of sets of component-specific low-level instructions that are installed in the devices. Merlin's policy language give permission to operators to delegate the control of a sub-network to tenants along with ensuring isolation. Although this delegated control can be further refined by each tenant owner, which allows them to customize policies for their particular needs.

Recent initiatives of system programming language detect aberrations to enhance the security of network protocols (e.g., Open-Flow), apart from it optimization of horizontal scalability to achieve high throughput on applications that are based on multicore architectures [197] Nevertheless, investigation and development on programming languages have a further lot of scope. As in a case where the researchers found that most of the modification of priority field [199] is done because of some rules update by current policy compilers.

The network management applications built on top of the infrastructure will yield value to SDN. A prolific SDN application development ecosystem can only succeed because of the advancement in high-level programming. Efforts on this are still undergoing to shape forthcoming standard interfaces (cf. [200]) and towards the realization of integrated development environments (e.g., NetIDE [201]) with the aim of fostering the development of a myriad of SDN applications.

4.4.8 Layer VIII: Network Applications

‘‘Network brains’’ that implement the control logic which gets translated into commands to be installed in the data plane, and it dictates the behavior of the forwarding devices. Like in a simple application as routing for instance where the logic of this application is to define the path through which packets will flow from point A to point B. So in order to achieve it, a routing application has to, based on the topology input, so as to decide on the path to use and whatever the chosen path, from A to B, it gives instruction to the controller to install the respective forwarding rules in all forwarding devices

SDN can be deployed on any traditional network environments, whether it is home and enterprise networks or data centers and Internet exchange points. A wide array of network applications such as routing, load balancing, and security policy enforcement are performed alongside exploring novel approaches, like reducing power consumption. Other features like fail-over and reliability functionalities to the data plane, end-to-end QoS enforcement, network virtualization, mobility management in wireless networks are also some functionalities among many others and these are combined with real case deployments.

The five categories on the basis of use cases SDN is grouped are traffic engineering, mobility, and wireless, measurement and monitoring, security and dependability and data center networking.

Traffic Engineering: ElasticTree [203], Hedera [204], OpenFlow-based server load balancing [205], Plug-n-Serve [206] and Aster*x [207], In-packet Bloom filter [208], SIMPLE [209], QNOX [210], QoS framework [212], QoS for SDN [211], ALTO [213], ViAggre SDN [214], ProCel [215], FlowQoS [216], and Middlepipes [33] are among the several traffic engineering applications that have proposed so far. The proposal also includes optimization of rules placement [217], efficient routing in data centers [218] through the use MAC as a universal label, various flow management techniques, fault tolerance, topology update, and traffic characterization [219]. Engineering traffic with the aim of minimizing power consumption, maximizing aggregate network utilization, optimizing load balancing, and other generic traffic optimization techniques is the main aim of such applications

SDN/OpenFlow envisioned Load Balancing as its first applications in which different algorithms and techniques were proposed for this purpose [205], [207], and [206]. Scalability of these solutions was one particular concern. The use of wildcard-based rules to perform proactive load balancing [205] was techniques applied for scaling applications. Wildcards aggregate client requests based on the ranges of IP prefixes, like for every new flow allowing the distribution and directing of large groups of client requests without requiring controller intervention. When traffic bursts are detected, it uses operation in reactive mode. Monitoring the network traffic and use some sort of threshold in the flow counters to redistribute clients among the servers is done by controller applications when bottlenecks are likely to happen.

Network services placed in the network [206] is simplified by SDN load balancing. The load balancing service takes the appropriate actions to seamlessly distribute the traffic among the available servers, considering both the network load and the availability of computing capacity of the respective servers, every time a new server is installed. Thus simplifying network management and providing better flexibility to network operators.

In order to actively monitoring the data plane load existing southbound interfaces are deployed. Optimizing the energy consumption of the network [203] can be leveraged upon such information. Specialized optimizing algorithm is deployed with the aim of meeting certain criteria like latency, performance, and fault tolerance, while reducing power consumption and moreover, diversified configuration options like applying simple techniques are in use that shuts downlinks and devices intelligently in response to traffic load dynamics, this saves approximately 50% of the network energy in normal traffic conditions for data center operators [203].

Avoiding or mitigating the effect of network bottlenecks on the operation of the computing services offered is the significant aim of data center networks. A technique for traffic patterns that stress the network by exploring path diversity in a data center topology is Linear bisection bandwidth which proposes in an SDN setting, allows the maximization of aggregated network utilization with minimal scheduling overhead [204]. A fully automated system that controls the configuration of routers is also provided by SDN which is useful in scenarios where virtual aggregation [220] is applied. This provides a way for network operators to reduce the data replicated on routing tables, which is the cause of routing tables' growth.

Another interesting and fruitful application for large-scale service providers is traffic optimization, which requires dynamic scale-out. Protocols such as ALTO [221] can be utilized for dynamic and scalable provisioning of VPNs in cloud infrastructures, and are simplified through an SDN-based approach [222]. Optimizing rules placement can increase network efficiency [217] according to the latest findings. ProCel [215] was a solution designed for cellular core networks, which was capable of reducing the signaling traffic up to 70% and was a significant achievement.

Routing and traffic engineering performed by other applications include application-aware networking for video and data streaming [223], [224] and employing multiple packet schedulers [225] for improving QoS and other techniques [226], [210], [212], [227]. Traffic engineering is a crucial issue in various kinds of networks, upcoming methods, techniques, and thus innovations can be expected in the context of SDNs.

Mobility and Wireless: Wireless networks' current distributed control plane is suboptimal for managing the limited spectrum, allocation of radio resources, implementations of handover mechanisms, managing interference, and for efficient load balancing between cells. SDN-based approaches provide a pathway making it easier to deploy and manage different types of wireless networks, such as WLANs and cellular networks [170], [228], [229], [230], [231], and [232].

Even though hard-to-implement traditionally but desired features are indeed becoming a reality with the SDN-based wireless networks which provide efficient handovers with seamless mobility [229], [231], [233], load balancing [170], [229], creation of on-demand virtual access points (VAPs) [234], [229], down-link scheduling (e.g., an OpenFlow switch can do a rate shaping or time division) [234], dynamic spectrum usage [234], enhancing inter-cell interference coordination [234], [231], device-to-device offloading (i.e., decide when and how LTE transmissions should be offloaded to users adopting the device to device paradigm [229]) [235], resource block allocations per base station or client (i.e., time and frequency slots in LTE/orthogonal frequency-division multiple access (OFDMA) networks, which are known as resource blocks) [170], [228], [232], controlling and assigning transmission and power parameters in devices or in a group basis (e.g., algorithms for optimizing the transmission and power parameters of WLAN devices defining and assigning transmission power values to each resource block, at each base station, in LTE/OFDMA networks) [170], [228], providing simplified administration [170], [229], [230], easily managing of heterogeneous network technologies [170], [230], [236], interoperability between different networks [232], [236], sharing wireless infrastructures [236], providing seamless subscriber mobility and cellular networks [231], QoS and making access control policies more feasible and easier [231], [232], and making deployment of new applications hustle free [170], [229], [236].

These features in wireless networks can only be realized by providing programmable and flexible stack layers for wireless networks [170], [237]. OpenRadio [237] a fine example that proposes a software abstraction layer that decouples the wireless protocol definition from the hardware, which allows shared MAC layers across different protocols using commodity multi-core platforms. It can be termed as the ''OpenFlow for wireless networks.'' Similarly, SoftRAN [170] reconsider the radio access layer of current LTE infrastructure with the goal of allowing operators to improve and optimize algorithms for providing better handovers, fine-grained control of transmit powers, allocation of the resource block, various management tasks.

Light virtual access points (LVAPs) a medium for improving the management capabilities of wireless networks, as in Odin framework [229] which works with existing wireless hardware and no change is imposed on IEEE 802.11 standards. It is a unique basic service set which is implemented as an identifier associated with a specific client, which implies one-to-one mapping between LVAPs and clients. The managing of client associations, authentication, handovers, and unified slicing of both wired and wireless portions of the network is simplified per-client access point (AP) abstraction. Control logic isolation between slices is achieved by Odin, while LVAPs are considered as the primitive type upon which applications make control decisions, and they cannot view LVAPs from outside their slice. Odin applications in a way empower infrastructure operators to provide services like mobility manager, client-based load balancer, channel selection algorithm, and wireless troubleshooting application within different network slices.

A situation where a movement of the user from one AP to another make the network mobility management application to automatically and proactively act and move the client LVAP from one AP to the other. Thus, wireless client will not even know it is using a different AP as there is no perceptive handoff delay, as it was previously experienced in traditional wireless networks.

SDN also targets a very dense heterogeneous wireless networks although these DenseNets have limitations due to constraints like radio access network bottle-necks, control overhead, and high operational costs [228].

To address some of these constraints [228], a dynamic two-tier SDN controller hierarchy can be adapted. Powering fast and fine-grained decisions by Local-Controllers, while making regional (or ‘‘global’’) controllers to have a broader, coarser grained scope, i.e., that take slower but more global decisions. In such a way, providing a feasible design with a single integrated architecture that encompasses LTE (macro/pico/ femto) and Wi-Fi cells, while challenging.

Measurement and Monitoring: Such solutions are defined into two classes: first, where new functionality is provided for other networking services by the applications; and second, to enhance features of OpenFlow-based SDNs, in order to reduce control plane overload because of the collection of statistics.

Improving the visibility of broadband performance [19], [238] is an example of this first class of application. New functions added in measurement systems such as BISmark [238] can be simplified by SDN-based broadband home connection, which makes the system to react to changing conditions in the home network [19]. Like a home gateway can perform reactive traffic shaping as in accordance with the current measurement results of the home network.

Variety of sampling and estimation techniques to be applied in the second class of solutions, so as to reduce the burden of the control plane with respect to the collection of data plane

statistics. Stochastic and deterministic packet sampling techniques [239], traffic matrix estimation [195], wildcard rules fined-grained monitoring [240], two-stage Bloom filters [241] to represent monitoring rules and provide high measurement accuracy without incurring in extra memory or control plane traffic overhead [242], in order to reduce traffic and processing load on the control plane [243] special monitoring functions(extensions to OpenFlow) in forwarding devices, are techniques are applied to achieve this goal. Network design and operational tasks like load balancing, anomaly detection, capacity planning, and network provisioning can be eased with point-to-point traffic matrix estimation. It makes it possible to construct a traffic matrix using diverse aggregation levels for sources and destinations [195] with the information on the set of active flows in the network, routing information (e.g., from the routing application), flow paths, and flow counters in the switches.

A stronger decoupling between basic primitives matching and counting) and heavier traffic analysis functions such as the detection of anomaly conditions attacks [244] are some other initiatives of this second-class. Portability and flexibility enhance stronger separation. For example, the basic primitives or specific hardware implementation should not constraint functionality to detect abnormal flows. That means developers should be empowering developers with streaming abstractions and higher-level programming capabilities.

Some of the data and control plane abstractions are specifically designed for measurement purposes. Flexibility for network measurements is provided by specially designed by OpenSketch [245] a special-purpose southbound API. In situations where multiple measurement tasks are executed concurrently without impairing accuracy. The internal design of an OpenSketch switch is a pipeline with three stages (hashing, classification, and counting). Hashing function passes the input packets. Then, matching rule classifies them accordingly. Finally, the counting index is identified with match rule, which calculates the counter located in the counting stage. While a TCAM with few entries in the classification stage, the flexible counters are stored in SRAM. This enhances the efficiency (fast matching) of OpenSketch's operation and cost-effectiveness (cheaper SRAMs to store counters).

OpenSample [246] and PayLess [247] other monitoring frameworks which provide with mechanisms for delivering real-time, low-latency, and flexible monitoring capabilities to SDN without impairing the load and performance of the control plane. Sampling technologies like sFlow [248] provides with a solution to monitor high-speed networks, and abstract network views yielding high-performance and efficient network monitoring approaches [246], [247], [245] are provided by flexible collections of loosely coupled (plug-and-play) components.

Security and Dependability: In the context of SDNs varied, diverse set of security and dependability proposals is emerging. Efficiently utilizing SDN for improving services required to secure systems and networks, like policy enforcement (e.g., access control, firewalling, middle pipes represented as middle boxes [33]) [33], [64], [249], [250], [251],

detection and mitigation of DoS attacks [252], [253], random host mutation [254] (i.e., mutation of the IP addresses of end-hosts randomly and frequently to break the attackers' assumption about static IPs) [255], monitoring of cloud infrastructures for fine-grained security inspections (i.e., automatically analyze and detour suspected traffic to be further inspected by specialized network security appliances, such as deep packet inspection systems) [256], traffic anomaly detection [252], [253], [239], flow-based network access control [257], fine-grained policy enforcement for personal mobile applications [258], and so on [64], [256], [252], [254], [250], [255], [251], [239]. While others OpenFlow-based solves networks issues like flow rule prioritization, security services composition, protection against traffic overload, and protection against malicious administrators [138], [259], [193], [360], [261].

Two different approaches, one where SDNs is used to improve network security, and another where improving the security of the SDN itself:

1) **SDN implemented to Improve the Security of Current Networks:** Enforcement on the first entry point to the network (e.g., the Ethernet switch to which the user is connected to) is applied by SDN. Similarly, security policy enforcement can be made on a wider network perimeter with the use of programmable devices (without the need to migrate the entire infrastructure to OpenFlow) [250] in a hybrid environment. These applications block malicious actions before entering the critical regions of the network. The detection (and reaction) against distributed denial of service (DDoS) flooding attacks [252], and active security [262] utilizes SDN successfully. The collection of a variety of information from the network, in a timely manner, is made easier by OpenFlow forwarding devices, which simplifies algorithms specialized in detecting DDoS flooding attacks.

   Collecting statistics data from the network and allowing applications to actively program the forwarding devices are capabilities offered by SDNs and are powerful for proactive and smart security policy enforcement techniques such as Active security [262] which proposes a methodology for novel feedback loop for improving defense mechanisms control for a networked infrastructure, and is based on five core capabilities: protect, sense, adjust, collect, and counter. So, with this view active security provides a centralized programming interface for simplifying the integration of mechanisms for detecting attacks by: 1) collection of data from different sources (to identify attacks); 2) convergence to a consistent configuration for the security appliances; and 3) blocking or minimizing the effect of attacks by enforcing countermeasures.

2) **Improving the Security of SDN Itself:** Identification of the critical security threats of SDNs and in augmenting its security and dependability [259], [193], [263] includes recent researches. Classification of applications and using rule prioritization, to ensure that rules generated by security applications will not be overwritten by lower priority applications [259] are the simple techniques applied in early approaches. There are proposals provide a

framework for developing security-related applications in SDNs [193]. Still, there is a long pathway to go in the development of secure and dependable SDN infrastructures [263].

3) **Data Center Networking:** Dependence on highly scalable and efficient data centers forms the strong basis of today's' networking world which includes small enterprises to large-scale cloud providers, most of the existing IT systems and services. Computing, storage, and networking are the significant challenges still posed by this infrastructure. Concerning the latter, designing and deployment of data center should be in such a way that offer high and flexible cross-section bandwidth and low latency, QoS based on the application requirements, resilience to be of the highest level, utilizing resources intelligently so as to reduce energy consumption and overall efficiency improvement, agility in provisioning network resources through the means of network virtualization and orchestration with computing and storage, and so forth [264]–[266]. And many of these issues still remain not figured due to the complexity and inflexibility of traditional network architectures.

   The current state of affairs is expected to change with the emergence of SDN. Data center networking can significantly be benefited from SDN according to earlier studies for solving different problems like live network migration [268], network management improvement [267], [268], avoiding failure eminently [267], [268], rapidly deploying from development to production networks [268], troubleshooting [268], [269], network utilization optimization [270], [271], [267], [269], dynamic and elastic provisioning of middle boxes-as-a-service [33], and minimizing latency of flow setup and reducing controller operating costs [272]. SDN offers Networking primitives for cloud applications, prediction of network transfers of applications [270], [271], mechanisms that provide fast reaction to operation problems, network-aware VM placement [273], [269], QoS support [273], [269], monitoring of real-time network and problem detection [271], [267], [269], enforcement of services of security policies and mechanisms [273], [269], and enabling programmatic adaptation of transport protocols [270], [274].

   Infrastructure providers can expose more networking primitives to their customers by allowing virtual network isolation, custom addressing, and middleboxes placements and virtual desktop cloud applications [273], [275] through the use of SDN. Exploring the potential of virtual networks in clouds to the fullest, virtual network migration another essential feature. Like a traditional virtual machine migrated to a virtual network, there may need to migrate when its virtual machines move from one place to another. Integration of live migration of virtual machines and virtual networks poses the forefront challenges [268]. It is essential to dynamically reconfigure all affected networking devices (physical or virtual) to achieve this goal. As seen possible with SDN platforms, like NVP [70].

   Detection of abnormal behaviors in network operation [267] is another potential application of SDN in the datacenter. Making use of different behavioral models and gathering necessary information from elements involved in data center operation (infrastructure, operators, applications), it is feasible to continuously build signatures for

applications by passively capturing control traffic. Then, the identification of differences in behavior can make use of signature history. Operators reactively or proactively take corrective measures. Every time a difference is detected. This solves the problem of isolating abnormal components and avoiding further damage to the infrastructure.

CHAPTER – 5

CENTRAL OFFICE RE-ARCHITECTED AS A DATACENTER (CORD)

5.1 INTRODUCTION

The rapid increase in bandwidth demand and service expectations poses a great challenge to network operators. As observed from the scenario of AT&T where data traffic increase by 100,000 percent over the period of last eight years, so it overcomes this ultrafast fiber is rolled out to provide access to 100 cities across the US [276]. It is essential to understand that the introduction of a new feature often takes months (delay due to the next vendor product release) and sometimes years (due to the standardization process to run its course).

In response to these challenges, Network operators find ways so as to benefit both the economies of scale (utilizing few commodity building blocks to for infrastructure construction) and the agility (rapid deployment and elastically scale services) that is being enjoyed by commodity cloud providers.

Cloud economies and agility are especially required at the edge of the operator network—In the Telco *Central Office (CO)* which has a diverse collection of purpose-built devices, which are assembled over fifty years, with little coherent or unifying architecture require Cloud economies and agility at the edge of the network operator. Looking at AT&T which currently operates 4700 Central Offices, with up to 300 unique hardware appliances, the significant source of CAPEX and OPEX, along with a barrier that hinders rapid innovation.

So, in order to overcome situation, network operators looking at CORD, an architecture for the Telco Central Office that is combination of Software Defined Networking (SDN),Network Functions Virtualization (NFV),and elastic cloud services, all running on commodity hardware, so as decrease cost, significantly lowering CAPEX/OPEX because agility in the network and enabling rapid service creation and monetization.

So, CORD stands for re-architect the Central Office as a data center. The fundamental idea behind this unify the centers, so the following three related but distinct technology trends:

- The first is about separation of the network's control and data planes in the SDN, making control plane programmable so that it fastens up the innovation. Simplification of forwarding devices that are built using merchant silicon, causing a decrease in expense due to white box switches.
- The second which describes the movement of the data plane from hardware devices to virtual machines causing a reduction in CAPEX costs (through server consolidation and replacing high margin devices with commodity hardware) and OPEX costs (through software based orchestration). In a way potentially enhancing operator agility and the opportunity for innovation.
- The third in which art of building scalable services is defined under a cloud to enable network operators to rapidly innovate, leveraging software based solutions, microservice architecture, virtualized commodity platforms, elastic scaling, and service composition.

Since all three factors (SDN, NFV, Cloud) are vital in reducing costs, it becomes significant to recognize that all three could be sources of innovative (and revenue generating) services that telcos can offer their subscribers. Control plane services (like content centric networking, virtual networks on demand, cloud network binding), data plane services (like, Parental Control, NAT, WAN Acceleration), and global cloud services (like, CDN, Storage, Analytics, Internet of Things) are included.

The CORD proposes to make the Central Office an integral part of every Telco's larger cloud strategy, by enabling them to offer more valuable services by replacing the current purpose built hardware devices with their more agile software-based counterparts that are making CORD's software architecture general enough to support a wide range of services. Access services (e.g., Fiber to the Home) and scalable cloud services (SaaS) are included; Services implemented in the data plane (NFV) and services implemented in the control plane (SDN) ; Trusted operator provided services and un-trusted third party services; And bundled legacy services and disaggregated greenfield services.

5.1.1 Commodity Hardware

CORD hardware is a collection of commodity servers which are interconnected by a fabric constructed from whitebox switches. The two salient features of the hardware configuration:

- The switching fabric is organized in a leaf spine topology (a topology composed of leaf switches (to which servers and storage connect) and spine switches (to which leaf switches connect). Leaf switches mesh into the spine, forms the access layer to delivers network connection points for servers.

(https://searchdatacenter.techtarget.com/definition/Leaf-spine)) So as to optimize traffic flow from east to west that is between the access network (connects customers to the Central Office) and the upstream links (connect the Central Office to the operator's backbone). So, the north-south traffic is obsolete.

- The proprietary and closed hardware that is traditionally used to connect millions of subscribers to the Internet is being replaced with an open software defined solution with the racks of GPON (Gigabit Passive Optical Network is a point-to-multipoint access network) OLT(a device which serves as the service provider endpoint of a passive optical network) MACS commoditize connectivity to the access network.

Fig5.1. Target hardware built from commodity servers, I/O Blades, and switches.

Hardware elements, organized into a rackable unit called a POD is a reference implementation of CORD. CORD POD is a component of:

1. QUANTA STRATOSS210X12RSIU servers with each one configured with 128GB of RAM, 2x300GB HDDs, and a 40GE dual port NICs that are qualified as Open Compute Project (OCP)

2. OCP qualified, and OpenFlow enabled Act on 6712 switches that are configured with 32x40GE ports that work on leaf and spine switch topology in CORD fabric.

3. ODM for "OLT pizza box is Celestica" PMC Sierra OLT MAC chip has I/O blades. that are 48x1Gbps GPON interfaces and 2x40GE uplinks.

Atrium software stack [277] consists of Open Network Linux, the Indigo OpenFlow Agent (OF 1.3), and the OpenFlow Data Plane Abstraction (OFDPA), layered on top of Broadcom merchant silicon I/O blades for 10GPON and G.Fast are not in the initial plan but are part of the near future.

Differently configuring the POD hardware is possible. As the selected leaf switches have sufficient capacity to support up to 24 dual port servers and similarly the spine switches can support up to 16 racks. Moreover, it is possible to configure a "micro POD" that includes only leaf/ToR switches and fits in a partial rack on the other end of the spectrum.

5.1.2 Software Building Blocks

The reference implementation of CORD exploits four open source projects with respect to software, as depicted:

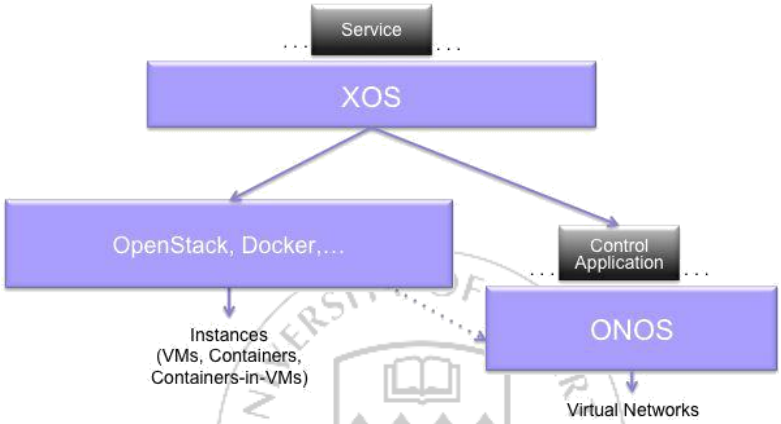


Fig 5 2. Open source software components used to build CORD.

- **OpenStack** [278] the core IaaS capability is provided by the cluster management suite, and creation and provision of virtual machines (VMs) and virtual networks (VNs) is also the other responsibility.

- Deployment and interconnect services are provided by **Docker** [279] through a container based means. It is used in configuring and deploying CORD itself (e.g., the other elements—XOS, OpenStack, and ONOS—are instantiated in Docker containers on the PODhead nodes.

- The management of the underlying whitebox switching fabric is done by **ONOS** [280], the network operating system. The control applications that implement services on behalf of Telco subscribers and embedded virtual networks in the underlying fabric are both hosted by it, which is in turn accessed via OpenStack's Neutron API.

- Assembling and composing services is the base of **XOS** [281]. Unification of infrastructure services (provided by OpenStack), control plane services (provided by ONOS), and any data plane or cloud services (running in OpenStack provided virtual machines and Docker provided containers) is the function of XOS.

The widest possible collection of services is only possible when the reference implementation supports services running in virtual machines (KVM) while containers are running directly on bare metal (Docker), and in containers nested inside virtual machines (Docker in KVM).

ONOS also interconnects VMs (this includes implementing VNs and managing flows across the switching fabric) and hosting control programs that implement first class CORD services are provided by the ONOS platform.

5.1.3 Transformation Process

Given this hardware/software foundation, Transformation of the current Central Office into CORD can be two step process because of the hardware/software foundation. The first step is virtualizing the devices, in which hardware device are turned into its software counterpart that is running on commodity hardware. Disaggregate and refactor the functionality bundled in the legacy devices is the essential key for this implementation, along with the decision of what is being implemented in the control plane and in the data plane.

In the second step, a framework is provided to the virtualized software elements along with any cloud services which the operator wants to run in the Central Office, and that can be plugged into for a coherent end to end system. This framework is a collection of hardware and software elements into a system that is economical, scalable, and agile.

5.2 Virtualizing Legacy Devices

The initial step is the virtualization of the existing hardware devices along with transforming each legacy device into its commodity hardware also the software service counterpart in pursue towards re-architecting the central office as a data center. Disaggregating and repackaging functionalities in new ways during this process.

Fig below highlights the process for the devices which includes Optical Line Termination (OLT), Customer Premises Equipment (CPE), and Broadband Network Gateways (BNG). The Ethernet switch is not virtualized, per se, but the switching fabric effectively replaces it effectively under the control of ONOS.

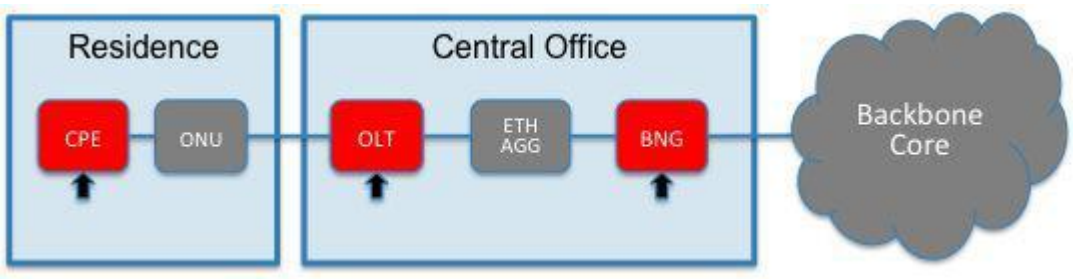


Fig 5 3. Legacy Central Office, including three physical devices to be virtualized.

5.2.1 Benefits and Challenges

A large capital investment which involves racks of closed and proprietary hardware that are terminate access for tens of thousands of subscribers per CO has been put into OLT (Optical Line Network). The challenge is virtualization of OLT as, unlike many network appliances which are implemented by software running on vendor branded commodity servers, while on the other hand OLT is implemented primarily in hardware. Thousands of customer sites per CO are also currently distributed with CPEs, which is also a significant operational burden. The burden is more when a service upgrade requires a hardware upgrade. Historically, most of the aggregation of the functionality provided by a Central Office was done by BNGs that are quite expensive and complex routers. Thus evolution in an agile and cost effective way seems difficult.

Systematically transforming such a diverse collection of devices into software running on commodity hardware is the real challenge faced today. But there is a simple template for how each physical device is virtualized. It possible with the combination of three elements: (1) merchant silicon, including both commodity servers and whitebox switches;

(2) A control plane function, a reference to SDN element; And

(3) A data plane function, with respect to the NFV element.

Both the SDN and NFV are implemented by software running on commodity servers, where if the packet processing is entirely in software then it is considered an NFV element, and if that software also controls commodity switches and me/O blades through an open interface like OpenFlow then it defines SDN.

So, to apply this pattern to OLT, CPE, and BNG, which result in virtual incarnations of each physical device. There is no need for preservation of a one to one mapping between physical and virtual devices, and but the opposite is true.

5.2.2 Virtualizing the OLT

The optical link in the Central Office is terminated by OLT, where each physical termination point is aggregating a set of subscriber connections. As there are a huge number and cost of OLT devices in a CO, virtualizing the OLT could potentially yield significant CAPEX and OPEX savings.

Creating an I/O Blade with the PON OLT MAC is the first challenge, and in order to develop an open specification for a GPON MAC 1RU “pizza box,” AT&T has worked with the Open Compute Project. This board includes a remote control program via OpenFlow controls the essential GPON Media Access Control (MAC) chip.

SDN based control paradigm as the whitebox based switching fabric is then implemented on these I/O blades. Virtual OLT (vOLT) that runs on top of ONOS, and implements all other functionality normally contained in a legacy OLT chassis (e.g., GPON protocol management, 802.1adcompliant VLAN Bridge) is the result of this control program. That is, vOLT authentication is implemented on a per subscriber basis, VLANs connecting the subscriber’s devices to the Central Office switching fabric is established and managed, and alongside other control plane functions of the OLT.

5.2.3 Virtualizing the CPE

A "home router" or "residential gateway," that is installed in the customer's premises is called CPE. As they are larger in numbers, they are an important aspect of CAPEX and OPEX costs, along with the hindrance they pose in introducing new services. Essential functions (like DHCP, NAT) and optional services (such as Firewall, Parental Control, VoIP) are run by them on behalf of residential subscribers. They also deal with, core sophisticated enterprise functions (e.g., WAN Acceleration, IDS). Extension of the capabilities of CPE in the cloud will help in adding new value services along with providing customer care capabilities that were impossible before because of limitations in the hardware.

Virtual Subscriber Gateway (vSG), a virtualized version of CPE, is also responsible for running subscriber selected functions, but on commodity hardware located in the Central Office rather than on the customer's premises. There is still a device in our home (referred to as the CPE), can be reduced to a bare metal switch, and all the functionality can be moved into CO that originally runs on CPE and running in a VM on commodity servers. Or we can say that a remote VM that resides in the central office includes the "customer LAN," in order to effectively provide every subscriber with a direct ingress into the Telco's cloud.

CORD provides a platform for implementation of such choices for subscriber bundles, with a full VM, a lightweight container, or a chain of lightweight containers. The bundle is treated as a whole (roughly corresponding to a VM image or a container configuration) as the standard representation and leaving behind the means which allows subscribers to select the set of functions that need to be included in their bundle for implementing their choice. Like, the reference implementation just allows the subscribers to make a selection from a small collection of functions (e.g., DHCP, NAT, firewall, parental filtering), but their implementation is done through the proper configuration of a container (as defined by a corresponding Docker file). It was observed experimentally that this approach could conservatively support 1000 subscribers per server.

5.2.4 Virtualizing the BNG

A complex and expensive device in a Central Office that allows subscribers to connect to the public Internet is known as BNG. A routable IP address is managed on behalf of each subscriber, along with providing that subscriber with some type of network connectivity. A massive collection of value added features and functions, like VPNs, GRE tunneling, MPLS tunneling, 802.1ad termination, and so on are provided by BNG.

Virtual Router (vRouter), CORD's virtualized BNG, is an implementation of ONOS hosted control program for managing flows through the switching fabric for subscribers. Not a lot of auxiliary functions historically were bundled into a BNG device, although in some cases (e.g., authenticating subscribers), except the functionality of providing another service (e.g., vOLT,)

Generally, it can be assumed that vRouter is the means to provide each subscriber with their own "private virtual router," where the underlying fabric is a distributed router with "line cards" and "backplanes" instantiated by bare metal switches. Then an IP network that routes between the attached, per subscriber subnets is created by the vRouter control program. Peering with legacy routers is also done by vRouter, which includes advertising BGP routes.

Authenticating the user is the prime responsibility of BNG, but that capability has been unbundled and moved to vOLT. As is necessary to authenticate subscribers before accessing vSG, which originally used to reside in their homes but now has been moved into the Central Office.

5.2.5 End-to-End Packet Flow

Assuming that subscriber already has a connection with Telco, a sketch of subscriber's packet flow through the CORD is concluded. An 802.1x authentication packet is sent via GPON to the CO as soon as the subscriber powers up the home router. The ONOS passes up the packet upon arrival at I/O blade port to the vOLT control program, where the subscriber is authenticated using an account registry like RADIUS. After the process of authentication is complete, VLAN tags are assigned by vOLT to the subscriber and the appropriate flow rules are installed in the I/O blade and switching fabric (via ONOS), a vSG spin up a container for that subscriber, and that container is bound to the VLAN. A routable IP address from the router is requested from vSG, which causes vRouter (via ONOS) to install the flow rules in the switching fabric and route packets are software switched to/from that subscriber's container.

As soon as the setup is complete, their starts a packets flow from the home router over a VLAN to the subscriber's container, the processing of packets is according to whatever bundle is associated with the subscriber's account and then using the assigned source IP address it is forwarded on to the Internet.

The high-level description is glossed over both many low-level details about each component (e.g., assignment of VLAN tags, flow rules installation, the exact composition of functions in each container) and the mechanisms that plumb various agents (ONOS, OpenStack, Docker, vOLT, vSG, vRouter) together.

5.3 Service Framework

The next step in rearchitecting the Central Office as a data center that involves orchestrating the software elements which resulted from the first step (plus additional cloud services that are needed by the operator) into a functioning and controllable end to end system.

5.3.1 Benefits and Challenges

The necessary initial step of replacing hardware devices with software running in virtual machines is not sufficient by itself. **Service Orchestration** process which states that just as all the hardware devices in a Central Office must be wired together in a sensible way, so as their software counterparts must also be managed as a collective, but that is possible only if network operators are to enjoy the same agility as cloud

providers, the underlining abstractions in the orchestration framework must fully embrace:

(1) Virtualized functionality should be elastically scaled out, and

(2) The resulting disaggregated (unbundled) functionality should be properly composed.

Introduction of the disparate functionality by virtualizing the hardware devices under a single coherent model is adoption out of **Everything as a Service (XaaS)** as a unifying principle [281]. The control functions run as scalable services run on top of ONOS, that is a scalable network operating system, the data plane functions run as scalable services that scale across a set of VMs, the commodity infrastructure is itself managed as a service referred to as IaaS, along with other global cloud services that are active in the Central Office are managed as scalable services.

5.3.2 Scalable Services, Not Virtual Devices

The virtualized counterpart is all three packaged as services and is termed as vOLT, vSG, and vRouter. These services are named according to their legacy counterparts, the bundling of functionality along the same boundaries as before is not required in the new architecture. Therefore, the virtualization process is outlined as resulting in three generic, multitenant services:

- A control program running on ONOS is vOLT implements AccessasaService for each tenant that corresponds to a Subscriber VLAN.

- vSG, a data plane function is scaled across as a set of containers. Subscriberas-aService where each tenant corresponds to a Subscriber Bundle is implemented under it.

- ONOS runs vRouter control program where the Internetasaservice for each tenant corresponds to a Routable Subnet is implemented here.

A Content Distribution Network (CDN) is a scalable cloud service that is deployed throughout the operator's network, which also includes caches in the Central Offices along with its addition to these three new services, the illustration of the three kinds of services are: a cloud service (CDN), a data plane service (vSG), and two control plane services (vOLT and vRouter).

Service abstraction and multitenant services are provided by CORD, as in the case of conventional cloud storage service which provides a "Volume" abstraction and a "KeyStore" abstraction that a No SQL DB service provides. The service graph below shows: "acquisition of subscriber VLAN from vOLT by the subscriber, which in turn acquires a subscriber bundle from vSG, and finally acquiring routable subnet from vRouter."

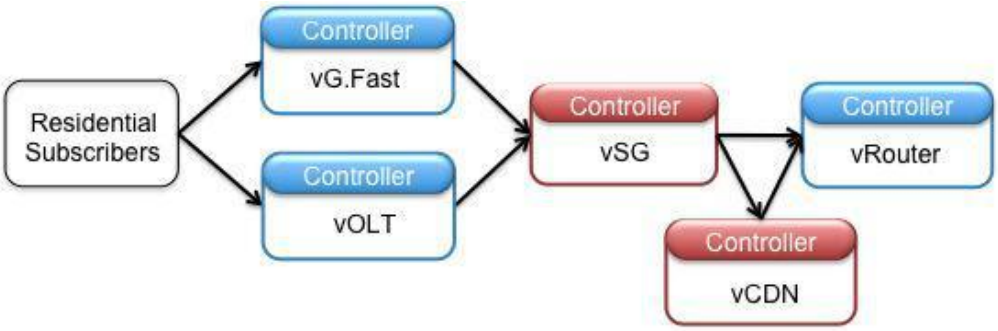


Fig 5.4 CORD service graph, including two access services: vOLT and vG.Fast.

Suggesting that the subscriber is a tenant of the service graph as a whole. Pragmatically, we can say that service graph is configured into CORD which provides the subscriber the facility to control his or her subscription (for instance the parental control feature like disallowing access to Facebook) on the subscriber object by invoking operations, without any prior knowledge of which service implements which feature. The XOS abstractions maps impose a structure in which a request onto the right set of components.

Service graph is simplified so as to focus on the services that provide direct value to end users. The services in service graph depends on the collection of building block service such as ONOS and OpenStack (for instance, we observe a tenant dependency between both vOLT and vRouter and ONOS), along with monitoring service that collects, and aggregates meters and delivers them to analytic engines for fault diagnostics and dynamic steering from each hardware and software element in CORD.

5.3.3 Layers of Abstraction

The high-level specification a network operator is represented through the service graph, but there is a need to map this specification onto the underlying servers, switches, and I/O blades. CORD forms the topmost layer of the building block components abstract the direct consequence of nested collection to make this happen. Operators express and enforce policies on them through the abstractions imposed on a structure on the set of services. CORD defines the following abstractions while working top down,

- Service Graph [120]: a set Dependent relationship among a set of Services is represented through this. Tenancy termed as a relationship between a provider service, and a tenant (i.e., client) service is the service composition of CORD models. Tenant Principal (e.g., subscriber) bounded to one or more User Accounts is defined under Service tenancy.
- Service [121]: represents an elastically scalable, multitenant program, which determines the ways to instantiate, control, and scale functionality are represented in this.

CORD models a service as a Service Controller that exports a multitenant interface and an elastically scalable set of Service Instances that collectively instantiated in a Slice are modeled as CORD service. A CORD ready service from both greenfield and legacy components are assembled through a mechanism which is included in XOS.

- Slice [122]: Represents a system-wide resource container in which services are executed Slice, also specifies ways resources are embedded in the underlying infrastructure. CORD utilizes slice as a set of Virtual Machines (VMs) and a set of Virtual Networks (VNs). Underlying IaaS components implements VMs (OpenStack and Docker), while ONOS implements VNs.

- Virtual Network [123]: Communication interconnection among a set of instances is represented in VNs. Several VN types, like Private (instances are connected within a Slice), Access Direct (tenant service uses it to access a provider service by directly addressing each instance in the provider service), and Access Indirect (tenant service uses it to access a provider service by addressing the service as a whole) are supported by CORD. Service composition is supported by the latter two types.

A pair of control applications run on top of ONOS implements a mechanism underlying CORD's support for Virtual Networks. VTN installs flow rules in the OvS that runs on each server to implement direct or indirect addressing is the initial one. The second is Segment Routing in which aggregate flows between servers across the switching fabric is implemented. Viewing the service graph with respect to NFV where each tenant abstraction in CORD corresponds to a Virtualized Network Function (VNF) in the NFV architecture [282]. The mapping of a sequence of such VNFs (a service chain) onto a sequence of VMs depends on three things:

(1) Whether the service implementation is on the network control plane or in the network data plane,

(2) How its tenants are service mapped onto one or more service instances, and

(3) What type of service instances are used to interconnect virtual network?

A linear chain of instances that corresponds to a single subscriber, many service orchestrators implementing service chaining is considered as just one of many possible outcomes of service composition in CORD. A wide variety of collection of services that span the full NFV, SDN, and Cloud space is that a more general model of service composition that is required, and this experience informs CORD's design.

5.4 Future Plans

Transforming legacy Central Offices in the Telco network is the revolutionary effort of CORD. The vision behind this is the new Central Office re-architected as a data center in which closed and proprietary hardware is replaced with software running on commodity servers and switches. This software will, in turn, be used to manage and orchestrate as a collection of scalable services. The main purpose of CORD is to demonstrate the feasibility of a Central Office that enjoys both the CAPEX and OPEX benefits of commodity infrastructure and the agility of modern cloud providers.

A reference implementation of CORD that plans to deploy it in a residential field trial at AT&T is being built. There is information on the specific requirements of the field trial

that is essential for the reference implementation; the design is a general platform that can be configured for a wide range of deployment scenarios. For instance, a configuration is targeting at Mobile users (MCORD), and another target is the Enterprise users (ECORD).

It should be understood that software plays a critical role in CORD, which leverages OpenStack and Docker a provision for the virtual compute instances, ONOS that manages the switching fabric and host control applications, and XOS that is used to assemble and compose services. Atrium and the Open Compute Project are considered as valuable parts of CORD.

CHAPTER – 6

M-CORD: MOBILE-CENTRAL OFFICE RE-ARCHITECTED AS A DATA CENTER

6.1 INTRODUCTION

The tremendous growth in wireless data traffic with a rapid increase in inexpensive investment on spectrum and LTE network deployment in order to support carrier network. The challenges for all the service provider is the continuation of the explosive growth of user demands alongside flattening of the revenues. Inefficient utilization of network resources is due to the way in which telecom infrastructure is built which is composed of proprietary vertically integrated devices. Further, customization of the current network according to different customer needs or locations has become hard enough, and the current architecture is not efficient enough for the creation of emerging services.

"Networked Society" enablement is the true idea behind 5G, and that is feasible by providing seamless connectivity for people and things [283], and this further adds a pile of challenging requirements for the mobile infrastructure and its providers. For instance, accessing information and ability for data sharing for people along with things at any time and in any location is the main goal of 5G. The only key for enabling the Internet of Things is 5G that is needed to provide connectivity to a massive number of devices alongside stringent energy and transmission constraints. Support mission critical services which require very high reliability and/or low latency is needed to be backed by 5G. Seamless working of a family of radio access technologies (RAT) and densification of small cells needs to be looked upon. Connectivity-based innovative services in numerous vertical sectors, such as health, automotive, home, energy, and many others are in the process of being offered by various mobile operators.

6.2 Overview of the CORD and M-CORD Architecture

As we have already discussed the CORD's mission of bringing data center economies and cloud agility using an open reference implementation with the active participation of the community to support service providers for their residential, enterprise, and mobile customers. Commodity servers, white-box switches, disaggregated access technologies and open source software to provide an extensible service delivery

platform is being used to build the reference implementation of CORD. This will provide network operators with the means to configure, control, and extend CORD in order to meet they're operational and business objectives. The reference implementation is sufficiently complete with supporting field trials. Two ambitious goals that CORD has for the reference implementation: The first being a complete solution, that is completely ready for evaluation in field trials on commercial operator networks and the second one being able to serve general-purpose platform that is able enough for delivering a wide range of innovative services from access services (e.g., 5G, LTE, Fiber-to-the-Home) to conventional cloud services (SaaS); from implementation of service in the data plane (NFV) to the implementation in control plane (SDN); from trusted operator-provided services to un-trusted third-party services; and from legacy services being bundled to greenfield services that are being disaggregated.

We already discussed the four open-source projects that integrate CORD software architecture:

1) IaaS capability being provided through *OpenStack* which is the cloud datacenter management platform,

2) The software platform that allows deployment and interconnection of services inside software containers is *Docker*,

3) *ONOS* is the Open Network Operating System [284] that controls the underlying white-box switch fabric using an organized in a leaf-spine topology, which hosts a collection of control applications and their implementation is necessary for ONOS-controlled services and embedding virtual networks into the underlying fabric for the OpenStack-controlled services,

4) *XOS* is the Anything-as-a-Service (XaaS) Operating System which makes assembling and composing services possible.

Similarly, the focus of M-CORD is to address the needs of mobile networks. The emergence of 5G use cases has influenced its use and has made it programmatically applicable to a range of performance targets on the same platform. M-CORD aims to transform the mobile network with decoupling of SDN control and data planes, making SDN control plane logically centralized, disaggregation and virtualization of cellular network functions as well as operator specific services, scalable services and the overall cellular network is the composition of virtualized functions and services that is orchestrated so that use case-specific set of services are on-boarded and dynamically scaled.

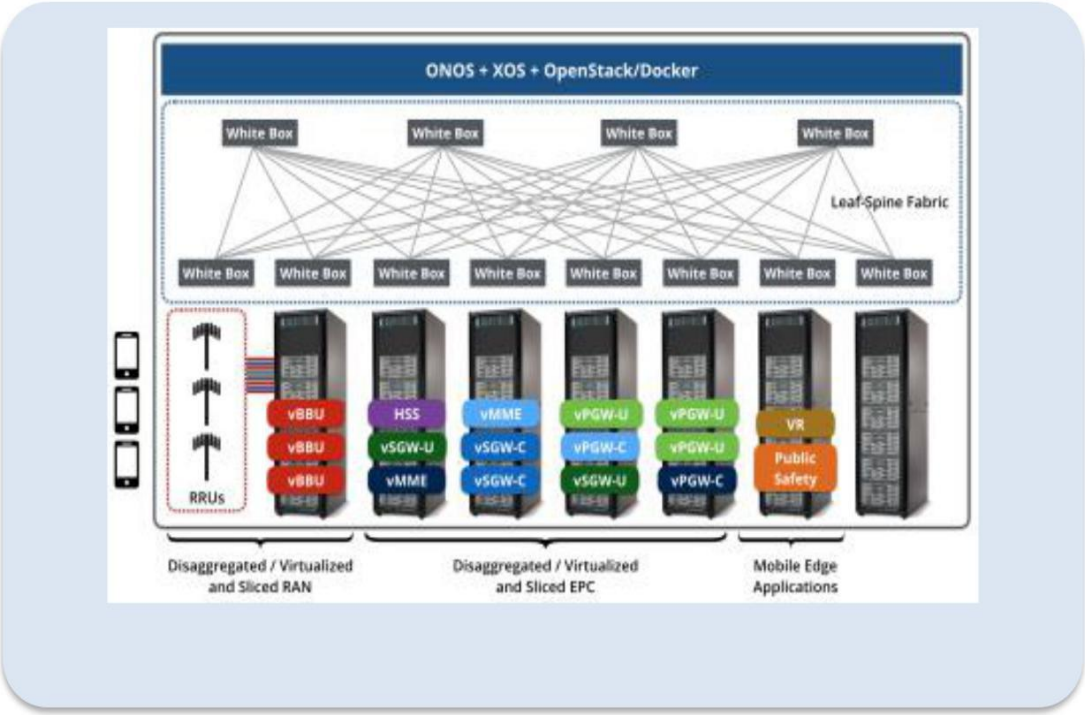


Fig 6.1. LTE over M-CORD allowing for Split RAN and End-to-End Network Slicing

The M-CORD helps by providing ways to allow for testing and development in a coordinated manner. There is a need for restructuring mobile infrastructure so as to enable 5G which is a resource-intensive task; hence M-CORD plays a role by rescuing in this regard and allows for resource utilization, especially spectrum. It helps by enabling the service providers to offer customized service which would lead to better QoE to the customers. The figure below provides the architecture for the M-CORD [285].

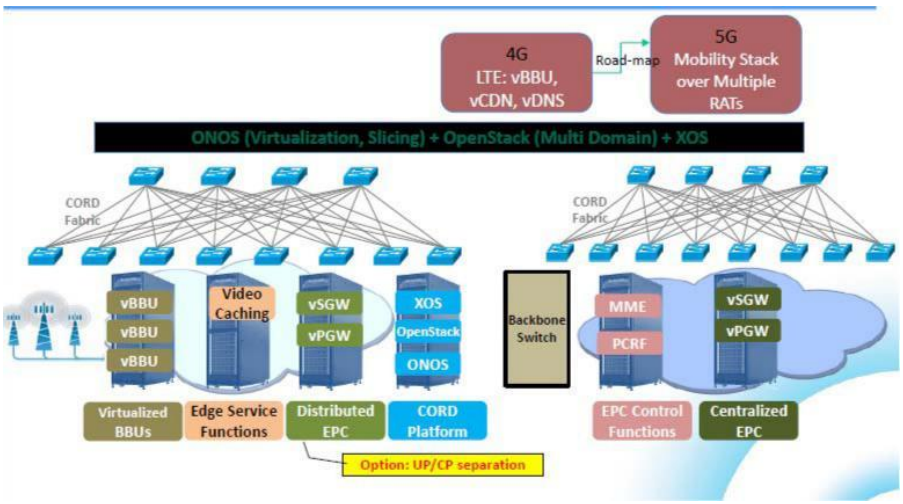


Fig 6.2. The M-CORD architecture

Significant reduction in the capital expenditure (CAPEX) is because of its agile and cost-efficient deployment process; it is known that how CAPEX could be a significant roadblock in the development of new technology. Reduction of capital and operational expenditure would allow the network providers to provide more services at the same cost which in turn enhances QoE. The figure below shows the basic overview of M-CORD [286].

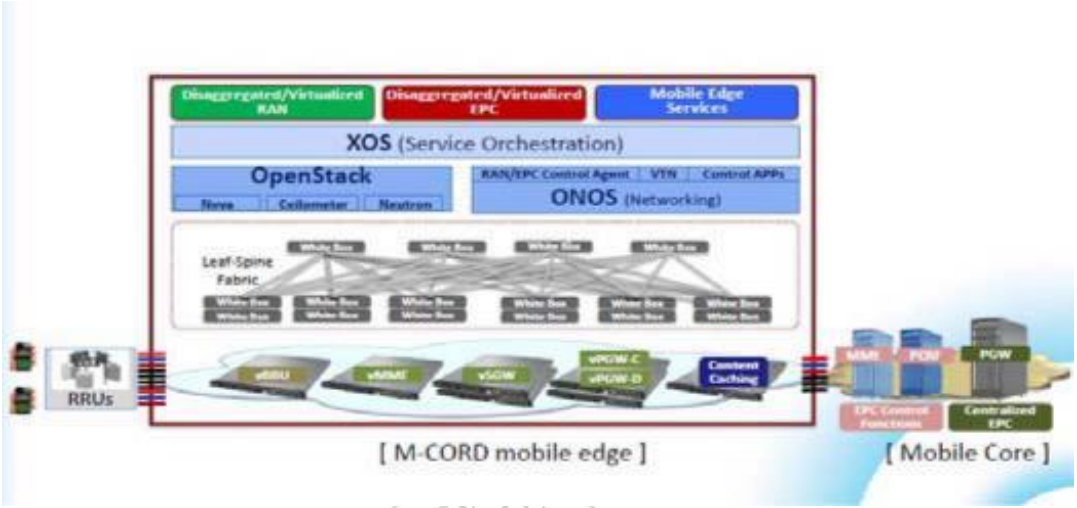


Fig 6.3. Overview of M-CORD

Monitoring as a service (MAAS) and the CORD Controller (XOS) are among some of the open networking solution provided by it.

6.3 USES

Safety as a service (SaaS) would be incorporated in upcoming 5G technology, which would play a crucial for bypassing credit checks and will help in providing maximum bandwidth irrespective of the user's affiliation to a network provider. Sharing location details and video calls [287] would be supported along with something beyond voice calls. The aim of the M-CORD is to enhance resource utilization by providing real-time resource management, the framework being monitored and exploiting the use of multiple Radio Access Technologies (RATs). Virtualization and Disaggregation of RAN and EPC use commodity hardware and open source software would result in a low cost and making deployment efficiently. The architecture of the CORD is the combination of open-source projects like OpenStack, Docker, XOS, and ONOS that are deployed for creating an integrated platform for providing a service delivery platform. Open Network Operation System is the reference of ONOS which applies the leaf-spine topology by controlling the underlying white box switch fabric. XOS operating system performs functions of assembling and composition services. Cloud datacenter management and interconnection

of services inside the software containers are performed by the use of OpenStack and Docker. The aim of this technology is not only at providing service to mobile users but also to a host of devices that would be expected to be connected to IoT in the future. The requirements of bandwidth are expected to grow manifold as a lot of devices would get connected, and that would require a network that would require less use of power. The 5G technology will provide the ability to enable low-power devices to run for up to 10 years without charging again. Significantly low latency would also be another feather which would help in enabling us to circumvent the primary roadblock of network delay in the realization of self-driving cars lowers also latency would make surgeons efficient enough to perform surgery with injected Nanoparticles in real-time. Revampification of the healthcare industry would prove to be a boon for innumerable users. The figure below highlights the uses of 5G

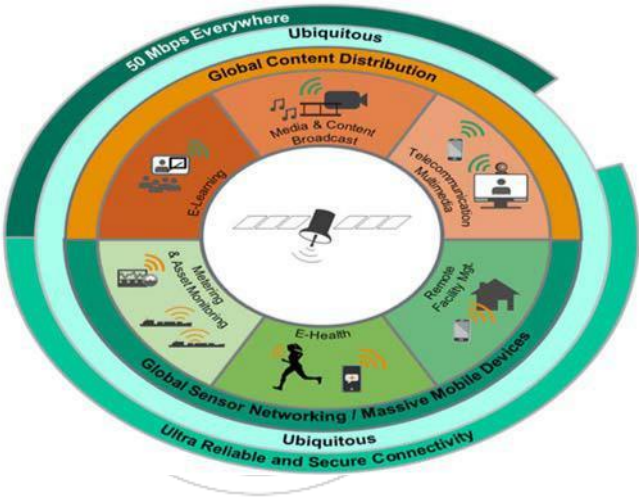


Fig. 6.4 5G Use Case

6.4 CONCLUSION

The demand and essentiality for faster and a better connection is at rising by the day. Going by the current infrastructure as we observe the number of devices that the internet connects today is already proving to be a series of concern to the network providers, and it's not going down by the day. Therefore, there is a requirement of a technology that can facilitate everything alongside without requiring a significant investment which would not only benefit the network providers but will be a lot useful in saving a lot of resources that could not be used possibly otherwise. The M-CORD platform facilitates along with allowing for collaborative development and leverages merchant silicon. Open source solutions are relied upon for this platform which in a way drives down the cost further. M-CORD is very attractive to the developers because it eases the use along with providing

a robust and economic environment. There are certain issues that require addressing, such as, the need for rapid development in the field is dependent on the development of the platform fast. Issues like platform downtime and build need addressing. Many companies have already Investment in this project have been already made by many companies, and they are supporting the development of 5G employing M-CORD. 5G technology would not only connect the world but also would aid technologies that haven't been realized yet because of the current limitations.

CHAPTER – 7

M-CORD Architecture for Traffic Offloading

The software that uses off-the-shelf hardware are the virtualized solution which aims to duplicate the high performance provided by any specific hardware. The RAN virtualization system performs one of the most arduous tasks of Real-time response to the radio frequency (RF) signals. A baseband unit (BBU) and a remote radio head typically form the base for a wireless base station. The remote radio head, also called, the remote radio unit (RRU) placement is usually on the base or the top of the tower whereas the BBU is usually in an equipment closet. In Centralized RAN architecture where all the BBUs are transferred to a centralized location where they share space with other BBUs.

Multiple partitioning approaches for the baseband in recent ways that are trending in the 5G environment which is suitable for small cells that cause a reduction in the setup and maintenance expenditure. The evolution of the current network architecture leading towards the 5G architecture that integrates Virtualized RAN solution as an integral part. An ability to manage and steer the capacity of the network towards high demand areas in a dynamic environment is possible with this setup. Further, this setup outperforms the traditional scenario where the network is set up for facilitating maximum demand at every node. Fig below illustrates the comparison of the traditional and virtualized RAN strategies.

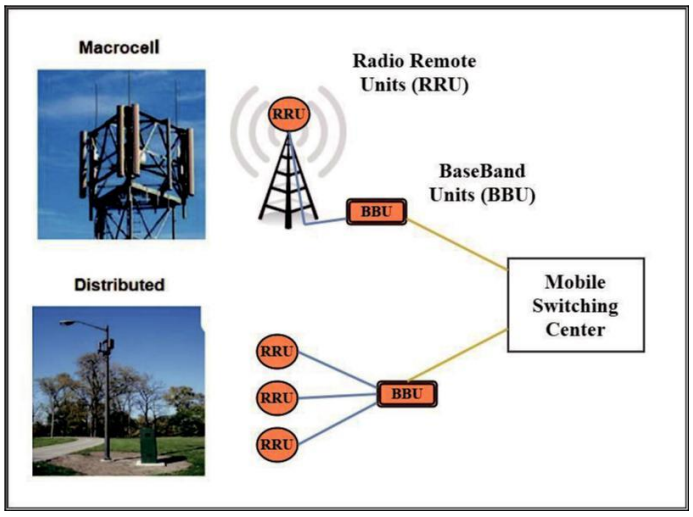


Fig.7.1 Traditional and virtualized RAN strategies

Evolved packet core (EPC) framework was the basis for the invention of 4G LTE Network. The voice packets and data packets in the traditional architecture (2G, 3G) were treated by different sub-domains. Voice traffic used Circuit switching (CS) and data traffic used packet switching (PS). Unification of both voice and data traffic is done by EPC and packet transmission is done by the use of internet protocol (IP) services. The critical components of the EPC are Mobility management entity (MME), Serving Gateway (SGW), policy and charging rules function (PCRF) and packet data node gateway (PGW). The session states and authentication along with user tracking is performed by MME. Routing of the data packets throughout the complete access network is the role of SGW. Data flow detection and enforces policies based on flow-based charging is done by PCRF. The PGW manages QoS by providing an interface between the LTE networks and the previous networks. The SGW and PGW are further divided into two parts based on their placement. The first being in the control plane (SGW-C and PGW-C) and others in the user plane (SGW-U and PGW-U). These functionalities of EPC are the basis of the MEC architecture in a 5G network environment. The traditional and the virtualized architecture of RAN and EPC can be illustrated as:

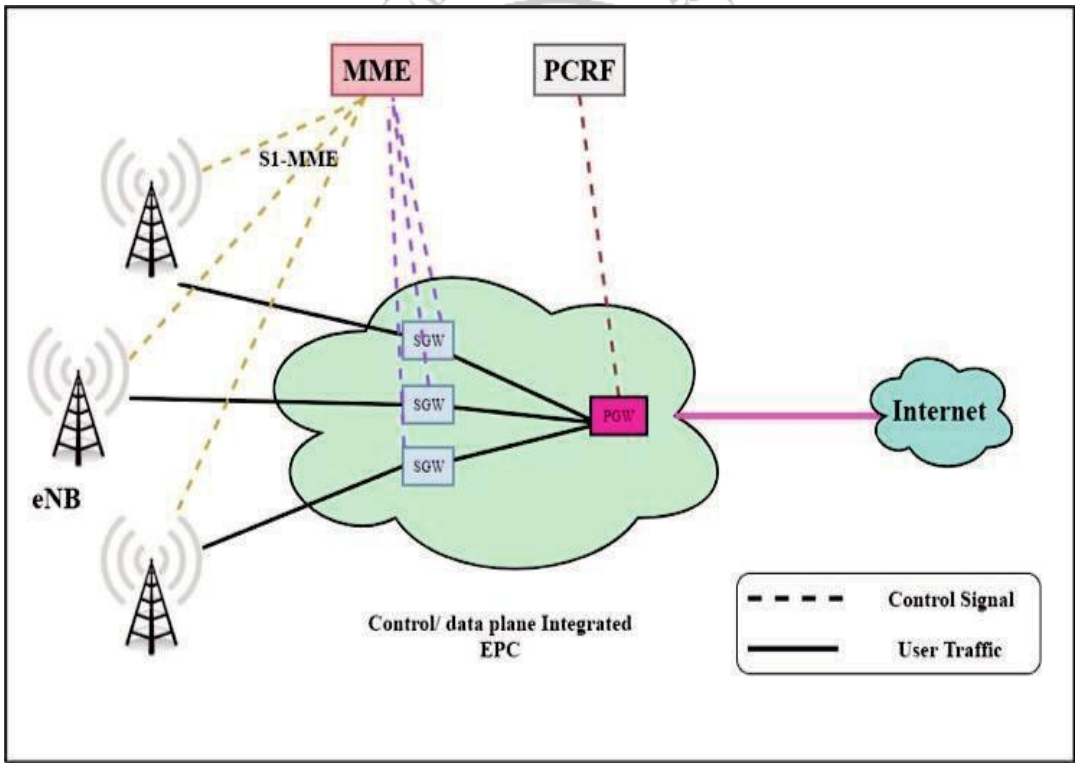


Fig.7.2 Traditional Architecture of RAN and EPC

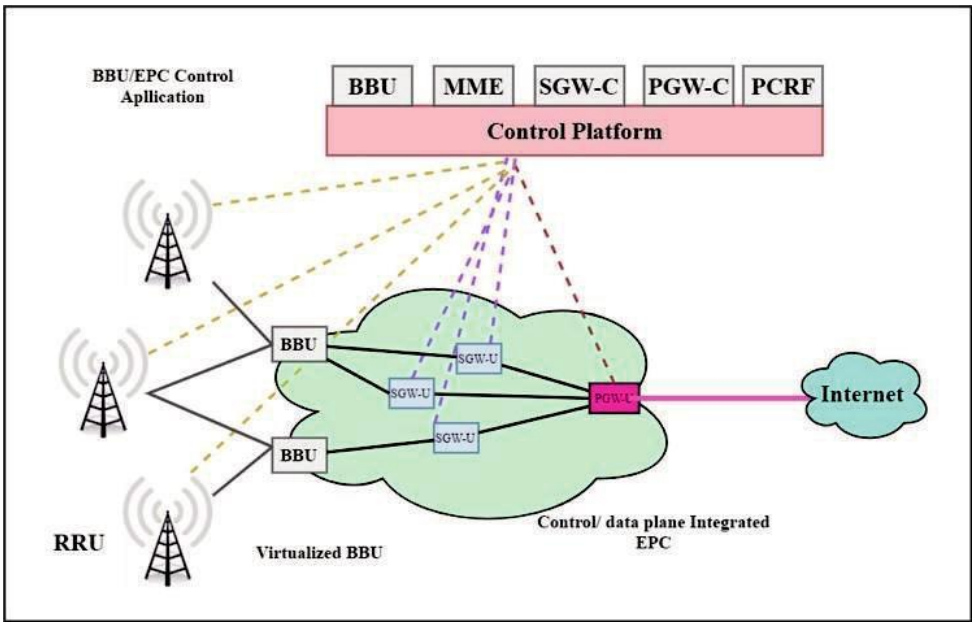


Fig 7.3. Proposed architecture of virtualized RAN and virtualized EPC

A highly flexible and scalable network is provided in the MEC architecture due to the Virtualization of RAN. The ability to coordinate the network at a central level is another aspect that RAN provides, which leads to optimal usage of the spectrum. Customization of the hardware according to the user requirements is made possible because of the disaggregation of EPC. The cost reduction and improvement in the output are the outcomes of Virtualized RAN and disaggregated EPC. Variety of innovations enabled by M-CORD [288] is due to MEC architecture which is a huge boost for the 5G environment. Some of them are elucidated as follows:

- **Optimized CORE for the Static Internet of Things (IoT) Devices:** The current LTE core networks will see excessive growth of signaling overhead in the control plane. M-CORD's open source core network elements solve this problem by providing significant flexibility which in another way helping the overall scope of 5G development. M-CORD model has a scalable, optimized core architecture that has the ability to handle a massive number of stationary IoT devices. M-CORD model has a core slicing feature that allows us to set up separate core slices for IoT devices and conventional mobile devices. The combination of SGW and PGW are done for the static IoT devices while the LTE connections are passed through the traditional pathway thereby reducing any periodic delay for the IoT devices.
- **Programmable and Scalable Connectionless Core:** Sophisticated functionalities would not be possessed by the static IoT devices which are needed by the mobile UEs. The ability to create a programmable core which will be customized for different types of UEs in the 5G network is provided by M-CORD. The signaling and state overhead will be reduced significantly by separating the cores for static IoTs that classifies the traffic at the RAN. There will be a significant improvement in the performance of the user plane because of the use of M-CORD supportable data plane development kit (DPDK).

- **Adaptive Analytical Service:** It is performed by using the M-CORD's model-driven service composition tool which allows us to initiate test and monitor agents performs this service. Exact and geographically specific results are provided because it is used at the edge of the network.

CHAPTER – 8

NETWORK SLICE MANAGEMENT INSIDE M-CORD BASED 5G FRAMEWORK

The core characteristics of 5G network architecture [289, 290] are Network slicing and slice management which is in trend these days. Several frameworks for network slicing (Access network (AN) and Core network (CN)) have been presented, but a detailed framework for slice management and slicing in transport network (TN) is still unavailable. The current working LTE technology treats each of the users, and each of the requested service types with equal resources, i.e., cannot differentiate on the basis the needs that are they have one physical network. There are some instances where IoT require low bandwidth but high availability while enterprise needs high bandwidth high availability. So the aim of the network slicing is to logically slice the network into multiple slices so that users are provided with a flexible network, which will serve them according to their requirements.

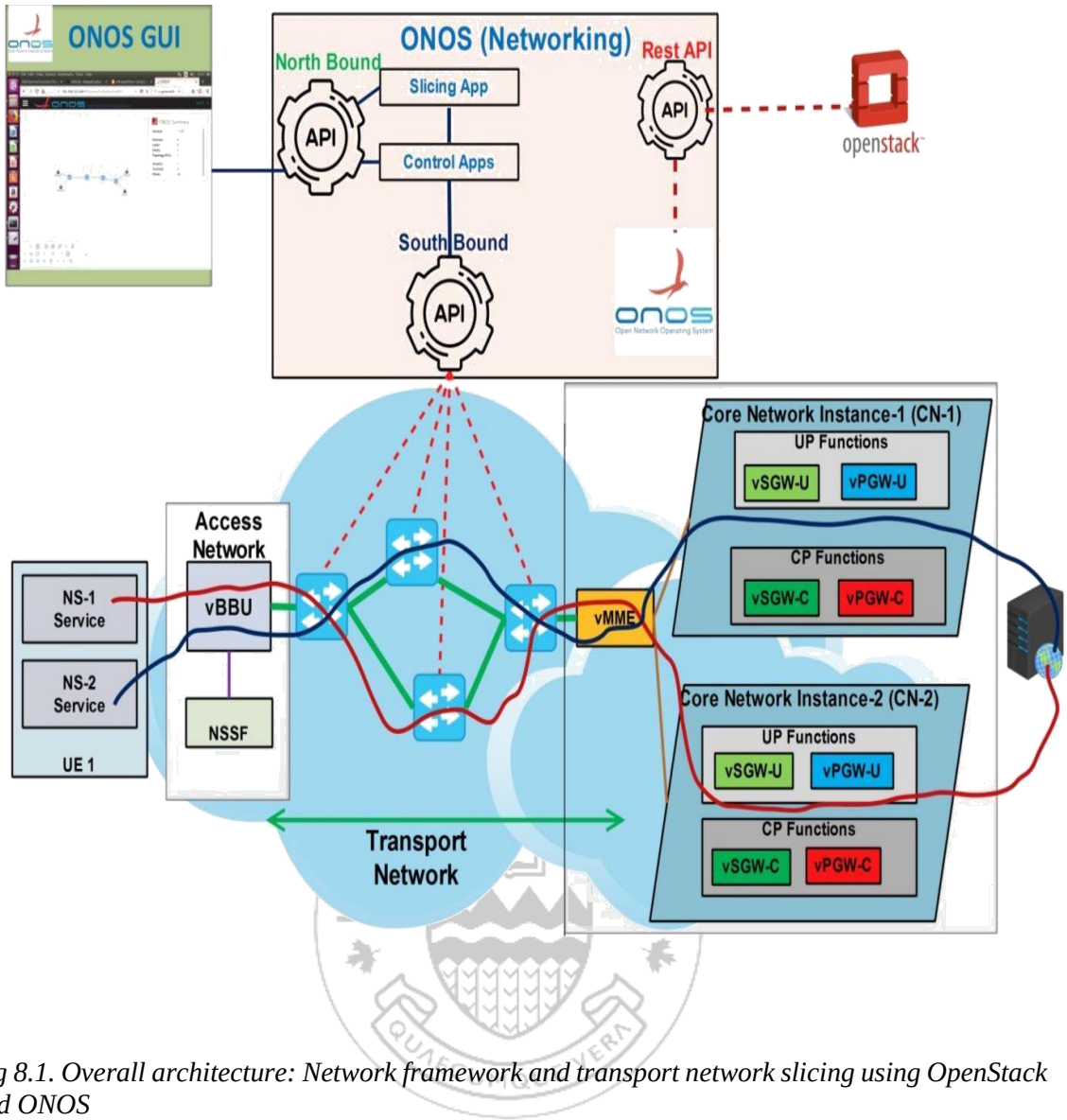


Fig 8.1. Overall architecture: Network framework and transport network slicing using OpenStack and ONOS

8.1 SYSTEM COMPONENTS

8.1.1 Virtualized access network

UE is accessed by CN with the help of this module which accepts its connection request. The development of a simulated module which exploits M-CORD vBBU functionality is included as a part of it. Furthermore, the allocation of different slices based on the service type request made by UE [291] is the responsibility of vBBU. Selection of appropriate core network slices to fulfill the user requirement is based on the communication with NSSF.

8.1.2 Virtualized core network

Virtualized core network (vCN) is a part of vMME that helps in keeping the core network stitched to access network by updating the state transition that is inside the vBBU. Also, it includes Core network slice instances (NSI) where each instance is the 3GPP based

control plane and user plane separated, evolved packet core network (CUPS-EPC) is included in this. Fig above shows how each NSI connects vMME to the video server or internet as per specification [291].

8.1.3 Transport network slice management

The development of the slice creation and management system for TN in between access/core network is the most significant and challenging task of this proposed system. Initially, it includes the development of a system which starts by Designing slices for TN as per service type along with considering the available resources is where the process of development of the system begins with. Furthermore, the deployment of slice management application is done on the top of ONOS. ONOS communicates with the neutron of OpenStack using REST API that is in the slicing application to get the information of the underlying framework so that it can create TN slices. Finally, communication with the created transport network begins by using southbound API that manages network flows and defines a path for traffic on the basis of selected service type [291]

CHAPTER – 9

NETWORK FUNCTION VIRTUALIZATION FOR HIGH-PERFORMANCE 5G

9.1 INTRODUCTION

Mobile core networks are vital to the mobile communication service. Managing connectivity sessions, handovers, idle mode and paging, security, policy, and charging are some of the key responsibility of mobile core networks. There is traditional standardization of these functions so as to ensure interoperability not only between the network and the terminals but also among networking devices. The functions, protocols, and procedures are governed by these standards along with a grouping of the functions into nodes that are connected by standardized interfaces. Traditionally, core networks implement these nodes in separate devices which results in low feature velocity, because new feature that impacts more than one no needs to be standardized first, then they are implemented by vendors (often as a new release of a monolithic device), and finally carefully rolled out, a long process taking years. So, it is essential to realize that any evolution of mobile core networks must significantly improve this problem. Modern IT software technologies and the cloud are finding new ways for the implementation of telecommunications equipment.

Deployment flexibility not only aids in efficient resource management and extensive standardization that permits use cases, such as local breakout are aided because of deployment flexibility [292]. Radio and service functions can be included and managed under a single platform due to the extensibility of the concept which means that some of the virtualized radio functions, such as PDCP or transport encryption employed in LTE are handled just like any other core network function. Functions deployed above (S)Gi interface can also be considered as a part of a single (mobile) service chain, these service functions can also be tied to the same control structure, so that all core network information (like user identity, policies, charging, location, etc.) is made available to them in a uniform manner.

Complexity, state, resource and security requirements are various User Plane Functions. The composition needs to be performance efficient; otherwise, flexibility enabled by the system would be costly. Small, simple UPFs, like Token bucket, policing, packet marking or encapsulation are some instances of simple UPFs that are invoked in the same thread, whereas isolation or the use of a different operating system is needed for complex UPFs.

Diversity in a mobile network is an essential goal that needs to be addressed. A limited prototype of mobile service chaining is in use to experiment with a distinct kind of composition method is developed. A chain compositor composing UPFs inside a Click modular router [293] was implemented particularly. Forwarding between nodes (carrying metadata between sites) and supporting (lossless and reordering-free) handovers via context transfer was also the part of the experiment. Section 9.3 describes the details of our prototyping environment establishing a baseline for performance. Section 9.4 there is a description of the prototype and report of the findings of the kind of performance price one pays for flexible compos-ability. Section 9.5 will demonstrate handover performance, another crucial aspect of a mobile environment. Finally, Section 9.6 we conclude by discussing how the composition method evaluated by our prototype fits among a wider set of composition methods. Along with a vision for a distributed packet processing execution environment for 5G.

9.2 RELATED WORK

9.2.1 Mobile Service Chaining preliminaries

Mobile Service Chaining 5G Core Network Architecture [294] is essentially an evolution of service chaining along with additional support for mobility and a formalized set of metadata. The specifics of the mobile core environment, such as bearers, requirements on grouped flow processing (e.g., to provide aggregated maximum bit-rate policing) and the associated scalability requirements are being catered.

The functions of modern Mobility Management Entity, Policy and Charging Rules Function and a chain controller are included in the architecture of the Control Plane (CP). Realization of the user plane is focused here alongside the evaluation of lossless handovers and chain forwarding which needs a simple Controller that is necessary for the coordination. The two types of functional elements for the actual packet processing in the User Plane (UP) are defined in this architecture: Forwarding Elements (FE) and User Plane Functions (UPF). Forwarding of packets between its ports based on the rules, which are configured by the CP is the role of FE. There is a connection of a port with either UPF or to a network interface. UPFs depends on the locally stored per-user contexts performs actual packet processing. It is not essential for UPFs to know about any detail about the higher level topology.

Information of how packets are sent through the system; accordingly they traverse in the service chain, this is added to the packets as tags by classifiers, this UPFs management is done by the CP. Header fields and packet direction (uplink or downlink) are based on simple classification, but we can also reuse tags assigned by prior UPFs (like Deep Packet Inspection). Few rules can only apply to the assigned tags or directly on the header fields of the packet. It is significant to understand that the assigned tags are carried with the packets through the network. For supporting handover, UP nodes have to be able to export and import the rules and UPF contexts related to the user handing over.

9.2.2 Other related work

Combination of OpenFlow and Virtualized Network Functions (VNF) realized in virtual machines (VMs), or Linux containers [295] refers to service chaining. A significant performance overhead [296] is represented when a packet is transferred from the OpenFlow switch to the VMs in the concluded studies. This especially causes a problem, with simple VNFs where the actual processing time can be compared with this overhead. Linux containers [297] have very less overhead, for a granular service chaining architecture because of a lot of small VNFs, so the network stack of the Linux kernel has the ability to cause bottlenecks. Configurations have also addressed the overhead of virtualization where the VNFs are simple processes on the host computer [298],[299] Nevertheless, the inter-VNF communication cannot be considered lightweight enough as it is processed via Ethernet soft switches and requires packet copies along the way.

Some performance overheads are used to provide a flexible packet processing environment in the Click modular router [293]. High I/O rate extensions [300] measures the overhead. Click can also be used for the construction of middleboxes beside general feasibility. The performance can be predicted with the CPU cache miss rates as seen in Dobrescu et al. [301]. also, present results for several middleboxes results are presented in Martins et al. [302], where there is variation in the complexity from simple forwarding to an intrusion detection system.

Live migration between virtual machines and Linux containers [303] is made possible with cloud-based mobility. However, 5G handovers do not need such a solution, because only the context of a single user needs to be migrated in 5G, not the whole user plane node which handles thousands of users. High accessibility for containers by exporting the internal states is the solution described by Wubin et al. [304], which is an important step towards our needs, but it is still incomplete to support the handover of a single user.

Most cases of packet processing and service chaining are evaluated involving simple rules or functions. Evaluation of more complex middleboxes do not consider the composition of these, neither chained with an OpenFlow switch nor inside the execution environment itself. Furthermore, there are no public results available about VNFs which support per-user context migration.

9.3 PROTOTYPING ENVIRONMENT

Click modular router environment is selected to investigate the performance of packet processing composition. Selecting one of the Click environments to build our prototype. The prototype is directly provided as a virtual machine or a Linux container which will ease the cloud-based deployment since this is needed for 5G systems. Dynamic composition of software routers [293] is enabled by the Click modular router. Assembly of the router in run-time is dependent on a configuration file from simple packet processing elements. Multiple inputs and output ports are possible for elements, and the internal logic can be written in pure C++, there is no limitation to the packet processing of a restricted set of actions as in the case of OpenFlow switches [305].

The traversing of packets is based on the chain of elements as sequences of function calls initiated by the active elements of the actual configuration (ingress interfaces or elements

with buffered packets, ready to send). Click supports multithreading, enabling the scheduling of active elements to different cores in parallel.

Three different environments are supported by Click framework, namely, Linux user space, kernel space and over the Xen hypervisor as a library operating system called ClickOS [302]. Netmap, a fast packet I/O solution available as a kernel module [306] in user space and for ClickOS is supported in most recent version Click which leads to a userspace deployment that has the ability to outperform the kernel based counterparts [300]. Netmap access in Docker containers is possible with mounting the related device file and granting privileged access for the container. A software switch called VALE is also a part of netmap framework, which provides high packet rates between netmap accelerated interfaces [307].

Linux (Ubuntu 14.04 LTS) servers, with each composed of 3.5 GHz 6-core Intel Xeon E5-1650 CPUs and having 32 GiB RAM used. For user plane traffic dedicated 10-Gigabit point-to-point links between the machines with Intel X540 network interface cards (NICs) was used for user plane traffic. The traffic on these links was directly handled by either a Click instance or a traffic generator/collector handled the traffic directly on these links, without being processed by the OS network stack. An exception was the ClickOS which used VALE switch. A separate 1-Gigabit network was needed for Control Plane traffic. Pkt-gen tool from the netmap framework was used for packet generation to achieve high packet per second rates.

A simple configuration could be used with Click router, which only rewrote the Ethernet addresses, thus realizing a simple low-level forwarder.

The peak performance estimates of other machines measured using packet forwarding in the Linux kernel just in order to provide a reproducible reference. This throughput is even lower than the simple user pace Click configuration because in Click only the Ethernet addresses are rewritten without going through the Linux protocol stack. We determine that the kernel space-based Click router improves the throughput but adds variability.

The Click router that runs in Docker container is used despite the slightly lower performance; this setup seamlessly fits into the envisioned cloud-based deployment of 5G systems.

9.4 THE 5G CORE NETWORK PROTOTYPE

The proposal for logical 5G Core Network Architecture to nodes on physical computers that each runs on a Click router instance. On how performance is evaluated:

9.4.1 Implementation details

Click elements such as the FEs and the UPFs, where UPFs are connected to the ports of the single FE. UDP encapsulation is used in simple tunneling protocol that carries the tag/value pairs besides payload is implemented between Click instances. As the encapsulated packet arrives at a Click node, there is the removal of extra headers and the placement of tags to the annotation area are defined by the Click for each packet, which makes tag handling efficient. The implementation has the capability of per-user handovers and utilizing a special buffering element in each node. This element performs nothing for users, not in a handover.

FE is connected to both the input and the output ports of the UPFs, where rules are applied for all the incoming packets. The actual rules can be defined with A rule language from the Control Plane is used for defining actual rules, where currently UPF names and packet tags are possibly matched. Obviously, the performance of the rule matching against each other. The performance of the system is determined especially for a large rule set. A hash table based caching component was implemented to speed up the lookup. Following are the fully functional, per-user handover capable UPFs:

- **Counter:** A separate packet counter for each user is maintained by this UPF, and the current value is written into the payload at an offset, this is chosen in order to avoid overwriting any previously placed counter values. Even though being a relatively simple function, payload parsing is still required, along with modification and the recalculation of the UDP checksum.

- **Marker:** The marker element uses Token buckets are used by Marker to check whether the traffic conforms to a per-user defined bandwidth limit. The packets that are marked with Non-conformant tags is to be dropped later.

- **Header compressor (IPHC):** IP/UDP compression and decompression as specified in RFC2507 [24] is implemented by this. It has the ability to handle multiple users with multiple flows per user, where the assignment of context identifiers is done automatically.

9.4.2 Evaluation of the prototype

The performance of the prototype that runs on a single node is evaluated assuming that it is an intermediate element of a longer chain, and it has already classified and tagged the incoming packets. The Pkt-gen tool was modified to be able to generate tunneled traffic, tagged with user-ID tag.

- **UPF Overhead:** The first setup where no encapsulation, just one FE with one rule to forward. In comparison to the baseline measurements of the previous section, this configuration removes the Ethernet header and validates the IP header before it enters into the FE. After the rule matching is complete, the IP addresses are rewritten, and a new Ethernet header is placed to the packet.

Tunneling and handover elements were included in the second setup. The encapsulation step also involves a configurable, next hop selection based on tags. Actual UPFs are added to the chain one after another, creating more and more complex configurations for the rest of the cases. Since these functions handle contexts, parse, validate and copy fields for each

incoming packet, it is expected that this CPU and memory intensive tasks affect the performance displayed in Fig. 9.1.

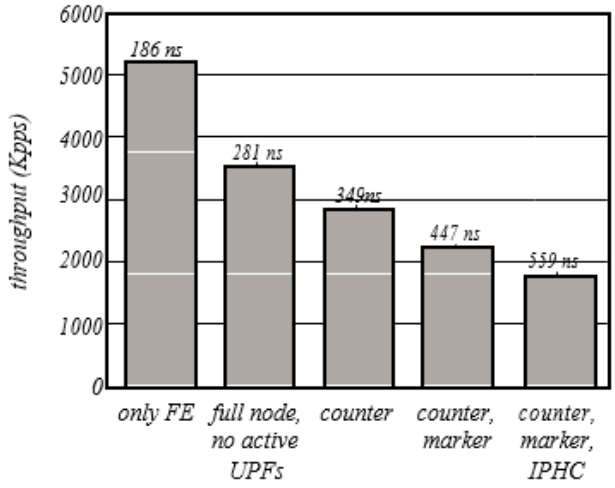


Fig. 9.1 Prototype Performance (throughput of different configurations)

- **Impact of additional users:** 10–100 k users are handled by Current custom-made core network nodes. In cloud-based deployments of future 5G systems per component, the load is expected to be lower but still in the thousands. Although the hash table based on lookups were used are near constant time, still, rise in the number of user's causes some performance degradation partly due to the L3 cache misses as shown in Fig. 9.2 below.

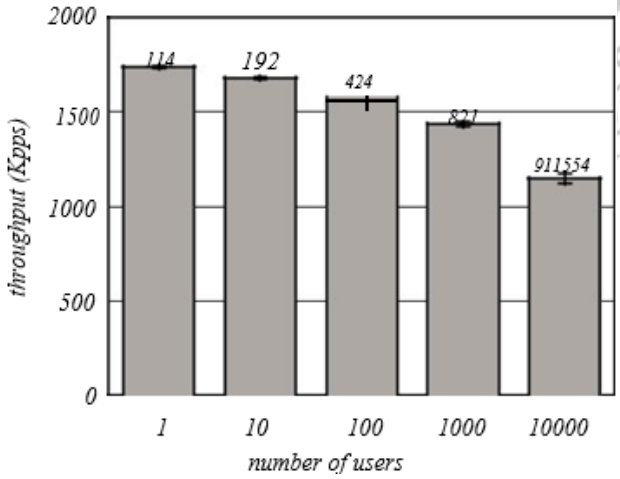


Fig. 9.2 Prototype Performance (throughput with multiple users)

- **Handling multiple chains:** 5G architecture proposes flexibility as its core strength. 1000 users and defined 2 chains, having one with three UPFs and one without any were used for this measurement. In contrast to FE rules, this requires only one extra rule, which defines the shortcut for packets arriving on the less complex chain. We generated the traffic with a set ratio of users taking the more and the less complex chains demonstrated in Fig. 9.3.

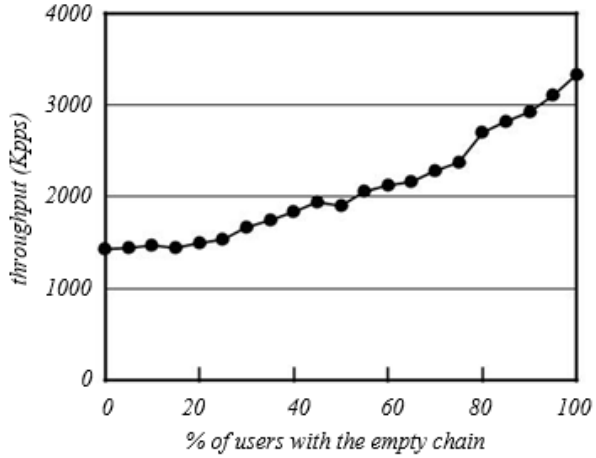


Fig. 9.3. Prototype Performance (throughput with mix of chains)

This demonstration of the prototype – as a general-purpose packet processing environment – supports more complex feature sets as well, the aggregated performance of serving multiple chains of various complexity remains predictable. The results also showcase that this kind of flexibility does not sacrifice the scalability: the same compute resources can be saturated either with more complex services at lower throughput, or with less complex ones at higher throughput.

9.5 HANDOVER

The proposed architecture supports the transfer of a user's UPF contexts from a source node to a target node which is the core requirement for mobile service chaining solutions. Buffering in nodes is done to avoid packet reordering, where traffic arrives on the new (post-handover) path until all the packets arriving on the old path are processed.

The process of handing over is described where a gateway node (GW) has two-way communication with an emulated UE via either one of the two base stations ($BS_{1, 2}$), with the choice of switching between the paths when Control Plane instructs to do so. BS_2 receives uplink packets from a buffer GW holds as well as one in BS_2 for packets already arriving on the new path. BS_2 also has a separate buffer for the packets arriving on the temporary path. So incoming packets are stored in this buffer until the migration of contexts completes. After the completion of all the preparatory steps, the path of downlink packets is switched in the GW, and no new packets take the old path, and the buffered packets can be consumed, which eventually makes the buffers get empty, at which point the node switches to processing incoming packets directly.

The bulk of the above functionality is realized in a special element chained before the FE, where lifecycle of per-user handover buffers is managed, and packet forwarding is handled. Functionality (like control-related communication or context transfer) spreads among the FE, the UPFs, the daemon of the user plane nodes and the Control Plane.

The UE emulates the behavior of a radio device in order to test by replying to the same base station it received the packet from.

Generally, the arrival of packet forms a diagonal line, the rate of the packets determines the slope. As handover is in progress, packets stop arriving for a short period (approximately 2 ms), due to buffering, during which the context is migrated. As soon as the packet processing is resumed, the nodes receive larger batches of packets at a time, because processing the buffer can be faster than the packet arrival rate. This makes buffer empty, from which point the arrival times follow their usual linear pattern.

9.6 CONCLUSION

The performance aspects of a certain composition model. This model characterized by the following attributes

- Function calls invoke the individual processing functions (UPFs);
- The execution of the next UPF based on the consultation of a (cached) rule database;
- UPFs free to process, duplicate, buffer, drop packets or even generate new packets; and
- Packets can carry Metadata between the UPFs.

Flexibility is the main advantage of the composition model. Single or multithreaded operations are allowed that has an internal scheduler that initiates elements that have work to do. There are some limitations because all Click elements run in the same process, and nothing much can be done to realize isolation either for resource control or security. Click represents a confined programming environment so as to manage relatively high performance and variety of situations. There are rules for handling of packets, allocating memory or doing I/O. This makes third party libraries vague, even in the STL. The STL makes the Development of complex UPFs, such as a full DPI suite or transparent HTTP proxies becomes quite cumbersome. A separate container or VM would be best for it since limitations would appear in any system using function calls as composition method.

Function calls represent the overhead and rule lookups can be essential for very small UPFs. Looking at flow counters, bandwidth policies, simple encapsulation modules (and in general most actions of an OpenFlow switch) are only complete in very few cycles. Lightweight composition model is better for these. Coping their code one after another in as many combinations as needed [308] is much simpler and more efficient.

A mixture of small and large UPFs has a versatile programmable packet processing execution environment, so it becomes easier if several composition methods are available to the programmers. The just-in-time (JIT) linker is used that dynamically copies small UPFs one after another. The fastest software switch for simple actions [308] can be achieved through this. Passing of packets among containers or VMs that are on the same server, a separate container/VM model is put into utilization, along with it, use of the third-party library is enabled or even OS (in the case of VMs) between contexts [296]. Enabling functions that reside on different hosts and uses encapsulation to carry chain ID and other metadata on the wire is implemented by chain forwarding [309].

The first two methods supported the run-to-completion mode of operation, whereas the second and third can split the processing of a single packet among multiple threads or processes (latter of which can even reside in different containers/VMs). Presumably, a

UPF needs development with its composition method. For instance, JIT linked components small fragments of code are required with linker metadata. Shared libraries should provide function call components, while the rest as apps or VM images.

The controller of this execution environment should have knowledge of the type of each UPF. So, implementing and optimizing user plane is left to the developer of the UPFs and the execution environment. A distinction should not be hardcoded in a standard. For instance, as an OpenFlow soft switch combined with VMs does not an ability to perform a wide variety of packet processing applications, rather it creates an artificial distinction between things that need to be done in OF switch. Even if a particular function is small enough that it can fit into the programming environment of the soft switch, should be placed in a VM, if it is not mentioned in the OpenFlow standard.

A user plane model, that is, individual UPFs is chained together inside and between machines is fairly future proof. There is an abstract view of the user plane that can mix and match the standard or de-facto components for the controllers. The ability to add components irrespective of their size boosts continuous feature innovation while enabling the abstraction itself from continuous performance innovation (hidden from controllers). A large number of composition models employed, high performance is expected even for small functions without the controller to understand the details of the user plane implementation. This level of abstraction has the ability to fully utilize hardware innovation without requiring the controller to understand the intricacies of the underlying user plane node.

CONCLUSION

The cause of concern for the network providers is due to the rise in the sheer number of devices that are connected to the internet, which further demands for faster and better connectivity by the day. Therefore, a technology that is determined to facilitate the rising demand without necessitating a significant investment by the network providers, along with saving a lot of resources.

An open source solution, the M-CORD platform that collaborates development with a robust and economic environment for the developers as discussed in the report. This open source reference solution stands on the pillars of open source SDN, NFV and cloud technologies, which helps in the integration of disaggregated and virtualized RAN and core functions of the wireless network along with mobile edge applications. The platforms allow operators to experiment and realize 5G technologies on an LTE network by implementing M-CORD without having to wait for all the 5G standards to be ratified. As we know with 5G, the world would be truly connected and would aid a lot of other technologies (IoTs) that are still to be realized.

Although this technology is offering a lot as compared to other network technologies we have gone through, it requires addressing certain issues like faster development and building of this platform across the globe. Even though M-CORD has attracted a lot of partners and collaborators through open source community to work on the innovations to empower the mobile network providers, but still, the effort is lagging on a global scale, which is causing platform downtime issues. Therefore, it has now become a necessity that its potential is realized globally and 5G standards are accredited for the faster development of the platform and related technologies.

REFERENCES

[1]URL: https://cdn.ttgtmedia.com/searchTelecom/downloads/SearchTelecom_Fundamentals_of_LTE_Chapter_1.pdf

[2] URL: http://dictionary.laborlatalk.com/ This site provides information on terms of various types such as encyclopedic, legal, medical, computer and science.

[3] URL: http://en.wikipedia.org/wiki/Main_Page This site hosts a free online encyclopedia.

[4] Creativyst, "Glossary of Industry Related Terms", URL: http://www.creativyst.com/cgi-bin/M/Glos/st/GetTerm.pl?fsGetTerm=802.11b Creativyst (pronounced "Kree.ATE.tiv.vist") is dedicated to providing the best and most reliable products and services to help automate your business, improve your web site, and develop your applications.

[5] 3G Americas. List of 3G deployments worldwide; URL: http://www.3gamericas.org

[6] UMTS Forum; URL: http://www.umts-forum.org

[7] Holma, H. et al. "High-Speed Packet Access Evolution in 3GPP Release 7." IEEE Communications Magazine, 45(12):29–35, December 2007.

[8] Holma, H. and A. Toskala. "High-Speed Downlink Packet Access." Chapter 11. WCDMA for UMTS. New York: John Wiley & Sons, Inc., 2002.

[9] 3GPP TR 25.913., "Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN)," v8.0.0, December 2008.

[10] IEEE Communications Magazine, Special issue on LTE—LTE Part II: Radio Access, April 2009.

[11] IEEE Communications Magazine, Special issue on LTE—LTE Part I: Core Network, February 2009.

[12] EURASIP Journal on Wireless Communications and Networking, Special issue on 3GPP LTE and LTE Advanced, August 2009.

[13] IEEE Communications Magazine, Special issue on LTE—LTE Part I: Core Network, February 2009.

[14] T. Benson, A. Akella, and D. Maltz, ‘‘Unraveling the complexity of network management,’’ in Proc. 6th USENIX Symp. Networked Syst. Design Implement., 2009, pp. 335–348.

[15] B. Raghavan et al., ‘‘Software-defined internet architecture: Decoupling architecture from infrastructure,’’ in Proc. 11th ACM Workshop Hot Topics Netw., 2012, pp. 43–48.

[16] A. Ghodsi et al., ‘‘Intelligent design enables architectural evolution,’’ in Proc. 10th ACM Workshop Hot Topics Netw., 2011, pp. 3:1–3:6.

[17] N. Mckeown, ‘‘How SDN will shape networking,’’ Oct. 2011. [Online]. Available: http://www.youtube.com/watch?v= c9-K5O_qYgA

[18] S. Schenker, ‘‘The future of networking, the past of protocols,’’ Oct. 2011. [Online]. Available: http://www.youtube.com/ watch?v=YHeyuD89n1Y

[19] H. Kim and N. Feamster, ‘‘Improving network management with software-defined networking,’’ IEEE Commun. Mag., vol. 51, no. 2, pp. 114–119, Feb. 2013.

[20]T. Koponen et al., ‘‘Onix: A distributed control platform for large-scale production networks,’’ in Proc. 9th USENIX Conf. Oper. Syst. Design Implement., 2010, pp. 1–6.

[21] S. Jain et al., ‘‘B4: Experience with a globally-deployed software defined wan,’’ in Proc. ACM SIGCOMM Conf., 2013, pp. 3–14.

[22]N. McKeown et al., ‘‘OpenFlow: Enabling innovation in campus networks,’’ SIGCOMM Comput. Commun. Rev., vol. 38, no. 2,
pp. 69–74, Mar. 2008.

[23]Open Networking Foundation (ONF), 2014. [Online]. Available: https://www. opennetworking.org/

[24] VMware, Inc., NSX Virtualization Platform, 2013. [Online]. Available: https://www. vmware.com/products/nsx/

[25] Open Daylight, A Linux Foundation Collaborative Project, 2013. [Online]. Available: http://www.opendaylight.org

[26] N. Feamster, J. Rexford, and E. Zegura, ‘‘The road to SDN,’’ Queue, vol. 11, no. 12,
pp. 20:20–20:40, Dec. 2013.

[27] R. Presuhn, ‘‘Version 2 of the protocol operations for the simple network management protocol (SNMP),’’ Internet Engineering Task Force, RFC 3416 (Internet Standard), Dec. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3416.txt

[28] J. Pan, S. Paul, and R. Jain, ‘‘A survey of the research on future internet architectures,’’ IEEE Commun. Mag., vol. 49, no. 7, pp. 26–36, Jul. 2011.

[29] R. Barrett, S. Haar, and R. Whitestone, ‘‘Routing snafu causes internet outage,’’ Interactive Week, vol. 25, 1997.

[30] K. Butler, T. Farley, P. McDaniel, and J.Rexford, ‘‘A survey of BGP security issues and solutions,’’ Proc. IEEE, vol. 98, no. 1, pp. 100–122, Jan. 2010.

[31] P. Newman, G. Minshall, and T. L. Lyon, ‘‘IP switchingVATM under IP,’’ IEEE/ACM Trans. Netw., vol. 6, no. 2, pp. 117–129, Apr. 1998.

[32] N. Gude et al., ‘‘NOX: Towards an operating system for networks,’’ Comput. Commun. Rev., vol. 38, no. 3, pp. 105–110, 2008.

[33] H. Jamjoom, D. Williams, and U. Sharma, ‘‘Don’t call them middle-boxes, call them middlepipes,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 19–24.

[34] H. Alkhatib et al., ‘‘IEEE CS 2022 Report (Draft),’’ IEEE Computer Society, Tech. Rep., Feb. 2014.

[35] M. Casado, N. Foster, and A. Guha, ‘‘Abstractions for software-defined networks,’’ ACM Commun., vol. 57, no. 10, pp. 86–95, Sep. 2014.

[36] A. Doria et al., ‘‘Forwarding and control element separation (ForCES) protocol specification,’’ Internet Engineering Task Force, Mar. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5810.txt

[37] H. Song, ‘‘Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane,’’ in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 127–132.

[38] K. Greene, ‘‘10 Breakthrough Technologies: Software-defined Networking MIT Technol. Rev., 2009. [Online]. Available: http:// www2.technologyreview.com/article/ 412194/tr10-software-defined-networking/

[39] T. D. Nadeau and K. Gray, SDN: Software Defined Networks, 1st ed. Sebastopol, CA, USA: O’Reilly Media, 2013.

[40] N. M. K. Chowdhury and R. Boutaba, ‘‘A survey of network virtualization,’’ Comput. Netw., vol. 54, no. 5, pp. 862–876, 2010.

[41] B. Davie and J. Gross, ‘‘A stateless transport tunneling protocol for network virtualization (STT),’’ Internet Engineering Task Force, Apr. 2014. [Online]. Available: http://tools. ietf.org/html/draft-davie-stt-06

[42] M. Mahalingam et al., ‘‘VXLAN: A framework for overlaying virtualized layer 2 networks over layer 3 networks,’’ Internet Engineering Task Force, Internet Draft, Nov. 2013. [Online]. Available: http://www. ietf.org/id/draft-mahalingam-dutt-dcops-vxlan-06.txt

[43] M. Sridharan et al., ‘‘NVGRE: Network virtualization using generic routing encapsulation,’’ Internet Engineering Task Force, Internet Draft, Aug. 2013. [Online]. Available: http://tools.ietf.org/id/draft-sridharan-virtualization-nvgre-03.txt

[44] F. Maino, V. Ermagan, Y. Hertoghs, D.Farinacci, and M. Smith ‘‘LISP control plane for network virtualization overlays,’’ Internet Engineering Task Force, Oct. 2013. [Online]. Available: http://tools.ietf.org/ html/draft-maino-nvo3-lisp-cp-03

[45] Y. Hertoghs et al., ‘‘A unified LISP mapping database for L2 and L3 network virtualization overlays,’’ Internet Engineering Task Force, Feb. 2014. [Online]. Available: http://tools. ietf.org/html/draft-hertoghs-nvo3-lisp-controlplane-unified-01

[46] J. Gross, T. Sridhar, P. Garg, C. Wright, and I. Ganga, ‘‘Geneve: Generic network virtualization encapsulation,’’ Internet Engineering Task Force, Internet Draft, Feb. 2014. [Online]. Available:
http://tools. ietf.org/id/draft-gross-geneve-00.txt

[47] R. Jain and S. Paul, ‘‘Network virtualization and software defined networking for cloud computing: A survey,’’ IEEE Commun. Mag., vol. 51, no. 11, pp. 24–31, Nov. 2013.

[48] E. Haleplidis et al., ‘‘SDN layers and architecture terminology,’’ Internet Engineering Task Force, Internet Draft, Sep. 2014. [Online]. Available: http:// www.ietf.org/id/draft-irtf-sdnrg-layer-terminology-02.txt

[49] Y. Rekhter, T. Li, and S. Hares, ‘‘A border gateway protocol 4 (BGP-4),’’ Internet Engineering Task Force, RFC 4271 (Draft Standard), Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4271.txt

[50] J. Vasseur and J. L. Roux, ‘‘Path computation element (PCE) communication protocol (PCEP),’’ Internet Engineering Task Force, RFC 5440 (Proposed Standard), Mar. 2009. [Online]. Available: http://www.ietf.org/rfc/ rfc5440.txt

[51] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman ‘‘Network configuration protocol (NETCONF),’’ Internet Engineering Task Force, RFC 6241 (Proposed Standard), Jun. 2011. [Online]. Available: http://www.ietf.org/rfc/ rfc6241.txt

[52] A. Corradi, M. Fanelli, and L. Foschini, ‘‘VM consolidation: A real case based on openstack cloud,’’ Future Generat. Comput. Syst., vol. 32, pp. 118–127, 2014.

[53] A. Shang, J. Liao, and L. Du, ‘‘Pica8 Xorplus, 2014. [Online]. Available: http://sourceforge.net/projects/xorplus/

[54] P. Jakma and D. Lamparter, ‘‘Introduction to the quagga routing suite,’’ IEEE Network, vol. 28, no. 2, pp. 42–48, Mar. 2014.

[55] NetFPGA, 2014. [Online]. Available: http://netfpga.org/

[56] Linux Foundation, ‘‘Open platform for NFV,’’ Sep. 2014. [Online]. Available: https://www.opnfv.org

[57] C. E. Rothenberg et al., ‘‘When open source meets network control planes,’’ IEEE Computer, Special Issue on Software-Defined Networking, vol. 47, no. 11, pp. 46–54, Nov. 2014.

[58] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden, ‘‘A survey of active network research,’’ IEEE Commun. Mag., vol. 35, no. 1, pp. 80–86, Jan. 1997.

[59] A. Lazar, K.-S. Lim, and F. Marconcini, ‘‘Realizing a foundation for programmability of ATM networks with the binding architecture,’’ IEEE J. Sel. Areas Commun., vol. 14, no. 7, pp. 1214–1227, Sep. 1996.

[60] A. Lazar, ‘‘Programming telecommunication networks,’’ IEEE Network, vol. 11, no. 5,
pp. 8–18, Sep. 1997.

[61] D. Sheinbein and R. P. Weber, ‘‘800 service using SPC network capability,’’ Bell Syst. Tech. J., vol. 61, no. 7, pp. 1737–1744, Sep. 1982.

[62] M. Caesar et al., ‘‘Design and implementation of a routing control platform,’’ in Proc. 2nd Conf. Symp. Netw. Syst. Design Implement., 2005, vol. 2, pp. 15–28.

[63] J. Van der Merwe, S. Rooney, I. Leslie, and S. Crosby, ‘‘The tempest-a practical framework for network programmability,’’ IEEE Network, vol. 12, no. 3, pp. 20–28, May 1998.

[64] M. Casado et al., ‘‘SANE: A protection architecture for enterprise networks,’’ in Proc. 15th Conf. USENIX Security Symp., 2006, vol. 15, Article 10.

[65] M. Casado et al., ‘‘Ethane: Taking control of the enterprise,’’ in Proc. Conf. Appl. Technol. Architect. Protocols Comput. Commun., 2007, DOI: 10.1145/1282380. 1282382.

[66] M. Macedonia and D. Brutzman, ‘‘Mbone provides audio and video across the internet,’’ Computer, vol. 27, no. 4, pp. 30–36, 1994.

[67] B. Chun et al., ‘‘PlanetLab: An overlay testbed for broad-coverage services,’’ SIGCOMM Comput. Commun. Rev., vol. 33, no. 3, pp. 3–12, Jul. 2003.

[68] L. Peterson et al., ‘‘Geni design principles,’’ Computer, vol. 39, no. 9, pp. 102–105, Sep. 2006.

[69] A. Bavier, N. Feamster, M. Huang, L.Peterson, and J. Rexford, ‘‘In VINI veritas: Realistic and controlled network experimentation,’’ SIGCOMM Comput. Commun. Rev., vol. 36, no. 4, pp. 3–14, Aug. 2006.

[70] T. Koponen et al., ‘‘Network virtualization in multi-tenant data centers,’’ in Proc. 11th USENIX Symp. Netw. Syst. Design Implement., Apr. 2014, pp. 203–216.

[71] U. Krishnaswamy et al., ‘‘ONOS: An open source distributed SDN OS,’’ 2013. [Online]. Available: http://www.slideshare.net/ umeshkrishnaswamy/open-network-operating-system

[72] Juniper Networks, ‘‘Junos OS architecture overview,’’ 2012. [Online]. Available: http://www.juniper.net/techpubs/en_US/junos12.1/topics/concept/junos-software-architecture.html

[73] Extreme Networks, ‘‘ExtremeXOS operating system, version 15.4,’’ 2014. [Online]. Available: http://learn.extremenetworks. com/rs/extreme/images/EXOS-DS.pdf

[74] Alcatel-Lucent, ‘‘SR OS,’’ 2014. [Online]. Available: http://www3. alcatel-lucent.com/ products/sros/

[75] A. T. Campbell, I. Katzela, K. Miki, and J. Vicente, ‘‘Open signaling for ATM, internet and mobile networks (OPENSIG’98),’’ SIGCOMM Comput. Commun. Rev., vol. 29, no. 1, pp. 97–108, Jan. 1999.

[76] H. Song, J. Gong, J. Song, and J. Yu, ‘‘Protocol oblivious forwarding (POF),’’ 2013. [Online]. Available: http://www. poforwarding.org/

[77] Open Networking Foundation (ONF), ‘‘Charter: Forwarding abstractions working group,’’ Apr. 2014. [Online]. Available: https://www.opennetworking.org/images/ stories/downloads/working-groups/ charter-forwarding-abstractions.pdf
[78] Centec Networks, ‘‘V350VCentec open SDN platform,’’ 2013. [Online]. Available: http://www.valleytalk.org/wp-content/uploads/2013/04/Centec-Open-SDN-Platform.pdf

[79] NoviFlow, ‘‘NoviSwitch 1248 high-performance OpenFlow switch,’’ 2013. [Online]. Available: http://205.236.122.20/ gestion/NoviSwitch1248Datasheet.pdf

[80] A. Weissberger, ‘‘VMware’s network virtualization poses a huge threat to data center switch fabric vendors,’’ 2013. [Online]. Available: http://viodi.com/2013/ 05/06/vmwares-network-virtualization-poses-huge-threat-to-data-center-switch-fabric-vendors/

[81] S. Shenker, ‘‘Stanford SeminarVSoftware-defined networking at the crossroads,’’ Jun. 2013. [Online]. Available: http://www.youtube.com/ watch?v=WabdXYzCAOU

[82] M. Casado, ‘‘OpenStack and network virtualization,’’ Apr. 2013. [Online]. Available: http://blogs.vmware.com/ vmware/2013/04/openstack-and-network-virtualization.html

[83] Big Switch Networks, ‘‘Project floodlight,’’ 2013. [Online]. Available: http://www. projectfloodlight.org/

[84] E. L. Fernandes and C. E. Rothenberg, ‘‘OpenFlow 1.3 software switch, SBRC’2014,’’ 2014. [Online]. Available: https://github.com/CPqD/ofsoftswitch13

[85] Open vSwitch, 2013. [Online]. Available: http://vswitch.org/

[86] OpenFlow Community, ‘‘Switching reference system,’’ 2009. [Online]. Available: http:// www.openflow.org/wp/downloads/

[87] Y. Yiakoumis, J. Schulz-Zander, and J. Zhu, ‘‘Pantou: OpenFlow 1.0 for OpenWRT,’’ 2011. [Online]. Available: http://www. openflow.org/wk/index.php/Open_ Flow1.0_forOpenWRT

[88] Open Network Install Environment (ONIE), 2013. [Online]. Available: http://onie.org/

[89] T. Kato et al., ‘‘Case study of applying SPLE to development of network switch products,’’ in Proc. 17th Int. Softw. Product Line Conf., 2013, pp. 198–207.

[90] B. Pfaff and B. Davie, ‘‘The Open vSwitch database management protocol,’’ Internet Engineering Task Force, RFC 7047 (Informational), Dec. 2013. [Online]. Available: http://www.ietf.org/rfc/ rfc7047.txt

[91] M. Smith et al., ‘‘OpFlex control protocol,’’ Internet Engineering Task Force,’’ Internet Draft, Apr. 2014. [Online]. Available: http:// tools.ietf.org/html/draft-smith-opflex-00

[92] G. Bianchi, M. Bonola, A. Capone, and C.Cascone, ‘‘OpenState: Programming platform-independent stateful OpenFlow applications inside the switch,’’ SIGCOMM Comput. Commun. Rev., vol. 44, no. 2, pp. 44–51, Apr. 2014.

[93] M. Sune, V. Alvarez, T. Jungel, U. Toseef, and K. Pentikousis, ‘‘An OpenFlow implementation for network processors,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, 2 pp.

[94] D. Parniewicz et al., ‘‘Design and implementation of an OpenFlow hardware abstraction layer,’’ in Proc. ACM SIGCOMM Workshop Distrib. Cloud Comput., 2014, pp. 71–76.

[95] B. Belter et al., ‘‘Hardware abstraction layer as an SDN-enabler for non-OpenFlow network equipment,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, 6 pp.

[96] B. Belter et al., ‘‘Programmable abstraction of datapath,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, pp. 7–12.

[97] R. G. Clegg et al., ‘‘Pushing software defined networking to the access,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, pp. 1–6.

[98] V. Fuentes et al., ‘‘Integrating complex legacy systems under OpenFlow control: The DOCSIS use case,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, 6 pp.

[99] T. J. Bittman, G. J. Weiss, M. A. Margevicius, and P. Dawson, ‘‘Magic quadrant for x86 server virtualization infrastructure,’’ Gartner, Tech. Rep., Jun. 2013.

[100] D. W. Cearley, D. Scott, J. Skorupa, and T.J. Bittman, ‘‘Top 10 technology trends, 2013: Cloud computing and hybrid IT drive future IT models,’’ Feb. 2013. [Online]. Available:http://www.gartnersummit.com/Gartnertop_10_technology_trends_2012_377 16.pdf

[101] C. Peng, M. Kim, Z. Zhang, and H. Lei, ‘‘VDN: Virtual machine image distribution network for cloud data centers,’’ in Proc. IEEE INFOCOM, Mar. 2012, pp. 181–189.

[102] Z. Zhang et al., ‘‘VMThunder: Fast provisioning of large-scale virtual machine clusters,’’ IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 12, pp. 3328–3338, Dec. 2014.

[103] R. Sherwood et al., ‘‘Can the production network be the testbed?’’ in Proc. 9th USENIX Conf. Oper. Syst. Design Implement., 2010, pp. 1–6.

[104] R. Sherwood et al., ‘‘FlowVisor: A network virtualization layer,’’ Deutsche Telekom Inc. R&D Lab, Stanford, Nicira Networks, Tech. Rep.,2009.

[105] S. Azodolmolky et al., ‘‘Optical FlowVisor: An OpenFlow-based optical network virtualization approach,’’ in Proc. Nat. Fiber Optic Eng. Conf., Mar. 2012.

[106] D. A. Drutskoy, ‘‘Software-defined network virtualization with FlowN,’’ Ph.D. dissertation, Dept. Comput. Sci., Princeton Univ., Princeton, NJ, USA, Jun. 2012.

[107] A. Al-Shabibi et al., ‘‘OpenVirteX: A Network Hypervisor,’’ 2014. [Online]. Available: http://ovx.onlab.us/wp-content/uploads/ 2014/04/ovx-ons14. pdf

[108] A. Al-Shabibi et al., ‘‘OpenVirteX: Make your virtual SDNs programmable,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 25–30.

[109] S. Racherla et al., Implementing IBM Software Defined Network for Virtual Environments. Durham, NC, USA: IBM RedBooks,
May 2014.

[110] C. Li et al., ‘‘Software defined environments: An introduction,’’ IBM J. Res. Develop., vol. 58, no. 2, pp. 1–11, Mar. 2014.

[111] A. Gudipati, L. E. Li, and S. Katti, ‘‘Radiovisor: A slicing plane for radio access networks,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp 237–238.

[112] H. Yamanaka, E. Kawai, S. Ishii, and S. Shimojo, ‘‘AutoVFlow: Autonomous virtualization for wide-area OpenFlow networks,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, 6 pp.

[113] Berlin Institute for Software Defined Networks (BISDN) GmbH, ‘‘The eXtensible OpenFlow Datapath Daemon (xdpd) bringing innovation into the fast path,’’ 2014. [Online]. Available: http://xdpd.org/

[114] R. Doriguzzi-Corin, E. Salvadori, M. Gerola, M. Sune, and H. Woesner, ‘‘A datapath-centric virtualization mechanism for OpenFlow networks,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, pp. 1–6.

[115] T. Szyrkowiec et al., ‘‘Demonstration of SDN based optical network virtualization and multidomain service orchestration,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, 2 pp.

[116] D. Depaoli, R. Doriguzzi-Corin, M. Gerola, and E. Salvadori, ‘‘Demonstrating a distributed and version-agnostic OpenFlow slicing mechanism,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, 2 pp.

[117] Z. Bozakov and P. Papadimitriou, ‘‘AutoSlice: Automated and scalable slicing for software-defined networks,’’ in Proc. ACM Conf. CoNEXT Student Workshop, 2012, pp. 3–4.

[118] D. Drutskoy, E. Keller, and J. Rexford, ‘‘Scalable network virtualization in software-defined networks,’’ IEEE Internet Comput., vol. 17, no. 2, pp. 20–27, Mar./Apr. 2013.

[119] X. Jin, J. Rexford, and D. Walker, ‘‘Incremental update for a compositional SDN hypervisor,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp 187–192.

[120] S. Ghorbani and B. Godfrey, ‘‘Towards correct network virtualization,’’ in Proc.3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 109–114.

[121] Juniper Networks, ‘‘Opencontrail,’’ 2013. [Online]. Available: http://opencontrail.org/

[122] HP, ‘‘SDN controller architecture,’’ Tech. Rep., Sep. 2013.

[123] K. Phemius, M. Bouet, and J. Leguay, ‘‘DISCO: Distributed multi-domain SDN controllers,’’ Aug. 2013. [Online]. Available: http://arxiv.org/abs/1308.6138

[124] D. Erickson, ‘‘The Beacon OpenFlow controller,’’ in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 13–18.

[125] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M.Casado, and R. Sherwood, ‘‘On controller performance in software-defined networks,’’ in Proc. 2nd USENIX Conf. Hot Topics Manage. Internet Cloud Enterprise Netw. Services, 2012, p. 10.

[126] Z. Cai, A. L. Cox, and T. S. E. Ng, ‘‘Maestro: A system for scalable OpenFlow control,’’ Rice Univ., Houston, TX, USA, Tech. Rep., 2011.

[127] Project Floodlight, ‘‘Floodlight,’’ 2012. [Online]. Available: http://floodlight. openflowhub.org/

[128] Y. Takamiya and N. Karanatsios, ‘‘Trema OpenFlow controller framework,’’ 2012. [Online]. Available: https://github.com/ trema/trema

[129] Nippon Telegraph and Telephone Corporation, ‘‘RYU network operating system,’’ 2012. [Online]. Available: http://osrg.github.com/ryu/

[130] M. Banikazemi, D. Olshefski, A. Shaikh, J.Tracey, and G. Wang, ‘‘Meridian: An SDN platform for cloud network services,’’ IEEE Commun. Mag., vol. 51, no. 2, pp 120–127, Feb. 2013.

[131] NEC, ‘‘Award-winning software-defined networking NEC ProgrammableFlow networking suite,’’ Sep. 2013. [Online]. Available: http://www.necam.com/ docs/?id=67c33426-0a2b-4b87-9a7a-d3cecc14d26a

[132] S. Shin et al., ‘‘Rosemary: A robust, secure, high-performance network operating system,’’ in Proc. 21st ACM Conf. Comput. Commun. Security, Scottsdale, AZ, USA, Nov. 2014, pp. 78–89.

[133] NEC, ‘‘ProgrammableFlow family of products,’’ 2013. [Online]. Available: http://www.necam.com/SDN/

[134] A. Tootoonchian and Y. Ganjali, ‘‘HyperFlow: A distributed control plane for OpenFlow,’’ in Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw., 2010, pp 3.

[135] M. Monaco, O. Michel, and E. Keller, ‘‘Applying operating system principles to SDN controller design,’’ in Proc. 12th ACM Workshop Hot Topics Netw., College Park, MD, USA, Nov. 2013, DOI: 10.1145/ 2535771.2535789.

[136] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, ‘‘Participatory networking: An API for application control of SDNs,’’ in Proc. ACM SIGCOMM Conf., 2013, pp. 327–338.

[137] F. Botelho, A. Bessani, F. M. V. Ramos, and P. Ferreira, ‘‘On the design of practical fault-tolerant SDN controllers,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, 6 pp.

[138] S. Matsumoto, S. Hitz, and A. Perrig, ‘‘Fleet: Defending SDNs from malicious administrators,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp 103–108.

[139] L. Richardson and S. Ruby, RESTful Web Services. Sebastopol, CA, USA: O’Reilly Media, 2008.

[140] T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell, and S. Shenker, ‘‘Practical declarative network management,’’ in Proc. 1st ACM Workshop Res. Enterprise Netw., 2009, DOI: 10.1145/1592681.1592683.

[141] N. Foster et al., ‘‘Frenetic: A network programming language,’’ SIGPLAN Notes, vol. 46, no. 9, pp. 279–291, 2011.

[142] C. Monsanto, N. Foster, R. Harrison, and D. Walker, ‘‘A compiler and run-time system for network programming languages,’’ SIGPLAN Notes, vol. 47, no. 1, pp. 217–230, Jan. 2012.

[143] Open Networking Foundation (ONF), ‘‘OpenFlow notifications framework OpenFlow management,’’ Oct. 2013. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-notifications-framework-1.0.pdf

[144] B. Pfaff et al., ‘‘Extending networking into the virtualization layer,’’ in Proc. Workshop Hot Topics Netw., 2009, pp. 1–6.

[145] A. Singla and B. Rijsman, ‘‘Contrail architecture,’’ Juniper Networks, Tech. Rep., 2013.

[146] Open Networking Foundation (ONF), ‘‘OpenFlow management and configuration protocol (OF-Config 1.1.1),’’ Mar. 2014. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1-1-1.pdf

[147] D. Harrington, R. Presuhn, and B. Wijnen, ‘‘An architecture for describing simple network management protocol (SNMP) management frameworks,’’ Internet Engineering Task Force, Dec. 2002. [Online]. Available: http://www.ietf.org/ rfc/rfc3411.txt

[148] H. Yin et al., ‘‘SDNi: A message exchange protocol for software defined networks (SDNS) across multiple domains,’’ Internet Engineering Task Force, Internet Draft, Jun. 2012. [Online]. Available: http://tools. ietf.org/id/draft-yin-sdn-sdni-00.txt

[149] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, ‘‘Analysis of operating system diversity for intrusion tolerance,’’ Softw., Practice Experience, vol. 44, no. 6, pp. 735–770, 2014.

[150] Z. Wang, T. Tsou, J. Huang, X. Shi, and X. Yin, ‘‘Analysis of comparisons between OpenFlow and ForCES,’’ Internet Engineering Task Force, Internet Draft, Dec. 2011. [Online]. Available: http://tools. ietf.org/id/draft-wang-forces-compare-open-flow-forces-00.txt

[151] K. Ogawa, W. M. Wang, E. Haleplidis, and J.H. Salim, ‘‘ForCES Intra-NE high availability,’’ Internet Engineering Task Force, Internet Draft, Oct. 2013. [Online]. Available: http://tools.ietf.org/id/draft-ietf-forces-ceha-08.txt

[152] F. A. Botelho, F. M. V. Ramos, D. Kreutz, and A. N. Bessani, ‘‘On the feasibility of a consistent and fault-tolerant data store for SDNs,’’ in Proc. 2nd Eur. Workshop Softw. Defined Netw., 2013, pp. 38–43.

[153] S. Vinoski, ‘‘Advanced message queuing protocol,’’ IEEE Internet Comput., vol. 10, no. 6, pp. 87–89, Nov. 2006.

[154] M. Canini, P. Kuznetsov, D. Levin, and S. Schmid, ‘‘Software transactional networking: Concurrent and consistent policy composition,’’ in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, DOI: 10.1145/2491185. 2491200.

[155] A. Ghodsi, ‘‘Distributed k-ary system: Algorithms for distributed Hash tables,’’ Ph.D. dissertation, Dept. Electr., Comp. Software Syst., Royal Inst. Technol. (KTH), Stockholm, Sweden, Oct. 2006. W. Stallings, ‘‘Software-defined networks and OpenFlow,’’ Internet Protocol J., vol. 16, no. 1, pp. 1–6, 2013

[156] Open Networking Foundation (ONF), ‘‘SDN architecture,’’ Jun. 2014. [Online]. Available: https://www.opennetworking.org/ images/stories/downloads/sdn-resources/ technical-reports/TR_SDN_ARCH_1.0_ 06062014.pdf

[157] R. Hand and E. Keller, ‘‘ClosedFlow: OpenFlow-like control over proprietary devices,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 7–12.

[158] M. Jarschel, T. Zinner, T. HoQfeld, P.Tran-Gia, and W. Kellerer, ‘‘Interfaces, attributes, use cases: A compass for SDN,’’ IEEE Commun. Mag., vol. 52, no. 6, pp. 210–217, Jun. 2014.

[159] E. Mannie, ‘‘Generalized multi-protocol label switching (GMPLS) architecture,’’ Internet Engineering Task Force, RFC 3945 (Proposed Standard), Oct. 2004, updated by RFC 6002. [Online]. Available: http://www. ietf.org/rfc/rfc3945.txt

[160] K. Pentikousis, Y. Wang, and W. Hu, ‘‘MobileFlow: Toward software-defined mobile networks,’’ IEEE Commun. Mag., vol. 51, no. 7, pp. 44–53, Jul. 2013.

[161] N. Foster et al., ‘‘Frenetic: A network programming language,’’ SIGPLAN Notes, vol. 46, no. 9, pp. 279–291, 2011.

[162] A. Voellmy and P. Hudak, ‘‘Nettle: Taking the sting out of programming network routers,’’ in Proc. 13th Int. Conf. Practical Aspects Declarative Lang., 2011, pp. 235–249.

[163] C. Monsanto, N. Foster, R. Harrison, and D Walker, ‘‘A compiler and run-time system for network programming languages,’’ SIGPLAN Notes, vol. 47, no. 1, pp. 217–230, Jan. 2012.

[164] A. Voellmy, H. Kim, and N. Feamster, ‘‘Procera: A language for high-level reactive network control,’’ in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 43–48.

[165] C. Monsanto, J. Reich, N. Foster, J.Rexford, and D. Walker, ‘‘Composing software-defined networks,’’ in Proc. 10th USENIX Conf. Netw. Syst. Design Implement., 2013, pp. 1–14.

[166] C. J. Anderson et al., ‘‘NetKAT: Semantic foundations for networks,’’ SIGPLAN Notes, vol. 49, no. 1, pp. 113–126, Jan. 2014.

[167] S. Narayana, J. Rexford, and D. Walker, ‘‘Compiling path queries in software-defined networks,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 181–186.

[168] B. Casemore, ‘‘SDN controller ecosystems critical to market success,’’ 2012. [Online]. Available: http://nerdtwilight.wordpress. com/2012/06/05/sdn-controller-ecosystems-critical-to-market-success/

[169] R. Kwan and C. Leung, ‘‘A survey of scheduling and interference mitigation in LTE,’’ J. Electr. Comput. Eng., vol. 2010, Jan. 2010, DOI: 10.1155/2010/273486.

[170] A. Gudipati, D. Perry, L. E. Li, and S. Katti, ‘‘SoftRAN: Software defined radio access network,’’ in Proc. 2nd Workshop Hot Topics Softw. Defined Netw., 2013, pp. 25–30.

[171] J. Dix, ‘‘Clarifying the role of software-defined networking northbound APIs,’’ May 2013. [Online]. Available: http://www. networkworld.com/news/2013/050213-sherwood-269366.html

[172] I. Guis, ‘‘The SDN gold rush to the northbound API,’’ Nov. 2012. [Online]. Available: http://www.sdncentral.com/technology/the-sdn-gold-rush-to-the-northbound-api/2012/11/

[173] B. Salisbury, ‘‘The northbound APIVA big little problem,’’ 2012.

[174] G. Ferro, ‘‘Northbound API, southbound API, east/north LAN navigation in an OpenFlow world and an SDN compass,’’ Aug. 2012.

[175] B. Casemore, ‘‘Northbound API: The standardization debate,’’ Sep. 2012. [Online]. Available: http://nerdtwilight.wordpress.com/2012/09/18/northbound-api-the-standardization-debate/

[176] I. Pepelnjak, ‘‘SDN controller northbound API is the crucial missing piece,’’ Sep. 2012. [Online]. Available: http://blog.ioshints. info/2012/09/sdn-controller-northbound-api-is.html

[177] S. Johnson, ‘‘A primer on northbound APIs: Their role in a software-defined network,’’ Dec. 2012. [Online]. Available: http:// searchsdn.techtarget.com/feature/A-primer-on-northbound-APIs-Their-role-in-a-software-defined-network

[178] R. G. Little, ‘‘ONF to standardize northbound API for SDN applications?’’ Oct. 2013. [Online]. Available: http://searchsdn. techtarget.com/news/2240206604/ONF-to-standardize-northbound-API-for-SDN-applications

[179] Austin Common Standards Revision Group, ‘‘POSIX,’’ 2014. [Online]. Available: http://standards.ieee.org/develop/wg/ POSIX.html

[180] M. Yu, A. Wundsam, and M. Raju, ‘‘NOSIX: A lightweight portability layer for the SDN OS,’’ SIGCOMM Comput. Commun. Rev., vol. 44, no. 2, pp. 28–35, Apr. 2014.

[181] R. Chua, ‘‘OpenFlow northbound API: A new Olympic sport,’’ 2012. [Online]. Available: http://www.sdncentral.com/sdn-blog/ openflow-northbound-api-olympics/ 2012/07/

[182] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, ‘‘Modular SDN programming with pyretic,’’ USENIX Mag., vol. 38, no. 5, Oct. 2013.

[183] K.-K. Yap, T.-Y. Huang, B. Dodson,

M. S. Lam, and N. McKeown, ‘‘Towards software-friendly networks,’’ in Proc. 1st ACM Asia-Pacific Workshop Syst., 2010, pp. 49–54.

[184] S. Gutz, A. Story, C. Schlesinger, and

N. Foster, ‘‘Splendid isolation: A slice abstraction for software-defined networks,’’ in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 79–84.

[185] D. Turull, M. Hidell, and P. Sjo¨din, ‘‘Evaluating OpenFlow in libnetvirt,’’ in Proc. 8th Swedish Nat. Comput. Netw. Workshop, Oct. 2012, pp. 1–5.

[186] Quantum Community, ‘‘OpenStack networking (‘Quantum’),’’ 2012.

[187] M. Guzdial, ‘‘Education: Paving the way for computational thinking,’’ Commun. ACM, vol. 51, no. 8, pp. 25–27, Aug. 2008.

[188] M. S. Farooq, S. A. Khan, F. Ahmad, S. Islam, and A. Abid, ‘‘An evaluation framework and comparative analysis of the widely used first programming languages,’’ PLoS ONE, vol. 9, no. 2, 2014, DOI: 10.1371/journal. pone.0088941.

[189] A. D. Ferguson, A. Guha, C. Liang, R Fonseca, and S. Krishnamurthi, ‘‘Hierarchical policies for software defined networks,’’ in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 37–42.

[190] T. Nelson, A. D. Ferguson, M. J. Scheer, and S Krishnamurthi, ‘‘Tierless programming and reasoning for software-defined networks,’’ in Proc. 11th USENIX Symp. Netw. Syst. Design Implement., Apr. 2014, pp. 519–531.

[191] N. P. Katta, J. Rexford, and D. Walker, ‘‘Logic programming for software-defined networks,’’ in Proc. ACM SIGPLAN Workshop Cross-Model Lang. Design Implement., 2012, pp. 1–3.

[192] P. Porras et al., ‘‘A security enforcement kernel for OpenFlow networks,’’ in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 121–126.

[193] S. Shin et al., ‘‘FRESCO: Modular composable security services for software-defined networks,’’ Internet Society NDSS, Feb. 2013.

[194] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, ‘‘Model checking invariant security properties in OpenFlow,’’ in Proc. IEEE Int. Conf. Commun., Jun. 2013, pp. 1974–1979.

[195] A. Tootoonchian, M. Ghobadi, and Yanjali, ‘‘OpenTM: Traffic matrix estimator for OpenFlow networks,’’ in Proc. 11th Int. Conf. Passive Active Meas., 2010, pp. 201–210.

[196] M. Reitblatt, M. Canini, A. Guha, and N. Foster, ‘‘FatTire: Declarative fault tolerance for software defined networks,’’ in Proc. 2nd Workshop Hot Topics Softw. Defined Netw., 2013, pp. 109–114.

[197] A. Voellmy, J. Wang, Y. R. Yang, B. Ford, and P. Hudak, ‘‘Maple: Simplifying SDN programming using algorithmic policies,’’ in Proc. ACM SIGCOMM Conf., 2013, pp. 87–98.

[198] R. Soule, S. Basu, R. Kleinberg, E. G. Sirer, and N. Foster, ‘‘Managing the network with Merlin,’’ in Proc. 12th ACM Workshop Hot Topics Netw., Nov. 2013, DOI: 10.1145/ 2535771.2535792.

[199] X. Wen et al., ‘‘Compiling minimum incremental update for modular SDN languages,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 193–198.

[200] P. Pereini, M. Kuzniar, and D. Kostic, ‘‘OpenFlow needs you! A call for a discussion about a cleaner OpenFlow API,’’ in Proc. 2nd Eur. Workshop Softw. Defined Netw., Oct. 2013, pp. 44–49.

[201] F. Facca et al., ‘‘NetIDE: First steps towards an integrated development environment for portable network apps,’’ in Proc. 2nd Eur. Workshop Softw. Defined Netw., Oct. 2013, pp. 105–110.

[202] E. Reinecke, ‘‘Mapping the future of software-defined networking,’’ 2014. [Online]. Available: http://goo.gl/fQCvRF

[203] B. Heller et al., ‘‘ElasticTree: Saving energy in data center networks,’’ in Proc. 7th USENIX Conf. Netw. Syst. Design Implement., 2010, p. 17.

[204] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, ‘‘Hedera: Dynamic flow scheduling for data center networks,’’ in Proc. 7th USENIX Conf. Netw. Syst. Design Implement., 2010, p. 19.

[205] R. Wang, D. Butnariu, and J. Rexford, ‘‘OpenFlow-based server load balancing gone wild,’’ in Proc. 11th USENIX Conf. Hot Topics Manage. Internet Cloud Enterprise Netw. Services, 2011, p. 12.

[206] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari, ‘‘Plug-n-serve: Load-balancing web traffic using OpenFlow,’’ 2009.

[207] N. Handigol et al., ‘‘Aster*x: Load-balancing web traffic over wide-area networks,’’ 2009.

[208] C. Macapuna, C. Rothenberg, and M. Magalhaes, ‘‘In-packet bloom filter based data center networking with distributed OpenFlow controllers,’’ in Proc. IEEE GLOBECOM Workshops, 2010, pp. 584–588.

[209] Z. A. Qazi et al., ‘‘SIMPLE-fying middlebox policy enforcement using SDN,’’ in Proc. Conf. Appl. Technol. Architect. Protocols Comput. Commun., 2013, pp. 27–38.

[210] K. Jeong, J. Kim, and Y.-T. Kim, ‘‘QoS-aware network operating system for software defined networking with generalized OpenFlows,’’ in Proc. IEEE Netw. Oper. Manage. Symp., Apr. 2012, pp. 1167–1174.

[211] S. Sharma et al., ‘‘Implementing quality of service for the software defined networking enabled future internet,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, pp 49–54.

[212] W. Kim et al., ‘‘Automated and scalable QoS control for network convergence,’’ in Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw., 2010, p. 1.

[213] M. Scharf et al., ‘‘Dynamic VPN optimization by ALTO guidance,’’ in Proc. 2nd Eur. Workshop Softw. Defined Netw., Oct. 2013, pp. 13–18.

[214] P. Skoldstrom and B. C. Sanchez, ‘‘Virtual aggregation using SDN,’’ in Proc. 2nd Eur. Workshop Softw. Defined Netw., 2013, pp 56–61

[215] K. Nagaraj and S. Katti, ‘‘ProCel: Smart traffic handling for a scalable software EPC,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 43–48.

[216] M. S. Seddiki et al., ‘‘FlowQoS: QoS for the rest of us,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 207–208.

[217] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, ‘‘Optimizing rules placement in OpenFlow networks: Trading routing for better efficiency,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp 127–132.

[218] A. Schwabe and H. Karl, ‘‘Using MAC addresses as efficient routing labels in data centers,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 115–120.

[219] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, ‘‘A roadmap for traffic engineering in SDN-OpenFlow networks,’’ Comput. Netw., vol. 71, pp. 1–30, Oct. 2014.

[220] H. Ballani, P. Francis, T. Cao, and J. Wang, ‘‘Making routers last longer with Viaggre,’’ in Proc. 6th USENIX Symp. Netw. Syst. Design Implement., 2009, pp. 453–466.

[221] R. Alimi, R. Penno, and Y. Yang, ‘‘ALTO protocol,’’ Internet Engineering Task Force, Internet Draft, Mar. 2014. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-alto-protocol/

[222] M. Scharf et al., ‘‘Dynamic VPN optimization by ALTO guidance,’’ in Proc. 2nd Eur. Workshop Softw. Defined Netw., Oct. 2013, pp. 13–18.

[223] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, ‘‘SDN-based application-aware networking on the example of YouTube video streaming,’’ in Proc. 2nd Eur. Workshop Softw. Defined Netw., Oct. 2013, pp. 87–92.

[224] T. G. Edwards and W. Belkin, ‘‘Using SDN to facilitate precisely timed actions on real-time data streams,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 55–60.

[225] A. Ishimori, F. Farias, E. Cerqueira, and A. Abelem, ‘‘Control of multiple packet schedulers for improving QoS on OpenFlow/ SDN networking,’’ in Proc. 2nd Eur. Workshop Softw. Defined Netw., Oct. 2013, pp 81–86.

[226] H. Egilmez, S. Dane, K. Bagci, and A. Tekalp, ‘‘OpenQoS: An Open-Flow controller design for multimedia delivery with end-to-end quality of service over software-defined networks,’’ in Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf., 2012, pp. 1–8.

[227] H. Kumar, H. H. Gharakheili, and V. Sivaraman, ‘‘User control of quality of experience in home networks using SDN,’’ in Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst., 2013, DOI: 10.1109/ ANTS.2013.6802847.

[228] H. Ali-Ahmad et al., ‘‘CROWD: An SDN approach for densenets,’’ in Proc. 2nd Eur. Workshop Softw. Defined Netw., Oct. 2013, pp.25–31.

[229]J. Schulz-Zander et al., ‘‘Programmatic orchestration of Wi-Fi networks,’’ in Proc. USENIX Annu. Tech. Conf., Jun. 2014, pp 347–358.

[230] K.-K. Yap et al., ‘‘OpenRoads: Empowering research in mobile networks,’’ SIGCOMM Comput. Commun. Rev., vol. 40, no. 1, pp.125–126, Jan. 2010

[231]L. Li, Z. Mao, and J. Rexford, ‘‘Toward software-defined cellular networks,’’ in Proc. Eur. Workshop Softw. Defined Netw., 2012, pp. 7–12.

[232] X. Jin, L. Erran Li, L. Vanbever, and J. Rexford, ‘‘SoftCell: Scalable and flexible cellular core network architecture,’’ in Proc. 9th Int. Conf. Emerging Netw. Exp. Technol., 2013, pp. 163–174.

[233] P. Dely, A. Kassler, and N. Bayer, ‘‘OpenFlow for wireless mesh networks,’’ in Proc. 20th Int. Conf. Comput. Commun. Netw., 2011, pp 1–6.

[234] J. Vestin et al., ‘‘CloudMAC: Towards software defined WLANs,’’ SIGMOBILE Mob. Comput. Commun. Rev., vol. 16, no. 4, pp 42–45, Feb. 2013.

[235] M. J. Yang, S. Y. Lim, H. J. Park, and N. H. Park, ‘‘Solving the data overload: Device-to-device bearer control architecture for cellular data offloading,’’ IEEE Veh. Technol. Mag., vol. 8, no. 1, pp. 31–39, Mar. 2013.

[236] K.-K. Yap et al., ‘‘Blueprint for introducing innovation into wireless mobile networks,’’ in Proc. 2nd ACM SIGCOMM Workshop Virtualized Infrastructure Syst. Architect., 2010, pp. 25–32.

[237] M. Bansal, J. Mehlman, S. Katti, and P. Levis, ‘‘Openradio: A programmable wireless dataplane,’’ in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 109–114.

[238] S. Sundaresan et al., ‘‘Broadband internet performance: A view from the gateway,’’ SIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 134–145, Aug. 2011.

[239] S. A. Mehdi, J. Khalid, and S. A. Khayam, ‘‘Revisiting traffic anomaly detection using software defined networking,’’ in Proc. 14th Int. Conf. Recent Adv. Intrusion Detection, 2011, pp. 161–180.

[240] P. Wette and H. Karl, ‘‘Which flows are hiding behind my wildcard rule?: Adding packet sampling to OpenFlow,’’ in Proc. ACM SIGCOMM Conf., 2013, pp. 541–542.

[241] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, ‘‘Theory and practice of bloom filters for distributed systems,’’ IEEE Commun. Surv. Tut., vol. 14, no. 1, pp. 131–155, 2012.

[242] Y. Yu, C. Qian, and X. Li, ‘‘Distributed and collaborative traffic monitoring in software defined networks,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 85–90.

[243] J. Kempf et al., ‘‘Scalable fault management for OpenFlow,’’ in Proc. IEEE Int. Conf. Commun., Jun. 2012, pp. 6606–6610.

[244] G. Bianchi, M. Bonola, G. Picierro, S. Pontarelli, and M. Monaci, ‘‘StreaMon: A software-defined monitoring platform,’’ in Proc. 26th ITC, Sep. 2014, pp. 1–6.

[245] M. Yu, L. Jose, and R. Miao, ‘‘Software defined traffic measurement with OpenSketch,’’ in Proc. 10th USENIX Conf. Netw. Syst. Design Implement., 2013, pp. 29–42.

[246] J. Suh, T. Kwon, C. Dixon, W. Felter, and J.Carter, ‘‘OpenSample: A low-latency, sampling-based measurement platform for commodity SDN,’’ in Proc. IEEE 34th Int. Conf. Distrib. Comput. Syst., Jun. 2014, pp.228–237.

[247] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, ‘‘PayLess: A low cost network monitoring framework for software defined networks,’’ in Proc. 14th IEEE/IFIP Netw. Oper. Manage. Symp., 2014, DOI: 10.1109/ NOMS.2014.6838227.

[248] sFlow.org Forum, 2012. [Online]. Available: http://www. sflow.org/

[249] G. Stabler, A. Rosen, S. Goasguen, and K-C. Wang, ‘‘Elastic IP and security groups implementation using OpenFlow,’’ in Proc. 6th Int. Workshop Virtualization Technol. Distrib. Comput. Date, 2012, pp. 53–60.

[250] K. Wang, Y. Qi, B. Yang, Y. Xue, and J.Li, ‘‘LiveSec: Towards effective security management in large-scale production networks,’’ in Proc. 32nd Int. Conf. Distrib. Comput. Syst. Workshops, Jun. 2012, pp. 451–460.

[251] G. Yao, J. Bi, and P. Xiao, ‘‘Source address validation solution with OpenFlow/NOX architecture,’’ in Proc. 19th IEEE Int. Conf. Netw. Protocols, 2011, pp. 7–12.

[252] R. Braga, E. Mota, and A. Passito, ‘‘Lightweight DDoS flooding attack detection using NOX/OpenFlow,’’ in Proc. IEEE 35th Conf. Local Comput. Netw., Oct. 2010, pp. 408–415.

[253] K. Giotis, G. Androulidakis, and V. Maglaris, ‘‘Leveraging SDN for efficient anomaly detection and mitigation on legacy networks,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, 6 pp.

[254] G. Stabler, A. Rosen, S. Goasguen, and L.-C. Wang, ‘‘Elastic IP and security groups implementation using OpenFlow,’’ in Proc. 6th Int. Workshop Virtualization Technol. Distrib. Comput. Date, 2012, pp. 53–60.

[255] J. H. Jafarian, E. Al-Shaer, and Q. Duan, ‘‘OpenFlow random host mutation: Transparent moving target defense using software defined networking,’’ in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 127–132.

[256] S. Shin and G. Gu, ‘‘CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?),’’ in Proc. 20th IEEE Int. Conf. Netw. Protocols, 2012, DOI: 10.1109/ICNP.2012.6459946.

[257] J. Matias, J. Garay, A. Mendiola, N. Toledo, and E. Jacob, ‘‘FlowNAC: Flow-based network access control,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, 6 pp.

[258] A. Sapio et al., ‘‘MAPPER: A mobile application personal policy enforcement router for enterprise networks,’’ in Proc. 3rd Eur. Workshop Softw. Defined Netw., 2014, 2 pp.

[259] P. Porras et al., ‘‘A security enforcement kernel for OpenFlow networks,’’ in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 121–126.

[260] S. Shin, V. Yegneswaran, P. Porras, and Gu, ‘‘AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks,’’ in Proc. ACM Conf. Comput. Commun. Security, 2013, pp. 413–424.

[261] Y. Wang, Y. Zhang, V. Singh, C. Lumezanu, and G. Jiang, ‘‘NetFuse: Short-circuiting traffic surges in the cloud,’’ in Proc. IEEE Int. Conf. Commun., 2013, DOI: 10.1109/ICC. 2013.6655095.

[262] R. Hand, M. Ton, and E. Keller, ‘‘Active security,’’ in Proc. 12th ACM Workshop Hot Topics Netw., Nov. 2013, 17 pp.

[263] D. Kreutz, F. M. Ramos, and P. Verissimo, ‘‘Towards secure and dependable software-defined networks,’’ in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 55–60.

[264]K. Kant, ‘‘Data center evolution: A tutorial on state of the art, issues, challenges,’’ Comput. Netw., vol. 53, no. 17, pp. 2939–2965, 2009.

[265] A. Greenberg, J. Hamilton, D. A. Maltz, and P Patel, ‘‘The cost of a cloud: Research problems in data center networks,’’ SIGCOMM Comput. Commun. Rev., vol. 39, no. 1, pp. 68–73, Dec. 2008.

[266] M. Bari et al., ‘‘Data center network virtualization: A survey,’’ IEEE Commun. Surv. Tut., vol. 15, no. 2, pp. 909–928, 2013.

[267] A. Arefin, V. K. Singh, G. Jiang, Y. Zhang, and C. Lumezanu, ‘‘Diagnosing data center behavior flow by flow,’’ in Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst., Jul. 2013, DOI: 10.1109/ICDCS.2013.18.

[268] E. Keller, S. Ghorbani, M. Caesar, and J Rexford, ‘‘Live migration of an entire network (and its hosts),’’ in Proc. 11th ACM Workshop Hot Topics Netw., 2012, pp 109–114.

[269] R. Raghavendra, J. Lobo, and K.-W. Lee, ‘‘Dynamic graph query primitives for SDN-based cloudnetwork management,’’ in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 97–102.

[270] G. Wang, T. E. Ng, and A. Shaikh, ‘‘Programming your network at run-time for big data applications,’’ in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 103–108.

[271] A. Das et al., ‘‘Transparent and flexible network management for big data processing in the cloud,’’ in Proc. 5th USENIX Conf. Hot Topics Cloud Comput., San Jose, CA, USA, 2013, pp. 1–6.

[272] A. Krishnamurthy, S. P. Chandrabose, and a.Gember-Jacobson, ‘‘Pratyaastha: An efficient elastic distributed SDN control plane,’’ in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 133–138.

[273] T. Benson, A. Akella, A. Shaikh, and S. Sahu, ‘‘Cloudnaas: A cloud networking platform for enterprise applications,’’ in Proc. 2nd ACM Symp. Cloud Comput., 2011, pp 8:1–8:13

[274] M. Ghobadi, ‘‘TCP adaptation framework in data centers,’’ Ph.D. dissertation, Grad. Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2013.

[275] P. Calyam et al., ‘‘Leveraging OpenFlow for resource placement of virtual desktop cloud applications,’’ in Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage., 2013, pp. 311–319.

[276] Delivering a Software-based Network Infrastructure.Krish Prabhu. AT&T Labs (October 2015).

[277] Atrium: A Complete SDN Distribution from ONF. http://onosproject.org/wp-content/uploads/2015/06/PoC_atrium.pdf(2016).

[278] OpenStack: Open Source Cloud Computing Software. https://www.openstack.org/(2016).

[279] Docker: Build, Ship, Run Any App, Anywhere. https://www.docker.com/(2016).

[280] ONOS: Towards an Open, Distributed SDN OS. ***HotSDN 2014***P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O’Connor, P. Radoslavov, W. Snow, G. Parulkar (August 2014).

[281] XOS: An Extensible Cloud Operating System. ***ACM BigSystems 2015***.L. Peterson, S. Baker, A. Bavier S. Bhatia J. Nelson, M. Wawrzoniak, M. De Leeneer, and J. Hartman (June 2015).

[282] Network Functions Virtualization—An Introductory White Paper. ***SDN and OpenFlow World Congress***(October 2012).

[283] J. Erfanian, B. Daly (eds.), “NGMN 5G Initiative White Paper,” v. 1.0, NGMN, February 2015.

[284] A. Al-Shabibi, “Basic ONOS Tutorial,” ONOS Wiki, https://wiki.onosproject.org/ display/ONOS/Basic+ONOS+Tutorial, August 2016.

[285] Yui Mao, Changsheng You and Jun Zhang, “A Survey on Mobile Edge Computing: The Communication Perspective” IEEE Communications Surveys & Tutorials, Vol. 19, Issue 4, 2017, page 2322 – 2358

[286]M-CORD Open networking foundation at https://www.opennetworking.org/solutions/m-cord/

[287] Nasir Abbas, Yan Zhang and Amir Taherkordi, “Mobile Edge Computing: A Survey” IEEE Internet of Things Journal, Vol.5, Issue 1, Feb 2018, Page 450 – 465

[288] CORD Project | Whitepaper: M-CORD as an Open Reference Solution for 5G Enablement

[289] Agyapong, Patrick Kwadwo, et al. "Design considerations for a 5G network architecture." IEEE Communications Magazine 52.11 (2014): 65-75.

[290] Rost, Peter, et al. "Benefits and challenges of virtualization in 5G radio access networks." IEEE Communications Magazine 53.12 (2015): 75-82.

[291] Third (3rd) Generation Partnership Project (3GPP), Website: https://5g-ppp.eu/.

[292] “Local IP Access and Selected IP Traffic Offload (LIPA-SIPTO),” (3GPP TR 23.829).

[293] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, “The Click Modular Router,” ACM Trans. Comput. Syst., vol. 18, no. 3, pp. 263–297, Aug. 2000.

[294] D. Roeland and Z. Fu, “Mobile Service Chaining 5G Core Network Architecture,” Submitted to IEEE International Conference on Communications,2016.

[295] J. Blendin, J. Ruckert, N. Leymann, G. Schyguda, and D. Hausheer, “Position Paper: Software-Defined Network Service Chaining,” in 2014 Third European Workshop on Software Defined Networks (EWSDN), Sept 2014, pp. 109–114.

[296] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, “Performance characteristics of virtual switching,” in 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), Oct 2014, pp. 120–125.

[297] R. Morabito, J. Kj¨allman, and M. Komu, “Hypervisors vs. Lightweight Virtualization: A Performance Comparison,” in 2015 IEEE International Conference on Cloud Engineering (IC2E), March 2015, pp. 386–393.

[298] I. Cerrato, G. Marchetto, F. Risso, R. Sisto, and M. Virgilio, “An efficient data exchange algorithm for chained network functions,” in 2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR), July 2014, pp. 98–105.

[299] I. Cerrato, M. Annarumma, and F. Risso, “Supporting Fine-Grained Network Functions through Intel DPDK,” in 2014 Third European Workshop on Software Defined Networks (EWSDN), Sept 2014, pp. 1–6.

[300] L. Rizzo, M. Carbone, and G. Catalli, “Transparent acceleration of software packet forwarding using netmap,” in INFOCOM, 2012 Proceedings IEEE, Mar. 2012, pp. 2471–2479.

[301] M. Dobrescu, K. Argyraki, and S. Ratnasamy, “Toward Predictable Performance in Software Packet-Processing Platforms,” in Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12). San Jose, CA: USENIX, 2012, pp. 141–154.

[302] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, “ClickOS and the Art of Network Function Virtualization,” in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). Seattle, WA: USENIX Association, Apr. 2014, pp. 459–473.

[303] W. Li and A. Kanso, “Comparing Containers versus Virtual Machines for Achieving High Availability,” in 2015 IEEE International Conference on Cloud Engineering (IC2E), March 2015, pp. 353–358.

[304] W. Li, A. Kanso, and A. Gherbi, “Leveraging Linux Containers to Achieve High Availability for Cloud Services,” in 2015 IEEE International Conference on Cloud Engineering (IC2E), March 2015, pp.
76–83.

[305] O. N. Foundation, “OpenFlow Switch Specification Version 1.5.0,” Dec 2014. [Online]. Available: https: //www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf

[306] L. Rizzo, “netmap: a novel framework for fast packet I/O,” in Proceedings of the 2012 USENIX Annual Technical Conference (USENIX ATC 12). Bellevue, WA: USENIX Association, Aug. 2012, pp. 101–112.

[307] L. Rizzo and G. Lettieri, “VALE, a Switched Ethernet for Virtual Machines,” in Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, ser. CoNEXT ’12.
New York, NY, USA: ACM, 2012, pp. 61–72.

[308] “Universal Node Initial Benchmarking Documentation,” D5.4 deliverable of EU FP7 project UNIFY (will be publicly available at https://www.fp7-unify.eu/index.php/results.html#Deliverables).

[309] P. Quinn and T. Nadeau. Problem Statement for Service Function Chaining. Visited Apr 2015. [Online]. Available: https://tools.ietf.org/ html/rfc7498

[310] M-CORD as an Open Reference Solution for 5G Enablement (CORD PROJECT – Whitepaper.)