SDN based service oriented control approach for future radio access networks

Gopalasingham Aravinthan

Ecole doctorale : Informatique, Télécommunications et Electronique de Paris
Spécialité : Informatique et Télécommunications

SDN Based Service Oriented Control Approach For Future Radio Access Networks

Approche de contrôle orientée services basée sur SDN Pour les futurs réseaux mobiles

Présentée le 16 juin 2017 par

ARAVINTHAN Gopalasingham

Pour obtenir le grade de **DOCTEUR DE TELECOM SUDPARIS**

Devant le jury composé de :

M. André-Luc Beylot	Professeur, ENSEEIHT, France	Rapporteur
M. Ruben H. Milocco	Professeur, Universidad Nacional del Comahue, Argentina	Rapporteur
M. Paul Mühlethaler	Professeur, Inria, France	Examinateur
Mme Selma Boumerdassi	Maitre de Conférences, HDR, le cnam, France	Examinatrice
M. Marcelo Dias de Amorim	Professeur, Directeur de Recherche, Université Paris 6 , France	Examinateur
M. Eric Renault	Maitre de Conférences, HDR, Telecom SudParis, France	Directeur de Thèse
M. Laurent Roullet	Responsable du groupe de Recherche, Virtualisation des Réseaux Mobiles, NOKIA Bell Labs, France	Superviseur

Numéro de la thèse :2017TELE0013

Contents

List of Figures

List of Tables

List of Publications

(1) Aravinthan Gopalasingham, Laurent Roullet, Nessrine Trabelsi, Chung Shue Chen, Abdelkrim Hebbar, Érick Bizouarn: Generalized software defined network platform for Radio Access Networks. IEEE Consumer Communications and Networking Conference (CCNC) 2016: 626-629.

(2) Aravinthan Gopalasingham, Quan Pham Van, Laurent Roullet, Chung Shue Chen, Éric Renault, Lionel Natarianni, Stephane De Marchi, Emmanuel Hamman: Software-Defined mobile backhaul for future Train to ground Communication services. IFIP Wireless and Mobile Networking Conference (WMNC) 2016: 161-167

(3) Aravinthan Gopalasingham, Quan Pham Van, Laurent Roullet, Chung Shue Chen, Eric Renault, Lionel Natarianni, Stephane De Marchi, Emmanuel Hamman: Demo/Poster: SDN for Future Train to Ground Communication Services. ACM MobiSys (Companion Volume) 2016: 134

(4) Aravinthan Gopalasingham, Dalia-Georgiana Herculea, Chung Shue Chen, Roullet Laurent. Virtualization of Radio Access Network by Virtual Machine and Docker: Practice and Performance Analysis. IFIP/IEEE International Symposium on Integrated Network Management (IM'17), 2017.

(5) Dora Boviz, Gopalasingham Aravinthan, Chung Shue Chen, Laurent Roullet: Physical layer split for user selective uplink joint reception in SDN enabled cloud-RAN. Australian Communications Theory Workshop (AusCTW) 2016: 83-88.

(6) Dora Boviz, Nivine Abbas, Gopalasingham Aravinthan, Chung Shue Chen, Mohamed Amine Dridi: Multi-cell coordination in Cloud RAN: Architecture and optimization. International Conference on Wireless Networks and Mobile Communications (WINCOM) 2016: 271-277.

(7) Abdelkader Outtagarts, Laurent Roullet, Bruno Mongazon-Cazavet, Gopalasingham Aravinthan: When IT Meets Telco: RAN as a Service. IEEE/ACM UCC 2015: 422-423.

(8) Matteo Artuso, Dora Boviz, Aleksandra Checko, Henrik Lehrmann Christiansen, Bruno Clerckx, Laura Cottatellucci, David Gesbert, Bobby Gizas, Aravinthan Gopalasingham, Faheem A. Khan, Jean Marc Kelif, Ralf R. Müller, Dimitrios Ntaikos, Konstantinos Ntougias, Constantinos B. Papadias, Borzoo Rassouli, Mohammad Ali Sedaghat, Tharmalingam Ratnarajah, Laurent Roullet, Stéphane Sénécal, Haifan Yin, Lin Zhou: Enhancing LTE with Cloud-RAN and Load-Controlled Parasitic Antenna Arrays. IEEE Communications Magazine 54(12): 183-191 (2016).

(9) Dora Boviz, Mohamed Amine Dridi, Nivine Abbas, Gopalasingham Aravinthan: Demo/Poster: Demonstration proposal: Multi-cell Coordination in Cloud RAN enabled by SDN, IEEE WCNC, 2017.

(10) Bogdan Uscumlic, Jesse Simsarian, Arnaud Dupas, Eric Dutisseuil, Gopalasingham Aravinthan, Thierry Zami, Sébastien Bigo and Yvan Pointurier: Cost Efficient Network Slicing for Optical Packet Switched Torus Data Centers. IEEE International Conference on Communications (ICC 2017).

Chapter 1

Introduction

1.1 Software-Defined Networking

The future Internet considered as a Next Generation Network (NGN) will provide integrated services, i.e. interactive, multimedia, data, real-time, and mobile services. With the rapidly growing data traffic over Internet, building and managing large IP/Ethernet networks become increasingly complex. Current network architectures partially meet only today's requirements of enterprises, carriers, and end users [10]. In today's network, there are many discrete sets of protocols (e.g, routing protocol, Ethernet, IP, etc.) used to connect the network devices over arbitrary distances, link speeds, and topologies. These protocols are defined in isolation to solve a specific problem without the benefit of any fundamental abstractions. This has resulted in one of the primary limitations of today's networks : **complexity.** For example, to add or move any devices, IT admins must (re)configure multiple hardware/software entities using device-level management tools considering the topology, vendor switch model, software version, etc. Due to this complexity, today's networks are configured in a relatively static way to minimize the risk of service disruption. **The static nature of networks** are the second limitation of today's network, because of today's dynamic server environment, server virtualization and virtual machines migration. Virtual machine migration challenges many aspects of traditional networking, from addressing schemes and namespaces to the basic notion of a segmented, routing-based design. In addition to adopting virtualization technologies, many enterprises today operate an IP converged network for voice, data, and video traffic [11]. While existing networks can provide differentiated QoS levels for different applications, the provisioning of those resources is highly manual. IT admins must configure each vendor's equipment separately, and adjust parameters such as network bandwidth and QoS on a per-session, per-application basis. Because of its static nature, the network cannot dynamically adapt to changing traffic, application, and user demands. In the other hand, **inconsistent policies** are also a limitation of today's network to implement network-wide policies. IT admins have to configure thousands of devices and mechanisms and the complexity of today's networks make it very difficult to apply a consistent set of accesses, security, QoS, and other policies. Furthermore, carriers and enterprises are always looking for new ways to deploy new capabilities and services in rapid response to changing business needs or user demands. However, their ability to respond is hindered by vendors equipment product cycles, which takes several years. Besides, the lack of standard limits the ability of network operators to design the network to their individual environments. All these limitations of the current network led the industry to a tipping point. In response, the industry has created the Software-Defined Networking (SDN) architecture

and is developing an associated standards [11].

Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth and dynamic nature of today's applications. The SDN architecture decouples the network control and forwarding functions in the network devices and enables the network to be directly programmable. The SDN, by abstracting the underlying infrastructure, provides necessary intelligence and control for applications and network services [12]. The Open Networking Foundation (ONF) takes the lead in SDN standardization, and it presents the SDN architecture model [11] as depicted in Fig. 1.1. According to the ONF, the SDN architecture is:

- *Directly programmable:* Network control is directly programmable because it is decoupled from forwarding functions.
- *Agile:* Abstracting control from forwarding allows dynamically adjust network-wide traffic flow to meet changing needs.
- *Centrally managed:* Control functions are (logically) centralized in software-based SDN controllers that maintain a global view of the network.
- *Programmatically configured:* SDN allows network managers to configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs.
- *Open standard-based and vendor-neutral:* implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

The ONF/SDN architecture consists of three distinct layers:

- *The Application layer* consists of the end-user business applications that communicate with the controller to provide different communication services.
- *The control layer* includes a centralized SDN controller, which maintains a global view of the network. It controls and supervises the network forwarding functionality through an open interface. Some of the open-source controllers are NOX, Floodlight, Ryu, OpenDayLight, etc.
- *The Physical layer* consists of the physical network devices, including Ethernet switches, Openvswitch and routers that provide packet switching and forwarding.

In a traditional network, network devices are composed of an embedded and distributed control plane that manages switching, routing and traffic engineering activities. This distributed control plane provides information used to build a forwarding table and configure packet handling policies (e.g. security, traffic shaping, forwarding, etc.). The data plane makes a decision based on the forwarding table on where to send frames and packets. Both of these planes are located in the same networking device as shown on Fig. 1.2. In contrast to traditional network, the SDN paradigm decouples the control plane from the network devices and becomes an external entity as shown in Fig. 1.2. The SDN control plane is implemented as a set of software modules with dedicated functionalities and can be seen as a logically centralized entity, controlling every data plane elements in the network. The SDN controllers use various network configuration protocols to interact with the forwarding tables of the routers, switches, and any SDN-compliant middle boxes in the network, thus bringing a level of programmability in the network. The distributed control plane protocols in the traditional network can be seen either as a network service or an application deployed in the controller. All applications can take advantage of the same network information (the global network view), leading (arguably) to more consistent and effective policy decisions while re-using control

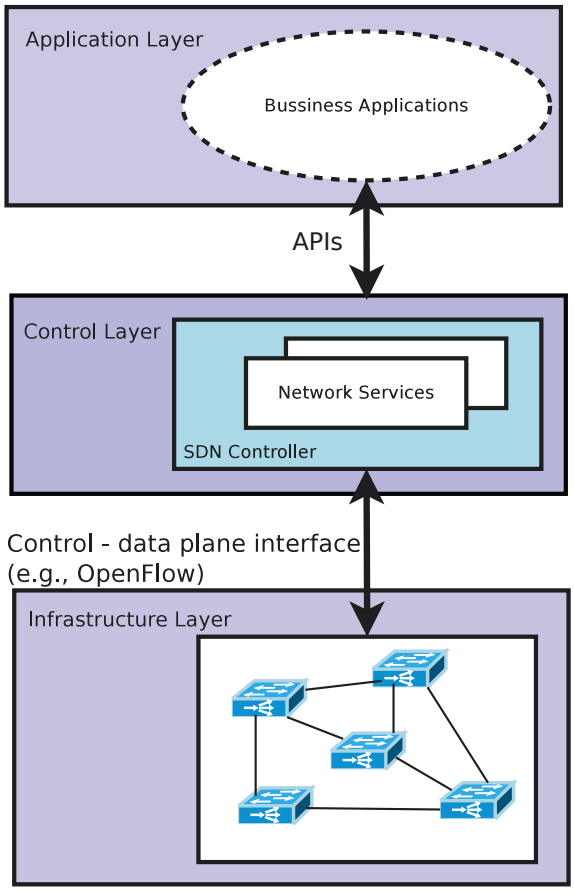


Figure 1.1 – Architecture: Software Defined Networking.

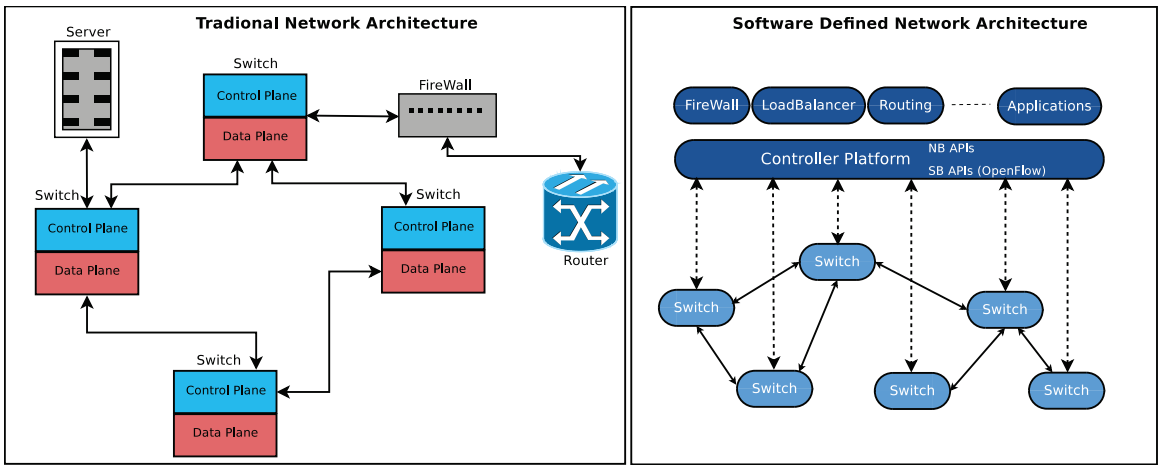


Figure 1.2 – Traditional networking vs. Software-Defined Networking (SDN).

plane software modules. These applications can take actions (i.e. reconfigure forwarding devices) from any part of the network. The integration of different applications becomes more straight forward. For example, load balancing and routing applications can be combined sequentially, with load balancing decisions having precedence over routing policies.

1.2 Next Generation Wireless Networks

The continuous growth in applications and services based on mobile broadband systems demands operators to anticipate for a network architecture that should be able to meet future extreme capacity and performance demands. In order to cope with such a demand, some network operators are now adopting cloud computing paradigm able to match capacity demand and reduce cost by offloading communication services from dedicated hardware in operator's core to server farms located in remote data-centers. But these services have different traffic characteristics and Quality-of-Service (QoS) constraints compared with conventional IT services that needs be considered in cloudification process [13].

Mobile Network operators and vendors believe that next-generation networks will be built out of existing and evolving technologies such as LTE, LTE-A and Wi-Fi along with recent revolutionary technologies to have gigabit user experience with virtually zero latency communications. The major principle requirements for next generation networks is to have stretched network performance with diverse use cases compare to its predecessors. In order to have such rich accessibility, the future wireless network is predicted to be heterogeneous in nature with dense deployment of macro and small cells (HetNet) [14]. Thus, the dense deployment of base stations will bring more complex interference scenarios in the network. The joint frequency and time scheduling techniques comes from the LTE-A are the standard techniques to mitigate inter-cell interference issues in HetNets [15]. However, the current distributed solutions for managing radio resources proposed in LTE are not scalable for dense small deployments. In LTE-A, Co-ordinated multiple point transmission (CoMP) and network multi-input multi-output (MIMO) are techniques to improve user experience and network utilization. The RAN in order to support such techniques relies heavily on network layer for coordinating between distributed base stations during transmission/reception. This introduces communication overhead (latency) in executing such techniques in the current design of RAN. In order to meet these requirements and various service demands, the network should be very flexible and easily reconfigurable. The evolution of SDN is being discussed [16] for enabling the programmability and flexibility of the radio access networks (RAN). The major benefit of such approach is to allow complete intelligence to the centralized controller so that the functionality of RAN can be better optimized.

Cloud RAN (C-RAN) [17] architecture can address the requirements of network self-optimization, self-configuration and self-adaptation in software control through SDN and NFV. In the context of C-RAN, virtualization plays a key role in deploying and managing the life cycle of mobile network functions and services in the central cloud in order to achieve efficient resource utilization, elasticity, and load balancing. The Network Function Virtualizatio (NFV) also results in a significant reduction in the CAPEX as well as OPEX by using automation techniques such as orchestration [18] together with SDN.

1.3 Research Motivation

Although the main technologies for the next generation wireless networks i.e. 5G yet to be specified, the important use cases and requirements have been listed [19, 20]. The common understanding among researchers, operators and vendors is that the 5G will be a unified platform to deploy services such as mobile broadband, critical machine type of communications, and Internet of Things (IOT). The variety of use cases certainly demands new network architecture to support

such wider mixture of services. In order to cover such use cases and requirements the 5G Radio Access Network (RAN) needs to be designed in such a way that it should efficiently utilize the available spectrum. The softwarization of RAN is a method to facilitate both orchestration and management of radio resources. The clear separation of control and data layer along with end-to-end network abstraction from the concept of SDN will be a key solution to bring such flexibility and programmability in future wireless networks. From the RAN perspective, the softwarization will facilitate not only the orchestration of radio resources but also the management and sharing of spectrum. The integration of SDN into wireless networks allows spectrum to be managed more efficiently, by spectrum awareness, spectrum mobility and effective implementation of spectrum sharing strategies in networks. Thus the software-defined RAN control architecture brings more efficient usage of radio resources. This approach also benefits operators by enabling RAN sharing and slicing feature for their primary users such as mobile virtual network operators (MVNOs) and content providers to better tune their network according to service requirements. End users will have enhanced network experience by the improved and optimized coordination between different network entities. Hence, the introduction of SDN will also have significant business impact on the value chain of future mobile industry by reducing both Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) due to it's inherent advantages. Hence, this thesis focus on designing and validating an appropriate **Software-Defined Radio Access Network Architectures** that brings flexibility, programmability and unified management for various use cases and deployment scenarios of next generation wireless networks.

1.4 Organization of Thesis

The rest of the thesis is organized as follows. Chapter 2 presents the state-of-the-art SDN and NFV technologies and their advantages, challenges along with performance validation of different SDN controllers. Chapter 2 describes also the state-of-the-art software-defined network approach to wireless communication systems and challenges. Chapter 3 begins with the presentation of main objectives of the thesis focusing three use cases of next generation wireless networks. Then it follows with the presentation of our proposed software-defined architectures corresponding to three main use cases such as SDN for Self-Organizing Network (SON), SDN for wireless backhaul, and SDN for dynamic user splitting. Chapter 4 presents our prototypes, performance validation and results corresponding to two of the three main use cases such as SDN for SON and SDN for wireless backhaul. Chapter 5 presents the conclusion and future work.

Chapter 2

SDN and NFV: State-of-the-Art

This chapter describes in detail the evolution of SDN and NFV technologies, advantages and motivation to apply such technologies in next generation wireless networks.

2.1 OpenFlow Protocol

OpenFlow [21] is the first standard SDN protocol defined and developed to demonstrate the concept of SDN. It originally defined the communication protocol in SDN environments that enables the SDN Controller to directly interact with the forwarding plane of network devices such as switches and routers, both physical and virtual, so it can better adapt to changing business requirements. Today, OpenFlow (OF) is widely used SDN protocol in research experiments as well as in industrial deployments. The original concept of OF begun at standford university in 2008 for the purpose of experimentation in their campus network.

An Openflow compliant switch consist of one or more flow tables with "match-action" or "match plus action" known as flow rules configure via OpenFlow applications from the controller. Each flow entry consists of a match rule to identify packets and an action that specifies the processing to be applied to matching packets. At the reception of packets, the switch compares it against flow entries in it's table and processes it according to the associated match actions. The set of available actions includes forwarding to one or more ports, dropping the packet, placing the packet in an output queues, and modifying, inserting, or deleting fields. Controllers program switches with the OpenFlow API by specifying a set of match-action entries. If there is no match found for particular packet (normally the first packet of a new flow) in it's table, the switch forwards it to the SDN controller for decision. The controller decision depends on the forwarding logic of network applications (routing, switching, load balancing) deployed by the network managers.

Recently, OpenFlow has garnered high interest from network managers, operators, vendors and researchers due to: i) it's flexible and low-cost implementation ii) it's ability to support broad range of research and deployment use cases. Although Ethane was the first protocol to introduce the concept of a logically centralized network architecture for managing enterprise network security policies, OpenFlow was the standard protocol created by researchers at Stanford and Berkley to provide a more generic network abstraction [21]. There are many series of OpenFlow compliant products in the market from vendors such as Nokia [22], Extreme Network [23], Dell [24], Hewlett-Packard [25], IBM [26], Juniper [27], Mellanox [28] and Pica [29].

Nowadays, OpenFlow has been widely used for data centers [30], cloud networking [31], network virtualization [32] and network management [33]. In general, the focuses is mainly on wired network for the purpose of packet switching, routing and flow isolation. The Open Networking Foundation (ONF) OpenFlow working group frequently releases updated versions of the OpenFlow protocol to support various switching and routing solutions and to align with the SDN paradigm. Beyond commercial use cases, OpenFlow is also used to experiment and validate new network protocols and applications at line rate with real traffic without requiring any custom hardware to be developed. OpenFlow has been demonstrated as a solution to provide : i) network slicing and isolation between different groups of users [34], ii) ability to utilize multiple networks simultaneously [35] iii) convergence between packet and circuit networks [36] iv) improvements in network fault recovery and debugging mechanism [37] v) a significant reduction in energy consumption of data center networks [38]. The ONF published several versions of OpenFlow protocol. OpenFlow version 1.0 is the first release to support the switch model with a single flow table and fixed set of match fields [39]. Version 1.1 extended the switch model to support group tables for executing sequential actions [40]. It supports MPLS networking, provides multipath routing and enables virtual port for creating tunnel endpoints. The support for IPV6 is introduced in version 1.2 [41]. OpenFlow 1.3 included the capability to have logical port abstraction along with support for provider backbone bridging (PBB), tunneling and extended QoS [42]. OpenFlow 1.4 is the first version that includes optical networking features by adding support for optical ports [43]. The complete road map explains the different versions of OpenFlow protocol is shown in the Fig. 2.1;

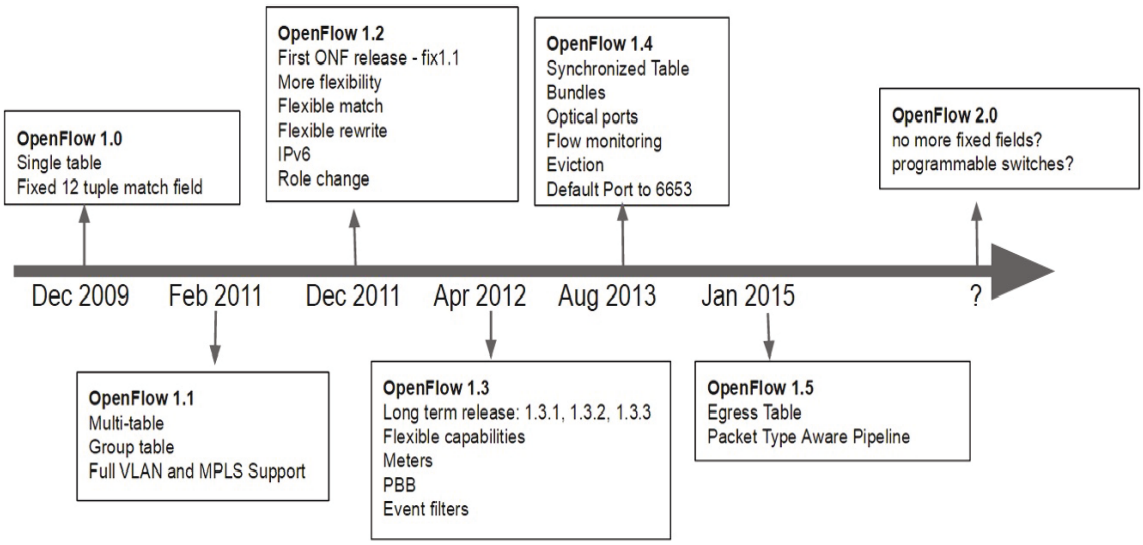


Figure 2.1 – OpenFlow Road Map.

In general, the OpenFlow protocol describes message exchanges between a controller and an OpenFlow switch. Typically, the protocol is implemented on top of SSL or Transport Layer Security (TLS), providing a secure OpenFlow channel. The OpenFlow protocol enables the controller to perform add, update, and delete actions to the flow entries in the flow tables. It supports three types of messages:

- **Controller-to-Switch:** These messages are initiated by the controller and, in some cases, require a response from the switch. This class of messages enables the controller to manage the logical state of the switch, including its configuration and details of flow- and group-table entries. Also

included in this class is the Packet-out message. This message is used when a switch sends a packet to the controller and the controller decides not to drop the packet but to direct it to a switch output port. Table 2.1 lists the set of messages between controller and OpenFlow switch.

Message	Description
Features	Request the capabilities of a switch. The Switch responds with a features reply that specifies its capabilities.
Configuration	Set and query configuration parameters. The Switch responds with parameter settings.
Modify-State	Add, delete, and modify flow/group entries and set switch port properties.
Read-State	Collect information from the switch, such as current configuration, statistics, and capabilities.
Packet-out	Direct packet to a specified port on the switch.
Barrier	Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations.
Role-Request	Set or query role of the OpenFlow channel. Useful when switch connects to multiple controllers.
Asynchronous-Configuration	Set filter on asynchronous messages or query that filter. Useful when the switch connects to multiple controllers.

Table 2.1 – Controller-to-Switch OpenFlow Messages.

- **Asynchronous:** These types of messages are sent without solicitation from the controller. This class includes various status messages to the controller. Also included is the Packet-in message, which may be used by the switch to send a packet to the controller when there is no flow-table match.

Message	Description
Packet-in	Transfer packet to controller.
Flow-Removed	Inform the controller about the removal of a flow entry from a flow table.
Port-Status	Inform the controller of a change on a port.
Error	Notify the controller of error or problem condition.

Table 2.2 – Asynchronous OpenFlow Messages.

- **Symmetric:** These messages are sent without solicitation from either the controller or the switch. They are simple yet helpful. Hello messages are typically sent back and forth between the controller and the switch when the connection is first established. Echo requests and reply messages can be used by either the switch or the controller to measure the latency or bandwidth of a controller-switch connection or just verify that the device is operating. The Experimenter message is used to stage features to be built into future versions of OpenFlow.

The OpenFlow protocol enables the controller to manage the logical structure of a switch, without regard to the details of how the switch implements the OpenFlow logical architecture. The fact that OpenFlow only defines the minimal set of hardware capabilities (via the switch

Message	Description
Hello	Transfer packet to the controller.
Echo	Echo request/reply messages can be sent from either the switch or the controller, and they must return an echo reply.
Experimenter	For additional functions.

Table 2.3 – Symmetric OpenFlow Messages.

specifications), and how to access them (via the OpenFlow channel), but not what to do with these switches, is what allows the network operator to define their own behaviors in their networks.

Although OpenFlow provides a powerful and vendor-independent approach to managing complex networks with dynamic demands, it explicitly specifies protocol headers on which it operates. This set has grown from 12 to 41 fields in a few years, increasing the complexity of the specification while not providing the flexibility to add new headers. The P4 [44], a high level language proposed to work in conjunction with OpenFlow to reduce the complexity in protocol specifications and allows programmers to take control of packet processors. Using P4, programmers can decide how the forwarding plane processes packets with out worrying about target hardware and implementation details. In summary, OpenFlow still requires languages such as P4 to well suits the programmable network of the future.

2.2 SDN Controllers

SDN controllers are available either as a proprietary or open source softwares. In this work, we focus on open source SDN controllers which provide opportunities for exploiting and also insight offered by the open source communities consuming the technology. Open source SDN controllers include, POX [45], FloodLight [46], OpenDayLight (ODL) [2], Ryu [47] and Open Networking Operating System (ONOS) [4], etc. The choice of a SDN controller depends on the services of the network operator or infrastructure owner. The Fig. 2.2 lists different SDN controllers and their use cases [1].

Controllers / Use-Cases	Trema	Nox/Pox	RYU	Floodlight	ODL	ONOS***
Network Virtualizaiton by Virtual Overlays	YES	YES	YES	PARTIAL	YES	NO
Hop-by-hop Network Virtualization	NO	NO	NO	YES	YES	YES
OpenStack Neutron Support	NO	NO	YES	YES	YES	NO
Legacy Network Interoperability	NO	NO	NO	NO	YES	PARTIAL
Service Insertion and Chaining	NO	NO	PARTIAL	NO	YES	PARTIAL
Network Monitoring	PARTIAL	PARTIAL	YES	YES	YES	YES
Policy Enforcement	NO	NO	NO	PARTIAL	YES	PARTIAL
Load Balancing	NO	NO	NO	NO	YES	NO
Traffic Engineering	PARTIAL	PARTIAL	PARTIAL	PARTIAL	YES	PARTIAL
Dynamic Network Taps	NO	NO	YES	YES	YES	NO
Multi-Layer Network Optimization	NO	NO	NO	NO	PARTIAL	PARTIAL
Transport Networks - NV, Traffic-Rerouting, Interconnecting DCs, etc.	NO	NO	PARTIAL	NO	PARTIAL	PARTIAL
Campus Networks	PARTIAL	PARTIAL	PARTIAL	PARTIAL	PARTIAL	NO
Routing	YES	NO	YES	YES	YES	YES

Figure 2.2 – SDN Controllers:Use Cases [1].

OpenDayLight Controller

The OpenDaylight project is a collaborative open source project that aims to accelerate the adoption of Software-Defined Networking and Network Functions Virtualization (NFV) for a more transparent approach that fosters new innovation and reduces risk. OpenDaylight (ODL) is a highly available, modular, extensible, scalable and multi-protocol controller infrastructure built for SDN deployments on modern heterogeneous multi-vendor networks. It provides a model-driven service abstraction platform that allows users to write applications that easily work across a wide variety of hardware and southbound protocols [2]. The goals of Opendaylight are: firstly, to create a robust, extensible, open source code base that covers the major common components required to build an SDN solution. Secondly, to get broad industry acceptance among vendors and users and finally, to have a thriving and growing technical community contributing to the code base, using the code in commercial products, and adding value above, below and around.

The OpenDaylight project is managed by the Linux Foundation and memberships covering over 40 companies, such as : Cisco, IBM, NEC, BROCADE, DELL, ERICSSON. It was founded in April, 2013 and so far, there are more than 40 sub-projects and five versions of OpenDaylight have been released, including Hydrogen, Helium, Lithium, Beryllium and the latest release is Boron. OpenDaylight project provides an open platform for developers to contribute, use, and even build commercial products as an individual. With 466 individuals contributing to the Boron release makes OpenDaylight one of the fastest-growing open source projects. It is being leveraged by a growing number of companies which offer solutions, applications, services, consulting and support to address a range of user needs.

In some parts of our research, OpenDaylight Hydrogen is used as an SDN controller that controls all functions and behaviors of network elements in the network. The detailed architecture setup and components of OpenDaylight Hydrogen is shown in Fig. 2.3.

The simplified architecture of ODL is shown in Fig. 2.4. The architecture mainly comprises three layers that include: the application Layer, the controller Layer, and the network device layer.

- **Application Layer** is formed by the Northbound API and the network application on top of the controller.
  - *Network application:* On top of OpenDayLight are the business and network logic applications that control and monitor the network behavior, such as VTN Coordination, OpenStack Neutron, etc.
  - *Northbound APIs:* they are used by applications. The provide two approaches to develop the application: the one based on OSGi framework and that must be deployed in the same address space as the controller and the other based on REST (Web-based) API that can be deployed on the same machine as the controller or on a different machine. In addition, northbound applications include more complex orchestration applications that engineer network traffic in accordance with the needs of environments such as the cloud and NFV.
- **The Controller Layer** consist of Service Abstraction Layer (SAL), base network functions and service modules. The controller in this part acts like a middleware in the OpenDaylight ecosystem. It is the framework that allows network applications and protocols to talk to the network devices for extracting services.
  - *Service Abstraction Layer:* In OpenDaylight, the SAL is the key design that enables the abstraction of services between the services consumers and producers. SAL acts like a large registry of

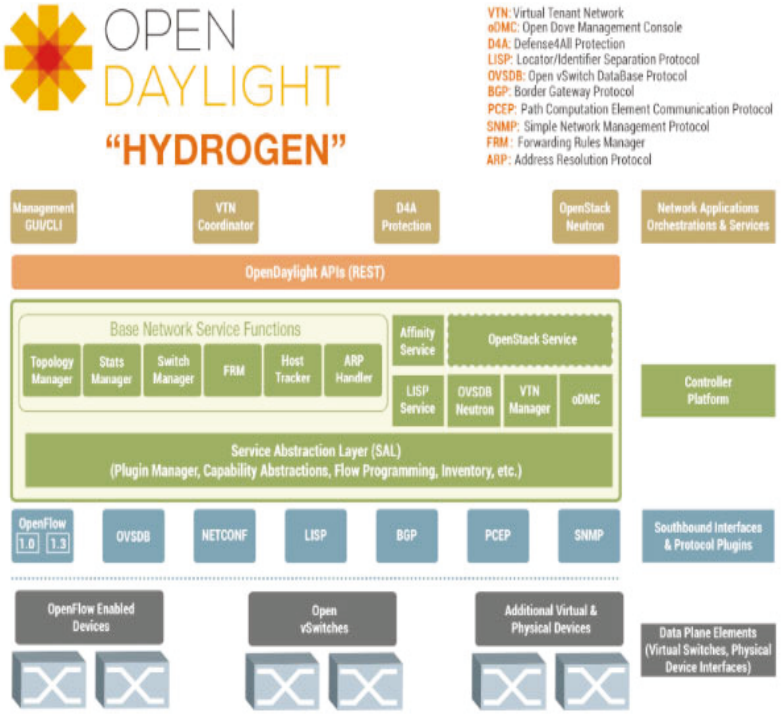


Figure 2.3 – OpenDaylight Hydrogen architecture and components [2].

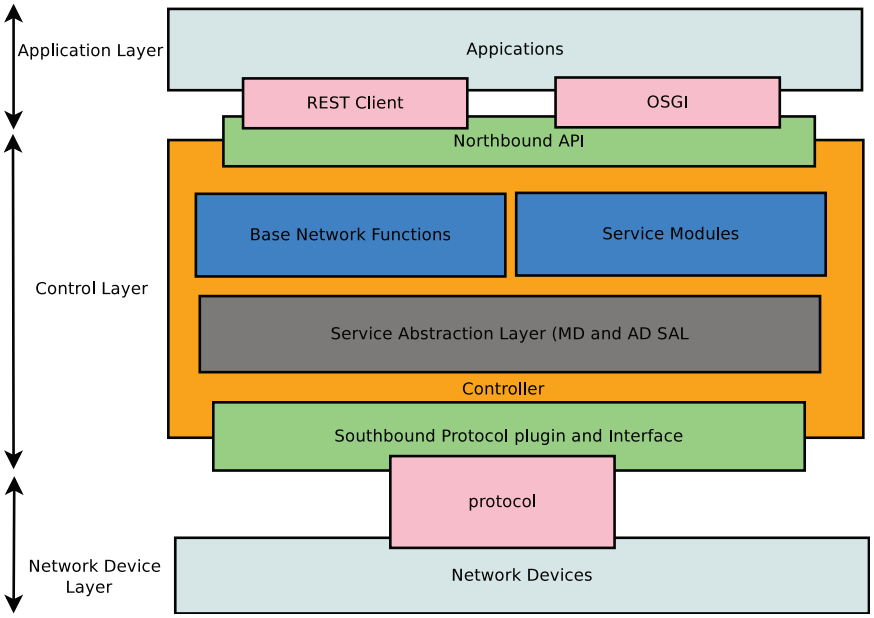


Figure 2.4 – Simplified Opendaylight Architecture [3].

services advertised by various modules and binds them to the applications that require them. Modules providing services, or producers, can register their APIs with the registry. When an application, or a consumer, requests a service via a generic API, SAL is responsible for assembling the request by binding producer and consumer into a contract, brokered and serviced by SAL. SAL has two architecturally different ways of implementing this registry: Application-driven SAL (AD-SAL) where the API is statically defined at compile/build time, and the provider

and consumer of events/data are directly plumbed to each other in code. In Model-Driven SAL (MD-SAL), the API code is generated from Yang model at compile time and loaded into the controller when the OSGI bundle plugin is loaded into the controller, all event calls and data go from provider to a consumer through a central datastore with MD-SAL [9].

◇ *Basic Network Functions:* The Base Network Service Functions section of Opendaylight Hydrogen includes six components shown in Table 2.4.

Network Funtions	Description
Topology Manager	Stores and handles information about the managed networking devices. When the controller starts, the Topology Manager creates the root node in the topology operational subtree. Then, it listens for notifications and updates this subtree with topology details, including all discovered switches and their interconnections. Notifications from other components, such as the Switch Manager or Device Manager, may also provide relevant topology information.
Statistics Manager	Implements statistics collection, sending statistics requests to all enabled nodes (managed switches) and storing responses in the statistics operational subtree. The Statistics Manager also exposes northbound APIs to return information on the following: node-connector (switch port), flow, meter, table, group statistics.
Switch Manager	Provides network nodes (switches) and node connectors (switch ports) details. As soon as the controller discovers network components, their parameters are saved to the Switch Manager data tree . One can use northbound APIs to get information on the discovered nodes and port devices.
Forwarding Rules Manager (FRM)	Manages basic forwarding rules (such as OpenFlow rules), resolves their conflicts, and validates them. The Forwarding Rules Manager communicates with southbound plugins and loads OpenFlow rules into the managed switches.
Inventory Manager	Queries and updates information about switches and ports managed by OpenDaylight, guaranteeing that the inventory database is accurate and up-to-date.
Host Tracker	Stores information about the end hosts (data layer address, switch type, port type, network address), and provides APIs that retrieve end node information. Host Tracker may work in a static or dynamic way. In case of dynamic operation, the Host Tracker uses ARP to track the status of the database. In static mode, the Host Tracker database is populated manually via northbound APIs.

Table 2.4 – Basic network functions of ODL [3].

◇ *Service Modules:* These are platform-oriented services and other extensions added into the controller platform for enhanced SDN functionality. Some platform-oriented service examples are: a VTN component to realize a multi-tenant network virtualization application using OpenFlow, affinity services that expose APIs to express workload relationships and service levels, or BGP-LS/PCEP to support traffic engineering with BGP-LS (BGP protocol library and topology model, etc.).

- **Network Device Layer:** It is comprised of southbound plugins and protocols. The southbound interface protocols are managing and controlling the underlying networking infrastructure. The

southbound interface is capable of supporting multiple protocols (as separate plugins), e.g. OpenFlow 1.0, OpenFlow 1.3, BGP-LS, LISP, SNMP, OVSDB, etc. These modules are dynamically linked to a service abstraction layer (SAL), which determines how to fulfill the service requested (by applications) irrespective of the underlying protocol used between the controller and the network devices.

OpenDayLight is written in JAVA and uses the following software tools/paradigms [9]:

- *Java interface* are used for event listening, specifications and forming patterns. This is the main way in which specific bundles implement call-back functions for events and indicate awareness of specific state.
- *Maven* is used by OpenDayLight for automated build process. Maven uses pom.xml (Project Object Model for this bundle) to script the dependencies between bundles and describe what bundles to load on start.
- *OSGi:* This framework in the backend of OpenDayLight that allows dynamically loading bundles and packaged Jar files, and binding bundles together for information exchange.
- *Karaf:* is a OSGi based runtime which provides lightweight container for loading different modules.

Bundle	Exported interface	Description
Arphandler	IHostFinder	Component responsible for learning about host location by handling ARP.
Hosttracker	IfIptoHost	Track the location of theopenflow-roadmap.jpg host relatively to the SDN network.
Switchmanager	ISwitchManager	Component holding the inventory information for all the known nodes (i.e. switches) in the controller.
Topologymanager	ITopologyManager	Component holding the whole network graph.
Usermanager	IUserManager	Component taking care of user management.
Statisticsmanager	IStatisticsManager	Component in charge of using the SAL ReadService to collect several statistics from the SDN network.
Sal	IReadService	Interface for retrieving the network nodes flow/port/queue hardware view.
Sal	ITopologyService	Topology methods provided by SAL toward the applications.
Sal	IFlowProgrammerService	Interface for installing/modifying/removing flows on a network node.
Sal	IDataPacketService	Data Packet Services SAL provides to the applications.
Web	IDaylightWeb	Component tracking the several pieces of the UI depending on bundles installed on the system.

Table 2.5 – List of Important Opendaylight Bundle [9].

The controller project offers certain core bundles, each of which export important services through Java interfaces. Table 2.5 is a brief list of the most important ones that come in handy while developing network services.

ONOS Controller

ONOS is an SDN operating system developed by ONOS Networking Laboratory and targeted at Service Provider Networks. ONOS provides high availability, performance and scalability to Service Provider Networks. The SDN controllers provides modular structure separating the various subsystems into independent modules. The architecture of ONOS is composed of a three tier structure namely; NorthBound, Distributed Core and SouthBound as shown in Fig. 2.5.

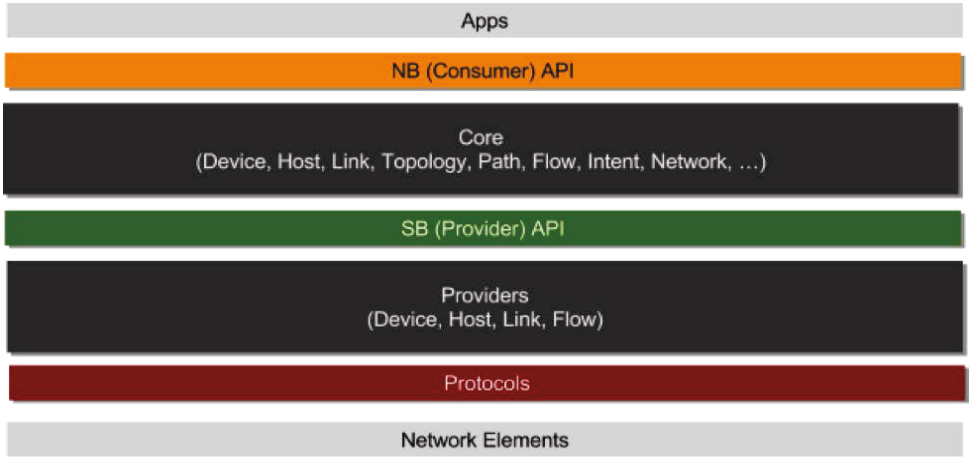


Figure 2.5 – ONOS-The System Architecture [4].

The ONOS NorthBound Interface is the top most layer of the three tier architecture responsible for reception and transmission of information and network requests respectively to the distributed core layer. SDN applications reside on the northbound and monitor network activities through a graphical user interface (GUI). Examples of such SDN applications include Topology Monitoring Applications, Device and Network Statistics Applications. These applications leverage on the northbound application programmable interfaces exposed by the subsystems or modules in the distributed core to synchronize the network state and present them to the user in real-time manner.

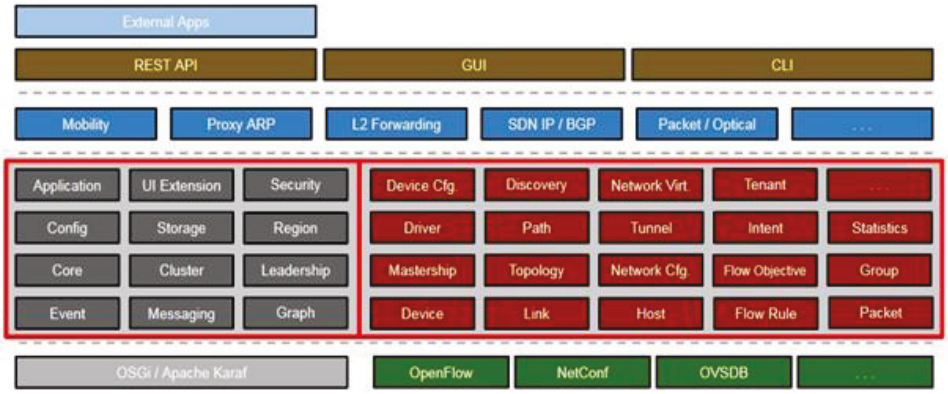


Figure 2.6 – ONOS-The Distributed Core System [4].

The distributed core is composed of a modular subsystem component exposing both NB and SB APIs to applications and infrastructure elements residing in the NB and SB respectively. The

subsystems are independent modules, however, the communicate with other subsystems for information pertaining to the network state. Network elements are abstracted into generic models including switches, hosts, and routers, and are not bound to have been specific protocols making them agnostic. Some of the subsystems include device, link, path, intent, etc. as indicated in Fig. 2.6. The core also leverages on Apache Karaf (java polymorphic container) for configuration and local or remote access to the ONOS instance through a Command Line Interface (CLI).

ONOS SouthBound Interface is the lowest tier of the ONOS architecture and communicates with the network infrastructure with specific device protocols. The SB APIs exposed by the distributed core communicate with the underlying network infrastructure to register and subscribe to the subsystem services to synchronise with network events. The available SB plugins in ONOS include Open Flow, NETCONFIG, OVSDB and SNMP for different protocol specific devices [4].

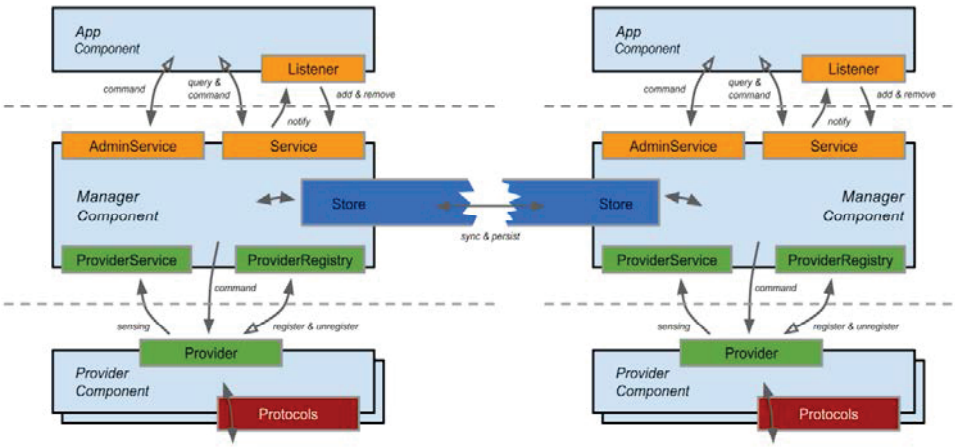


Figure 2.7 – ONOS-The Subsystem Structure [4].

The Generic Subsystem Model is the subsystem available in ONOS distributed core that follows a three layer structure with each layer residing in one of the main tier of the ONOS architecture as shown in Fig. 2.7. The layers are made of application, manager and provider components located in the NB interface, distributed core and SB interface respectively. Applications residing in this part of the APP component layer communicates with network infrastructure through NB APIs (AdminService and Service) exposed by the ONOS core. The applications register with the AdminService API of the subsystem and subscribe to services of these subsystems according to the concept of publish/subscribe. The application or user is updated about network events or any change relating to the subscribed service. The application can simultaneously send requests for networks or get service updates through the NB APIs. The Manager Component serves as the "broker" between the Application and the Provider. The Management Component exposes two North APIs (AdminService and Service) and two South APIs (ProviderRegistry and ProviderService). Northbound APIs permit applications to subscribe and receive network event changes by means of the event triggers within subsystems or modules. As part of the manager component, a store is responsible for tracking and persisting network state changes and synchronization with other instances of ONOS in a clustered environment. Network elements communicate with the provider with device specific protocols like Open Flow, NETCONFIG and SNMP in the southbound. Provider components register these network devices through a southbound API (ProviderRegistry) exposed by the Manager component. Registered devices could subscribe to services of the subsystem through its manager API (ProviderService) in order to be synchronized with event changes on the network.

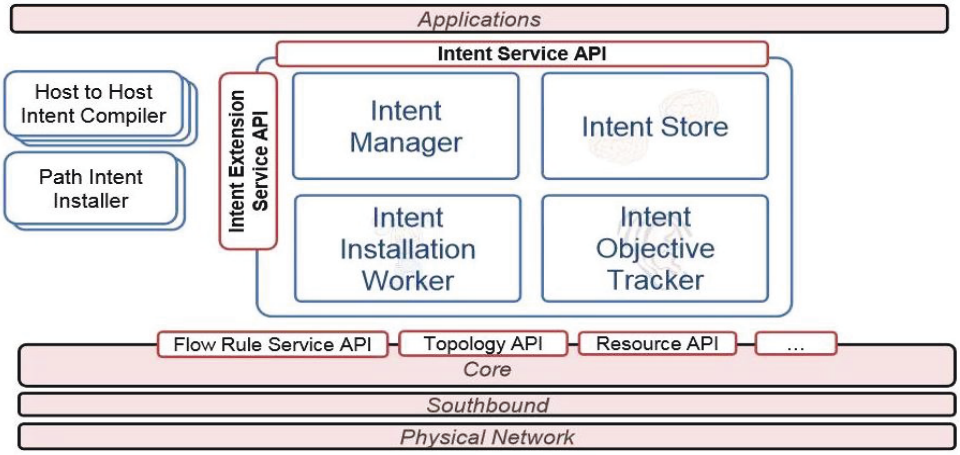


Figure 2.8 – ONOS-Intent Subsystem Architecture [4].

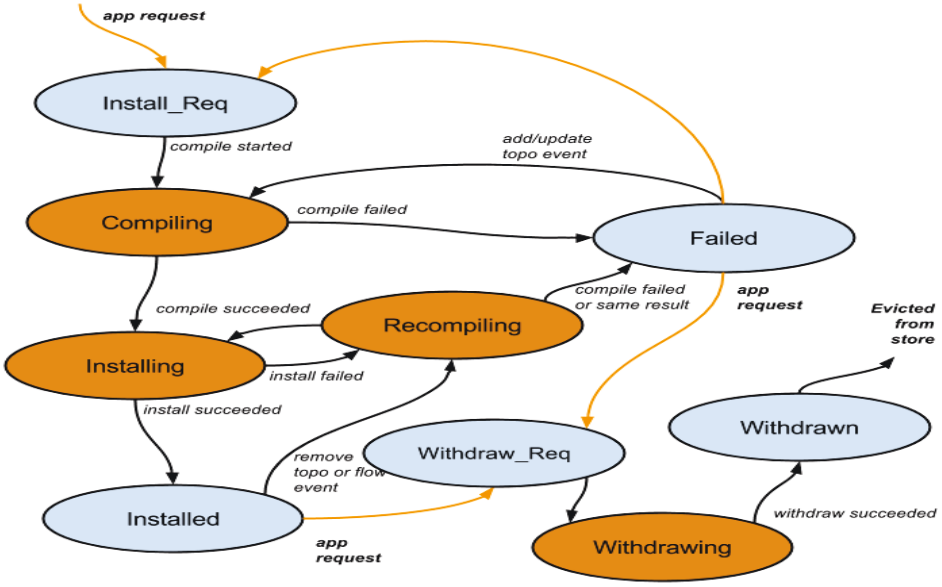


Figure 2.9 – ONOS-Architecture of Intent State Machine [4].

The intent framework is one of the subsystems allowing network applications or orchestrators to express network control desires, known as intents [4]. The subsystem follows the generic subsystem model as shown in Fig. 2.8. The intent framework has additional components, compilers and installers for intents. The functions of the various sub-modules of the framework are discussed as below. The core accepts intents from the NB interface and translates them into device instructions. Intent requests are transformed via a compilation process into installable (non-decomposable or primitive) intents. Installers are then responsible for actionable operations on the network. The intent framework is composed of default intents, compilers and installers based on defined use cases. Components of the intent framework designed by the Open Network Lab (ON Lab) are extensible and provide room for enhancement of intents or development of new intents depending on the network context or use case. The intent model follows the idea of the basic English language, i.e. subject, verb, object and predicate and the intent model includes subject, criteria (conditions), actions and constraints. One can relate subject to network resources including links, devices and hosts affected by the intents. The verb is equivalent to actions or instructions to be performed

based on a set of defined criteria. Criteria, answers the question of WHEN, i.e. a set of defined policies targeted for a network. Constraints or Properties explain the level of accessibility to network resources e.g. bandwidth, type of link and optical frequency for optical networks. Before an intent is executed into actionable instructions or into low-level device specific instructions, it is compiled and followed by an installation phase. The intent state-machine diagram in Fig. 2.9 shows the flow of the intent compilation and installation phase.

The intent state machine composed of four transitional states and five parking states. Upon receipt of the intent on the NB interface, depending on the type of intent request, the associated compiler is invoked. For example, to create an end-to-end connection or host-to-host intent, a host-to-host compiler is invoked. After successful compilation, an installation phase comes into completion. If the compilation fails due to changes in the network resources, intent lands into a failed parking state. A successful installation phase transits the intent into an installed state. Otherwise, the intent moves into a recompilation phase. Upon a successful re-compilation phase, the intent shows as an installed state and as a failed state otherwise. An installed intent can be removed through a withdrawal request which moves it to a transitional state, the "withdrawing state", and then finally into a "withdrawn state".

Performance Validation of SDN Controllers

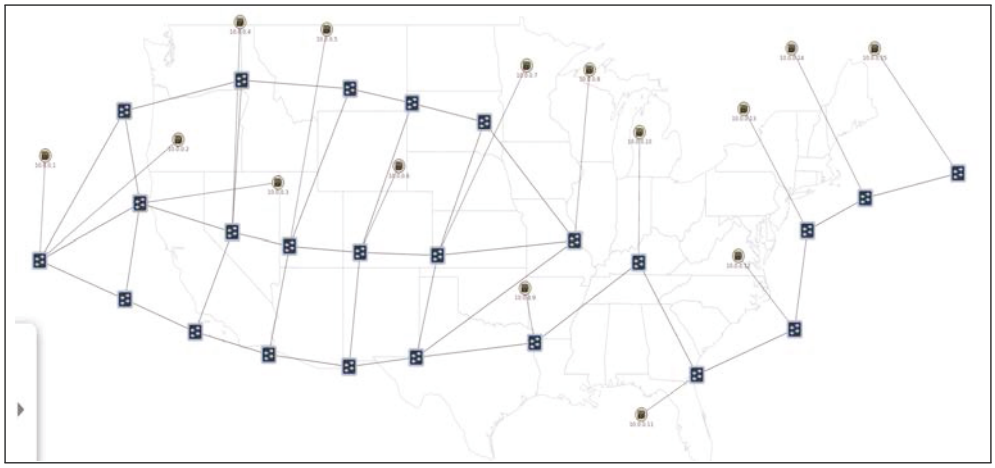


Figure 2.10 – Network Topology for Performance Validation.

In order to select suitable SDN controllers for our research, basic performance evaluation in terms of flow setup time i.e. the time each controller takes to setup path for new flow on its arrival were conducted. The network topology presented in Fig. 2.10 was used for test under different SDN controllers such as ODL, ONOS, FloodLight, RYU and POX. The opensource SDN controllers along with mininet network emulators were used for replicating the OpenFlow network deployed on the same PC. The measurements were taken by increasing number of intermediate network nodes i.e. switches between two emulated hosts sending and receiving traffic flow between them.

The performance comparison between different controllers are shown in Fig. 2.11 , where ONOS and ODL performs well, i.e. lowest average flow setup time as compared to other SDN controllers.

This is mainly due to the fact that ODL and ONOS are implemented for production network standard, where dynamic network abstraction and statistics are built to be in real time as compared to other controllers. As a result, the flow setup process (flow setup application) in ODL and ONOS can use the available real-time network abstraction details in the controller which results in reduction of flow setup time. Hence, for the implementation of our proposed architectures, we chose both ODL and ONOS respectively for software and hardware based prototypes.

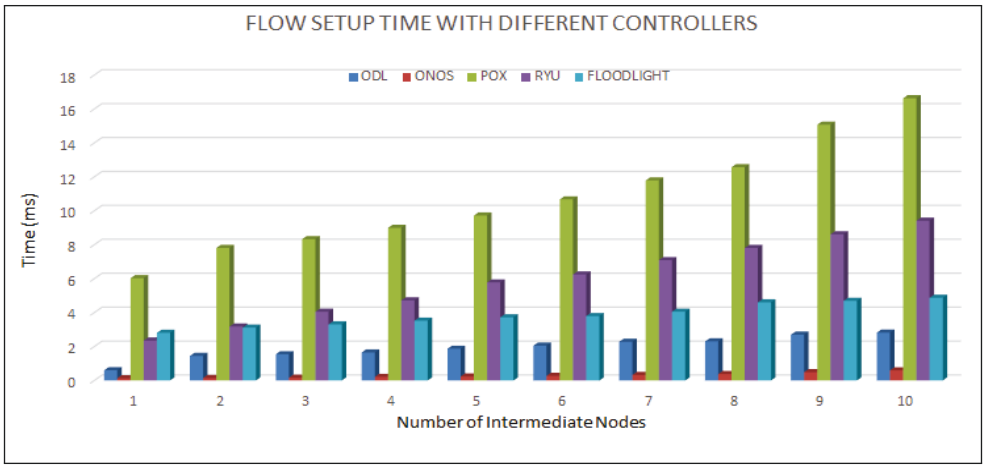


Figure 2.11 – SDN Controllers Performance Comparison.

In order to estimate the confidence interval to confirm the stability of the experimentations, the measurements were repeated 50 times with the same setup. The mean and standard deviation of each set of measurements under different controllers are calculated using formulas (2.2.1), and (2.2.2), where $\overline{x}$ denotes the mean of measurements, σ is the standard deviation and n is the number of samples, 50 in the present case.

$$\overline{x} = \frac{\sum x}{n} \tag{2.2.1}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n}} \tag{2.2.2}$$

Finally the confidence interval for the confidence level of 95% is calculated using the formula (2.2.3), where $Z_{\frac{a}{2}}$ is the confidence coefficient, a is the confidence level, σ is the standard deviation, and n is the sample size. The performance of each controllers is presented in Fig. 2.12.

$$C.I = [\overline{x} + Z_{\frac{a}{2}} * \frac{\sigma}{\sqrt{n}}, \overline{x} - Z_{\frac{a}{2}} * \frac{\sigma}{\sqrt{n}}] \tag{2.2.3}$$

The confidence interval in each of the controller performance is very narrow i.e. 95% of the measured flow setup time lies closer to the measured average value.

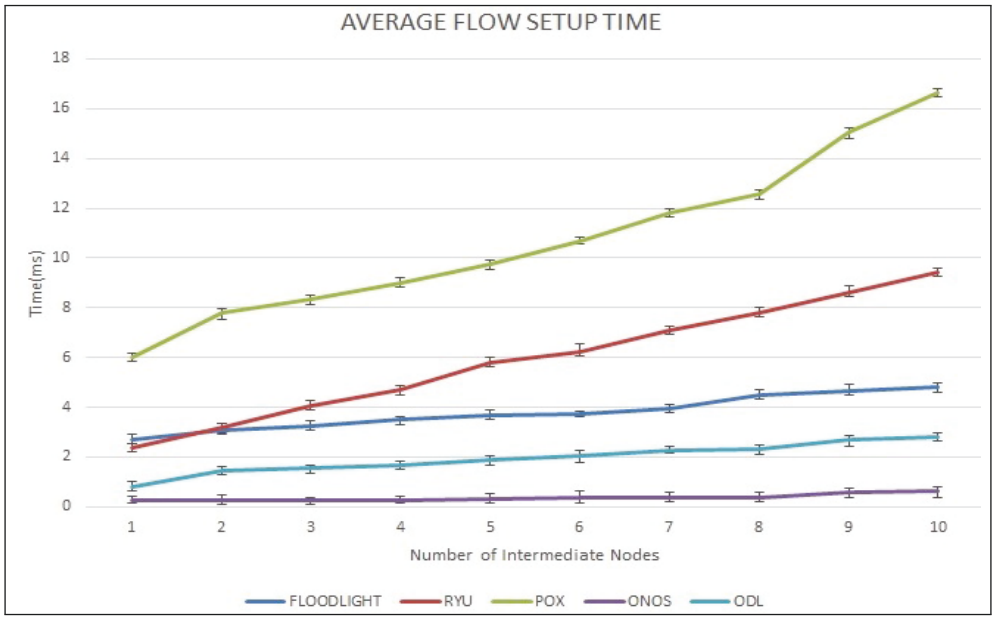


Figure 2.12 – SDN Controllers Performance Measurement: Confidence Interval.

2.3 Server Virtualization

The cloud computing paradigm extensively uses virtual machines (VMs) to share and isolate computing resources between different users. In particular, the Infrastructure-as-a-Service (IaaS) concept is built on top of VMs. Leading cloud service providers such as Amazon EC2 utilizes VMs as a way to provide resources to users on demand. Also, other cloud computing services such as Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS) are deployed on top of IaaS using VMs. Due to the fact that all cloud work loads are currently running in VMs, VM performance is a key aspect behind the overall cloud performance. As per the design, VM uses a limited number of hypercalls or emulated devices through hypervisor to establish communications with outside world with certain level of isolation and security. But the hypervisor adds measurable overheads on the cloud performance that can not be bypassed or optimized from higher layer. In the Cloud environment, VMs utilize emulated block devices backed by image files to access storage which is time consuming, and also disk images with duplicated content can waste storage space.

The virtualization based on containers has become an interesting solution in the cloud to provide operating system-level virtualization through a virtual environment that has its own process and network space, instead of a VM that runs full operating system as shown in Fig. 2.13 [48]. Linux or Unix is the preferred operating system for the cloud due to its zero price, large ecosystem, good hardware support, good performance, and reliability. The kernel namespace feature needed to implement containers in Linux has only become mature in the last few years[49]. The kernel namespace feature was originally found to be a solution to various difficulties in high performance computing clusters [49]. The clone () system call using kernel namespace can create multiple separate instances of previously global namespaces. Linux implements filesystem, PID, network, user, IPC, and hostname namespaces. For example, each filesystem namespace has its own root directory and mount table, similar to chroot() but more powerful. Though the concept of namespaces can be used in different ways, the most interesting and common approach is to have containers that are isolated, light-weight and elastic in nature. In order to instantiate containers, a container layer such as libcontainer or LXC has to installed on top of the OS which is usually a Linux variant.

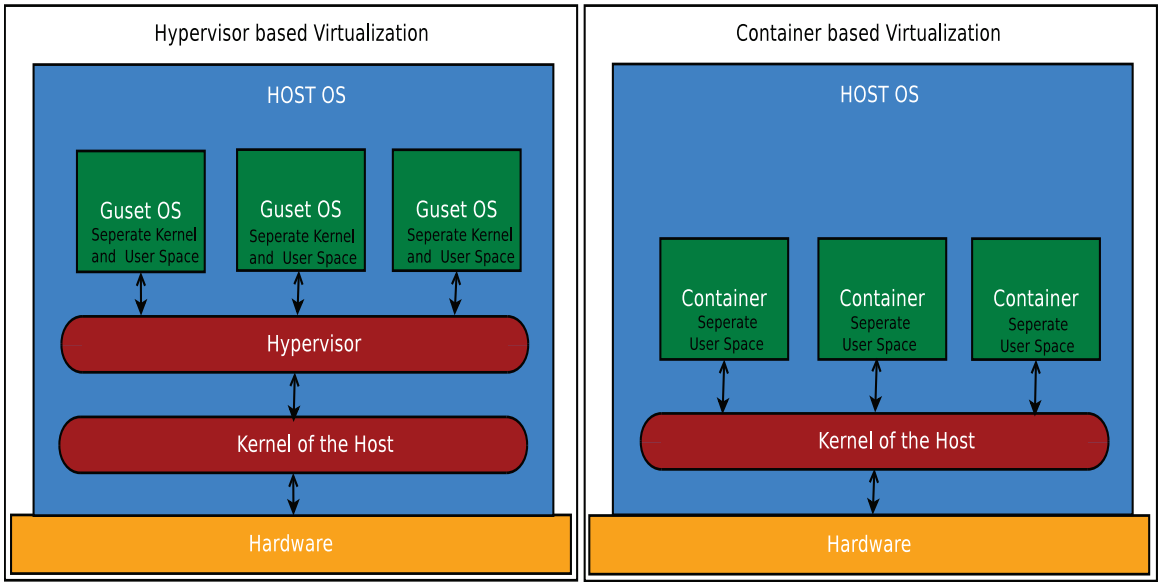


Figure 2.13 – Hypervisor Virtualization vs Container Virtualization.

Since all containers in the system share the same Kernel , the container instances are faster with almost zero performance overhead as compared to native VMs. Hence, the containers are considered to be an attractive solution for cloud service providers which can deploy a large number of container instances across the same hardware investment than using VMs. However the inherent dependency of container on the underlying kernel leads to the possibility of a single point of failure for all the deployed containers in the same system especially in the context of OS crashes or malware attacks. Also, containers running on the same physical machine have inherent dependencies on the underlying kernel. As a result, though the containers are easy to migrate, they can only be instantiated in another system with compatible OS kernels which is also a potential limitation to migration options.

There are several tools such as LXC [50], Docker [51], systemd-nspawn [52], Warden [53] and lmctfy [54] for creating and managing containers. Docker has been used widely as a container management tool due to features such as ease of use and layered file system that can be reused between containers with a reduction in space usage. Docker container images require less disk space and I/O as compared to VM disk images. This helps in faster container deployment in the cloud environment where images have to be copied to local disk before instantiation.

A container can be deployed in two different ways i.e. as an OS container or as an application container. In general, OS containers are used for running multiple applications, services, libraries, etc. Also, OS containers are used to deploy identical or different flavor of a container instance as shown in Fig. 2.14, where containers have to be created and instantiated from templates or images that decides the structure and contents of the containers. As depicted in Fig. 2.15, OS containers are used for deploying together multiple applications and services in a single container instance, where as application containers are designed to package and run a single service per container.

Although OS and application containers share the same kernel, application containers use the layering concept of dockers to create multiple containers from the same base image with a new layer for each container. Docker is a widely used tool for creating and managing application containers. In dockers, dockerfile defines the property and content of the container i.e. OS, packages, libraries,

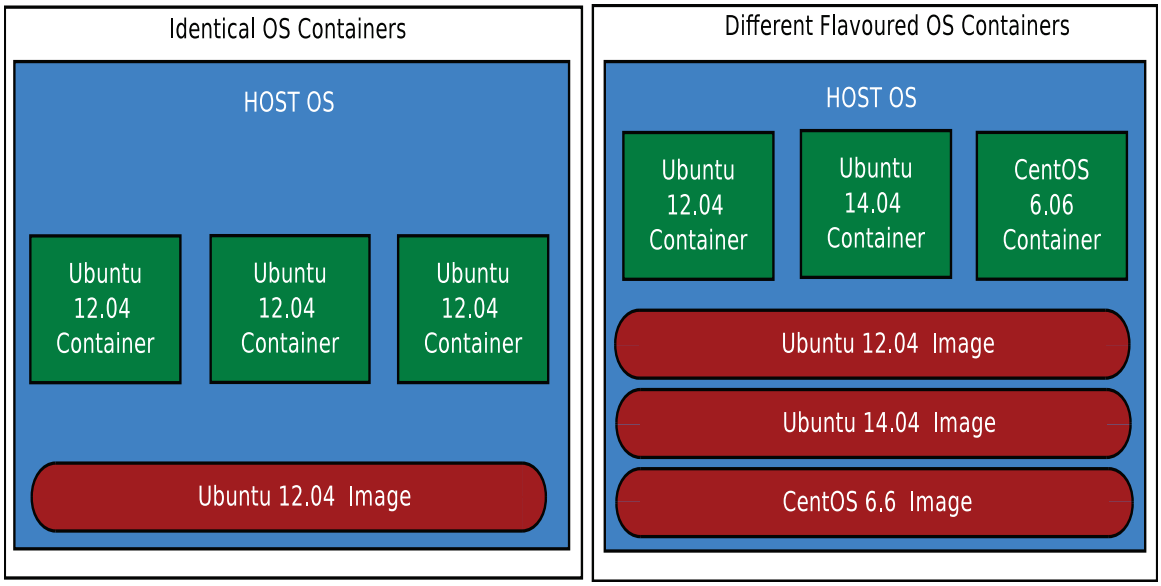


Figure 2.14 – Identical & Different Flavored OS containers.

etc. The *RUN* command specified in the dockerfile creates a new layer every time the dockerfile is executed, and during the instantiation the docker combines these layers to deploy the specific container.

The motivation behind application container is to use different isolated containers for each of the services in any application. As shown in Fig. 2.15, a typical web application consists of three different services such as presentation, application and database that can be deployed either in the same container (OS container) or in separate containers (application container) and they communicate with each other through dedicated APIs.

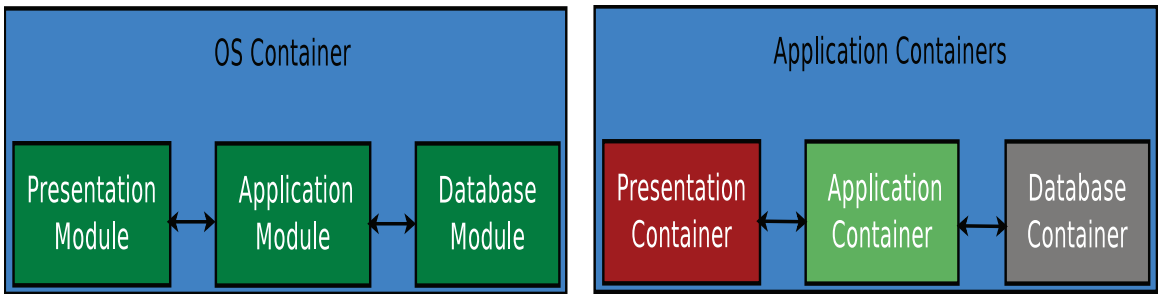


Figure 2.15 – OS containers vs Application containers.

The application container fits well for deploying multicomponent system as a micro-service architecture in a distributed computing environment. The major benefit for using application container is to create various versions and size of the same component from the base OS image using layering concept and choose them according to the application and user requirements. Since micro-service architecture and cloud computing paradigm plays a key role in the virtualization of resources in future 5G system [55], the container would be a suitable solution for deploying and running various processes, applications and services more efficiently.

2.4 Open vSwitch and Hypervisor Bridge

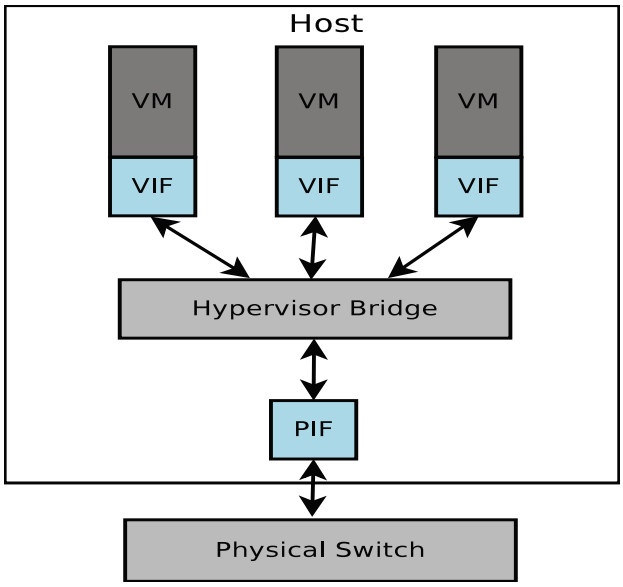


Figure 2.16 – Virtual Network using Hypervisor.

In the recent years, the rapid growth in server virtualization techniques has brought significant challenges in datacenter networking. In general, server virtualization imposes requirements on network mobility, scaling and isolation that is far beyond the characteristics of tradional networks. Also, tradional networking has significant drawbacks in order to adapt server virtualization such as classical IP does not support mobility in a scalable manner with out segmenting it into multiple logical networks through Virtual LANs (VLANs). However, in a datacenter environment, in order to deploy hundreds or thousands of VMs per physical host along with the possibility of migrating VMs between physical hosts that are shared between different tenants, network scalability, mobility and isolation become necessary. In the beginning, the networking in the virtual environment is supported through L2 switching [56] and IP routing [57] functionality implemented within the hypervisor or hardware management layer. These virtual swithing and routing features in the hypervisor supports limited networking such as communication between co-located VMs and their connectivity to the Physical Network Interface Card (NIC) of the host. As shown in Fig. 2.16, networking in a virtual environment is achieved through simple L2 switches within the hypervisor which connects the virtual interfaces (VIFs) of VMs to the Physical Interface (PIF) of the host and also provides networking between VMs located on the same host.

Though the networking through a hypervisor provided early solution to network provisioning in the virtual environment, it has many drawbacks in terms of network manageability. Normally, the traffic between VMs travel through the hypervisor without passing through the physical switch where the host is connected, so that the network administrator does not have any opportunity to monitor or control the traffic in the virtual environment. It was considered to be a minor issue until server virtualization became a potential driver for cloud computing solutions. Since then, the number of VMs deployed in a single physical host started to grow and 40 or more is not uncommon. For such situations, network administrators started to think about mechanisms to manage networking in the virtual environment, especially for an efficient utilization of the underlying physical resources, reliability and network security. These challenges resulted in the

emergence of network virtualization [33], where software switches provides primary network services to VMs by establishing IP tunnels between hypervisors through physical datacenter networks. This approach decouples the virtual network from the underlying physical networks and provides logical network abstractions, services and tools to administrators and users. The usage of software switches for network virtualization led to the term virtual switch and pave the way for adoption of software switches in datacenter networking. In oder to meet requirements of a virtual environment, new virtual switches have been developed to be highly flexible, vendor agnostic and low cost. Open vSwitch is one of the widely accepted and opensource virtual switch supporting common standards such as OpenFlow, SNMP and IPFIX. Today, Open vSwitch works with hypervisors and container systems such as Xen, KVM and Docker on its original Linux platform [58].

Open vSwitch supports OpenFlow feature since its inception. It is a multi purpose software switch that is re-programmable through the OpenFlow protocol [59]. This is an advantage as compared to the data path model of other virtual switches in which packet processing pipelines are fixed and only configuration of prearranged feature is possible. The flexibility and programmability of Open vSwitch has become an essential element in the SDN paradigm due to the intrinsic feature of OpenFlow support. However, the advanced use cases of SDN such as network virtualization results in higher packer classification loads that demands far longer packet processing pipelines in Open vSwitch than available in the traditional virtual switches. In order to accelerate the packet processing and also to reduce the consumption of hypervisor resources, the latest versions of Open vSwitch are implemented with flow caching feature in the kernel module to accelerate forwarding decisions for any packets belong to existing flows [60] rather than moving all the packets to user-space for decisions. This is similar to switching decision based on flow rules in the forwarding tables of OpenFlow switches that is setup previously by the centralized SDN controller. Open vSwitch is an opensource and multi-platform where the user has freedom to choose the OS distribution and hypervisor. This makes Open vSwitch modular and portable.

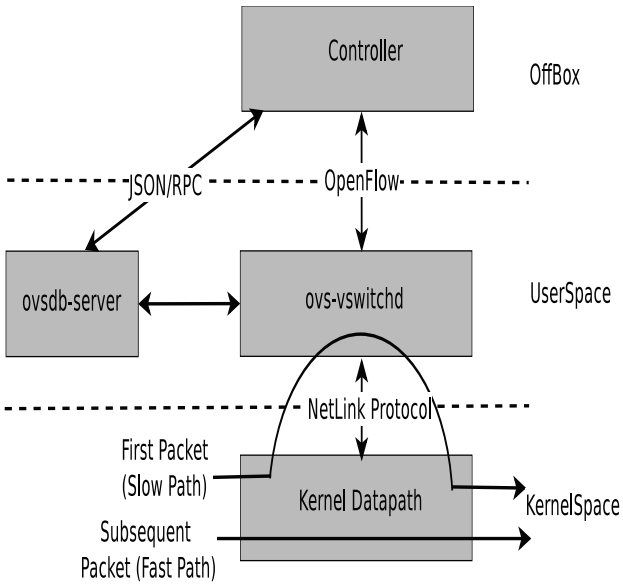


Figure 2.17 – Open vSwitch Architecture.

As depicted in Fig. 2.17, Open vSwitch consists of two main components such as ovs-vswitchd and a data-path kernel module for managing packet forwarding. The ovs-vswitchd daemon located

in the userspace that works in co-ordination with data-path module of kernel to take forwarding decisions. In oder to treat any incoming packet to the switch, the kernel module looks for associated matching rule in it's cache table, and if no action specified for the particular packet, it will be forwarded to ovs-vswitchd daemon for decision. In the userspace, the ovs-vswitchd daemon decides how the particular packet should be handled, then it passes the packet to the datapath daemon along with the instruction to store the forwarding action in its flow cache table for future decisions on packets belonging to the same flow. This is similar to the OpenFlow controller providing switch actions to OpenFlow switches for packet treatment.

The flow caching technique in Open vSwitch has greatly evolved over time with more features to ease and accelerate packet switching decisions. The initial release of Open vSwitch supports data-path with micro-flow cache that essentially caches per transport connection forwarding decision, but the latest version of Open vSwitch has two layer of caching: micro-flow cache as well as mega-flow cache which caches forwarding decisions for traffic aggregation more than individual connections.

Open vSwitch is widely used as an open source OpenFlow supported SDN switch, especially in virtual networking solutions. As shown in Fig. 2.17, the ovs-vswitchd daemon communicates with SDN controller using the OpenFlow protocol in order to receive flow tables. Using these flow tables it matches packets arriving from the datapath daemon and finally caches the resulted action in the kernel data-path. The kernel module and the ovs-vswitchd daemon communicate via the NetLink Protocol, OVS-DB Server and the controller or any external configuration modules communicate via JSON/RPC. OVSDB module in the controller is responsible for backing up the status of the ovsdb server time to time and also can configure the ovsdb server from the controller. The configurations are related to creating virtual ports, bridges, tunnels, etc. According to the controller's view, the implementation details and separation between user and kernel space modules in the Open vSwitch is completely isolated. It treats packet according to the detail of the network topology and packet's header information.

Open vSwitch, with its centralized control capability, enables the creation of an automated network infrastructure for virtualized environments along with the management of network policies that migrate with VMs. Open vSwitch also supports programmatic control of address remapping (L2 and L3 via OpenFlow), and programmatic control of tunnels as well as multiple tunnel types (e.g. GRE, IPsec, and CAPWAP) [61]. These come in handy for various network virtualization functions such as supporting mobility or an L2 service model across subnets. Open vSwitch is already used in many production environments. It is most commonly used as a vswitch in large cloud deployments (many thousands of servers) for automated VLAN, policy, and tunnel management. Open vSwitch has many extensions to standard OpenFlow to accommodate various use cases in network virtualization.

There are three common use cases of Open vSwitch in the virtualized environment: (i)flexibility in Centralized management, (ii) creating Virtual Private Networks, and (iii) enabling Mobility between IP subnets [62]. The dedicated interfaces for configuration and synchronous notification in Open vSwitch can be used to create a single logical view of the multiple switch instances (e.g. in the particular VLAN) running in physically separated servers. This facilitate network admins to manage the VMs during their migration process and ensure any configuration state remains coupled to the particular logical entity for which they are defined. As a result, each VM has a unique virtual port through out the particular network.

Providing isolation and security for particular tenant who requires multiple and inter-connected VMs in the distributed cloud computing environment is an important and crucial task for cloud solution providers. Open vSwitch can be a potential solution for creating virtual private networks which support dynamic overlay creation for interconnecting all the VMs of the particular tenant deployed in physically separated servers.

2.5 Software-Defined Wireless Networks

2.5.1 Ongoing Research

Over recent years, software-defined wireless network (SDWN) has become an emerging and significant research branch for SDN, and captured increasingly attentions. Orienting towards the characteristics of mobile and wireless network, SDWN aims to study the network architecture and a series of relative crucial technologies for the future mobile and wireless network. Consistent with the core idea of SDN, SDWN separates the control plane and the data plane. The network devices, including radio accessing devices, forwarding devices, and etc. have been simplified and behave according to the rules scheduled by the logically centralized control plane. Possessing the global information dynamically reported by the network devices, the control plane allocates the resources, schedules the devices behaviors, and configures the wireless parameters.

OpenFlow [21] is the prominent SDN protocol initially introduced for separating the control plane of a network from its forwarding using abstraction and to enable advanced traffic management and various access and routing protocols. Nowadays, OpenFlow found it's usage in several other areas such as data centers [30], cloud networking [31], network virtualization [32] and network management [33]. In general, it focuses mainly on wired network for the purpose of packet switching, routing and flow isolation. It supports various switching and routing solutions aligned with the SDN paradigm. However the logically centralized control view and the unprecedented data-plane programmability of SDN offers key benefits to mobile operators by aligning mobile and wireless networks. The summary of various research study on SDWN are listed in in table 2.6

OpenRoad [63] is the first work about SDWN. OpenRoad envisages a world in which users can move freely between any wireless infrastructure by separating the network service from the underlying physical infrastructure. They propose an OpenFlow based open and backward compatible wireless network infrastructure, and deploy it in college campuses [71]. Network devices are controlled by NOX. They also introduce virtualization by FlowVisor [72, 73]. Finally, they verify the handover performance among Wifi and WiMax. However, OpenRoad manly orients towards WiFi and provides no special support for cellular networks. Note that our work is heavily inspired by and follows the vision of OpenRoads.

Odin [64] is a prototype software defined wireless network framework for enterprise WLANs by introduces programmability. Odin builds on a light virtual access point (AP) abstraction, virtualizing association state and separate them from the physical AP. The decision module in Odin is an application on top of the OpenFlow controller. Odin benefits several applications such as seamless mobility, load balancing and hidden terminal mitigation.

The fast-varying and vulnerable mobile and wireless environment makes the PHY and MAC layers technologies in mobile networks much more complicated than in the wired scenarios. In recent years, some studies concentrate on investigating lower layer wireless network technologies

	Network Type	Summary
OpenRoad [63]	Wifi	OpenFlow based open-source platform for moving freely between any wireless infrastructure.
Odin [64]	WLAN	Prototype software-defined wireless network framework for enterprise WLANs by introduces programmability.
OpenRadio [65]	General	A programmable wireless data plane for the continuous evolution of wireless network.
OpenRF [66]	WLAN	A software-defined cross-layer architecture for managing MIMO signal processing with commodity Wi-Fi cards.
SoftRAN [16]	LTE	A software-defined centralized control plane of the radio access network by introducing a virtual big-base station.
OpenRAN [67]	Heterogeneous	A cloud computing based software-defined radio access network architecture for heterogeneous wireless networks.
CellSDN [68]	LTE	Abstract the control functions from both accessing and forwarding devices to meet the needs for fast and frequent updates.
SoftCell [69]	LTE	Software-defined cellular core network by aggregating traffic from multi-dimensions and simplifying the P-GW.
MobileFlow [70]	LTE	Carrier-grade flow-based forwarding and a rich environment for innovation at the core of the mobile network.

Table 2.6 – Summary of Research Study on SDWN.

in the SDWN. OpenRadio [65] approaches the wireless SDN through a software abstraction layer that enables wireless protocol programmability at both physical and MAC layers. OpenRadio decomposes any specific wireless protocols into processing plane and decision plane components with these APIs. Concretely, the decision plane schedule the "rules" by encapsulating all the decision logic functionality. The processing plane encapsulates a group of signal processing algorithms used in a PHY processing chain such as FFT, 64-QAM, convolutional encoding, Viterbi decoding, and etc. Therefore, OpenRadio implements the PHY and MAC layers of different wireless protocol stacks such as WiFi, LTE and WiMax by flexibly programming.

OpenRF [66] proposes a software-defined cross-layer architecture for managing MIMO signal processing in today's networks with commodity Wi-Fi cards. OpenRF adopts the SDN idea and enables access points to control MIMO signal processing at physical layer, such as interference nulling, coherent beamforming and interference alignment. After converting high-level quality of service requirements of downlink flows to low-level physical-layer techniques, OpenRF controller makes cross-layer decision to guarantee the transmission rate and control interference across access points.

Software-defined radio access network is also an attractive direction of SDWN since radio access network (RAN) provides ubiquitous wireless connectivity to mobile end users and integrates the mobile and wireless features. SoftRAN [16] is a software-defined centralized control plane of the radio access network by introducing a virtual big-base station. The virtual big-base station consists of a central controller and radio elements (individual physical base stations). To achieve the tradeoff between the optimal centralized control and the sensitive inherent delay, SoftRAN redesigns

the control plane functionalities cooperatively between the the controller and the radio elements. Specifically, the centralized controller handles the cross radio elements decision, while the individual radio elements deal with the frequently varying parameters. SoftRAN mostly focuses on the one-tier situation, consisting of microcells, which is not suitable for the scenarios of heterogeneous networks.

OpenRAN [67] proposes a software-defined radio access network from another perspective by introducing cloud computing inspired from Cloud RAN [17]. OpenRAN consists of wireless spectrum resource pool, cloud computing resource pool and a SDN controller. The wireless spectrum resource pool covers multiple heterogeneous wireless networks. The cloud computing resource pool implements the baseband process-ing of these heterogeneous networks. According to the dynamic network requirements, SDN controller establishes the virtual base station in the wireless spectrum resource pool, and corresponding virtual baseband processing unit in the cloud computing pool by installing appropriate PHY and MAC layer protocols.

As core network in mobile and wireless centralizes almost all data-plane functionalities and traffic, CellSDN [68] and SoftCell [69] aim at studying the software-defined core network in the LTE network. CellSDN covers both access and core network, and deploys a network operating system to abstract the control functions from both accessing and forwarding devices. To meet the demand for fast and frequent updates, it introduces a local agent to make the realtime decision. SoftCell is successive research of CellSDN, and mainly focuses on the challenges of core network. SoftCell aggregates traffic from three dimensions-the service policy, the base station location, and the UEs. SoftCell simplifies the packet gateway (P-GW) through two ways: introducing the software-defined access switches to implement the packet classification, and configuring optimal forwarding paths across various specific middleboxes.

MobileFlow [70] studies the carrier network and proposes blueprint for flow-based forwarding and a rich environment for innovation at the core of the mobile. The EU FP7 "Connectivity management for eneRgy Optimized Wireless Dense networks" (CROWD) project identified that the SDN approach can be a potential solution for MAC layer reconfiguration, dynamic backhaul reconfiguration, and connectivity management in mobile networks [74].

2.5.2 Challenges

From the ongoing research on SDWN, It is clear that SDN will have a significantly impact on wireless networks. However, SDN still needs further research attention to integrate it in wireless products and solutions. In the following, open issues covering the SDN life cycle from standardization, implementation, to deployment with regard to SDWN are listed.

- *Standardization:* Although OpenFlow specification evolves rapidly, it so far supports only fixed or wired networks. There is no definitive standard protocols for SDWN. SDN is not yet integrated into RAN equipments. OpenFlow is not the only SDN protocol or implementation to cover the concept of SDN. The ETSI Industry Specification Group (ISG) for Network Functions Virtualization (NFV) has been formed recently to promote NFV, which is highly complementary to SDWN [75]. A comprehensive comparison among all potential SDN implementation should be conducted in order to define and standardize SDN protocol for wireless systems.
- *Implementations:* SDN controllers in general requires extension as well as addition of features and protocols to support the functionalities of wireless networks e.g. X2AP [7] is the typical interface

standardized for exchanging control messages between eNodeBs. All of the SDN controllers [2, 4, 46, 76, 45, 77, 47, 78] so far designed/developed to bring programmability and flexibility only for transport networks. Also the latency in communication between controllers and the network devices are currently suitable only for transport networks that has higher latency budget for control decisions compare to wireless networks that has stringent latency budget e.g. 5G use cases such as Autonomous Car Robotics, Video Streaming, Immersive Experience require round trip time (RTT) ranging between 100s usec to 100 msec [79]. Some other design drawbacks in the current SDN implementation do exist. For example, in OpenFlow, a long header is used for matching to allow more granular control, i.e. twelve tuples in OpenFlow 1.0 and more in other versions. This requires more space in rule storage and more time in lookup. This leads to negative effect on performance. This requires definition/implementation of new protocols as well as optimization of existing protocols for SDWN e.g. OpenFlow to support those limitations of wireless networks.

- *Deployment:* Although the concept of SDN is built around the idea of control and data plane separation, it is not completely feasible in wireless networks due to it's stringent latency budgets for transmission/reception. For example, in LTE the typical transmission time interval (TTI) is 1 ms [80], it means the scheduling decision has to be made within 1 ms for a user. As a consequence of these limitations, the SDWN requires partial decoupling of control plane i.e. further study and analysis need to be conducted in order to identify the list of functions and processes that can be centralized without affecting the required QoS. Issues related to SDN controller reliability and security need further investigation and solutions for carrier-grade SDN deployments [81].

Chapter 3

Software-Defined Wireless Networks

This chapter outlines the main objectives of the thesis. It presents also the proposed software-defined architectures corresponding to each objective and their use cases.

3.1 General Architecture and Research Objectives

The software-defined networking is an important aspect of next generation wireless network architecture. The SDN allows for a fast reconfiguration to dynamically influence and optimize the performance of mobile networks. The SDN in general brings set of advantages to wireless networks as summarized below:

- **Flexibility:** A current problem for mobile network operators is the high amount of capital and operational expenditures (CAPEX and OPEX) of their networks independent of the actual traffic load and service usage, and thus the earning for products they sell to customers. By means of SDN approach, operators would be able to fit the network to their needs by simply re-programming the controller and thus reducing costs.
- **Programmability:** It allows third parties to acquire network resources on-demand satisfying their individual SLAs. In addition, programmability can enhance the user perceived QoE by customizing the network resources accordingly.
- **Unified management:** Adopting a logically centralized control unifies heterogeneous network platforms and provides a simplified operation of the wireless network: With SDN, network operators only need to control a set of central entities (namely, the controllers) that control the entire network, which possibly includes heterogeneous radio technologies.
- **Enabling new services:** By modifying the behavior of applications that run on top of the SDN controller (northbound interface), many new services that were not included in the initial architecture design can be enabled by modifying the network behavior and adapting its capabilities for the introduction of new services within few hours instead of weeks.

As shown in the Fig. 3.1, the global research focus of this thesis falls under two main use cases of next generation mobile networks i.e. **Teleco** and **Vertical**. We follow the 5G mobile network architecture that consists of **Front End Unit**, **Edge Cloud** and **Central Cloud** [82]. In the telco use cases, we exploit the advantages of SDN to have flexible control framework for Self-Organizing Network (SON) and dynamic user processing split. In vertical use case, we apply various advantages of SDN and OpenFlow protocol to efficiently utilize the scare radio resources of wireless backhaul network in the train-to-ground communication systems. SDN controller in

our global architecture consist of three set of functionalities supporting SON, control for wireless backhaul and control for dynamic user processing split. Orchestrator is placed at the top of SDN controller to represents the mobile network orchestration framework for the deployment of virtual network functions (VNFs).

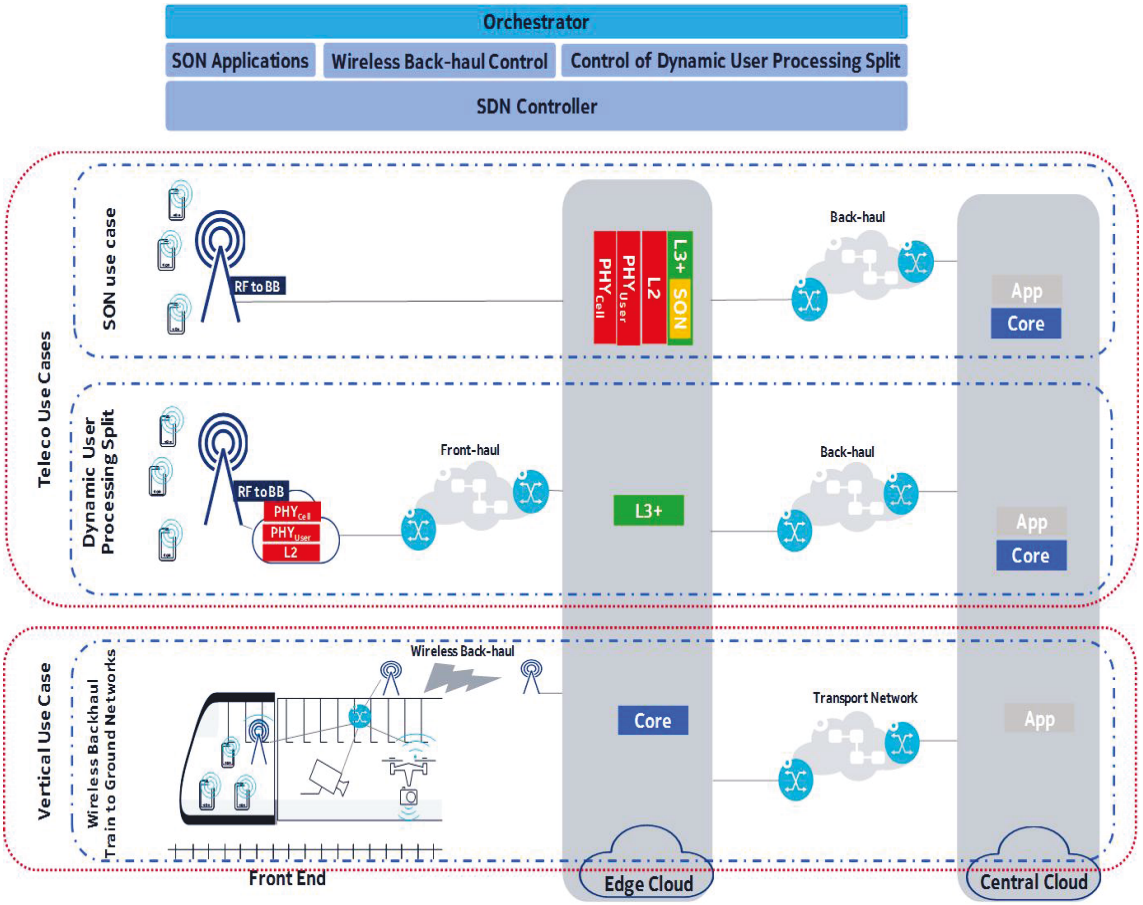


Figure 3.1 – Software-Defined Wireless Network: Global View.

- **SDN for SON:** The introduction of HetNet in 3GPP LTE/LTE-A [7] is the well known solution for increasing network coverage and capacity through higher spectrum spatial reuse by deploying low power small cells (SC) mixed with high power macro cells (MC). It is expected that for achieving the performance needs of future applications, 5G will be deployed as heterogeneous network, consisting of multiple radio access technologies including LTE-A, Wi-Fi and 5G schemes. In addition, there will be a combination of macro, small and super small cells in 5G [83]. The increase of network and traffic density will lead to an inter-cell interference between adjacent cells and as a result, there will be significant degradation in user throughput and service quality. Overlapping adjacent cells need to coordinate their resource allocations (time/frequency/spatial dimension) to reduce harmful interference to users in overlapping area. This would generally bring additional management and operational complexity due to system heterogeneity and lack of flexible interoperability among network devices. 3GPP SON introduced in LTE is known for simplifying and reducing network operations and also optimizing radio access network (RAN) through automated mechanisms (e.g. self-configuration, self-optimization, self-healing) [15]. According to 3GPP standard there is a X2AP interface defined between base stations for exchanging informations to have coordination between them. SON supports low frequency mech-

anisms (for instance self-configuration at power up, per day self-optimization of antenna tilt) and also high frequency mechanisms (for instance distributed interference coordination optimization via X2 interface). This distributed coordination mechanism do not scale well as they need to work with larger number of base stations, especially in terms of latency. This leads to poor performance, reducing capacity significantly due to the in-ability to manage interference and balance load. The SON techniques requires dedicated controllers deployed by operators called autonomic servers to realize centralized coordination for implementing SON techniques. SDN can be a flexible mechanism for adapting SON solutions. The logically centralized controller enables radio resource allocation decisions to be made with global visibility across many base stations, which is far more optimal than the distributed management. By centralizing network intelligence and computation resources, resource allocation decisions can be adjusted based on the dynamic power and sub-carrier allocation profile of each base station. In addition, scalability is improved because as new users are added, the required computing capacity at each base station remains low because these processing is centralized in the Sofware-defined RAN controller. In this research, we design, implement and validate the SDN controller framework that supports the deployment of SON solutions and also to bring programmability in the radio access network.

- **SDN for Wireless Backhaul:** The mobile Xhaul (backhaul and fronthaul) network [84] is an essential milestone towards realizing a 5G network. Backhaul networks in 5G require more capacity, less latency, synchronization, security, and resilience. Current backhaul networks are mostly built with microwave links (often operator owned) and fibre/copper based links (often leased) with different proportions per operator and country [85]. 5G is partly an evolution of existing technologies but is also based on disruptive technologies affecting all parts of the network, nonetheless the backhaul, and revolutionizing the traditional approaches to network design. As a result, the 5G back-haul challenges are manifold: >10 Gbps capacity, <1 msec end-to-end latency, high security and resilience, time and frequency synchronization, low energy consumption and low cost [86]. None of the current backhaul solutions can deliver all of the above as a stand-alone solution; perhaps fibre optic based backhauls rank the best in all aspects, except cost.

  Intelligent Transportation Systems (ITS) uses both wired and wireless backhaul networks to handle communication between various sub systems. Train-to-Ground communication network is one of the important use case of ITS. Although there are several technologies and solutions to provide wireless backhaul in train-to-ground communication systems, the rail industry still faces several issues such as packer re-ordering due to frequent handover, difficulty in providing guaranteed services for QoS constraint services,limitation to wireless infrastructure due to tunnels, etc. The evolution of SDN that provides global network visibility, would complement the backhaul architecture, thus, allowing fully coordinated backhaul operation. Due to the advantages such as centralized network abstraction, fine control of en-to-end network resources, dynamic re-reconfigurability of devices, SDN based wireless backhaul networks can be a suitable solution for above challenges. In this research, we design, implement and validate the software defined wireless backhaul framework that can addresses various challenges in the current as well as in the future wireless backhaul networks.
- **SDN for Dynamic User Processing Split:** Cloud Radio Access Networks (C-RAN) is the evolving paradigm in next generation mobile networks where base stations are migrated and placed in a central location called BBU pool leaving only the antenna unit in the cell site connected via optical or high speed ethernet based transport network. The evolution of this architecture leads the mobile network to the cloud paradigm. The motive behind the centralization of RAN is to have benefits such as resource allocation on demand, load balancing between base stations,

resource sharing and feasibility in implementing joint signal process techniques for better user experience. Although the centralization of RAN brings several advantages, there is an inherent latency in such techniques. This leads to the proposition of partially centralized architecture i.e. Partial C-RAN [17], that enables the operators to deploy RAN functions in the distributed cloud environment such as Front End Unit, Edge Cloud and Central Cloud depends on the use cases e.g. RAN for service corresponding to augmented reality, autonomous vehicles, and robotics requires RTT in micro seconds.

In general, the concept of virtualization in cloud is characterized by the ability to either aggregate or share distributed and multiple physical resources to achieve required level of performance. This requires the implementation of control framework that brings such dynamicity in RAN deployments. SDN [16] is considered to be a suitable approach for manage virtualization especially to generate global view of available resources for management, load balancing and networking between distributed processing units. So SDN approach can benefit C-RAN for implementing dynamic resource allocation process and traffic load balancing between different BBUs. In this research we proposes a control framework for C-RAN based on SDN architecture. The main focus of this framework is to provide unified management for dynamic user processing split in C-RAN. This framework also supports various features of C-RAN such as flexible and dynamic resource allocation, resource sharing, implementation of joint signal processing techniques.

3.2 SDN for Self Organizing Networks

This section presents our proposed software-defined RAN framework for SON management, and it's advantages and use cases.

3.2.1 Self Organizing Networks

Like every networking systems, Long Term Evolution (LTE) based mobile network also need to be optimized and managed for maximizing the utility without degradation in the quality. In Universal Mobile Telecommunications System (UMTS) or 3rd Generation mobile systems, Radio Network Subsystem (RNS) is the management entity standardized to manage the overall air interface. In the case of LTE, due to the possibility of heterogeneity i.e. the deployment of small cells adjacent to macro cells for capacity improvement, brings extra complexity in management. SON in 3GPP standardization is considered to be the promising approach for the optimization and management of LTE networks. SON framework aims to configure and optimize the RAN in LTE automatically, so that any human interaction can be reduced and the capacity and efficiency of the network can be improved. The main functionalities of SON include: self-configuration, self-optimization and self-healing [87].

- **Self-configuration** is defined as the process where newly deployed nodes (eNBs) are configured by automatic installation procedures to get the necessary basic configuration for system operation. Self-configuration process works in pre-operational state, which starts from when the eNB is powered up and has backbone connectivity until the RF transmitter is switched on. As shown in Fig. 3.2, self-configuration includes two stages: basic setup and initial radio configuration.
- **Self-optimization** is a process where UE and eNB measurements and performance measurements are used to auto-tune the network. This process works in operational state, which starts when the RF interface is switched on. The self-optimization process collects measurement infor-

mation from UE and eNB and then with the help of external optimization tool, it auto-tunes the configuration data to optimize the network. A typical example is the neighbor list optimization.

- **Self-healing** function aims at automatic detection and localization of most of the failures and applies self-healing mechanisms to solve several failure classes, such as reducing the output power in case of temperature failure or automatic fall-back to previous software version.

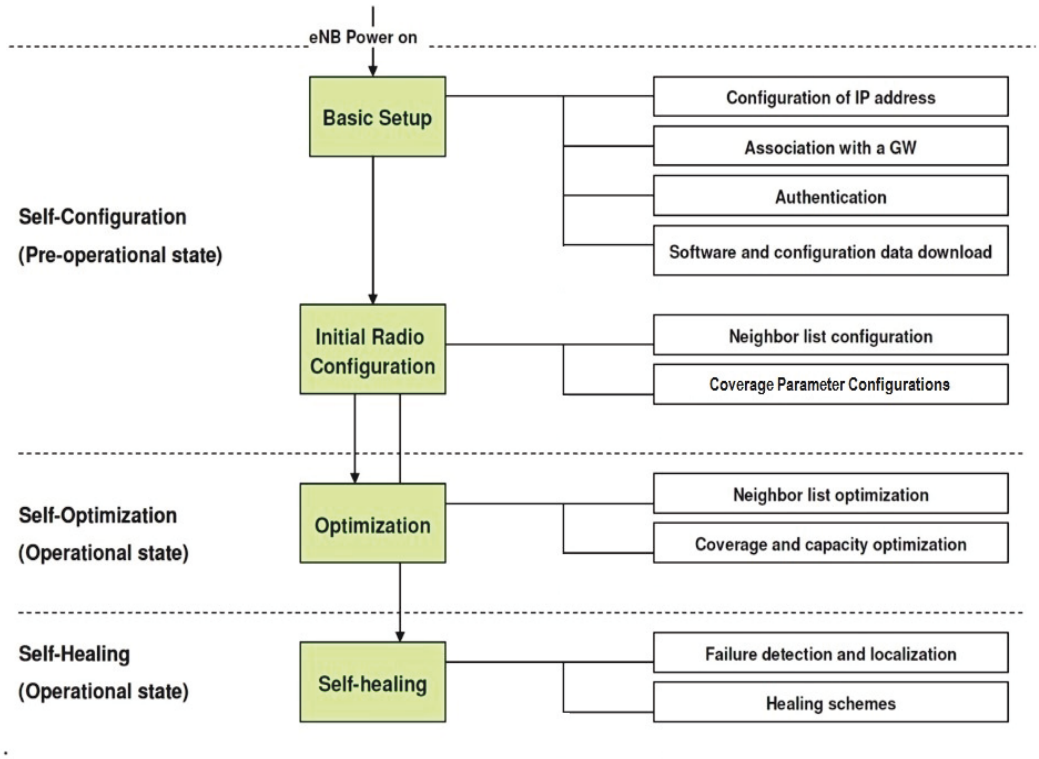


Figure 3.2 – General Framework of Self Organizing Networks.

Self Organizing Network Architecture

A self-configuration Subsystem is created in Operation and Management (OAM) system to be responsible for the self-configuration of eNBs. For self-optimization functions, they can be located in OAM or eNB or both of them. So, according to the location of optimization algorithms, SON can be divided into three classes: Centralized SON, Distributed SON and Hybrid SON [5].

- **Centralized SON:** In Centralized SON, optimization algorithms are executed in the OAM System. In such solutions, SON functionalities reside in a small number of locations, at a high level in the architecture. Fig. 3.3 shows an example of Centralized SON. In Centralized SON, all SON functions are located in OAM systems, so that it is easy to deploy them. However, since different vendors have their own OAM systems, there is low support for optimization cases among different vendors. It also does not support those simple and quick optimization cases. To implement the Centralized SON, existing Itf-N interfaces need to be extended.

- **Distributed SON:** In Distributed SON, optimization algorithms are executed in eNBs. In such solutions, SON functionalities reside in many locations at a relatively low level in the architecture. Fig. 3.4 shows an example of Distributed SON. In Distributed SON, all SON functions are located in eNBs, so that it causes a lot of deployment work. It is also difficult to support complex

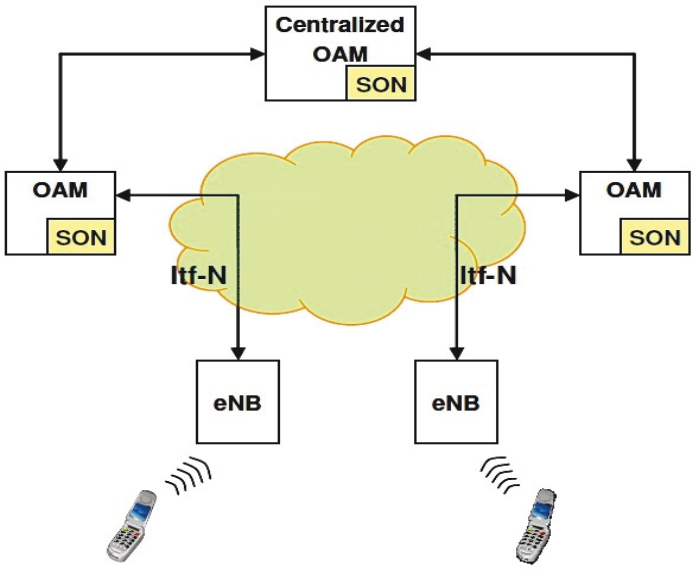


Figure 3.3 – Architecture of Centralized SON [5].

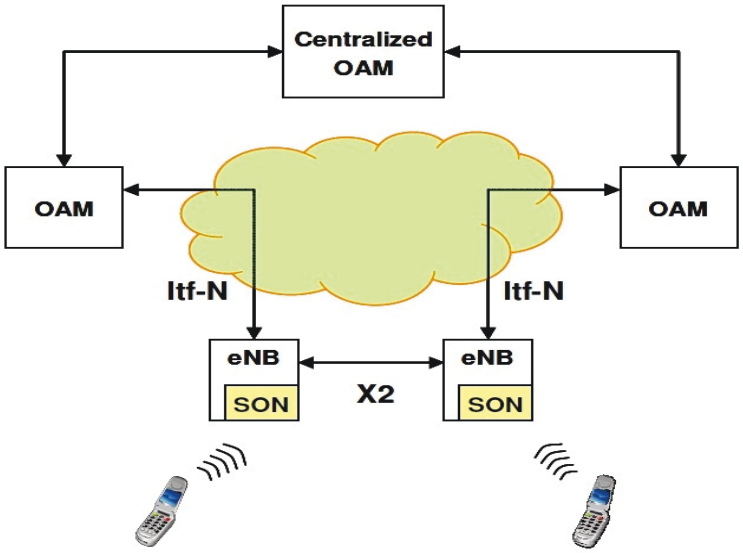


Figure 3.4 – Architecture of Distributed SON [5].

optimization schemes, which require the coordination of lots of eNBs. However, in Distributed SON, it is easy to support those cases, which only concern one or two eNBs and require quick optimization responses. For Distributed SON, the X2 interface needs to be extended.

- **Hybrid SON:** In Hybrid SON, a part of optimization algorithms are executed in the OAM system, while others are executed in eNBs. Fig. 3.5 shows an example of Hybrid SON. In Hybrid SON, simple and quick optimization schemes are implemented in eNBs and complex optimization schemes are implemented in OAM. As a result, it is very flexible to support different kinds of optimization cases. It also supports the optimization between different vendors through the X2 interface. However, it costs lots of deployment effort and interface extension work.

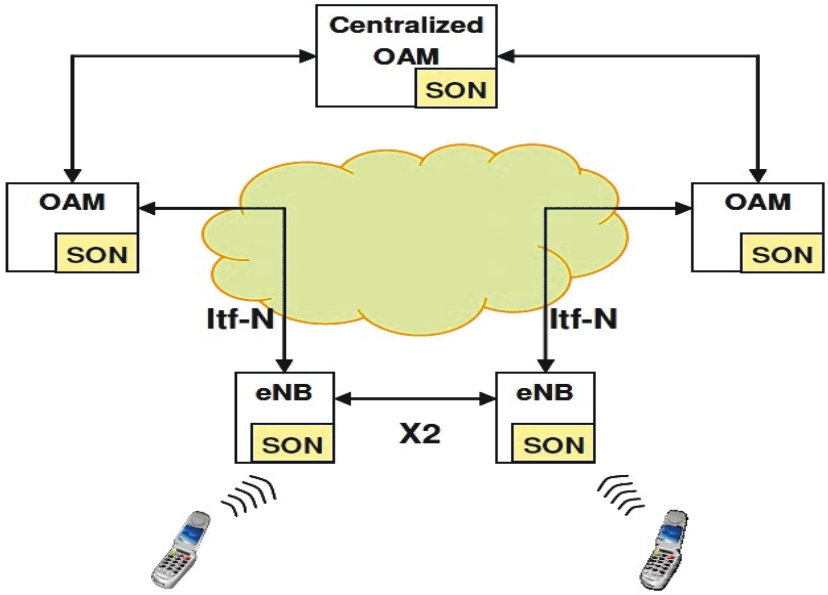


Figure 3.5 – Architecture of Hybrid SON [5].

SON Use Cases

Although there are several use case of SON listed in 3GPP LTE-Advanced specification, our research focus is mainly on use cases related to interference management and performance improvement (e.g. Load Balancing, Inter Cell Interference Coordination (ICIC),and Coordinated Multi Point (CoMP) based ICIC.).

- **Load Balancing**

  Load Balancing refers to the process whereby similar network elements that are intended to share traffic, share the load. The similar network elements can be anything from packet gateways to MMEs to base stations and sectors. In LTE, MME pools are expected to share user traffic load across different MMEs as load increases, while eNBs may have RRM functions that share/offload traffic to neighboring cells in order to increase the system capacity. As a result, different real-time algorithms at different nodes can simultaneously provide Load Balancing of user traffic per network element as required. Additionally, long term traffic behavior of each node can be monitored so that traffic may be "directed" a priori by a centralized entity in the network. For instance, this could be a desirable feature for markets where periodic or scheduled concentrations of users regularly occur (e.g. sporting events, conventions, daily commutes, etc.) [88, 89].

  The decision to re-balance a cell or move a particular user must take in to consideration the target for the user(s). It is not desirable to send a user to an alternate location (i.e. neighbor or co-located frequency) if that user will then have a reduced QoS or lower performance than remaining in the source, or if the re-balance will result in reduced system capacity/utilization. The objective of Mobility Load Balancing (MLB) is to intelligently spread user traffic across the system's radio resources as necessary in order to provide quality end-user experience and performance, while simultaneously optimizing system capacity. Additionally, MLB may be desirable to shape the system load according to operator policy, or to "offload" users from one cell or carrier in order to achieve energy savings. The automating of this minimizes human intervention in the network management and optimization tasks.

- **Inter Cell Interference coordination (ICIC)**

  The frequency usage in cellular networks results in interference between cells. Given the orthogonal nature of intra-cell transmissions, the source of interference in LTE is inter-cell interference. Within the OFDM and SC-FDMA based LTE system, interference has to be coordinated on the basis of the physical resource blocks (PRBs). The ability to schedule users over variable portions of the carrier bandwidth allows for inter-cell interference coordination techniques to be utilized which can shape the interference in frequency. Inter-Cell Interference Coordination (ICIC) involves the intelligent co-ordination of physical resources between various neighboring cells to reduce interference from one cell to another. Each cell gives up use of some resource in a coordinated fashion to improve performance especially for cell edge users which are impacted the most by inter-cell interference. ICIC is accomplished as follows: i) Users can be scheduled in units of a physical resource block (PRB) which is 180 kHz, ii) Neighbouring cells can coordinate which portions of the bandwidth are used in each cell and the transmission powers across various frequency resource blocks, and iii) Inter-cell Self Organizing Networksinterference can be reduced or avoided in uplink and downlink by a coordinated usage of the available resources (PRBs) in the related cells which leads to improved SIR and corresponding throughput. This coordination is realized through the restriction and the preference of the resource usage in the different cells. This can be achieved by means of ICIC related RRM mechanisms employing signalling of the following standards defined metrics such as High interference indicator, Overload indicator, Relative Narrowband Transmit Power (RNTP) [90, 91, 92].

  Exchanging the above information would allow an advantageous RRM resource usage by means of preferences with respect to the available time/frequency resources, neighbourhood relations of the cells, and QoS requirements targeted by the operator, etc. Expected results of automated ICIC parameter settings include the automatic configuration or adaptation, with respect to cell topology, of: i) ICIC reporting thresholds/periods, ii) Resource preferences in eNBs and iii) the RSRP threshold for ICIC. ICIC RRM might be configured by ICIC related configuration parameters like reporting thresholds/periods and preferred/prioritized resources. Then, these have to be set by the operator for each cell. Setting and updating these parameters automatically is the task of a SON mechanism. The objective of SON is the self-configuration and self-optimization of control parameters of RRM ICIC schemes for uplink (UL) and downlink (DL) ICIC. SON based ICIC requires the exchange of messages between the eNBs of various cells, via the X2 interface, for interference coordination. By means of ICIC related Performance Measurements (PM) analysis, the SON function may properly tune ICIC configuration parameters like reporting thresholds/periods and resource preference configuration settings in order to make the ICIC schemes effective with respect to Operator's requirements [92].

  Shaping the inter-cell interference allows an improved SINR to be realized in certain portions of the spectrum. This can be exploited by the scheduler to provide an improved trade-off between sector throughput and cell-edge bit rate. The result is an improved SINR for cell edge UEs that achieve better edge rates and cell throughputs, improved hand-off performance for cell edge UEs (lower hand-off delays and lower handover failure rate), etc. The goal of a SON based ICIC is to have minimized human intervention in network management and optimization tasks.

- **Coordinated Multi Point (CoMP) based ICIC**

  LTE contains several coordinated multipoint transmission techniques (CoMP) to perform a dynamic coordination of transmissions and receptions over a variety of different base stations. Its aim is to enhance the overall system performance, effectively utilize the resources, and improve

end user service quality (e.g. cell-edge user throughput). However, CoMP requires close coordination between geographically separated eNBs. The main techniques of CoMP [93] are coordinated scheduling or beamforming (CS/CB), dynamic point selection (DPS), and joint transmission (JT). In first two cases UE's data would be available only at serving cell but coordination is necessary between all of the interfering cells. Whereas in later two cases, UEs data might be available at more than one cells; by default at all coordinating cells, but transmission may be occurred at one or more cells. In addition, depending on coordination architecture CoMP are also divided into two main schemes, namely Central CoMP and Distributed CoMP. There Several techniques associated with CoMP technology [94, 95], but all the approaches require to share some scheduling information regarding the UEs data at coordinating cells. This increases complexity and signaling overhead on X2 interface.

3.2.2 Architecture

The main objective of our architecture is to bring programmability and flexibility to the RAN using SDN. The major benefit of such approach is to decouple network intelligence to the logically centralized controller(s) so that the functionality of RAN can be better optimized or reprogrammed. We define our architecture with reference to ODL controller. It is illustrated in Fig. 3.6. Since the main focus of the architecture is to provide vendor-agnostic North Bound (NB) APIs for supporting SON algorithms and optimization, the controller has to provide abstraction of network Key Performance Indicators (KPI) and configuration parameters. Here, RAN applications would use the controller's NB APIs to:

- discover existing network elements and topology.
- retrieve RAN measurements and configuration parameters.
- re-configure RAN using optimal parameters.

Radio Net Flow (RNF) is the South Bound (SB) protocol introduced in the controller as a RAN configuration protocol for (i) collecting radio network measurements and configurations and (ii) re-configuring parameters of those radio elements. We implemented RNF using SCTP because it is the standard transport layer protocol in the mobile network architecture which facilitates the integration of our SDN controller with standard RAN [96]. We developed the corresponding protocol agent called Radio Net Flow Agent (RNFA). RNFA can be integrated into any eNodeB (i.e. 3GPP-LTE base station) as a communication interface to the SDN controller. We introduced three service modules in the Service Abstraction Layer (SAL) of ODL: the RAN Configuration Manager, the RAN Statistics Manager, and the RAN Topology Manager. All these service modules interface with the RNF SB protocol using specific libraries. Through the RNF, they can collect data from the RAN and also send configurations to the RAN.

Since the SAL imposes transient data storage, we introduced the RAN Inventory that is an external database to maintain long term network state history. We used a No-SQL database [97] for performance considerations. During operations, RAN service modules update the RAN Inventory. On the other hand, NB applications access the RAN Inventory for information retrieval and settings via NB REST (HTTP RESTful compliant) APIs.

The content of RAN Inventory for a SON use case i.e. eICIC optimization is illustrated by Fig. 3.7. It consists of the following information for th NB application to deploy an eICIC optimization:

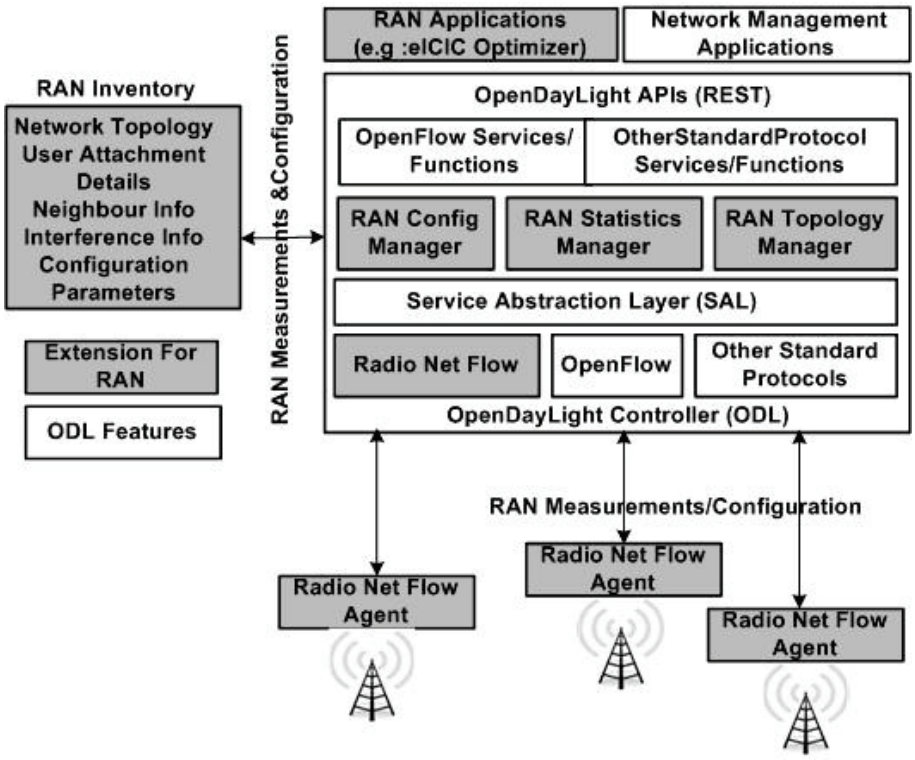


Figure 3.6 – Software Defined Radio Access Network Architecture.

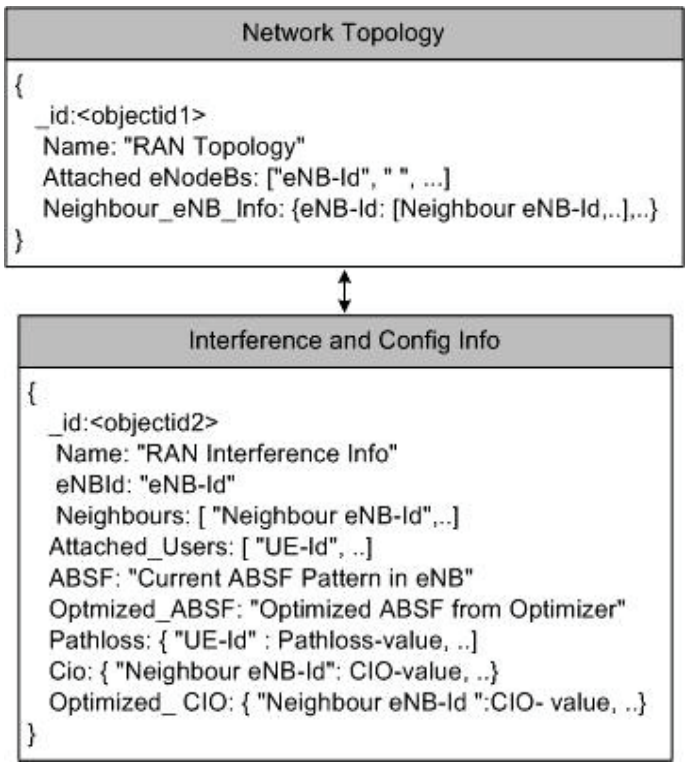


Figure 3.7 – RAN Inventory in MongoDB for eICIC network optimization.

- Network Topology: A complete record of network topology contains all the macro and small cells connected to the controller, together with their unique IDs and neighboring cell information.

The RAN Topology Manager service in the controller generates such a graph in the logical table format in the RAN Inventory. The record remains static until there is a change in the network topology (addition/removal/failure of a base station). The RAN Topology Manager updates the table once the network topology changes.

- User Attachment and Interference Information: the RAN Statistics Manager is responsible for populating this information in the RAN Inventory. User Attachment information is updated via the RNF protocol once a user performs handover or enters/exits the network. Interference information refers to the interference situation experienced by each user: it is in terms of the user ID, its base station ID, interfering base station ID, and the level of the interference. Each base station sends the information to the controller periodically.
- Configuration Parameters: The RAN Configuration Manager handles configuration parameters in the RAN Inventory. NB applications store optimized parameters for each base station in the RAN Inventory. The RAN Configuration Manager sends these parameters periodically to the base stations.

Use Case	Description
eICIC/ICIC (More generally dynamic/fast SON)	Different optimization algorithms can be plugged as NB applications in the SDN controller which has global network view of RAN.
CoMP (e.g. Coordinated scheduling, Dynamic point selection)	Coordinated scheduling and dynamic point selection can be directly implemented as NB applications where they get the required input parameters from the RAN Inventory via REST APIs.
Load balancing (Between base stations)	SDN can trigger hand over for cell edge users to be better served by lightly loaded neighboring base stations.

Table 3.1 – SDN platform for SON use cases.

3.2.3 Use Cases

As shown in Table I, there are three potential use cases of SDN for SON: ICIC/eICIC, CoMP and Load Balancing. The following describes in details three use cases of our SDN platform and the design for mobile network inter-cell interference management.

SDN for eICIC

Enhanced inter-cell interference coordination (eICIC) introduced in 3GPP-LTE is to conduct a time repartition of the physical resource blocks (PRBs) such that during some time slots called Almost Blanked Subframe (ABS), macro cells mute and do not transmit any traffic data. This technique allows to mitigate strong inter-cell interference that may severely degrade cell-edge user performance. In addition to interference management, a user-cell association optimization is also very important. In HetNets, due to the large difference in the transmit power between macro and small cells, user-cell association should be conducted in a more intelligent manner but not simply be decided due to the strongest received signal. In order to favor user association to small cell against high-power macro cell, LTE applies Cell Individual Offset (CIO) during the comparison of received signal strengths from neighboring cells for the user-cell association decision. This can also be supported by our SDN.

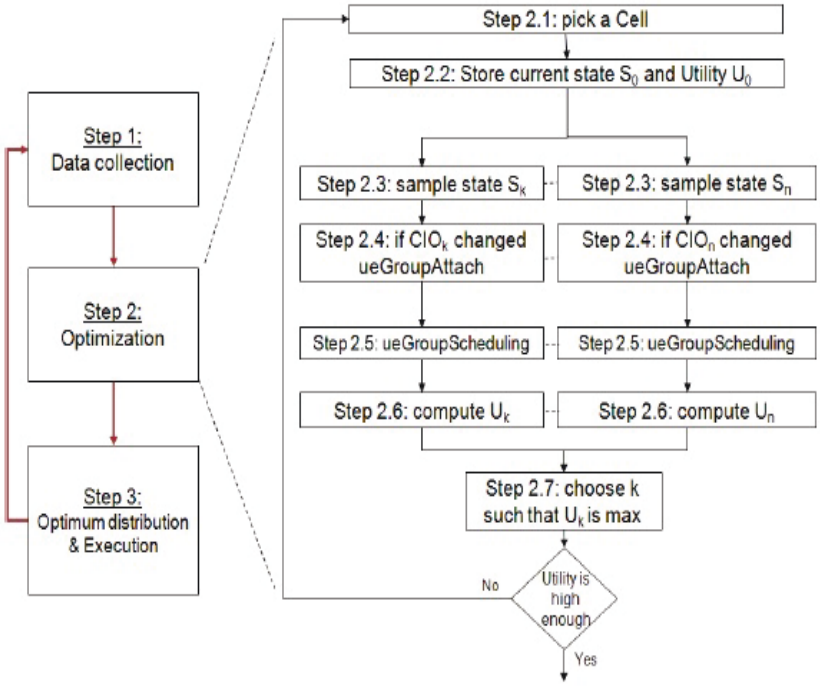


Figure 3.8 – eICIC Optimization Algorithm.

Our eICIC optimizer is based on a two-tier model composed of macro and small cells that perform dynamic user-cell association optimization and transmission scheduling which aims to determine optimal CIO and ABS settings for each base station. Fig. 3.8 shows our best-response potential game approach where the CIO optimization with respect to a predefined network utility is discussed. Let $\{S_k\}$ be the states referring to the possible choices of CIO settings and $\{U_k\}$ be the utility values indicating the network performance. During the optimization iteration for each base station, the CIO is chosen to maximize $\{U_k\}$ which involves base station k and its neighboring cells. To facilitate the above optimization, each UE reports to its serving base station (eNB) their experienced channel quality, signal level and interference power. These values are sent via the SDN controller to the logically centralized eICIC optimizer (see Fig. 4.1). The eICIC optimizer runs optimization iterations during which various eNBs are taken into account for determining their configurations for the whole network. Notice that the iterations consist in the successive selection of various eNBs. Once an eNB has been selected, a sampling (testing various options of CIO and ABS settings) is performed and the best one is then selected (best response in potential game). Through iterations, the global utility of the network improves gradually and converges to an optimal setup. Note that here different optimization methods (e.g. heuristic or strict global optimization) can be plugged as applications to the SDN controller, which can select for various criteria, functions and also performance trade-off.

It is worth noting that this two-tier model permits the separation of control decisions between the SDN controller and the local operation of radio element (the eNB), so as to facilitate flexible and dynamic network management and optimization. The parameters that would influence the neighboring eNBs need some central coordination in the controller shown in Fig. 4.1. Here, CIO re-configuration can trigger handovers and maintain a load balanced network. Determining ABS

ratio can coordinate the use of physical radio blocks (PRBs) and mitigate interference between cells. Finally, the local scheduler in each eNB allocates its provided resources blocks according to predefined local utility or scheduling policy.

SDN for CoMP

The dynamic point selection technique in CoMP requires a coordination between base stations so as to transmit from multiple cells where the UE can select the best cell in a very fast time scale or even on a subframe basis [93] for optimal performance. This would require a communication and instruction from which cell the network should direct its data to and also transmission synchronization. SDN can be the right choice for DPS such that one can maintain a fast switching and dynamic backhauling between the eNBs and benefit from the diversity gain when choosing the best serving point dynamically for example according to channel conditions.

In our proposed architecture, coordinated scheduling and dynamic point selection can be directly implemented as NB applications where they get the required input parameters such as neighbor cell list (NCL), radio resource availability, backhauling information (data traffic and also CSI/control traffic) and transmission power setting from the RAN Inventory via REST APIs. Since in ODL, the NB applications can use multiple SB protocols for producing the global abstraction of LTE networks, coordinated scheduling and dynamic point selection can easily use our RNF protocol (built on SCTP) for abstracting RAN and use OpenFlow protocol for backhaul network configuration to achieve coordinated multipoint transmission and enhanced QoS.

SDN for Load Balancing

In LTE deployments, handover is initiated only when a UE observes higher received power from neighboring base station compared to its serving base station. So in the distributed RAN architecture, no handover will be made even if the cell-edge users continue to receive poor service. The SDN controller, with complete knowledge of RAN, can easily recognize that the edge users can be better served by being handed over to lightly loaded neighboring base stations. Eventhough the power level, the users receive from neighboring base stations is lower than the served one, the users will be served with more resource blocks and better throughput. This is the typical load balancing use case the RAN can benefit by integrating SDN approach.

3.3 SDN for Wireless Backhaul Networks

This section presents our proposed software-defined wireless backhaul framework for train-to-ground communication system and it's advantages and use cases.

3.3.1 Wireless Backhaul in ITS

Intelligent transportation system

Intelligent Transportation System (ITS) is a framework which allows the information, data processing communication, and sensor technologies to be applied to vehicles, transport infrastructure and users with the goal to improve economic performance, safety, mobility, environmental sustainability, and convenience [98]. ITS has the ability to provide information in real time to the users of all transport modes, especially to those planning, designing, constructing and operating the transport system. Thereby, ITS makes transport easier, more efficient and safer. It also provides

the tools to help users to make intelligent transport choices in the way that they use the system and adapt themselves to improve transport outcomes without the need for user intervention [99].

Figure 3.9 – Intelligent Transport Systems [6].

For the last two decades, ITS have emerged as an efficient way of improving the performance of transportation systems. As an essential element of ITS, High-Speed Railway (HSR) has been developed rapidly as a fast, convenient and green public transportation system and would become the future trend of railway transportation worldwide [100]. For instance, a high-speed rail plan has been outlined in America and the length of HSR lines in China will reach 18,000 km by 2020 [101]. With the continuous construction of HSR in recent years, the issue of train operation safety has attracted more and more attention. The train operation control system plays a key role in train operation safety and is regarded as the nerve center of the HSR system. A standard has been set up for the train operation control system, which is known as European Train Control System (ETCS) [102]. The increasing demand for HSR communications leads to significant attention on the study of Radio Resource Management (RRM), which enables efficient resource utilization and improved system performance. RRM design is a challenging problem due to heterogenous quality-of-service requirements and dynamic characteristics of HSR wireless communications [103, 104].

Wireless Back-hauling in Train-to-Ground systems

The early train-to-ground communication systems such as Private Mobile Radio (PMR) were operated using Terrestrial Trunked Radio (TETRA) to carry mainly voice operations and narrow-band data services. TETRA is operated at Ultra High Frequency (UHF) in the range of 420-470 MHz band. However, due to the limitation in bandwidth and the excessive usage of UHF spectrum, the migration to higher bandwidth digital system has become necessary. Global System for Mobile Communications-Railway (GSM-R), a standard based on GSM (2G+) was introduced by the European Rail Traffic Management System (ERTMS) to carry both signaling information and voice communication. The GSM-R system has many advantages including support for fast handover with train speed up to 500 km/h and it was an important development in the train-to-ground

communication systems [105, 106]. GSM-R has been widely used in HSR communications and can maintain a reliable communication link between the train and the ground. However, GSM-R has some major shortcomings, such as insufficient capacity, low network utilization, and limited support for data services [107].

In the railway industry, Telecommunications are becoming a key aspect not only for deploying services related to its operation and management but also for providing real-time safety and security for passengers on board (on-board CCTV) along with value added services such as multimedia entertainment, passenger information, etc. Also, the growth of Internet and the introduction of smart devices (smart phones, tablets, etc.) necessitates the availability of the seamless Internet connections to users independent of their locations. Currently, there are several solutions based on recent technologies such as WLAN, WiMAX, Satellite Link and LTE to deploy high-speed multi-service train-to-ground communication networks [108]. Among all the solutions, LTE has an immense potential in-terms of high uplink and downlink data rate along with end-to-end QoS for mission critical CBTC traffic and real-time surveillance systems [109].

A broadband wireless communication system for HSR called long-term evolution for railway (LTE-R) has been presented in [110, 111] and determined in the 7th World Congress on High-Speed Rail [112]. Broadband wireless communications can enhance the train operation by allowing an operation center to monitor real-time train-related data information, such as safety information and track diagnostic information [113]. In addition to the train control data transmission, LTE-R is also expected to provide passenger services such as Internet access and a high-quality mobile video broadcasting [102, 114]. With the benefit of it, passengers can treat their journey as a seamless extension of their working or leisure environment.

To improve the capacity for wireless communications on the train, the future HSR communication networks are expected to be heterogeneous with a mixture of different networks and radio access technologies that can be simultaneously accessed by hundreds of users on the train [115, 116]. For instance, the heterogeneous network architecture can be considered as a combination of satellite network, cellular network and wireless data network [113], where the advantage of each access network can be taken into consideration. This architectural enhancement along with the advance communication technologies such as multiple-input multiple-output (MIMO), orthogonal frequency-division multiplexing (OFDM) and radio over fiber (RoF), will provide high aggregate capacity and high spectral efficiency. Nevertheless, the demand for HSR wireless communications is increasingly growing. For example, the estimated wireless communication requirement could be as high as 65 Mbps per train [117]. To further relieve the contradiction between the increasing demand and limited bandwidth of HSR wireless communications, it is necessary to implement radio resource management (RRM) to improve resource utilization efficiency and ensure quality-of-service requirements. However, the traditional RRM methods (e.g. handover, power control and resource allocation) for common cellular communications may not be efficient in HSR wireless communications due to the following reasons, which are closely related to the characteristics of the HSR scenario.

- **High mobility**: The dramatic increase of train speed will cause frequent handover. Given a cell size of 1-2 km, a high-speed train of 350 km/h experiments one handover every 10 to 20 seconds [118]. Solving the frequent handover problem is one of the main functions of RRM in HSR wireles communications. Moreover, the fast relative motion between the ground and the train leads to large Doppler shift and small coherence time. The maximum speed of HSR in China is currently

486 km/h, which induces a Doppler shift of 945 Hz at 2.1 GHz [117]. Thus, when implementing resource allocation for HSR communications, it is necessary to consider the fast-varying channel and inter-carrier interference (ICI) especially for the OFDM technology.

- **Unique channel characteristics**: The moving train encounters diverse scenarios (e.g. cuttings, viaducts and tunnels) with different channel propagation characteristics [102, 119], which causes that a single channel model could not depict features of HSR channels accurately. It brings a big challenge to RRM schemes, which should be adaptive to diverse scenarios along the rail with different channel models. Furthermore, the line-of-sight (LOS) component is much stronger than the multipath components especially in viaduct scenario, which implies that the propagation loss mainly depends on the distance between the base station (BS) and the train [120]. Since the distance varies with the train's position, the power control along the time has a large influence on system transmission performance [121].
- **Heterogeneous QoS requirements**: Many types of services with heterogeneous QoS requirements and priorities will be supported on the train [113]. The QoS performance in HSR wireless communications will be degraded because of high mobility and unique channel characteristics, especially for real-time services and critical-core services that are critical for the train operation [122]. In order to improve system performance and satisfy heterogeneous QoS requirements, it is critical to design effective RRM schemes and resource optimization methods for multiple services transmission in HSR communications.

All these unique characteristics make it challenging to facilitate RRM design for HSR wireless communications. Thus, a new look into the RRM problem in HSR communications is urgently required, where the network architecture and unique characteristics of HSR scenario should be fully taken into consideration.

3.3.2 Architecture

Our architecture is defined using the SDN principle to bring programmability in mobile backhaul networks. The main challenge is to design a controller that provides NB APIs for applications to control/re-configure the backhaul network in real time and to integrate wireless devices to operate under SDN. Our solution consists of different components including a SDN controller, software-defined wireless modems, and OpenFlow-supported traffic Aggregation Switch for connecting to internet gateway and the mobile core network, as shown in Fig. 3.10. There are two kinds of software-defined wireless modems in our architecture: (i) Infra Modem (IM), a fixed modem in the ground connected to the service provider networks, and (ii) User Modem (UM), a mobile modem placed in the train connected to the ground network via IM. UM is attached to different service nodes (WiFi, LTE Femto, CCTV, CBTC, etc.) through LAN in the train. Each IM can simultaneously serve multiple UMs.

Since our framework can provide service-level QoS, we place train control systems (CCTV, CBCT, etc.) and Passenger Information Systems (PIS) in the global Internet rather than as separate entity as in traditional metro train operations. Due to the advantage of high uplink/downlink data rate for mobile broadband services, we back on LTE as a technology for providing mobile phone connectivity in the train. The passengers on board can use LTE femto to have seamless mobile connectivity as well as WiFi hotspots for the internet access. Taking the benefits from QoS provision in OpenFlow [103, 104], our software-defined wireless backhaul can dynamically adopt QoS for different traffic flows. This is necessary for certain services in trains, for example CBTC is

the most time-critical service in the train that needs accurate information about the train location in order to efficiently manage the operations from the control room.

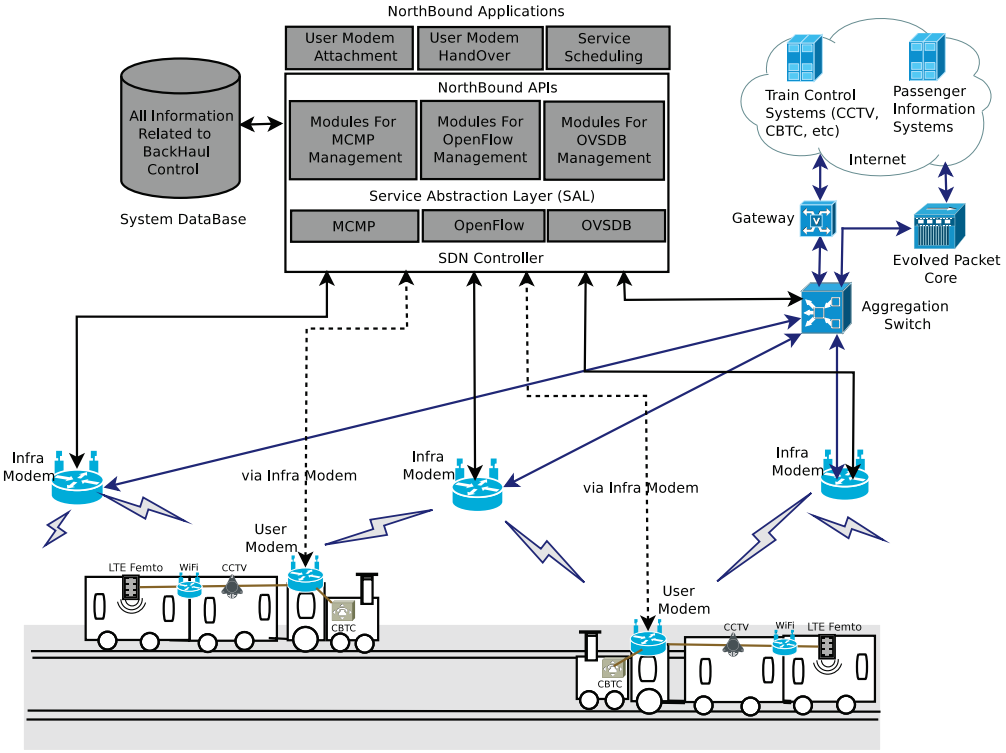


Figure 3.10 – Software-Defined Wireless Backhaul for the Train-to-Ground Networks.

The controller design closely follows our SDN architecture for SON in Fig. 3.6, where we used an external database (DB) together with the ODL controller as a way to store measurements and configuration details to be used by NB applications. Modem Control and Management Protocol (MCMP) is the southbound protocol introduced in addition to standard OpenFlow (OF) and Open vSwitch Database Management Protocol (OVSDB) [104] to handle messages related to mobility management procedures such as User Modem Attachment and User Modem Handover. These messages are originated from the Low Level Radio Control (LLR-C) messages between IM and UM. MCMP is defined using ZeroMQ (ZMQ) [123] due to it's high performance and it's ability to establish many-to-many connection between endpoints. This is necessary because the controller needs to handle request/response from multiple Infra Modems in parallel to avoid latency in control decisions. NB applications use MCMP for sending and receiving messages to/from IM for mobility management. MCMP management modules in the controller interact with system DB to maintain mobility-related information to be used by applications. OF and OVSDB protocols are used by applications to interact with OVS for establishing QoS aware service scheduling in both IM and UM.

Software-Defined Radio Modem

Radio Modems such as IM and UM in our architecture is an embedded Software Defined Radio (SDR) modem designed using pulsar boards [124]. Pulsar has a scalable and programmable hardware architecture that includes variable field-programmable gate arrays (FPGAs) for medium-to-high performance data needs and can integrate 2×2 Radio Frequency (RF) Transceivers to

support bandwidths from 5 to 20 MHz at carrier frequency from 60 MHz to 6 GHz. It has an inbuilt Ethernet and USB3 ports for external connectivity. Radio Modems can support bit rate up to 80 Mbit/s with variable uplink and downlink capacity on demand [125]. Radio Modem is Time Division Duplex (TDD) with variable uplink and downlink frame size with dedicated time slots for broadcasting, initial user connectivity, and synchronization. The modems are configured to operate at carrier frequency of 2.5 GHz with useful bandwidth of 4 MHz and a data rate of 80 Mbit/sec. Radio Modem exposes socket based application programming interface (API) through which external devices can send/receive both user and control data called as Low-Level Radio Data (LLR-D) and Low-Level Radio Control (LLR-C) messages.

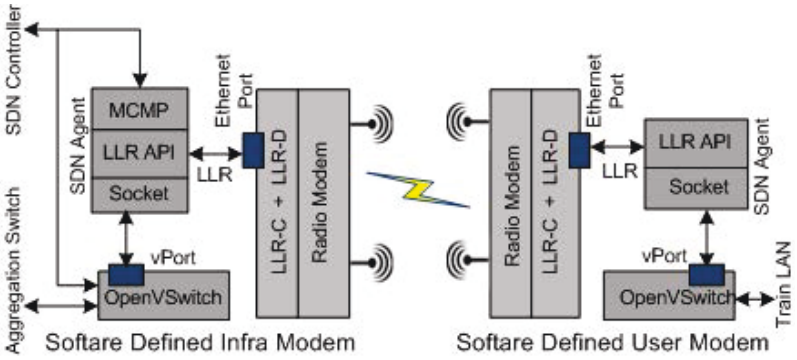


Figure 3.11 – Software-Defined Radio Modem Architecture.

As shown in Fig. 3.11, Radio Modems are integrated in to our SDN framework using SDN-Agent located in the external CPU boards (e.g. Raspberry Pi). A SDN-Agent consists of two sub-functional modules such as a MCMP Agent (MCMP-A) for communication with controller (e.g. for mobility management) and a socket for binding virtual ports (vPorts) of OVS to Radio Modem for sending/receiving LLR messages. OVS in IM dynamically creates and deletes vPorts upon receiving instructions via the OVSDB protocol in SDN controller during UM attachment and handover procedures. OVS in IM allocates one vPort per UM as shown in Fig. 3.12. On the UM side, each OVS has one vPort statically created to distribute data over the connected LAN. OpenFlow control messages between UM and SDN controller are communicated via IM.

OpenFlow for QoS aware Flow Scheduling

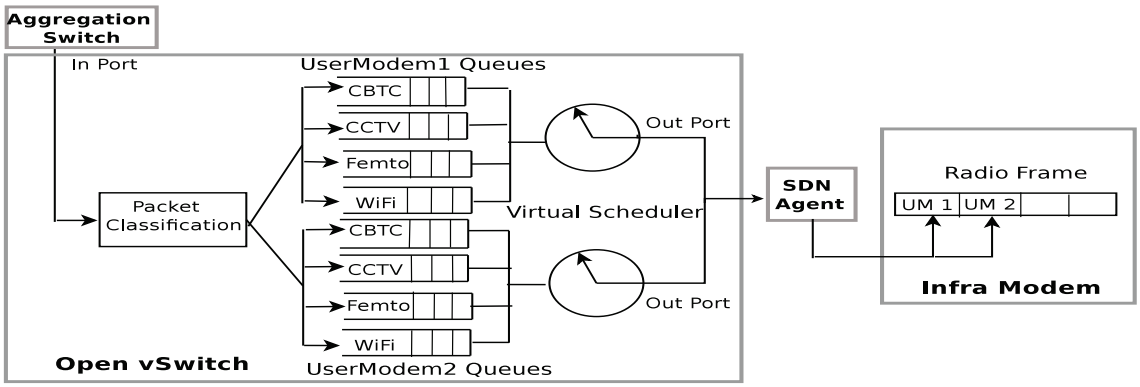


Figure 3.12 – Packet Scheduling using Priority Queuing in OVS.

The main objective behind integrating OVS with Radio Modem is to have QoS-aware flow scheduling using OpenFlow priority queue feature as shown in Fig. 3.12. Hence, traffic flows belonging to different services are scheduled inside the OVS before packets arrive at Radio Modem for sending over the air. The service scheduling application on the top of the SDN controller completely manages packet scheduling inside OVS using OpenFlow and OVSDB plugins. As explained in Fig. 3.12, the purpose of using OpenFlow protocol in our architecture is to enforce networking rules by assigning flows to traffic shaping queues behind each outport (queue/service) inside OVS. OpenFlow starting from version 1.2 supports QoS such as "set-queue" action to forward packets to each queue based on their predefined rules, query queue statistics from the controller for NB applications to take scheduling decisions dynamically, etc.[103]. Although the controller can assign flows to queues and query queue statistics using OpenFlow protocol, it cannot perform certain configuration related actions such as creating queues and modifying queue parameters (e.g. maximum and minimum data rate for each queue). In fact, configuration protocols for OpenFlow devices such as OpenFlow Management and Configuration protocol (OF-CONFIG) or OVSDB were introduced in order to perform those missing configuration features from OpenFlow [104]. Even though the OVSDB protocol was not supported in any early SDN controllers, ODL controller from Linux foundation exclusively supports the OVSDB protocol [2].

3.3.3 Use Cases

As shown in Table 3.2, there are two main use cases of SDN for wirelessbackhaul: Mobility Management and Service Scheduling.

Use Case	Description
Mobility Management	Attachment and Handover procedures for the mobile part (modems in the train) of the wireless backhaul are managed by respective application in the NB of the SDN controller.
Service Scheduling (QoS improvement)	Priority queuing feature of OpenFlow can be used for enabling pre-scheduling by integrating OpenFlow into the wireless nodes (user and infra modems) and manage them by a service scheduling application in the NB of the SDN controller.

Table 3.2 – SDN platform for wireless backhaul use cases.

In the following, we describe in detail the three main procedures describing the use cases of our architecture.

User Modem Attachment

This procedure enables the User Modem to register to the network during the initial power on as shown in Fig. 3.13. All the IMs always broadcast their availability and the location of their downlink *Ranging* slot (a dedicated slot in the frame for connection request to be sent by UM). IMs send broadcast message during the dedicated uplink *Synch* slot known by UM. The UM receives the broadcast message and sends *Hello* messages to IM, which includes its identity and received power level from the IM. On the reception of *Hello* message from UM, IM sends *User-Detected* messages via SDN-Agent using the MCMP protocol to the UM Attachment application in the NB of the SDN Controller. This message includes the information received from UM such as identity and power level. After receiving this message, the UM Attachment application executes lookup process in the

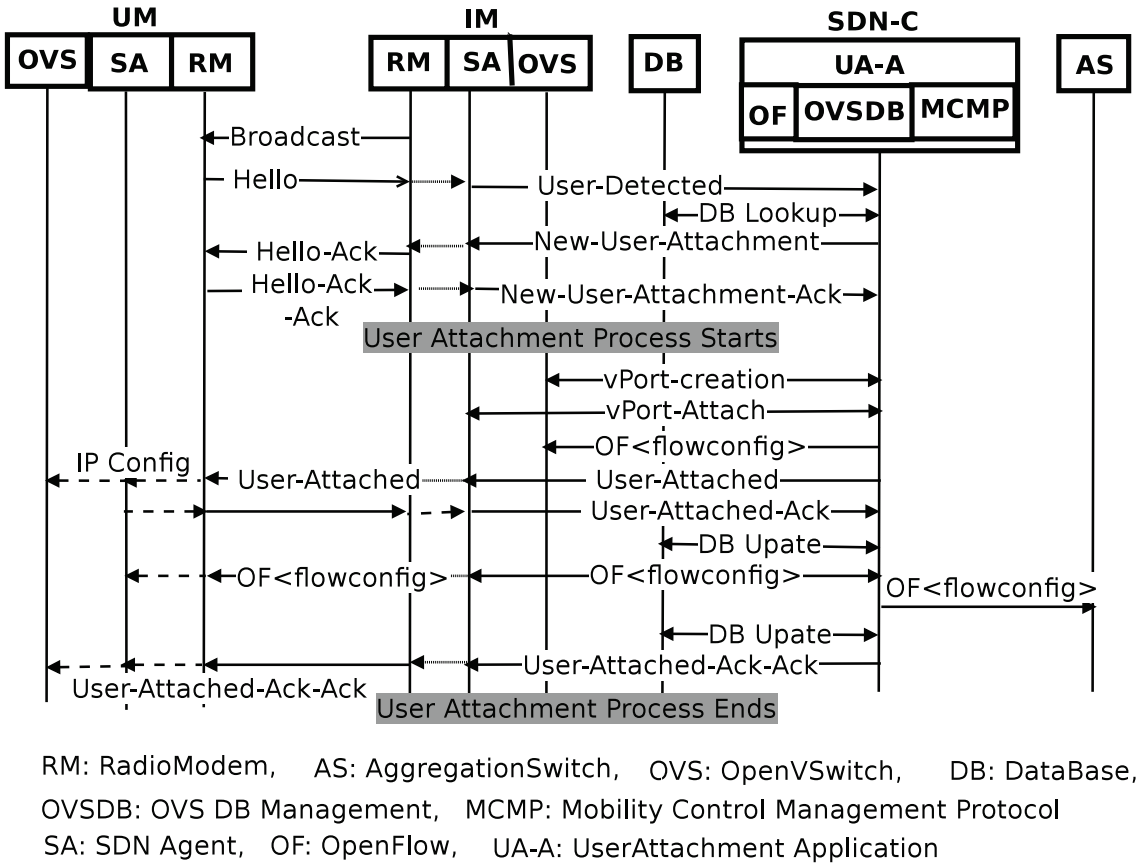


Figure 3.13 – User-Attachment Scenario.

DB to verify if the UM already exists in the network. If the UM is a new user in the network, the application sends back *User-Attachment* message to IM using the MCMP protocol. Then, the IM sends back *Hello-Ack* message to UM. UM then acknowledges IM by sending back *Hello-Ack-Ack* message. Upon receiving this message, IM sends *User-Attachment-Ack* message to UM Attachment application in the SDN controller. Now the application begins the User-Attachment process by creating a new entry for UM in IM's table in the DB. Using the OVSDB protocol, the application creates a vPort for the new UM in the IM's OpenVwsitch (OVS) and binds this port to a Radio Modem (RM) using dynamic socket connection process by sending *vPort-creation* message to OVS and a *vPort-Attach* message to SDN-Agent. After the vPort is established for the UM, the UM Attachment application using the OpenFlow protocol sends an OF<*flowconfig*> message to OVS in IM to configure the initial flow setup (DHCP discovery messages) for attached UM traffic to reach the ground network. This is necessary since all the devices connected in the train LAN network needs to get an IP address from the DHCP server located at the edge of the network for end-to-end IP connectivity.

After the initial flow configuration for the attached UM, the UM Attachment application sends a *User-Attached* message to the UM including the controller's IP address in order to configure OVS to establish connection with the SDN controller. After the successful configuration, UM sends a *User-Attached-Ack* message to IM and UM Attachment application in the controller. Then, the application updates the DB with the new user attachment details. Finally, the application sends an OF<*flowconfig*> message to OVS in UM and also to Aggregation Switch (AS) to create flow

entry in order to support initial DHCP discovery related traffic to reach the ground network via the IM. This completes the UM Attachment procedure and now all the devices in the train starts to discover their services and establish connections via IM. Once all the devices have discovered their IP addresses, AS needs to have a flow rule based on source destination IP addresses so as to route UM traffic to the respective IM. This is automatically done by routing applications running in most of the SDN/OpenFlow controllers using OpenFlow related services such as topology and switch manager.

Service Scheduling

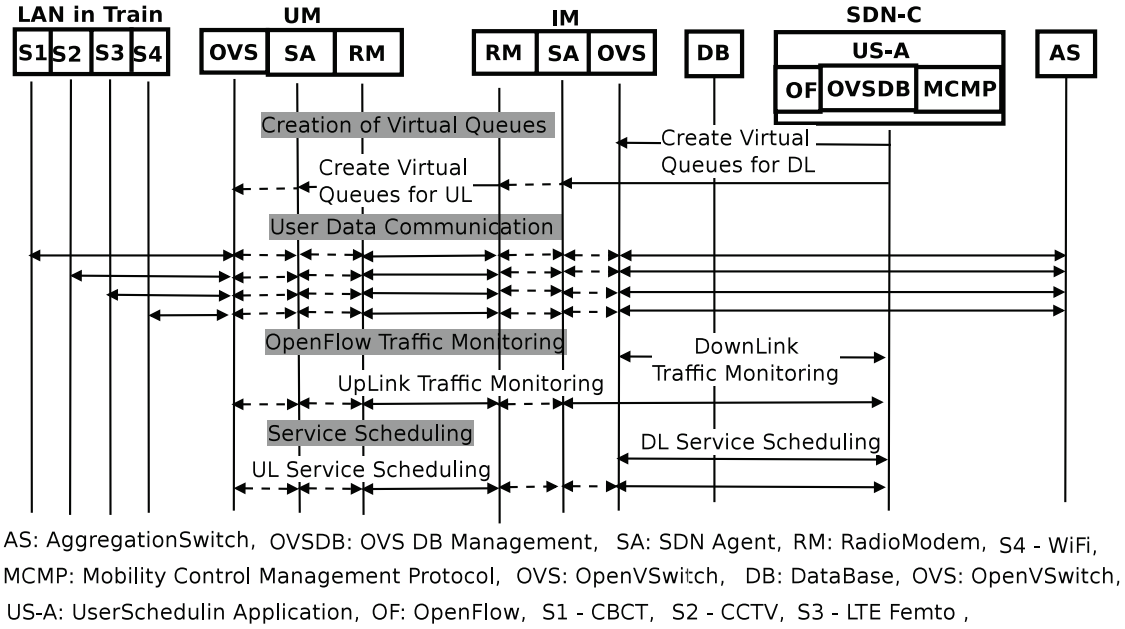


Figure 3.14 – Service Scheduling Procedure.

The Service Scheduling procedure follows immediately a successful UM Attachment procedure. Service scheduling is required to prioritize certain services especially those related to train control and surveillance systems. It is initiated by the Service Scheduling application in the controller. OpenFlow priority queuing feature is used here for service scheduling as shown in Fig. 3.14. The *Service Scheduling* application sends OVSDB messages (ovs-vsctl) to IM OVS and UM OVS to create a number of virtual queues for scheduling both downstream and upstream traffic to and from UM. Service Scheduling application monitors upstream and downstream traffic in the network using OpenFlow traffic monitoring features such as statistics per port, queue and flow in each OVS. The application attaches traffic flow of each service to different virtual queues and also configures the maximum and the minimum data rate for each virtual queue depending on their priority level. In our design, there are four different traffic flows as shown in Fig. 3.12. The priority decision and the maximum and minimum data rate for each queue completely depend on the algorithm implemented inside the Service Scheduling application. This procedure is repeated periodically as defined in the service scheduling algorithm.

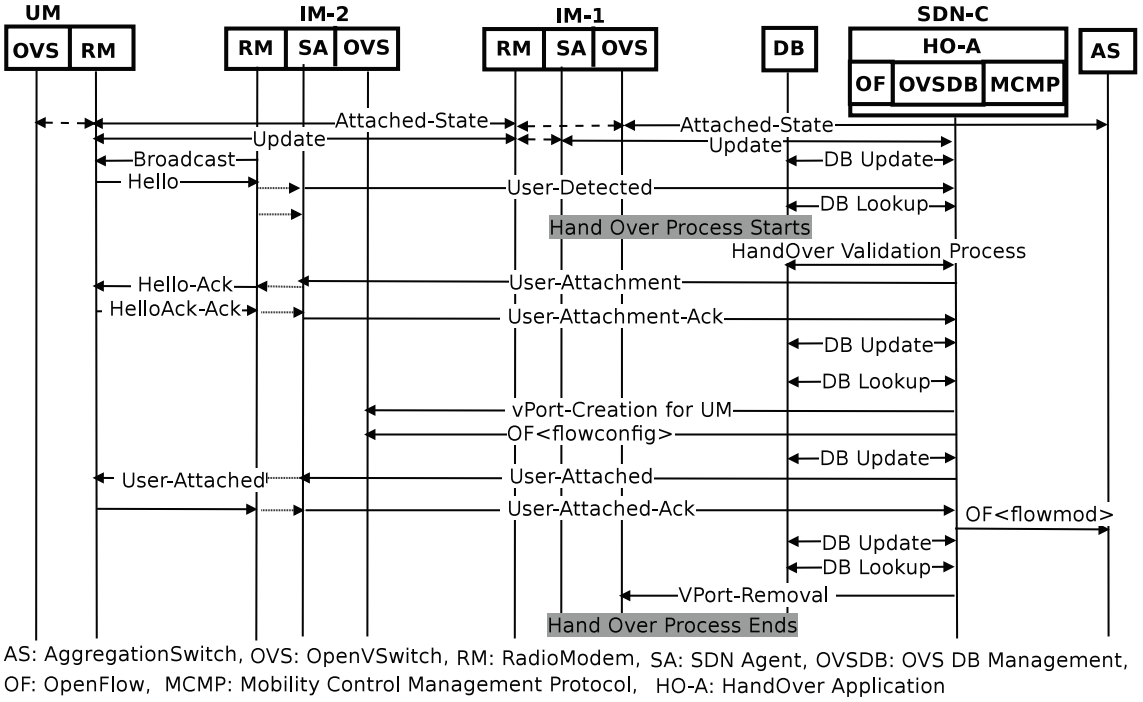


Figure 3.15 – User Modem Handover Procedure.

User Modem Handover

The handover procedure follows both UM Attachment and Service Scheduling procedures when the train is moving into the signal coverage of another IM, called as IM-2. As shown in Fig. 3.15, this procedure follows the same message sequence as the User Modem Attachment procedure until the NB *User Modem Handover* application begins the Handover validation process. The User Modem Handover application depends on its handover criteria (power level experienced by UM, load, etc.) to decide if the UM has to be attached to IM-2. In case the application decides to execute the handover, it sends a *User-Attachment* message to IM confirming the handover decision. Then, IM-2 acknowledges UM by sending a *Hello-Ack* message. On the reception of the hello acknowledgment, UM follows the handover procedure by sending back *HelloAck-Ack* to IM-2. Then, IM-2 proceeds the handover procedure by sending a *User-Attachment-Ack* message to the User Modem Handover application. After receiving an acknowledgment from IM-2, the handover application creates a vPort in the IM-2 OVS for the UM that is under the handover procedure by sending a *vPort-creation* message to OVS and a *vPort-Attach* message to the SDN-Agent. In order to avoid packet drops due to the handover, the application queries the DB for flow details corresponding to existing services in the UM and sets up th flow in IM-2's OVS by sending OF<*flowconfig*> message. Finally, the application sends a *User-Attached* message to IM-2 and IM-2 forwards it to UM. After the User Modem Handover application receives the *User-Attached-Ack* message from the UM, it modifies the flow rules in the Aggregation Switch for all services in UM to be routed via IM-2 by sending a OF<*flowmod*> message. It removes the vPort for UM in IM-1 by sending a *vPort-Removal* message to IM-1's OVS. This completes the handover procedure and the UM is now served by IM-2.

3.4 SDN for Dynamic User Processing Split

This chapter briefly describes the literature study and analysis carried out in LTE stack in order to define the architecture of software-defined Cloud Radio Access Networks (C-RAN). Then, it presents our software-defined C-RAN framework that enables the execution of dynamic user processing split decisions from the logically centralized SDN controller. This chapter also describes various advantages and use cases of our architecture.

3.4.1 Evolution of Cloud RAN

LTE Architecture and Functions

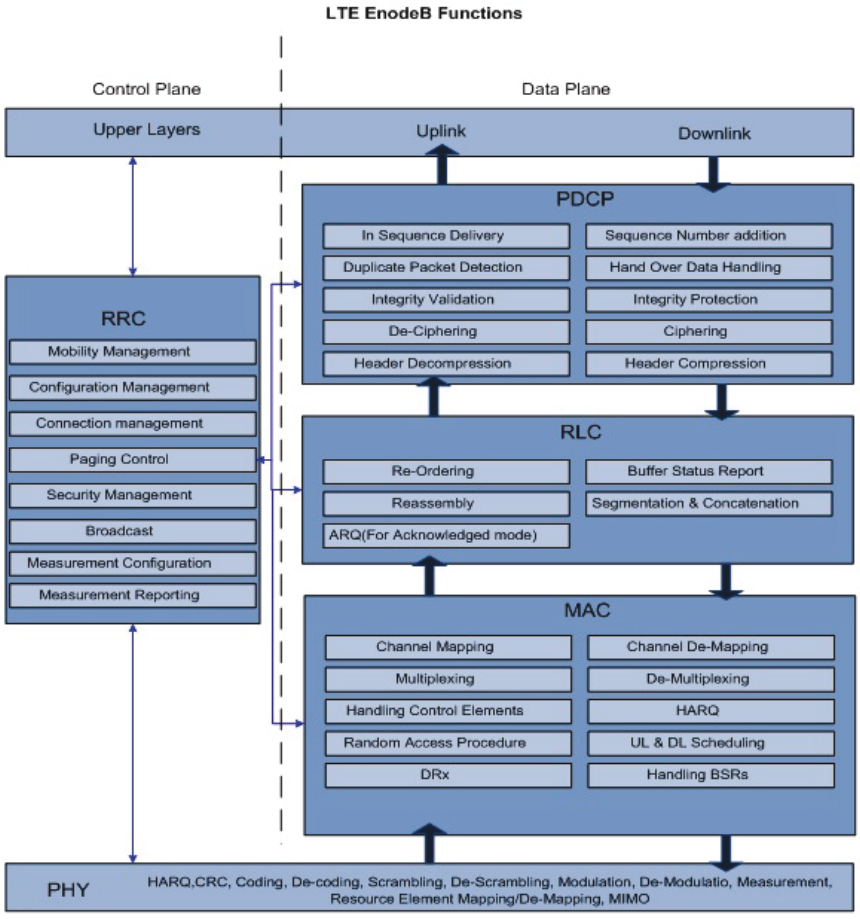


Figure 3.16 – LTE Architecture and Functions [7].

As shown in the Fig. 3.16, the radio protocol architecture of LTE can be separated into a control-plane and a user-plane architecture. At the user-plane side, the application creates data packets that are processed by protocols such as TCP, UDP and IP, while in the control plane, the Radio Resource Control (RRC) protocol writes the signaling messages that are exchanged between the base station and the mobile. In both cases, the information is processed by the Packet Data Convergence protocol (PDCP), the Radio Link Control (RLC) protocol and the Medium Access Control (MAC) protocol, before being passed to the physical layer for transmission.

On the user-plane, packets in the core network (EPC) are encapsulated in a specific EPC protocol and tunneled between the Packet Data Network Gateway (PGW) and the eNodeB. Different

tunneling protocols are used depending up on the interface. The GPRS Tunneling Protocol (GTP) is used on the S1 interface between the eNodeB and SGW and on the S5/S8 interface between the Serving Gateway (SGW) and PGW. The control-plane additionally includes the Radio Resource Control layer (RRC) which is responsible for configuring the lowest layers. The control-plane handles radio-specific functionality which depends up on the state of the user equipment which are either idle or connected.

Mode	Description
Idle	The user equipment camps on a cell after a cell selection or re-selection process where factors like radio link quality, cell status and radio access technology are considered. The UE also monitors a paging channel to detect incoming calls and acquire system information. In this mode, control plane protocols include cell selection and re-selection procedures.
Connected	The UE supplies the E-UTRAN with downlink channel quality and neighbor cell information to enable the E-UTRAN to select the most suitable cell for the UE. In this case, the control plane protocol includes the Radio Link Control (RRC) protocol.

Table 3.3 – Radio Resource Control States.

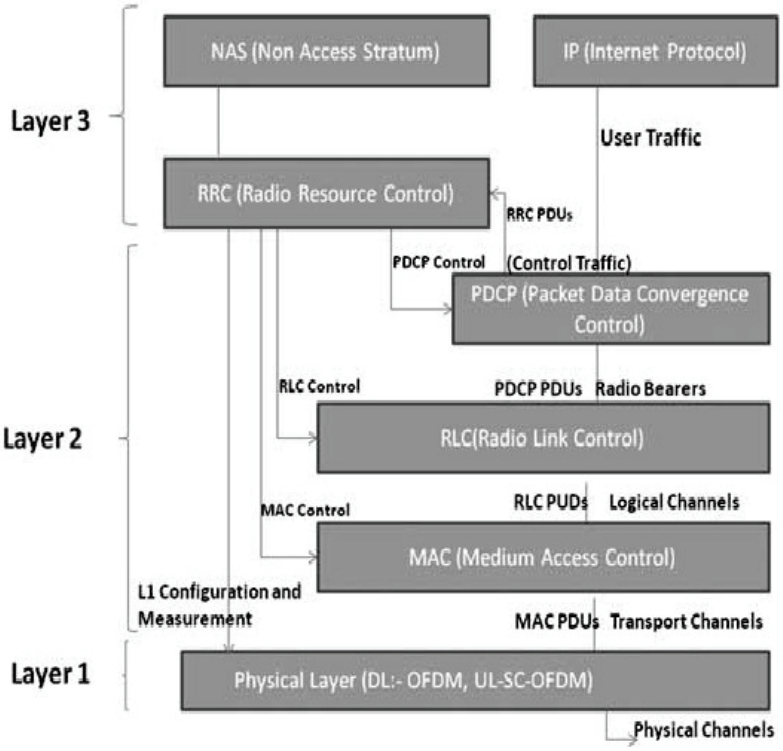


Figure 3.17 – Different Layers of LTE [7].

- Physical Layer (Layer 1): The Physical Layer carries all information from the MAC transport channels over the air interface. It takes care of the link adaptation, power control, cell search (for initial synchronization and handover purposes) and other measurements (inside the LTE system

and between systems) for the RRC layer.

- Medium Access Layer (MAC): The MAC layer is responsible for mapping between logical and transport channels, multiplexing MAC service data units (SDUs) from one or different logical channels into transport blocks (TB) to be delivered to the physical layer on transport channels, de-multiplexing MAC SDUs from one or different logical channels from transport blocks (TB) delivered from the physical layer on transport channels, scheduling information reporting, error correction through HARQ, handling priority between UEs by means of dynamic scheduling, handling priority between logical channels of one UE, prioritization of logical Channels.
- Radio Link Control (RLC): RLC operates in three modes of operation such as Transparent Mode (TM), Unacknowledged Mode (UM), and Acknowledged Mode (AM). RLC layer is responsible for transfer upper layer protocol data units (PDUs), correcting error through ARQ, concatenation, segmentation and reassembly of RLC SDUs. RLC is also responsible for re-segmentation of RLC data PDUs, re-ordering of RLC data PDUs, duplicate detection, RLC re-establishment, and protocol error detection.
- Radio Resource Control (RRC): The main services and functions of the RRC sublayer include broadcast of System Information related to the non-access stratum (NAS), broadcast of system information related to the access stratum (AS), paging, establishment, maintenance and release of an RRC connection between the UE and E-UTRAN, Security functions including key management, establishment, configuration, maintenance and release of point to point radio bearers.
- Packet Data Convergence Control (PDCP): The PDCP Layer is responsible for the header compression and the decompression of IP data, data transfer (user-plane or control-plane), maintenance of PDCP sequence numbers (SNs), in-sequence delivery of upper layer PDUs at re-establishment of lower layers, duplicate elimination of lower layer SDUs at re-establishment of lower layers for radio bearers mapped on RLC AM, ciphering and de-ciphering of user-plane and control-plane data, integrity protection and verification of the control-plane data, timer based discard, and duplicate discarding. PDCP is also used for mapping signaling radio bearers (SRBs) and dedicated radio bearers (DRBs) mapped on DCCH and DTCH type of logical channels.
- Non Access Stratum (NAS) Protocols: The non-access stratum (NAS) protocols form the highest stratum of the control plane between the user equipment (UE) and MME. NAS protocols support the mobility of the UE and the session management procedures to establish and maintain IP connectivity between the UE and a PDN GW.

Overview of Cloud RAN

RAN is the most important asset for mobile operators to provide high data rate, high quality, and 24/7 services to mobile users. A traditional RAN architecture has the following characteristics: first, each base station (BS) only connects to a fixed number of sector antennas that cover a small area and only handle transmission and reception signals in its coverage area; second, the system capacity is limited by interference, making it difficult to improve the spectrum capacity; and last but not least, BSs are built on proprietary platforms as a vertical solution. These characteristics have resulted in many challenges. For example, the large number of BSs requires the corresponding initial investment, site support, site rental and management support. Building more BS sites means increasing CAPEX and OPEX. Usually, the utilization rate of base stations is low because the average network load is usually far lower than that in peak load, while the BS processing power can not be shared with other BSs. Isolated BSs prove costly and difficult to improve the spectrum capacity. Lastly, a proprietary platform means mobile operators must manage multiple

non-compatible platforms if service providers want to purchase systems from multiple vendors, causing operators to have more complex and costly plan for network expansion and upgrading. To meet the fast increasing data services, mobile operators need to upgrade their network frequently and operate multiple-standard network, including GSM, WCDMA, TD-SCDMA and LTE. However, a proprietary platform means mobile operators lack the flexibility in network upgrade, or the ability to add services beyond simple upgrades.

Cloud Radio Access Network (C-RAN) proposed by the China Mobile Research Institute (CMRI) addresses the challenges faced by network operators [17]. As shown in Fig. 3.18, C-RAN consists of Remote Radio Heads (RRHs) with antenna equipment installed in the cellular sites connected to the centralized baseband unit pool (BBU pool) using microwave or optical connections. Centralizing the baseband processing reduces the rack space required to set up base stations that covers the same area. The virtualization of base stations enables dynamic on-demand allocation of resources, which in turn reduces the overall power consumption and increases the utilization rate [17].

There are three stages involved in the advance implementation of C-RAN:

- The centralization of base band unit(s) (BBU) or digital unit(s) (DU) in the central cloud called BBU or DU pools.
- A high-speed and low-latency interconnection between BBUs is required to exchange information to facilitate the implementation of collaborative radio to achieve a reduction in interference issues with improved system performance.
- A Centralized coordination and management framework is required to efficiently manage the BBU resources in the central cloud to achieve load balancing, efficient resource allocation on demand, etc.

The deployment of small cells technology with C-RAN improves the quality-of-experience and greatly reduces the power consumption in the radio access network. Virtual base stations are being co-located in the same BBU pool; co-operative signal processing techniques can be easily implemented to reduce the inter-cell interference and improve the spectral efficiency. Although the C-RAN architecture has lots of advantages, it still has many technical challenges that need to be solved in order to deploy them in mobile networks. In order to take benefits from the various features and to mitigate the identified challenges in the C-RAN, it is important to have a tight control framework that is flexible, dynamic and yet predictable.

Cloud RAN is not a replacement for 3G or 4G standards. It is only an alternative approach to gain advantages of IT technologies as well as addressing problems that arised after overwhelming mobile network usage. From a long-term perspective, C-RAN provides a low-cost and high-performance green network architecture to operators. In turn, operators are able to deliver rich wireless services in a cost-effective manner for all concerned. C-RAN is not the only RAN deployment solution that will replace all today macro, micro, and pico cell stations, indoor coverage system, and repeaters. Different deployment solutions have their respective advantages and disadvantages and are suitable for particular deployment scenarios. C-RAN is targeting to be applicable to most typical RAN deployment scenarios such as macro, micro and pico cells and indoor coverage. In addition, other RAN deployment solutions can also serve as complementary for certain mobile network use cases.

Cloud RAN can be deployed in different architectures with respect to the chosen process splitting options. In general, there are two main deployment options for CRAN, which namely are partially or completely centralized C-RAN. In the partially-centralized C-RAN option, layer 1 or

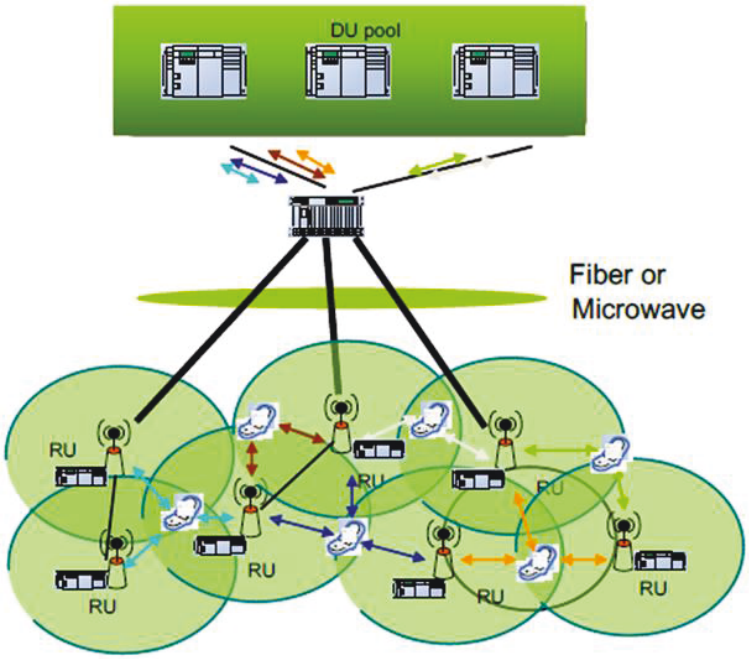


Figure 3.18 – General Architecture of Cloud RAN [8].

physical-layer functionality of baseband is still located adjacent to the cell site antenna unit or RRH, and MAC and PDCP processes are located in the central cloud. In the case of a fully centralized C-RAN, the complete stack of BBU functions are centralized or located in the central cloud leaving RRH in the remote cell sites.

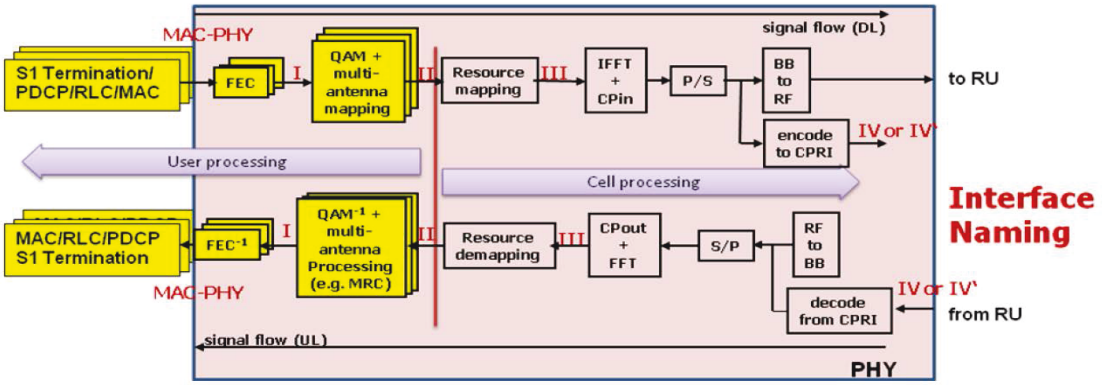


Figure 3.19 – Potential Interfaces for process split of Physical Layer [8].

The functional blocks of the base band processing and potential interfaces for a functional split at the interface between central cloud (BBU Pool) and Remote Unit (RU) is explained in the Fig. 3.19. In both Uplink (UL) and Downlink (DL) baseband processing, the baseband part can be subdivided into traffic load dependent user processing part and traffic independent cell processing part. The User processing part of the BBU includes entities such as S1 termination, PDCP, RLC, MAC, PHYuser with Forward Error Correction (FEC), quadrature amplitude modulation (QAM) and multi antenna mapping for DL, PHYuser for downlink (Forward Error Correction (FEC), quadrature amplitude modulation (QAM) and multi antenna mapping) and PHYuser for uplink

(inverse Forward Error Correction (FEC), inverse quadrature amplitude modulation (QAM) and multi antenna processing for uplink). The Cell processing part includes entities such as Resource Mapping (framer), Resource De-mapping (De-framer), FFT, CPin (Cyclic Prefix insertion) for UL, CPout (Cyclic Prefix Out) for DL, Common Public Radio Interface (CPRI) encoding for DL and CPRI decoding for UL. The fronthaul, an interface between the Remote Unit (RU) and the BBU Pool is an important aspect of the C-RAN deployment in order to fulfill the timing requirements of LTE without any compromise. This latency requirement of the front-haul interface can vary according to the splitting option i.e the decision on part of the baseband processes that need to be centralized.

	MAC-PHY	I	II	III	IV'	IV
DL	139,9 Mb/s	298,9 Mb/s	2,9 Gb/s	3,02Gb/s	4,9Gb/s	14,7Gb/s
UL	123,2 Mb/s	1,94Gb/s	4,17/3,74* Gb/s	4,78/4,3* Gb/s	4,9Gb/s	14,7Gb/s

Figure 3.20 – Data Rate for Downlink [8].

According to the 3GPP standard the latency requirement for physical layer processes (L1) is more stringent as compared to that of MAC and PDCP layers mainly due to the 1 ms scheduling interval. As per the analysis of Next Generation Mobile Networks (NGMN) consortium, the required data rate for each interfaces between entities in LTE base band processing both in uplink and downlink is listed in Fig. 3.20, and it claims that the required data rate on the interface is comparatively lower when the splitting decision is done between MAC and PHY. For the analysis they have considered the configuration of a 20 MHz LTE system with 3 sectors and 4 antennas. This is the scenario of a partial C-RAN where the Physical layer is still located adjacent to the RU and the MAC layer is cloudified or virtualized in the central location connected by the fronthaul network interface. Similarly, the required data rate is relatively higher in the fully centralized C-RAN scenario, where both Physical and MAC processes are centralized in the cloud leaving only the radio functions in the remote unit. Such a scenario requires a high-speed optical or ethernet-based fronthaul network (14.7 Gb/s) to satisfy the underlying latency requirement from LTE standards.

3.4.2 Architecture

The main focus of this research is to implement control framework for dynamic user processing split i.e. to virtualize the User Processes (UP) using SDN approach. The capability of SDN to have complete view of the network from the logically centralized controller simplifies the overall network management process. Alignment of SDN with C-RAN also provides other solutions such as programmable fronthaul network, centralized radio resource management and coordinated joint signal processing. The simulation study in [126] shows that the resource utilization would improve up to a factor of 3 by the distribution of user load (between the uniform to non-uniform user distribution cases). Precisely, the virtualization in our approach considers the distribution of work load corresponding to user dependent functions of RAN during allocation of data radio bearers (DRBs) i.e. L1-UP, L2 and L3 functions as shown in Fig. 3.21. This approach does not consider the virtualization of cell dependent functions such as Parallel to Serial (P/S) and Serial to Parallel (S/P) conversions, Fast Fourier Transform (FFT), Inverse Fast Fourier Transform (IFFT), Cyclic Prefix In (CPin), Cyclic Prefix Out (CPout), Resource Mapping and De-mapping functions.

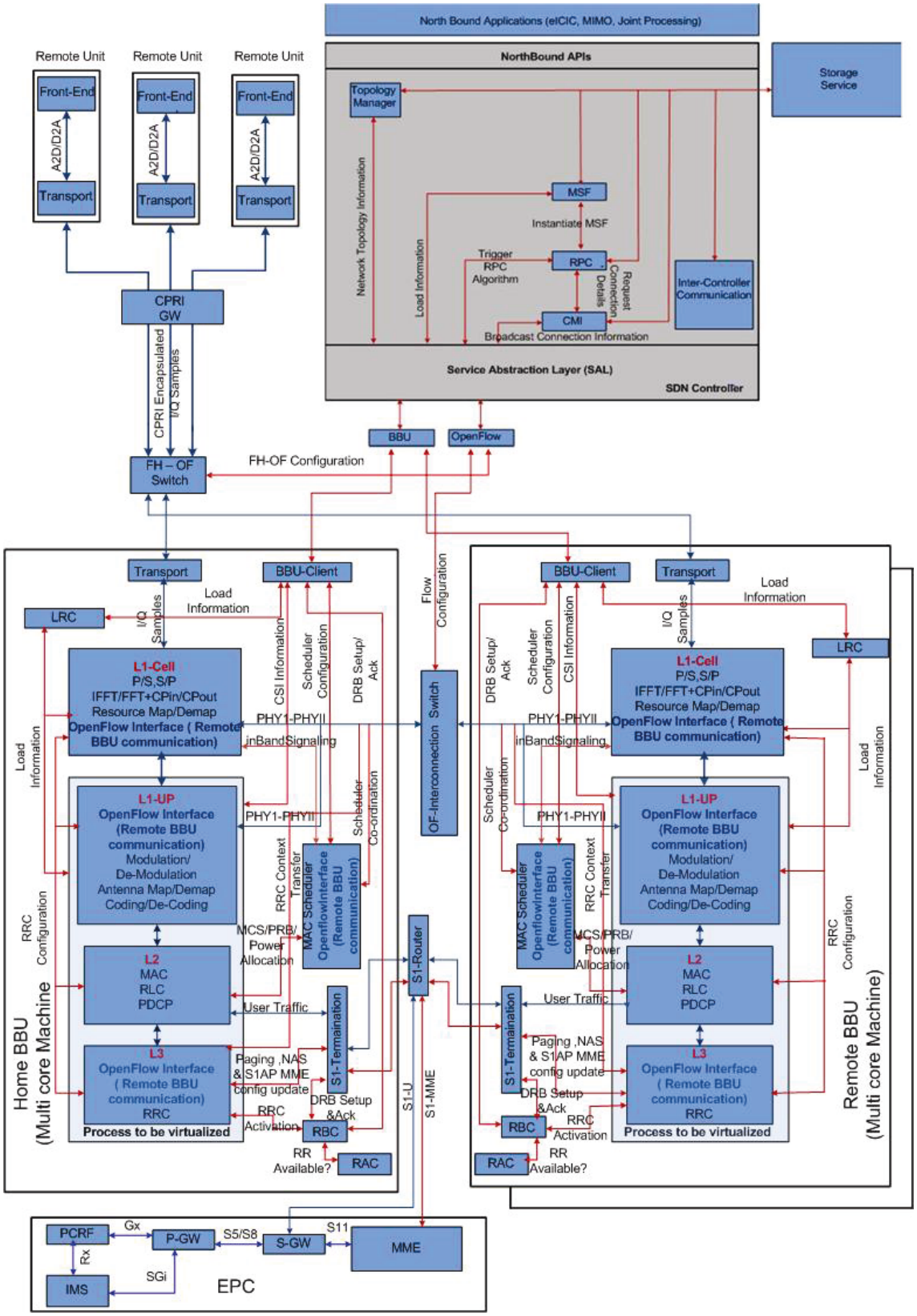


Figure 3.21 – Software-Defined Cloud RAN Architecture.

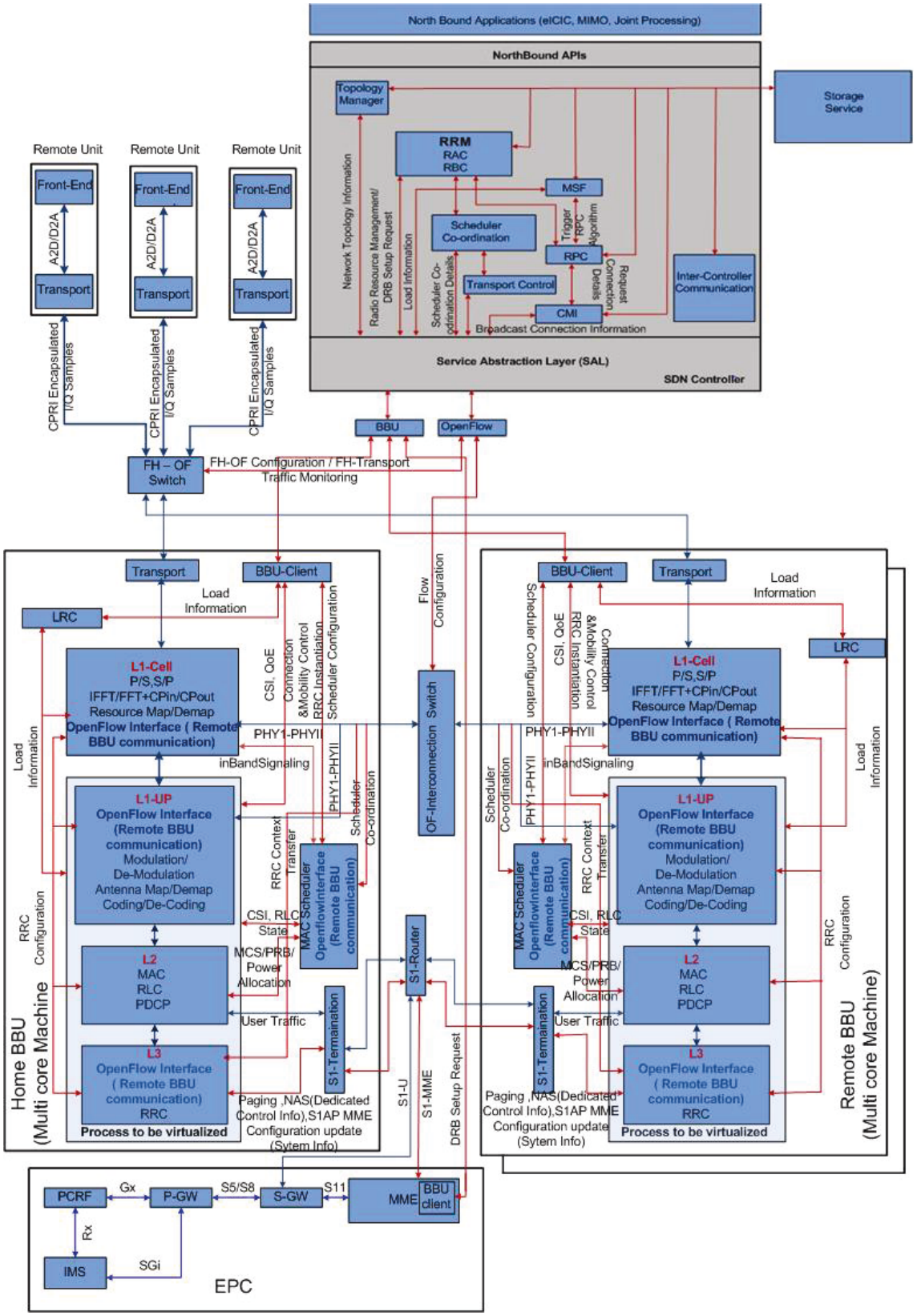


Figure 3.22 – Software-Defined Cloud RAN with complete centralization of RRM.

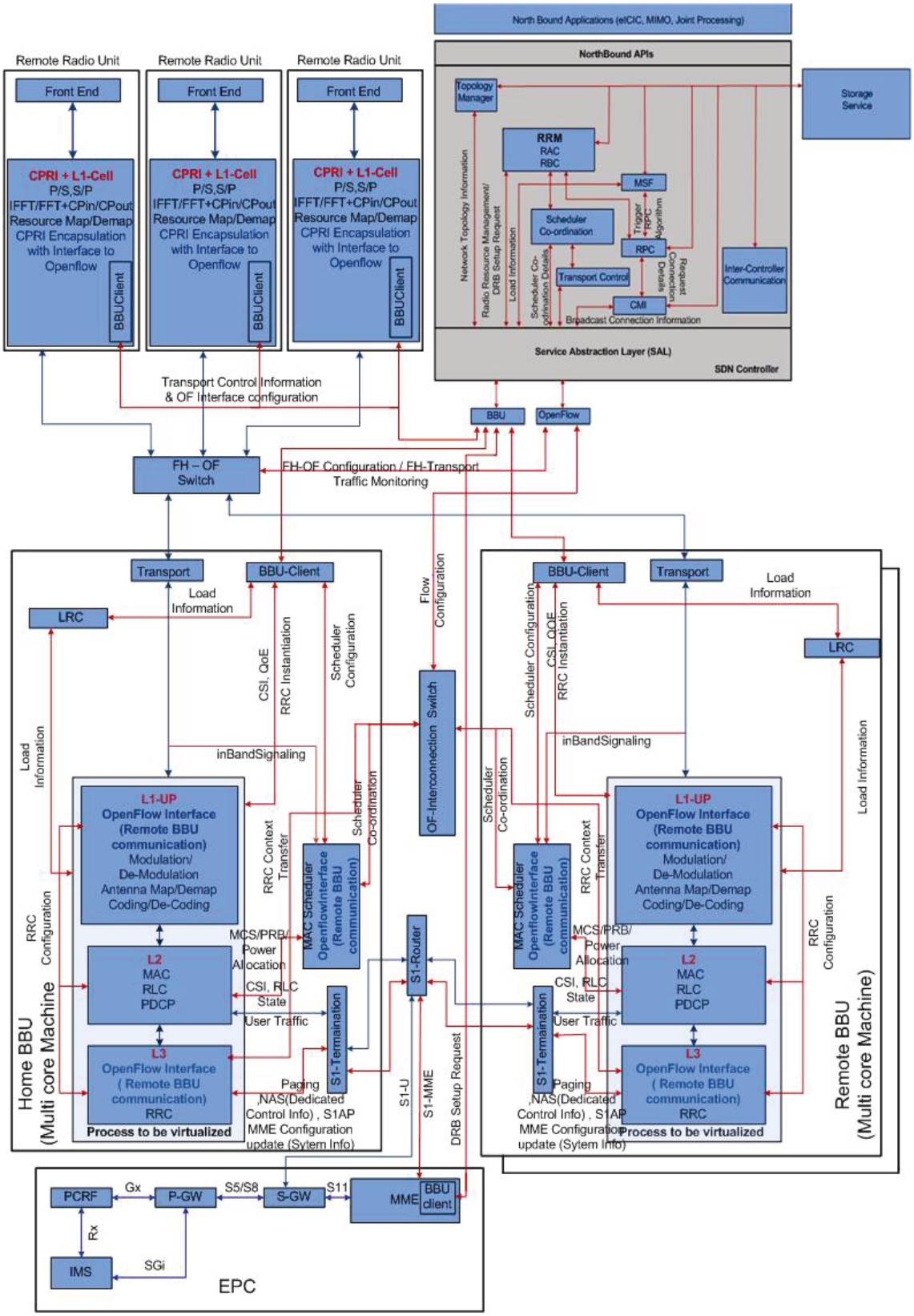


Figure 3.23 – Software-Defined Cloud RAN with PHY-Cell in the RRH and complete centralization of RRM.

The general architecture of software defined Cloud RAN is described in Fig. 3.21, where SDN plays a key role in managing network as well as computing resources. The architecture consist of six main components such as SDN controller, Remote Unit or RRH, CPRI Gateway, base band units (BBUs), OpenFlow enabled transport networks (for fronthaul and BBU interconnection) and core network. The key focus of this architecture is to have control framework that distributes work load corresponding to User Processes (UP) functions of Data Radio Bearers (DRBs) across one or more BBUs. In other words, the SDN framework provides unified management framework to deploy resource pooling algorithm that improves overall resource utilization while maintaining QoS for each user. This architecture also supports re-allocation or process migration during overload situations. The Remote Unit (RU) consist of radio functions and transport units for sending and receiving I/Q samples to and from the CPRI Gateway unit. The CPRI Gateway unit is responsible for Encapsulation/De-Encapsulation of I/Q samples to ethernet packets and vice versa. This is necessary for achieving both ordering as well as sending data over an ethernet-based fronthaul network. Since the fronthaul network is OpenFlow complaint, it can send downlink traffic to appropriate BBUs that processes the data from the particular RU, and also the uplink data from the BBUs to the appropriate RU.

Our architecture consist of one home BBU and number of remote BBUs. It means that the home BBU serves the particular cell or group of cells, and remote BBUs can be used in case of overload conditions. In the base band unit, the LTE stack is split into number of functions such as L1-Cell, L1-UP, L2, L3, Radio Admission Control (RAC), Radio Bearer Control (RBC), and MAC Scheduler. The physical layer functions are grouped according to user-independent (L1-Cell) and user-dependent (L1-UP) functions, i.e. the various physical layer functions such as modulation/de-modulation, coding/de-coding and data mapping/de-mapping depend on the QoE of the user. As highlighted in the Fig. 3.21, the virtualization is focused only on L1-UP, L2 and L3 functions in the LTE stack. Each entity in the data-plane part of the cloud RAN consists of an OpenFlow interface for enabling Layer-2 or Layer-3 packet communications via the OpenFlow inter connection switch. The switching and routing between different baseband units are carried out through the inter-connection switch that is configured via suitable switching and routing applications in the NB of the controller.

In order to make use of the complete advantages of software-defined C-RAN, it is necessary to centralize the management of radio resources, i.e. the centralization of RAC and RBC. This is necessary for the SDN controller to have a complete abstraction of radio resources in the BBU pool (availability, usage, etc.) to well precise radio admission control decisions. The architecture of software-defined Cloud RAN with centralization of RRM is shown in Fig. 3.22. The various solutions for inter-cell interference issue such as ICIC and eICIC involve the management of radio resources between interfering neighbours. The possibility of joint radio resource management from the centralized SDN controller paves the way for the realization and deployment of ICIC and eICIC algorithms as an application at the NB of the SDN controller. In order to achieve the centralization of RRM, one of the control flow in the LTE standard is changed in our approach, i.e. the DRB request messages coming from SGW to MME need to be sent directly to the SDN controller since the RBC is now located in the controller. The MME of the EPC is now integrated with the SDN framework to achieve the centralization of RRM, but it is limited to the DRB setup procedure. The architecture in Fig. 3.23 demonstrates the complete centralization of RRM along with the integration of L1-cell or non-virtualized user independent physical layer functions in the RRH. The advantage of such an approach is to bring down the required data rate in the fronthaul network

segment that connects the RRH to the BBU pool as explained in the Table 3.19. This approach also allows the entire BBU pool to be completely virtualized according to our approach of user process virtualization. The functionality of each component in our architecture is summarized below:

FrontHaul OpenFlow Switch

The openflow switch in the front haul establishes a switching network between RRHs and BBUs. It helps to recover transport links between RRHs and BBUs in case of BBU failure with the direct control from the central SDN controller. The OpenFlow switch has the ability to be re-programmed dynamically from the SDN controller to re-direct traffic corresponding to one destination to another. This is necessary in order to distribute radio traffic from RRHs to different processing units in the BBU pool depending on the load conditions.

OF-Interconnection Switch

The openflow switch is used for interconnecting physically separated BBUs or processing machines located in the BBU pool. It is directly configured by the centralized SDN controller to direct flows according to various header information such as source and destination addresses of each flow. The OpenFlow switch has the ability to work both on L2 switching and L3 routing mode and is a beneficial aspect for RAN context where the networking between both L2 and L3 interfaces is required.

BBU Client

The BBU Client is a communication module located in each of the Base Band Unit (BBU) for establishing communication with the controller. It sends and receives data and configuration details to and from the controller. In the proposed architectures, two different protocols are defined for the purpose of communication and configuration:

- BBU protocol: Southbound BBU protocol, which is a dedicated SDN protocol that is defined in our architectures for the integration of C-RAN to our SDN framework. All the BBUs in the MSS-BBU pool can establish a communication with the controller using the BBU protocol. BBU protocol has a set of defined rules to send and receive data to and from the BBUs for DRB setup procedure, resource usage monitoring, failure detection, network abstraction, resource allocation and re-allocation procedure, configuration procedures during joint signal processing and joint transmission/reception setups, etc.
- OpenFlow protocol: It can be used for configuring the OpenFlow based fronthaul and the interconnection switch since they have a built-in OpenFlow agent for the controller communication and execution of the commands from the controller.

The following are the typical functions of the **BBU Client**:

- During the initialization process it sends the MAC/IP address of the corresponding BBU to the SDN controller.
- It reads the resource and load status from the Local Resource Controller (LRC) every pre-defined interval of time and sends them to the controller if there is any change in their value as compared to the previous one. This avoids any duplication of resource status sent to the controller.
- The BBU Client sends the quality-of-experience and the Channel State Information (CSI) reports from UP functions to the SDN controller.

- It receives connection details and other information from the SDN controller during DRB setup and configures various functions (Scheduler, UPs) in the BBU to establish remote BBU coordination and communication.
- In the use case of the DRB setup request that involves Radio Bearer Control (RBC), the RBC sends the request via the BBU Client to the controller.
- The BBU Client is the generic module which can be extended further with any new functional extension in the BBU unit depending on the C-RAN deployment scenarios and splitting options.
- The Local Resource Controller (LRC) is a module located in each BBU to generate the resource usage report and store it in the local database associated with it. The BBU Client reads this database in a defined interval of time and report it to the SDN controller if there is a variation in the resource usage report.

OpenFlow Interface

It is a specialized module in each BBU to support communication over OpenFlow interconnection switch in the BBU pool, especially the DRB established across distributed BBUs i.e during remote BBU communications. It is used to packetize the extracted data from the de-mapping function to discriminate data corresponding to different flows. In order to establish a communication between the home and the remote scheduler there is a need for the Open Flow (OF) interface to define the message corresponding to the particular flows. During the initial DRB setup phase, the OF interface get configured by the controller along with the DRB id and the corresponding IP address of those BBUs that serve the particular DRB.

InBand Signaling for Mapping, De-Mapping and OpenFlow Interface

There is an in-band signaling from the scheduler that goes to the Mapping and De-Mapping function as well as to the OF Interface to inform about the Physical Resource Block (PRB) allocation for remote DRBs.

Software Architecture of SDN controller:

The detailed software architecture of SDN controller for C-RAN is depicted in Fig. 3.24. Since ODL was highly supported and recognized by Telco vendors and operators at the time of this research, we used ODL as reference SDN controller to define software-defined Cloud RAN architectures. The controller uses two different protocols such as BBU (our specified protocol for C-RAN management) and the OpenFlow protocol. This is the unique feature of OpenDayLight SDN controller that has multiple southbound protocols and that can be extended further to have more SB protocol plugins as per different requirements. The service abstraction layer (SAL) in the controller is the management module that translates various commands and requests coming from the BBU pool and initiates the respective controller services. As an example, if it is a request for the DRB setup, then SAL triggers Resource Pooling Control (RPC) to initiate the DRB setup procedure or else Measurement Supporting Function (MSF) for storing the load information in the controller database. The MSF is responsible for storing the measurement information coming from the BBU pool, and triggering Resource Pooling Control (RPC) for initiating dynamic resource allocation in case of overload indication. In all scenarios considered in this document, Radio Resource Management (RRM) consists of only Radio Admission Control (RAC) and Radio Bearer Control (RBC). However in reality, centralizing the RRM can support the implementation of LTE-A features such as eICIC, CMC and MIMO from the top of the controller. Connection

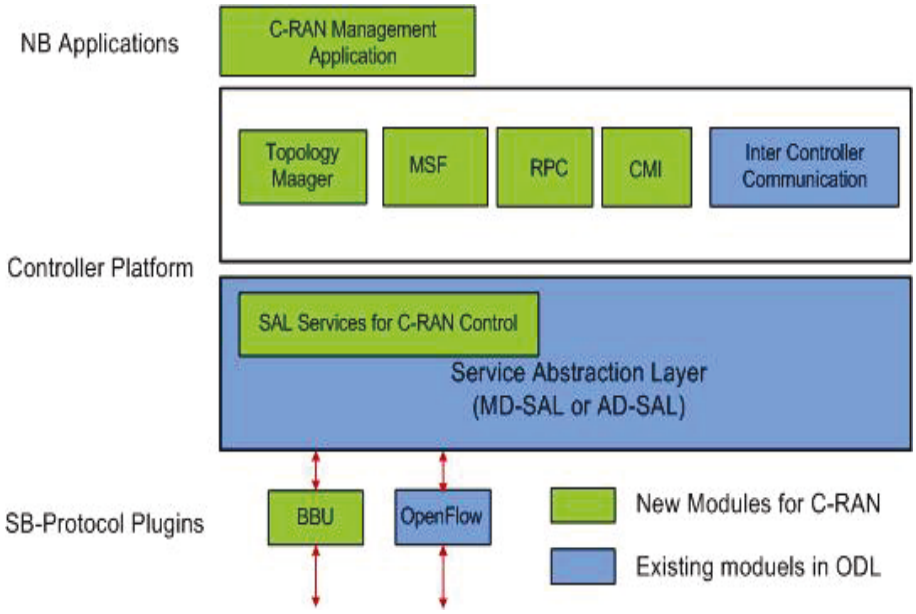


Figure 3.24 – SDN Controller Architecture of software-defined Cloud RAN.

Management and Instantiation (CMI) is an inventory responsible for the identification of all the connection and configuration details for the newly setup DRBs or for the reallocated ones. The topology Manager is responsible for collecting network information of all the processing and networking nodes in the Cloud RAN architecture to generate the end-to-end network topology that is needed during the process of resource allocation and load migration. Controller interacts with the Storage Services for fetching information. The Scheduler Coordination module is specified for the implementation of CoMP and eICIC. Inter Controller Communication is a service used for controller-to-controller communications especially in the scenario of having distributed as well as standby or backup controllers.

3.4.3 Use Cases

Though the main focus of the architecture is to implement dynamic user processing split (virtualization) using SDN approach, the architecture can also support other use cases such as failover recovery, fronthaul control, and ICIC shown in Table III. In the following, we present in detail the three use cases such as dynamic user processing split, failover recovery, fronthaul control. The SDN approach for ICIC is already presented as a use case for "SDN for SON" in page 41.

Dynamic User Processing Split

In the following, we describe in detail the procedures (DRB allocation and dynamic re-allocation) corresponding to dynamic user processing split. The DRB setup procedure is explained in the Fig. 3.25, where the SGW initiate the DRB setup procedure by sending a DRB setup-request message to the MME soon after it receives a service request from the attached user. The MME upon receiving the DRB setup-request message along with the necessary QoS requirement parameters. It sends it to the SDN controller via the BBU-Client for the RRM decision. Once the SAL of the SDN controller detects the DRB setup-request, it routes this message to the RRM service in the controller. The RRM service module consists of two sub modules: RBC and RAC. The RBC in RRM interacts with the RAC in order to find the availability of radio resources. RRM always

Use Case	Description
Dynamic User Processing Split (Virtualization)	SDN controller with the global view of radio and computing resources in the underlying C-RAN infrastructure can control the placement of DRBs in the appropriate BBUs.
Failover Recovery	Monitoring application in the NB of the SDN controller is able to detect and trigger recovery procedure for affected DRBs during hardware or software failures.
Fronthaul Control	SDN controller using OpenFlow protocol can implement routing algorithm to protect radio traffic in the fronthaul transport network. This is most suited for scenario where telco and non-telco service providers share the same transport network infrastructure.
ICIC (eICIC, CoMP)	The possibility of joint radio resource management and coordination from the centralized SDN controller enable the realization of ICIC algorithms as applications at the NB of the SDN controller.

Table 3.4 – SDN platform for C-RAN use cases.

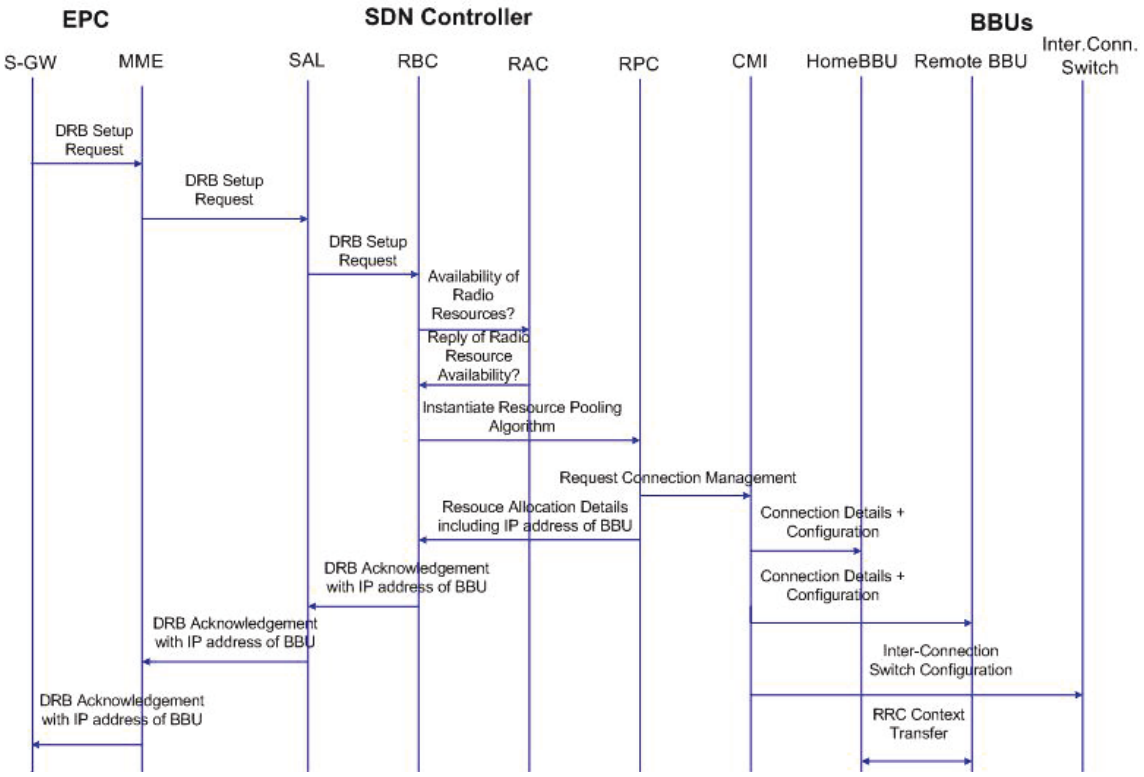


Figure 3.25 – Data Radio Bearer setup procedure for complete centralization of RRM.

interacts with the storage services (e.g. Data Base, XML, etc.) to get the radio resource status in the BBU pool under its control. If the RBC receives a positive response for the radio resource availability verification from the RAC, it triggers the RPC with the DRB requirement information to initiate the resource allocation procedure. RPC, with the help of the complete resource status information available in the storage services, decides the allocation of processing as well as the networking resources. Once RPC decides the location and amount of resources to be allocated to

the particular DRB, it sends the resource allocation details along with the IP addresses of those BBUs involved in the DRB service to the RBC as an acknowledgement of the resource allocation procedure. The RBC in turn forwards this message coming from RPC to the MME via SAL as an acknowledgement of the DRB setup-request message. The RPC in parallel triggers the CMI to manage the connectivity and networking between virtually and physically separated base band functions. After fetching the connection details along with the addresses of the inter-networking devices, CMI sends those details to both home and remote BBUs as well as to the inter-connection switch. The BBUs and the networking devices upon the reception of the configuration details, auto-configure itself for supporting the upcoming DRB traffic in the network. The home BBU, i.e. the BBU where the user is attached shares the user-context information with the remote BBU in case of remote resource allocation.

As shown in Fig. 3.22, the architecture of the software-defined C-RAN consists of both home BBU and remote BBUs. This is the preliminary step towards virtualization, where the BBU of neighbors or remote locations can be utilized or shared. As explained earlier, the SDN controller frequently monitors or receives the resources status from the C-RAN network, especially the BBU pool. The Measurement Supporting Function (MSF) service in the SDN controller receives the BBU resource updates from the Local Resource Controller (LRC) module in the BBU pool. These information are stored in the attached storage service with the controller, and shared between other controller services such as RPC and CMI. The dynamic resource re-allocation procedure during overload can be either initiated from the BBU by sending the over-load indication message to the SDN controller, or from the SDN controller by predicting the arrival of the over-load situation in advance using the monitoring information.

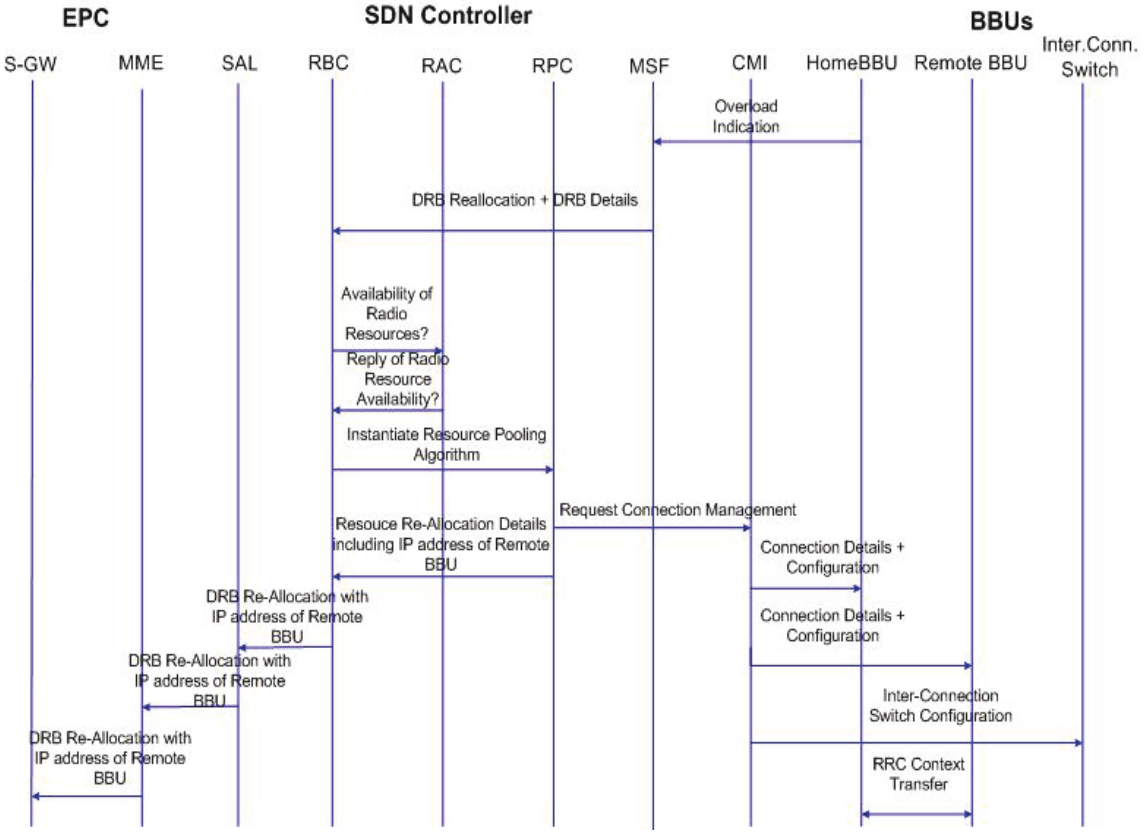


Figure 3.26 – Dynamic Resource allocation procedure for complete centralization of RRM.

The dynamic resource allocation procedure in case of an over-load situation is explained in the Fig. 3.26, where the DRB is allocated previously in the home BBU. The home BBU upon detection of an over-load situation, i.e. the LRC responsible for keeping track of the local resources in the BBU sends the over-load indication message to the SDN controller. The over-load indication message is routed to the MSF module in the controller by the SAL. The MSF investigates the over-load condition by interacting with the storage services and retrieve those details of those DRBs that have to be re-allocated. The MSF re-launch the DRB setup procedure by sending DRB allocation information to RPC. Then, the re-allocation procedure continues with the same steps as in the DRB setup procedure. The MSF repeats this procedure for the number of affected DRBs in the BBU due to over-load conditions. The process completes with the re-allocation of DRBs along with the context transfer from their home BBU to the newly allocated BBU.

Failover Recovery

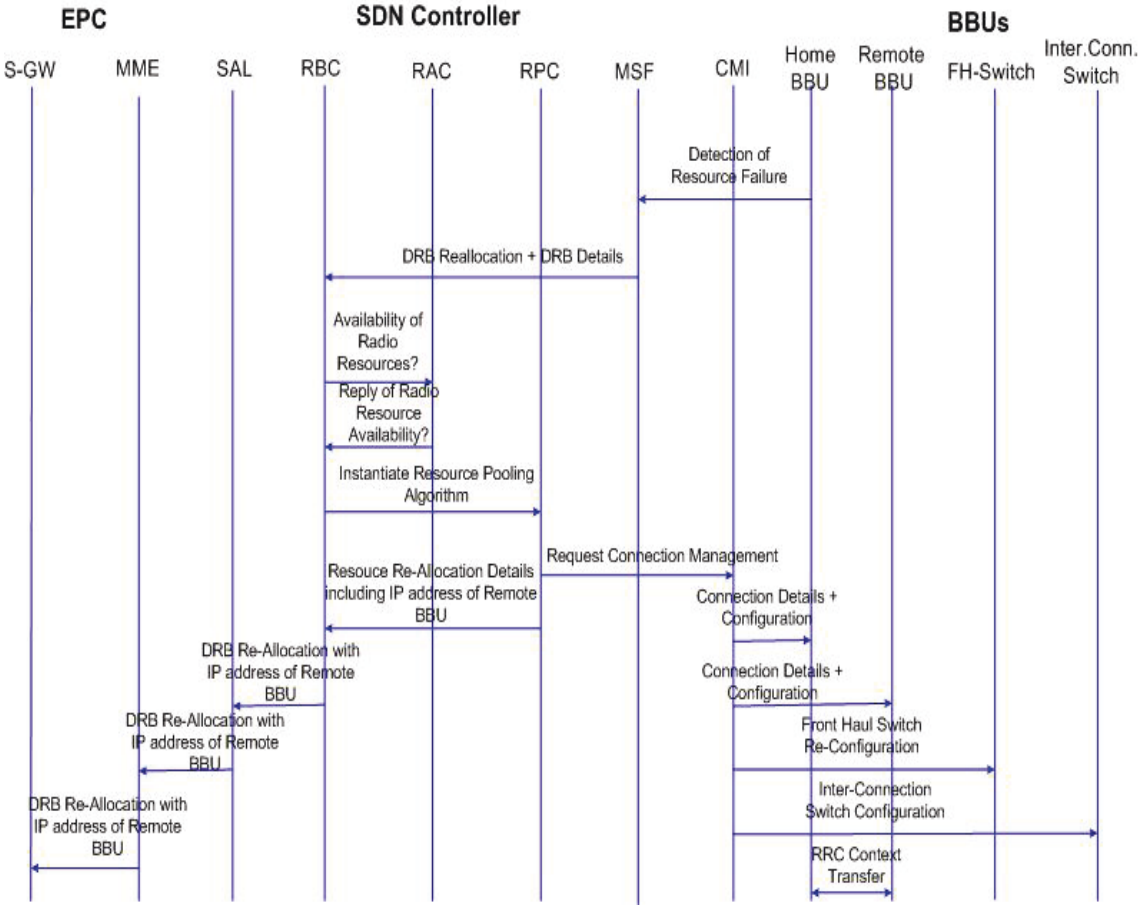


Figure 3.27 – Resource Failure Recover Procedure.

The resource failure procedure is similar to the dynamic re-allocation procedure, but it differs in the initiation. In all the software-defined Cloud RAN architectures, all BBUs sends frequent messages to the SDN controller via the LRC regarding their resource status. The SDN controller also acknowledges those status information from the LRC located in each BBU. As shown in the Fig. 3.27, the SDN controller upon the detection of a resource failure, i.e. no messages from the home BBU for a defined interval of time, initiates the service recovery procedure. Since MSF is the

module responsible for statistics retrieval, it begins the recovery procedure by interacting with the storage service and collects information about all the DRBs previously allocated to the affected BBU. The RPC continues the procedure with the same set of steps as in the DRB allocation procedure. The RPC service of the SDN controller completes the procedure by interacting with both storage service and CMI service.

Fronthaul Control

The architecture in 3.22 consist of single fronthaul OpenFlow switch that connects RRHs and BBUs. However, in deployment scenarios, C-RAN [8] consist of either optical or ethernet fronthaul network ranges between 20-40 Kms. The fronthaul network can be shared between telco and non-telco cloud service providers. In such scenarios, it is mandatory to satisfy the requirement of radio traffic i.e. fronthaul needs to transmit mobile data within the radio-link sub-frame interval. Hence, the fronthaul network requires Traffic Engineering (TE) mechanism to route/prioritize radio traffic through the best path in the network to satisfy QoS of mobile network services. The intrinsic features of OpenFlow such as provision of per flow QoS, network slicing, traffic metering, and availability of per flow statistics [21] can be used to implement such TE mechanism in the fronthaul to achieve the required criteria of mobile network services. Hence, SDN/OpenFlow based fronthaul network can address those objectives such as network sharing and TE.

Chapter 4

Architecture Validation

This chapter presents the prototypes and performance validation of software-defined SON and wireless backhaul architecture proposed in the previous chapter. The prototype of SDN for SON is demonstrated using two different virtualization technologies such as VMs and docker containers along with performance analysis. This architecture is validated with centralized SON algorithm as a NB application in the controller. For the use case of SDN for wireless backahul, both software and hardware prototypes are demonstrated with user-attachment scenario. This architecture is also integrated in the docker-swarm orchestration framework that deploys software-defined wirless backhaul As a Service.

4.1 SDN for Self Organizing Networks: Prototype, Implementation and Validation

In the proposed architecture in Fig. 3.6 on page 40, 4G/5G eICIC optimizer is deployed as a RAN application of the controller which is to coordinate MAC scheduling over LTE networks. In the testbed and in experimentations, Matlab based LTE-complaint simulator from TU Wien's Institute of Telecommunications is used to emulate LTE mobile network scenarios. It simulates typical 3GPP heterogeneous cellular network (HetNet) scenarios consisting of macro and small cells, mobile users, radio propagation and pathloss. For the purpose of experimenting the end-to-end functionality of our architecture, light-weighted eNBs that implement RNFA are emulated and fed with eNB related information (measurements and configuration) by Matlab using standard User Datagram Protocol (UDP) socket connections. Then, the emulated eNB acts like real base station to the SDN controller. The Matlab simulator and the SDN environment (emulated eNBs, SDN controller and MongoDB) are implemented in two physically separated machines. The SDN controller, MongoDB and eNBs are deployed as virtual entities in one physical machine and used another dedicated physical machine for the Matlab simulator.

The architecture is experimented using two virtualization techniques, virtual machines and Docker containers, under the same hardware configuration of 16-core CPUs and 16 GB RAM operating with Ubuntu Linux 14.04 version. The motivation of validating our framework under two different virtualization techniques is to analyze the performance and behavior of the system under the two different virtual environments and to provide a comparative study of their suitability for enabling 5G SDN/NFV.

4.1.1 Prototype: VM Based Virtualization

To justify the proposed SDN and system design, The first use case is tested: the eICIC optimizer over LTE HetNet delivering optimized CIO and ABS values. The network is iterated per Transmission Time Interval (TTI), which generates time varying link quality and traffics, and each eNB scheduler is running real-time on resource allocation.

As shown in Fig. 4.1, the SDN controller, the eICIC optimizer, the MongoDB and the emulated eNBs are deployed in separate VMs using lightweight Ubuntu 14.04 images. The Kernel-based Virtual Machine (KVM), that uses Quick Emulator (QEMU) to emulate hardware for virtual machine deployment is used for this experimentation. Number of virtual machines are launched in parallel by executing VM execution command through Python script. The networking between VMs is achieved through network address translation (NAT) through the virtual switch of the hypervisor. Though, VMs use limited number of hypercalls or emulated devices via hypervisor to establish communications with the outside world with a certain level of isolation and security, it is known that the hypervisor would add substantial overheads on the performance. These cannot be bypassed or optimized from higher layer. Note that the performance of VM based testbed is often lower when compared to native machines but this can be compensated by the advantage of virtualizing hardware resources of the same physical machine for several entities.

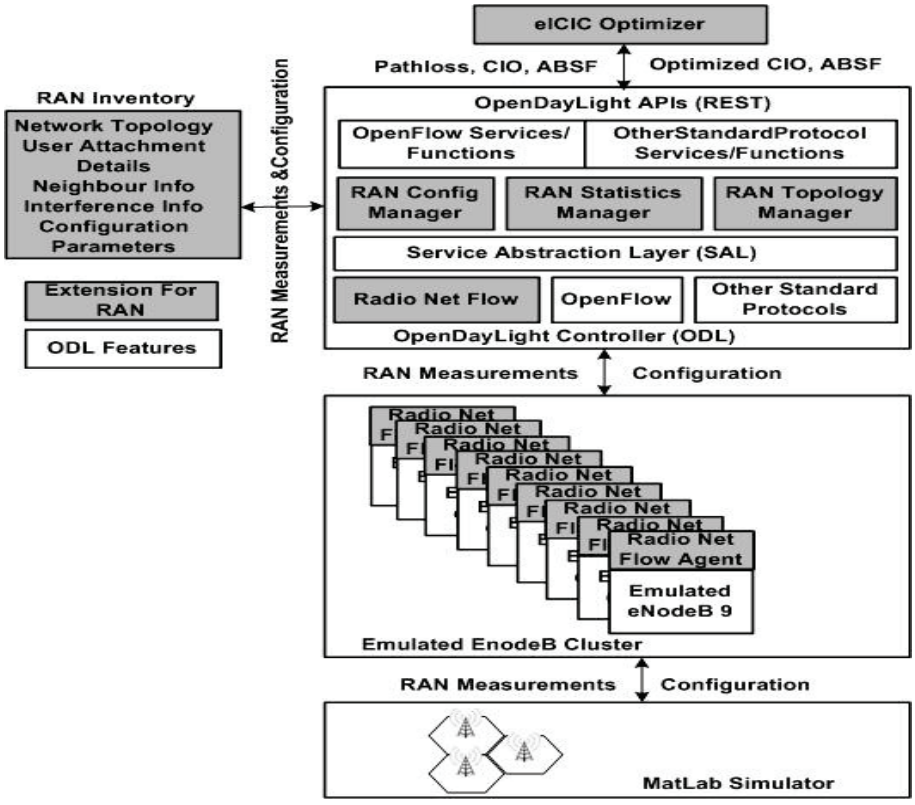


Figure 4.1 – Prototype for performance validation.

During optimization iterations, the Matlab simulator creates UDP sockets and sends any updated messages for each eNB: the CIO values, the ABS patterns, the attachment of mobile users, and the channel conditions (by the CQI from each user to its serving cell and neighboring cells). All these messages are then sent via SCTP from the eNBs to the ODL to be stored at the RAN Inventory. The eICIC optimizer therefore sends requests using REST APIs to retrieve the infor-

Radio	10MHz LTE Release 8 SISO, 3GPP TS36.942 -recommended pathloss and shadow fading models
Topology of eNBs	Hexagonal, a macro (MC) has 3 sectors & 6 SCs Antenna (MC/SC): Kathrein/omnidirectional Max Power (MC/SC): 40W/1W
Mobile users	25 UEs per MC, 10 UEs per SC, total 135 UEs, No mobility, Full buffer model
Optimization values	CIO choices =[0, 10, -10] ABS choices = [0%, 20%, 40%]

Table 4.1 – eICIC simulation parameters.

mation needed to perform optimization iterations. The parameters of the tested LTE HetNet for simulation are given in Table 4.1. Fig. 4.2 illustrates the network topology.

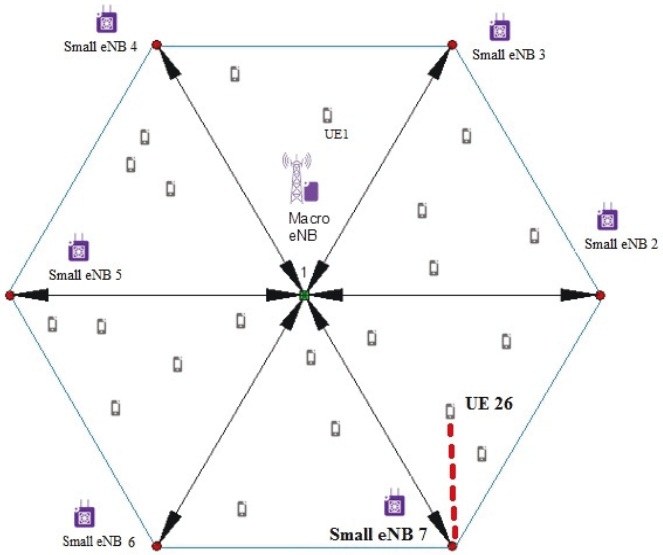


Figure 4.2 – Network topology.

The optimization delivers the best CIO values for each neighboring cell. For instance, before optimization, the CIO vector at eNB-7 (a small cell) for its three neighboring cells, i.e. eNB-6 (a small cell), eNB-1 (a macro cell) and eNB-2 (a small cell), in Fig. 4.2 is [0, 0, 0]. After eICIC optimization, the CIO vector becomes [0, -10, 0], which means a 10 dB of cell selection power bias favorable to attach to small cell eNB-7 when deciding a UE to attach to eNB-7 or attached to eNB-1 by comparing received signal strengths from these two cells. As a result, UE-26 which is attached initially to macro cell eNB-1 therefore handovers and associates with small cell eNB-7. Besides, after eICIC optimization, the ABS ratio of the macro cell increases from 0% to 50% which aims to reduce its interference to neighboring small cells and especially offer a higher throughput for cell edge users at small cells by macro cell muting.

Fig. 4.3 shows a substantial increase of the global average throughput in the system (bolded red curve) after the optimization. Thanks to the SDN controller, at each TTI, new CIO and ABS values are sent to the different eNBs to update the user attachment and inter-cell interference coordination and tune the system dynamically. Note that in the above optimization, optimization

are performed using a LTE proportional fairness policy for a balance of global throughput and service fairness.

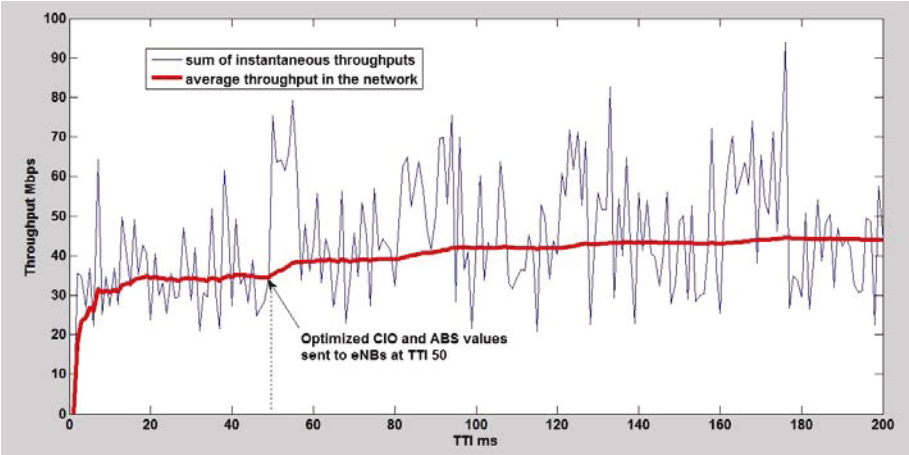


Figure 4.3 – Network throughput before/after eICIC optimization at TTI 50.

4.1.2 Prototype: Docker Container Based Virtualization

In order to compare the performance variation between two virtualization techniques, the same framework is validated in containers instead of virtual machines. As shown in Fig. 4.4, the SDN controller, the eICIC optimizer, the MongoDB and emulated eNBs are deployed in isolated Docker containers using Alpine Linux version 3.2 that is 5 MB in size built around the musl libc [127] and busybox [128]. In order to achieve the containerization of each entity in our architecture, separate docker-files are created along with the necessary libraries, includes Java 1.7, Maven [129], SCTP for the deployment of ODL controller, json [130], and gcc for eICIC optimizer which is developed using C++. We have used the MongoDB Docker file from Docker hub for the deployment of DB container in our platform. All the docker-files are built using docker daemon version 1.11 to produce corresponding docker images and stored them in the local repository before launching our experimentation testbed. During the experimentation, all the containers such as ODL controller, MongoDB, eICIC optimizer and emulated eNBs are launched using a single shell script. The instantiation of all the containers of our experimentation testbed appears very fast: it is less than one minute compared to the VM implementation that may take several minutes. The advantage

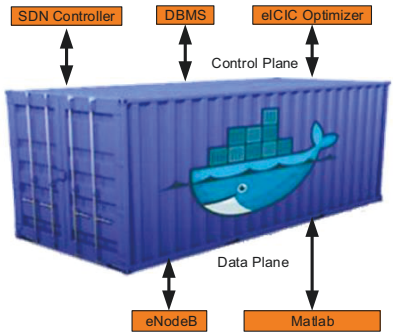


Figure 4.4 – Testbed using Docker containers.

of using docker containers is that a container virtualizes the kernel so that each container uses underlying kernel with its own process and network space which is light weighted and efficient as compared to VMs that run full operating system as shown in Fig. 4.5. Besides, the communication latency between Docker containers is much lower as compared to VMs which have hypervisor overheads. Processes running within containers are more elastic in nature as compared to those deployed within VMs which are more static in resource utilization. The entire booting time for our testbed with several Docker containers is of the order of seconds whereas the booting time with VMs is of the order of tens of seconds or even more than a minute. This is due to the fact that Docker containers share the single OS and require to load only the necessary software packages and libraries for particular processes whereas VMs require the entire OS to be loaded irrespective of the requirement of the deployed processes.

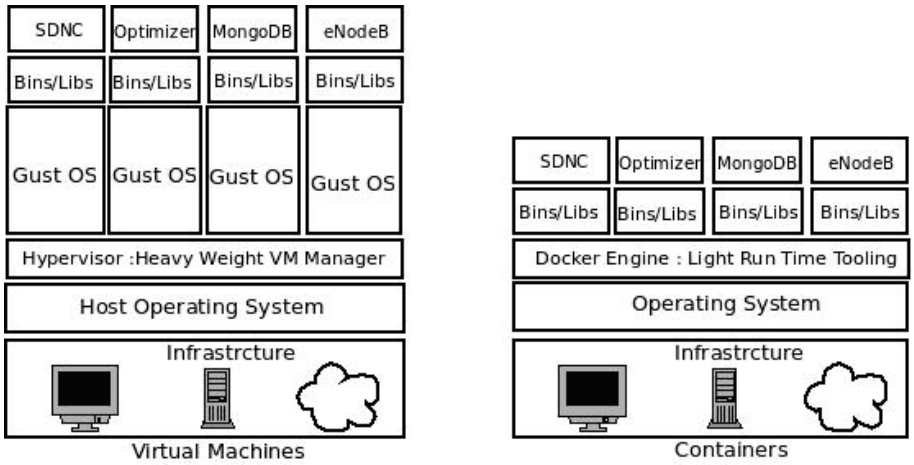


Figure 4.5 – Architecture: Virtual machines versus Docker containers.

4.1.3 Controller Queuing Model

According to the previously described architecture in Fig.4.1, the SDN controller manipulates the flows of information (measurements and configuration information) received from the eNBs. The flows are collected in a round robin manner. After the flows are collected and aggregated into a single flow, this flow is sent to the central entity. The size of the flow and consequently the load would depend on the number of eNBs for which the SDN controller conduct operations. The i-th flow is assumed to receive in conformity to a Poisson arrival process with parameter λ_i. Note that parameter λ_i stands for the expectation of the arrival rate of the packet-in messages at the controller.

In order to characterize the processing discipline of the SDN controller, the M/M/1 queuing model [131] is adopted. In a M/M/1 queuing model, the length of packet-in message queue at a SDN controller $N(t)$ is a birth and death process on the countable infinite state set $E = \{0, 1, 2, \ldots\}$. We denote its instantaneous probability by $p_j(t)$ and the stationary probability by p_j such that

$$p_j(t) = P\{N(t) = j\}, \tag{4.1.1}$$

and

$$p_j = \lim_{t\to\infty} p_j(t), j \in E. \tag{4.1.2}$$

Let ρ be the traffic intensity of the controller, which is given by λ/μ, where λ is the inter-arrival rate and μ is the service rate. According to Little's Law, the length of the waiting queues:

$$\overline{N}_q = \frac{\rho}{1-\rho}. \tag{4.1.3}$$

Notice that $p_0 = 1 - \rho$ is the probability for the controller to be in idle state, and ρ is the probability for the controller in busy state, where $0 \leq \rho \leq 1$. It is easy to observe that the controller becomes busier with the increase of ρ. The controller processes packets on a first-come-first-served basis. Therefore, the distribution of the waiting time of a packet-in message is given by:

$$W_{q(t)} = P\{W_q \leq t\} = 1 - \rho e^{-(\mu-\lambda)t},\ t \geqslant 0. \tag{4.1.4}$$

Using (4.1.4), the average waiting time of a packet-in message becomes:

$$\overline{W}_q = E[W_{q(t)}] = \frac{\rho}{\mu(1-\rho)}. \tag{4.1.5}$$

By adding the expected processing time of a packet, which is given by $1/\ \mu$, we have the sojourn time of a packet in the controller is given by:

$$\overline{W}(t) = \overline{W}_q + \frac{1}{\mu} = \frac{1}{\mu-\lambda}. \tag{4.1.6}$$

From (4.1.6), the processing rate is:

$$\mu = \frac{1}{\overline{W}(t)} + \lambda. \tag{4.1.7}$$

The above result allows to estimate the processing rate in the testbeds by mapping the behavior of the testbeds to the queuing model and the measurements to metrics in terms of inter-arrival rate and average waiting time.

4.1.4 Performance Comparison

Numerical Evaluation

Fitting the Model to the Measurements: The measurements are performed in the two testbeds described in Section 4.1.1 and 4.1.2. Fig. 4.6 illustrates the inter-arrival time of the packets collected from the eNBs served by the SDN controller. The number of eNBs are increased gradually. Results show a significant difference in the behavior of the two testbeds. It is due to the fact the hardware resource of the physical machine that is virtualized using VMs would easily get over-loaded with the increase in the number of virtual nodes. The VM-based architecture has the intrinsic nature that requires more resources to deploy the whole operating system stack comparing to docker containers that run only the necessary packages and libraries on top of the existing kernel. As a result, the communication latency between docker containers is significantly lower as compared to the one of VMs which has one extra layer of hypervisor, that generates additional delay during the data arrival to the centralized controller. The non-linearity of the VM testbed is due to the extra time required by the VM platform instantiation. Then, the average waiting time is measured for the two testbeds. Note that this waiting time refers to $\overline{W_q}$ in (4.1.5). Fig. 4.7 shows the experimental results. As expected, in both testbeds the average waiting time increases with the number of eNBs due to the increase of the load results in a request of more physical resources. However, the docker container architecture shows clearly its superiority over the VM.

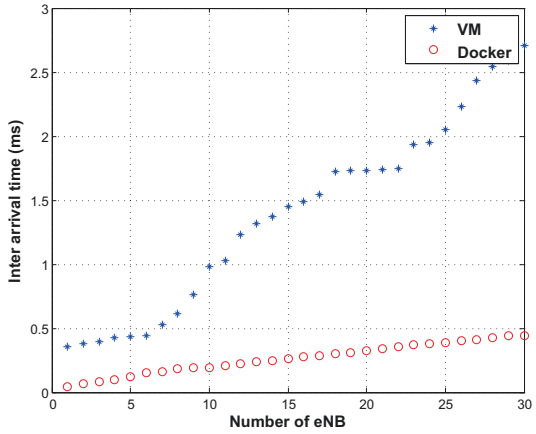


Figure 4.6 – Inter-arrival time of packets under VM architecture and Docker architecture.

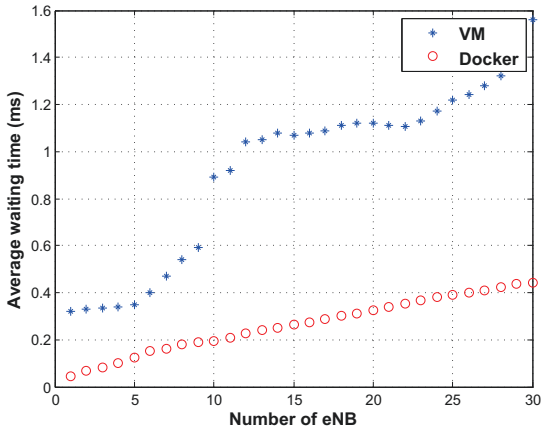


Figure 4.7 – Average waiting time for the VM-based architecture and for docker-based architecture.

VM architecture vs docker architecture service processing time: In order to estimate the processing time required for the eNBs information update, an analytical model is described in Section 4.1.3, that illustrates the behavior of the two testbeds. Expressing the processing rate by (4.1.7) and using the measured inter-arrival rate (see Fig. 4.6) and waiting time (see Fig. 4.7), the service rate for the VM and docker based testbeds can be obtained in order to plot the results. The Fig. 4.8 shows that both the average service rates of docker container testbed and VM testbed decrease with the number of virtual nodes running in the same physical system. However, the performance using docker containers is always higher than that of VMs due to its lightweight and lower resource occupancy in achieving the same amount of tasks. Note that the docker container testbed shows a very high service rate when the system is started for a small number of eNBs. This behavior is a consequence of the container's virtually zero performance overhead and fast kernel resource allocation. However, the VM testbed's performance is nearly flat since for any number of eNBs the testbed loads the whole operating system with large overhead and the resource allocation is comparatively slow.

4.1.5 Discussions and Conclusion

In the proposed SDN architecture, we have deployed three different entities in the control node: the controller, the database, and the application. The motivation for using the centralized database

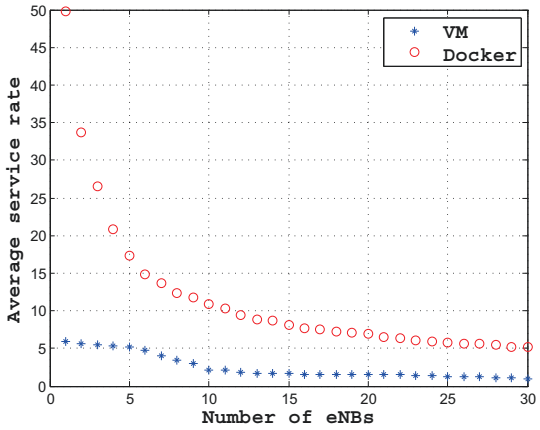


Figure 4.8 – Service rate for the VM-based architecture and for Docker-based architecture.

as an inventory rather than using the XML-based yang data model in ODL is to generate historical data as an input for data analytic tools and applications, which can facilitate big data analytics [132] so as to extract information for future 5G intelligent end-to-end mobile network management. The implementation of REST APIs for north bound (NB) application enables the deployment of RAN applications in remote servers which can be developed using various programming languages. The southbound (SB) protocol plug-in RNF is implemented using SCTP because SCTP has the unique feature of creating multiple and parallel message streams between two end points which makes SCTP robust in nature in comparison to TCP and UDP [96].

This work presents two implementations of Software Defined RAN architecture, one using Virtual Machines and another one using the recently introduced technology of dockers. The architecture is described using an analytical model that is used to derive the service time of the two deployment options such as VMs and Dockers. Measurements and computations have shown that the docker-based architectures provides better service rate than the Virtual-Machine-based architecture. This is helpful for the future development of Software-Defined Mobile Networks. Though our SDN platform is primarily developed for supporting LTE SON methods, it can be further extended to support different use cases in Cloud RAN and future 5G for dynamic resource allocation and traffic load balancing between different base-band units (BBUs). Note that that the analytical method introduced here could be also useful for studying other similar systems such as SDN for Cloud RAN.

4.2 SDN For Wireless Backhaul Networks: Prototype, Implementation and Validation

4.2.1 Software Prototype

In order to perform initial validation of our architecture, a software prototype is implemented as described in Fig.4.9, using ODL controller, MongoDB, Open vSwitches, and Wireless Modem emulator from Simpulse [125]. Two PCs having Linux Ubuntu distribution with low latency kernel 3.13 running on Intel i5 at 3.2 GHz and 8GB RAM are used for the experimentation. The control-plane components such as SDN controller, applications and DBMS are deployed in one dedicated PC, and data-plane nodes such as modems and Open vSwitches in another PC as shown in Fig.4.9. OpenDayLight is chosen for the experimentation, since it supports both OF and OVSDB protocols

together and provides flexible framework to integrate any new protocol plugins. The choice of MongoDB is due to high performance as compared to SQL-based DBMS systems [133]. The MCMP plugin is implemented using ZeroMQ [134] messaging library in the SB of ODL to exchange LLR messages between Infra Modems and the mobility control applications in the NB of the controller. Set of new service modules are integrated in the controller for MCMP management. The introduction of MCMP plugin in ODL required necessary modifications to both Service Abstraction Layer (SAL) and NB API layer to provide necessary interfaces for NB applications. Our modem emulator software is able to emulate any number of Infra and User modems. The SDN-Agent is implemented Modem APIs to interface Radio Modems to the SDN environment (SDN controller, OVS).

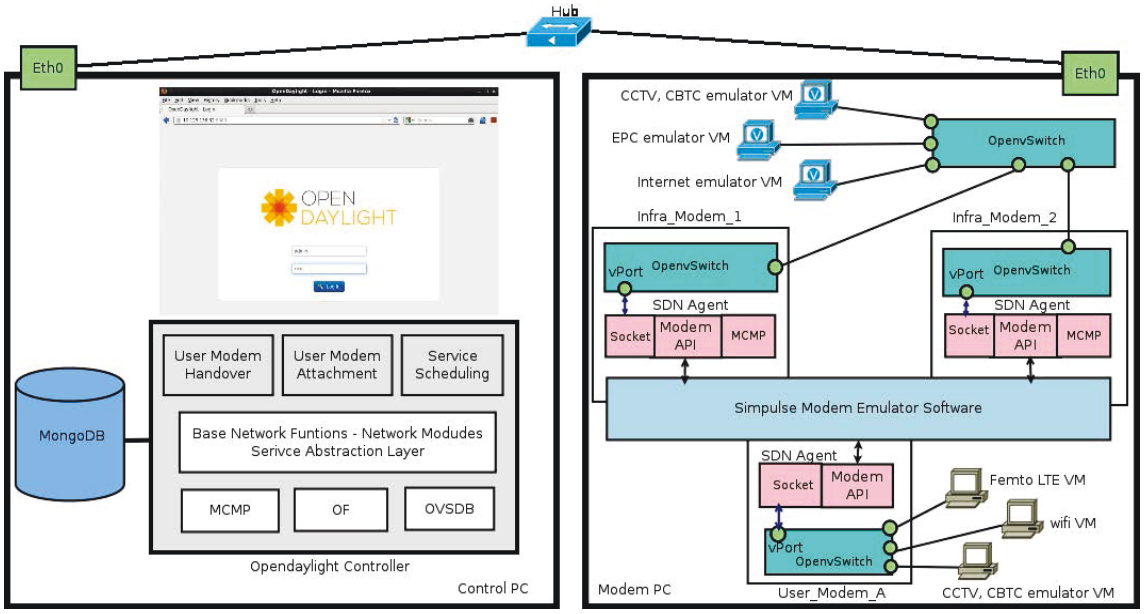


Figure 4.9 – Software Prototype Architecture.

For the demonstration of the mobility management scenario (User Modem Attachment and User Modem Handover procedures), two Infra Modems and one User Modem are emulated as in Fig.4.9. Three Open vSwitches are integrated with emulated modems using our SDN-Agent module. Open vSwitch deployed at the edge of the network acts as an Aggregation Switch (AS) which connects wireless backhaul network and networks of service providers in the ground. In order to emulate the end-to-end traffic, Virtual machines are attached to the AS and to the User Modem. As a first step towards validation of the framework, applications managing User Modem Attachment and User Modem Handover are deployed in the NB of the controller. All the information regarding UM Attachment and UM Handover such as the ID of UM and the received power level are stored in the DB and the NB applications can access them through APIs in the NB of the controller.

According to our scenario, the initial network configuration looks as in Fig. 4.10(a), where two Infra Modems are connected to Aggregation Switch and attached to the controller. Also, three Virtual Machines are connected to the Aggregation Switch to represent three different services such as Internet, LTE core network, and Transport Control Systems (CCTV, CBTC). Initially, the IMs started to broadcast and listen for new UMs to be attached. As defined in our validation scenario, after five seconds the emulated UM started to send attachment request to the Infra Modem. As described in Fig. 4.10(b), the UM attachment procedure is completed in 3 ms and we validated it by successfully sending ICMP messages between VMs connected to the UM and

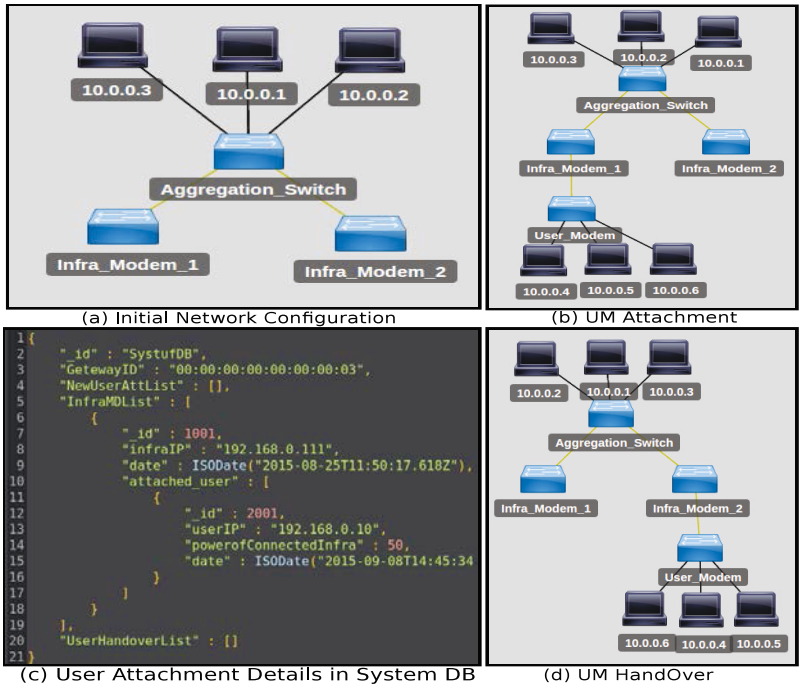


Figure 4.10 – Software Prototype Scenario.

to the Aggregation Switch. The user attachment details in system DB is shown in Fig. 4.10(c). After 1 minute, the handover procedure begins when the UM started to attach with the IM-2. The handover application begins to execute the procedure when the power received by UM from IM-2 is higher than that from IM-1. As shown in Fig. 4.10(d), the handover procedure is completed in 6 ms. ICMP messages between VMs connected UM while VMs connected to Aggregation Switch continuously flows but through IM-2 instead of IM-1 after handover. Hence by this process both the attachment and handover procedures are validated.

4.2.2 Hardware Prototype

The software prototype is then migrated to hardware by replacing modem emulators with embedded software-defined radio modems, and real time service via LTE femto software. The architecture of the hardware prototype is depicted in Fig. 4.11. The complete hardware-based prototype setup of our proposed architecture consist of an ONOS SDN controller with necessary extended features for mobile backhaul, software-defined radio modems implemented on Pulsar board, OpenFlow-based ALU omni switch 6800, ALU EPC box and LTE femto (OpenAirInterface) from Eurecom. OmniSwitch 6800 is used as an Aggregation Switch, which has high performance and extremely low latency. Since these switches also support open flow protocol, the network between core network and Inframe Modems is controlled by the SDN controller for routing and dynamic mapping of flows between ground services and Infra modems during user attachment and handover procedure.LTE femto and webcam client is depicted as service use cases in the train. In the hardware architecture we used ONOS SDN controller due to improved performance in the usage of OpenFlow protocol and well defined graphical user interface. The SDN-Agent module and SDN applications from software prototype implementation are reused for the hardware prototype. The hardware architecture is validated with the use case of User-Attachment along with LTE-Femto as a service in the train. The validation architecture as shown in Fig. 4.14 represents the user

attachment scenario in the demonstration platform, where EPC (LTE core network) is connected to LTE femto in the train using Software-defined wireless backhaul network. The OAI femto establish a connection to the use via the USRP+LTE dongle setup.

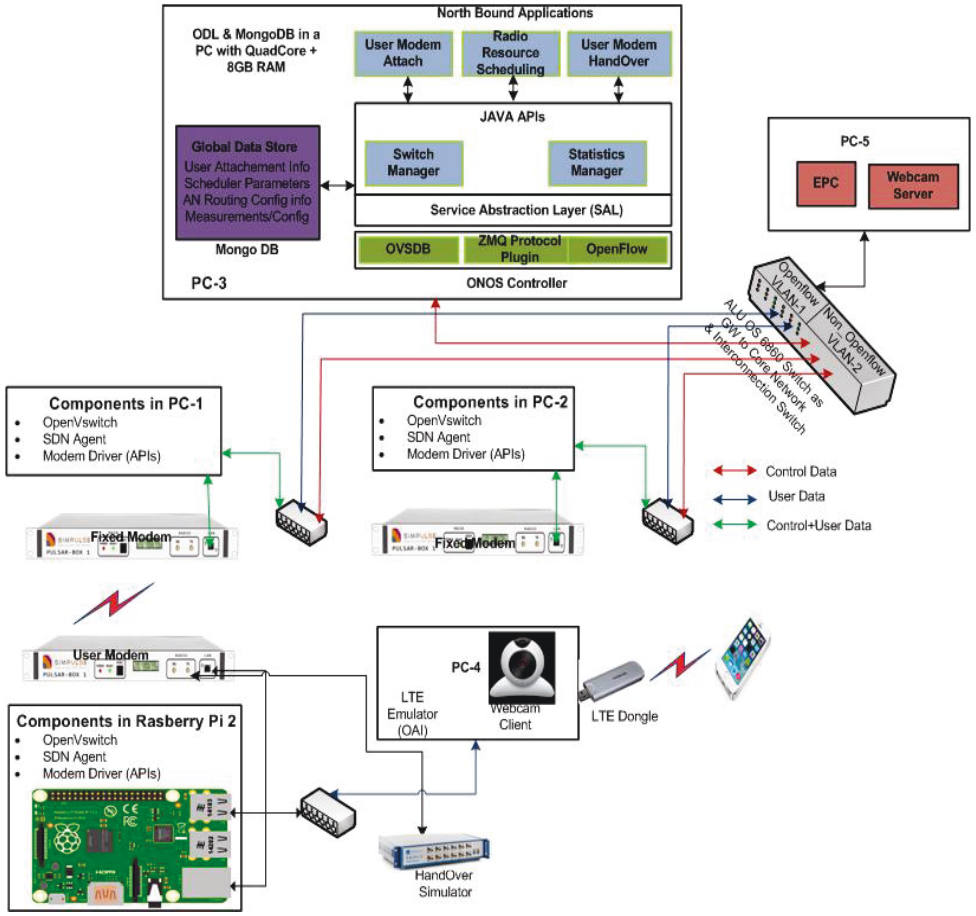


Figure 4.11 – Hardware Prototype Architecture.

Software-Defined Radio Modem on Pulsar Board

As shown below in Fig. 4.12, the software-defined radio modem from simpulse is integrated to the SDN controller via SDN-agent and an Open vSwitch instance deployed in the raspberry pi. SDN-Agent uses the exposed APIs of Radio Modem. Both in Infra and User Modem uses mini helical omni directional antennas operate at the frequency of 2.4 GHz with 2 dB gain.

OpenAirInterface Femto and Nokia EPC Box

The OAI-based Femto includes a full software implementation of 4th generation mobile cellular systems compliant with 3GPP LTE standards under realtime Linux optimized for x86. User PC (UE) has a dongle modem that interconnects the OpenAirInterface (OAI) softmodem software through USRP B210 platform as shown in Fig. 4.13. Nokia Bell Labs EPC Box software, which is the LTE complaint core network software runs in a stand-alone PC is connected with OAI softmodem to have end-to-end mobile network connection.

Figure 4.12 – Radio Modem - Hardware Prototype.

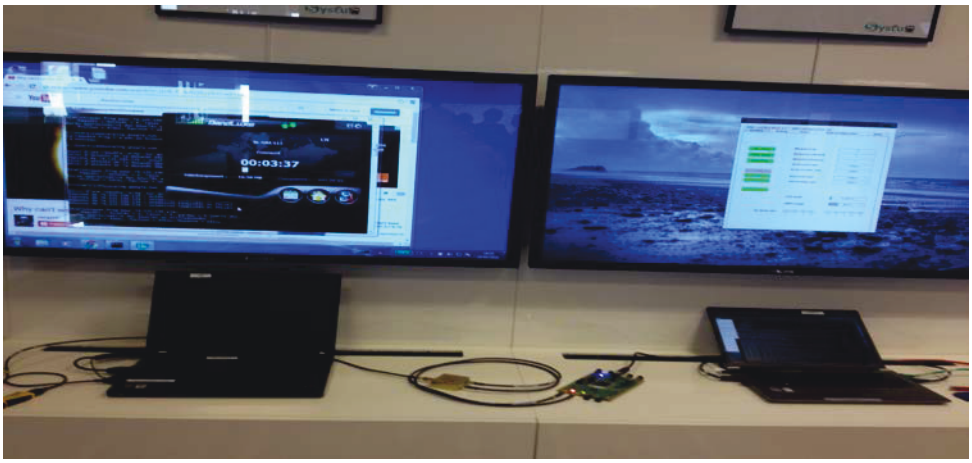


Figure 4.13 – OpenAirInterface Based Femto Setup.

Demonstration of Hardware Prototype

The ONOS SDN controller have a list of default applications for the network management and that can be initiated during the start up procedure. As shown in Fig. 4.15, the ONOS controller activates those applications related to the network management using the OpenFlow protocol. Since the user attachment application is not initiated and the SDN controller does not have the provision to manage the mobile backhauling, the User Modem is not attached to the Infra Modem as shown in Fig. 4.16. At this point, the SDN controller connects to the OVS of the Infra Modem only and does not have applications to handle the user-attachment related messages coming from the SDN-Agent of the Infra Modem. The internet service connection is attached to the OVS of the Infra Modem. Hence the SDN controller discovers all the other connected hosts on the network. By default three flows are installed on the OVS due to the openflow and drivers application responsible for the default flow configuration. In the next step of the validation process, The User-Attachment application is deployed in the running instance of the ONOS SDN controller. The application package is "User Modem Attachment App". After the successful deployment and activation of the application, the User Modem Attachment application can be viewed from the list of applications in the ONOS app store. Fig. 4.17 shows the list of all the applications available in the store

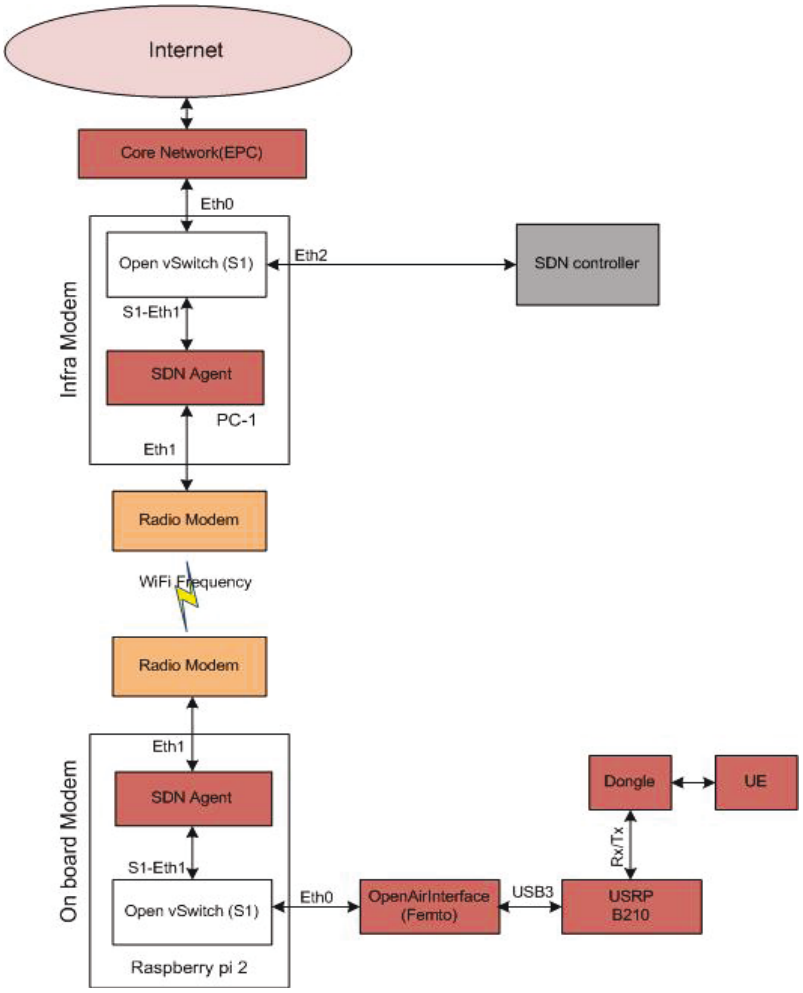


Figure 4.14 – Network Architecture of the Hardware Prototype Demonstration.

with eight applications activated (ticked in green). The Fig. 4.18 shows the appearance of the User Modem few seconds after the deployment of "User Modem Attachment Application". The IM is constantly broadcasting and discovers new User Modem within its broadcasting range and when it receives the attachment request from User Modem, it sends the request to SDN controller for authentication and attachment. As a first step soon after the User Modem attachment, all end users connected to the User Modem start to discover their service network and get allocated with IP address. The application in the controller responsible for IP address allocation is "Proxy ARP/NDP App". Fig. 4.19 displays the detection of the end user on the train and provisioned with internet services from the Mobile Backhaul i.e. infrastructure side. It can also be observed from the Fig.4.19 that five intents have been installed i.e. flow rules installed on the physical devices, OVS on the SDN-Agents to allow the user to have access to internet in the backhaul.

4.2.3 Software-Defined Wireless Backhaul As a Service

In the next step of the prototype implementation, the software-defined wireless backhauling framework is integrated into the docker-swarm orchestration framework. Both data and control plane functions of software-defined wireless backhaul framework are containerized. This enables each functions to be independent from each other and can be deployed and chained by the orchestration frame work. This falls under the micro-services approach in software engineering to

Figure 4.15 – The ONOS Command Line Interface.

Figure 4.16 – ONOS GUI-Before the User Attachment.

develop an application as a set of small independent services. Each service is composed of self and independent processes that communicate with each other using very light mechanisms like HTTP. Also, the centralized management of those services are again another independent and completely separate service[135]. As shown in Fig. 4.20, docker swarm[136] is used as an orchestration framework to deploy micro-services that implements the software defined mobile back-hauling features. In order to achieve this, the User modem components includes OpenvSwitch and modem APIs

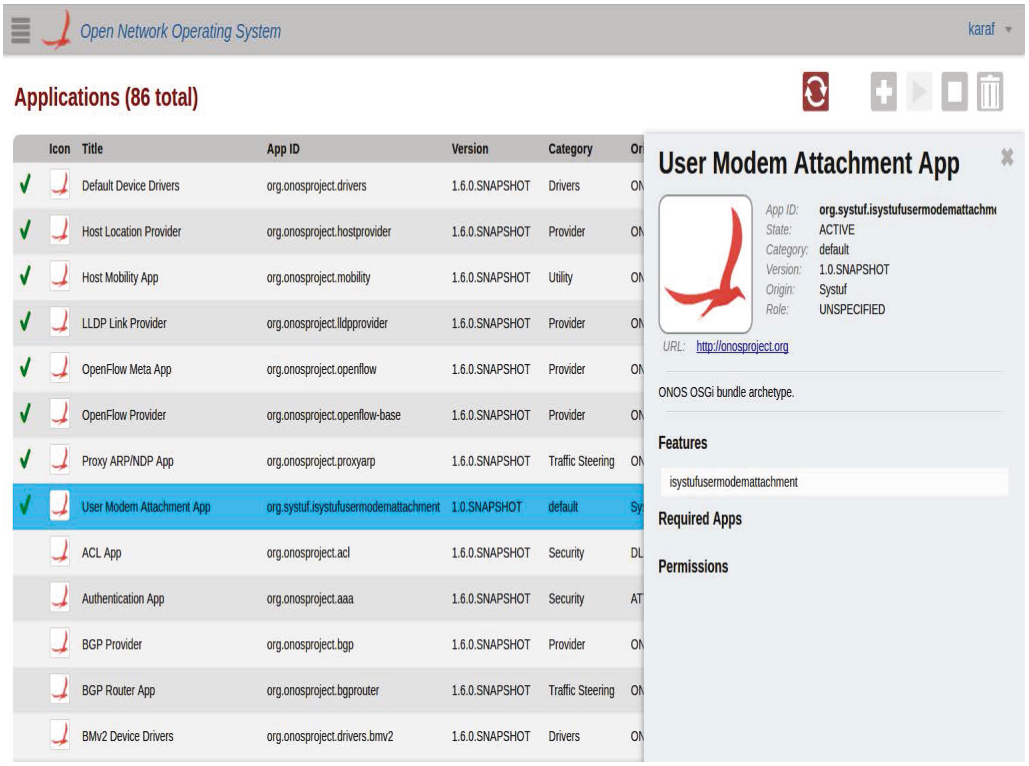


Figure 4.17 – ONOS GUI-User-Attachment Application deployment.

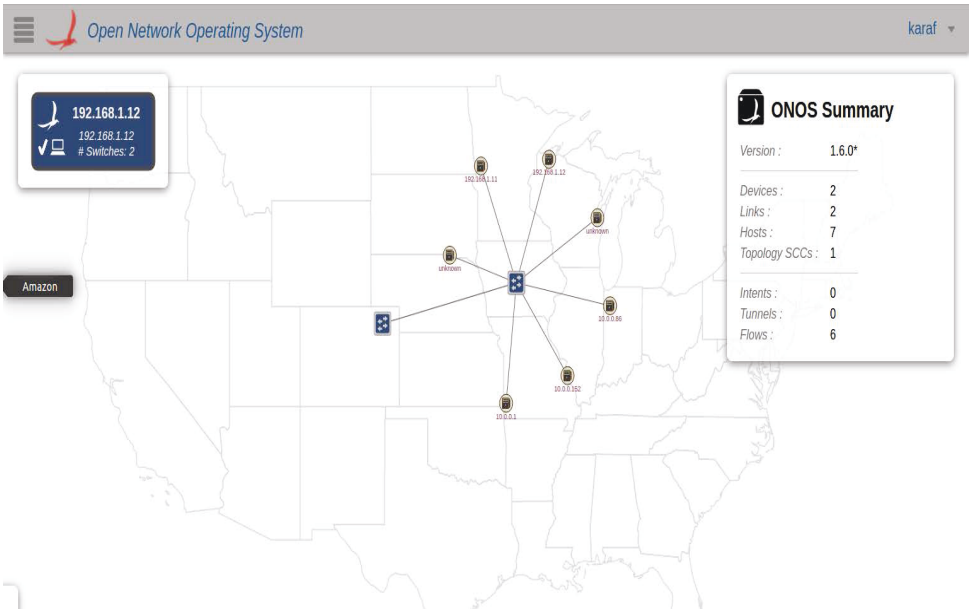


Figure 4.18 – ONOS GUI-After the User Modem Attachment.

are containerized as a docker containers and stored in the local docker hub. Also the ONOS SDN controller is containerized as a docker component and stored in the local docker hub. For the simplicity of the experimentation, there is only one client connected to the User Modem. The Infra modem is directly attached to the broad band service network. Initially the client attached to the User Modem will not have any connectivity to the network because the User Modem is not attached to the Infra Modem or in other words there is no mobile back-hauling network activated

Figure 4.19 – ONOS GUI: After end-to-end network connectivity for Attached Users.

to establish a connection between the end user and the service network. By default, the User Modem is always running and listening for any Infra Modem in its frequency range for connection. As a first step, using the docker swarm User Interface, ONOS SDN controller and the Infra Modem are deployed using containers in the targeted hardwares i.e. PCs. Upon deployment, the ONOS SDN controller and Infra Modem started to listen for new connectivity. The User Modem as soon as receives the broadcasting signal from the Infra Modem, it sends connection request to the Infra Modem and the Infra Modem transmit those request to the ONOS controller for attachment procedure to complete. The ONOS controller then sends initial user attachment procedure completion to the Infra Modem to give access to the particular User Modem who has sent the connection request. Up on completion of the User Attachment procedure, the client attached to the User Modem sends DHCP request via User Modem to the Infra Modem for acquiring an IP address for communication. Soon after the acquiring an IP address, the client starts to send user data over the mobile back-hauling framework. At this point, the OpenFlow protocol comes into playing its role in flow configuration in Open vSwitches located both in Infra and User Modem. The ONOS reactive forwarding application is responsible for the configuration of data flows across the mobile back-hauling network. We were able to stream online video as a test use case over the mobile back-hauling framework upon deployment. The approach of integrating software-defined wireless backhauling with the orchestration framework is a way to deploy wireless backhaul network on demand from users.

In the next step of the validation process, The User-Attachment application is deployed in the running instance of the ONOS SDN controller. The application package is “User Modem Attachment App”. After the successful deployment and activation of the application, the User Modem Attachment application can be viewed from the list of applications in the ONOS app store. Fig. 4.17 shows the list of all the applications available in the store with eight applications activated (ticked in green).

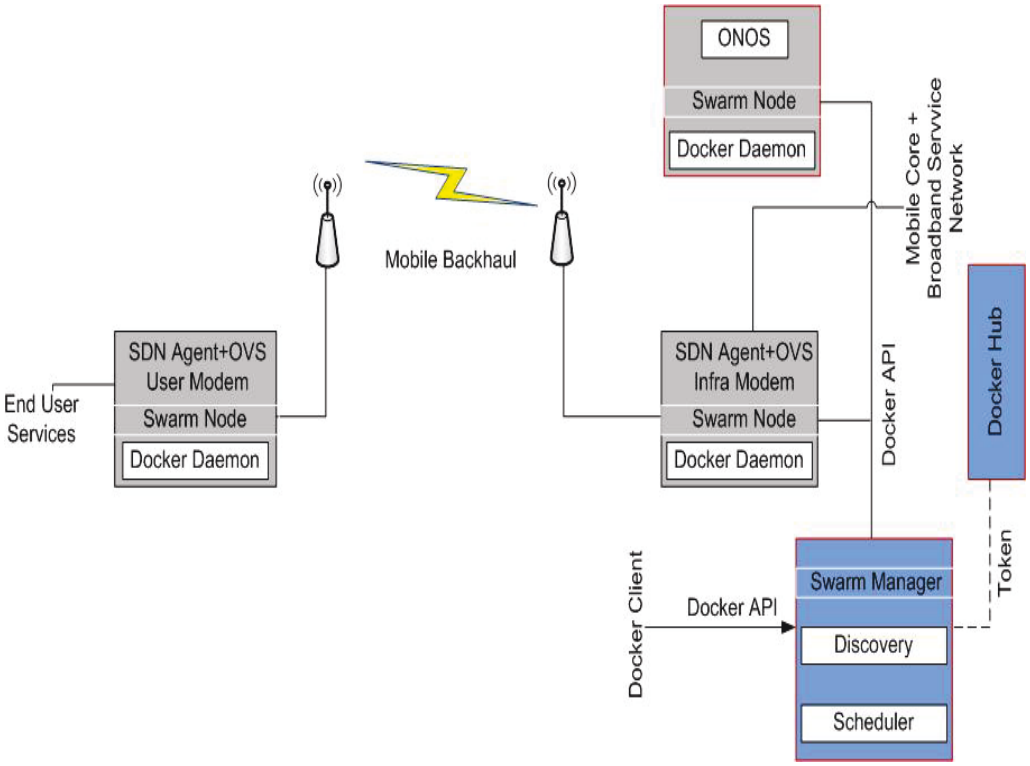


Figure 4.20 – Software-Defined wirless backaul-as-a-Service.

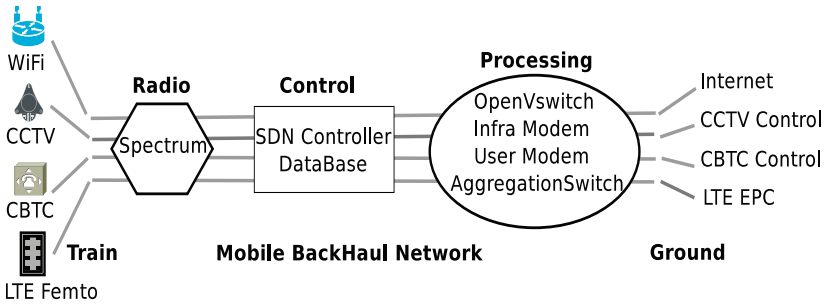


Figure 4.21 – Network Slicing in wireless backhaul.

4.2.4 Discussions and Conclusion

Although there are various solutions to provide high speed communication services for transport systems, they cannot offer guaranteed QoS for certain services in dynamically varying traffic conditions. In this work, we introduced the concept of SDN in the train-to-ground communication systems in order to bring programmability in the mobile backhaul. As depicted in Figure. 4.21, the primary goal of this work is to slice end-to-end backhaul network resources among different services based on their level of priority and QoS requirements. The proposed framework enables the delivery of network as a service on-demand and tailors resources accordingly. The main advantages from our SDN based framework especially dynamic programmability and reconfigurability enable mobile backhaul to have mobility management driven directly from locally deployed SDN controller rather than from dedicated mobility management entity deployed in core networks as in traditional wireless systems. Taking the advantages from OpenFlow and OVSDB protocols, we are also able to efficiently efficiently manage scarce radio resources using pre-scheduling via Open

vSwitches integrated with Radio Modems. We have implemented our architecture in a software as well as hardware prototype and validated scenarios related to mobility management. In the hardware based prototype the modems can make a mobile backhaul connectivity with data rate up to 10 Mb/s completely controlled by SDN controller. We have demonstrated the complete user attachment procedure along with LTE-Femto supporting live video transmission in downlink.

Chapter 5

Conclusion and Future Work

In the recent years, Mobile Networks have made significant growth in terms of technological evolution and number of users. The 5G mobile network is being seen as a user-centric concept instead of service centric as in 4G. The 5G wireless network should enable the development and exploitation of massive capacity and connectivity of complex and powerful heterogeneous infrastructures. The diverse use cases and requirements of 5G pushes mobile network performance and capabilities to their extremes. Although there is no standard description or specification for next generation networks, current research activities claim that 5G will be re-configurable and programmable end-to-end network built on top of a cloud infrastructure that provides processing power on-demand for end-user applications.

Although current SDN controllers support programmability and flexibility for fixed networks, they still lack or have no definitive provisioning for mobile networks. None of the main stream SDN controller frameworks provide control as well as programmability for RAN. Currently, the southbound protocols in the controllers are specified for managing only transport networks e.g., IP routing, MAC switching, VLAN management, etc. OpenFlow, the prominent SDN protocol, is realized so far only in the fixed network and to the best of our knowledge there is no implementation of standard SDN protocols to manage and program wireless networks. Also, NFV technologies which are developed for the virtualization of resources in the transport networks require optimization and performance improvement to cope up with the stringent latency requirements of mobile networks. The realization of SDN/NFV paradigm for wireless networks requires the definition, the implementation and the validation of such architectures using several use cases.

In this thesis, we proposed Software-Defined architectures for three different use cases of next generation wireless networks such as Self-Organizing Networks, Dynamic User Processing Split, and Wireless Backhaul. Through out the thesis, our main focus was to address challenges related to capacity expansion, QoS improvement, flexibility and system re-programmability. In the first part of our research, we presented two implementations of Software-Defined RAN architecture for Self-Organizing Networks, one using Virtual Machines and another one using the recently introduced dockers. The architecture is described using an analytical model that is used also to derived the service time of the two deployment options such as VMs and Dockers. Measurements and computations have shown that the docker-based architecture provides better service rate than the Virtual Machine-based architecture. This is helpful for the future development of Software-Defined Mobile Networks. Although our SDN platform is primarily developed for supporting SON management, it can be further extended to support different use cases in Cloud RAN and future

5G for dynamic resource allocation and traffic load balancing between different base-band units (BBUs). Note that we also expect the analytical method introduced here to be also useful for studying other similar systems such as Cloud RAN.

In the second part of our research, our focus was to bring programmability in the wireless backhaul networks especially in the train-to-ground communication network that often failed to adapt itself to stringent QoS requirements for time sensitive devices on board (e.g. automatic train control system). Although there are various solutions to provide high speed communication services for transport systems, they can not offer guaranteed QoS for certain services in dynamically varying traffic conditions. The proposed framework enables the delivery of Network as a Service on-demand and tailors resources accordingly. The main advantages from our SDN-based framework is to provide dynamic programmability and reconfigurability for wireless backhaul to have mobility management driven directly from locally deployed SDN controller rather than from dedicated mobility management entity deployed in core networks as in traditional wireless systems. Taking the advantages from OpenFlow and OVSDB protocols, we are able to manage efficiently the scarce radio resources using pre-scheduling via OpenvSwitches integrated with Radio Modems. We have implemented our architecture in a software as well as hardware prototype and validated scenarios related to mobility management. In the hardware based prototype the modems can make a mobile backhaul connectivity with data rate up to 10 Mb/s completely controlled by the SDN controller. We have demonstrated the complete user attachment procedure along with LTE-Femto supporting live video transmission in downlink. In the final part of the thesis, we analysed the various functions and protocols in the LTE architecture as a preliminary step towards understanding the benefits along with the problems that cloud RAN can address. we summarized the state-of-the-art cloud RAN architectures, advantages and challenges. As a final step, we positioned the software-defined networking approach to cloud RAN and listed possible architectures and benefits. We also discussed those architectures in details along with message sequence charts describing virtualization of user processes, overload management and resource recovery during failure situations.

Although we demonstrated and validated the possible SDN/NFV-based architectures and solutions for future Radio Access Networks, the consideration of such approach in practical deployments of large scale networks require improvements in terms of performance and scalability. Since SDN/NFV technologies are basically developed in the scope of transport networks, the implementation of such a solution in mobile networks requires further optimization of their functionalities using modern software engineering methodologies. There are some studies focusing on the scalability of wired SDN [81, 137], but it is difficult in the wireless scenario. As the size of a network enlarges, more packets are sent to the controller. There is no doubt that the controller can hardly handle all these incoming requests [81]. Simply improving the performance of the sole centralized controller, without designing from the architecture level, can impossibly adapt to the wide-ranging and increasing dense network scale. Therefore, SDWN might consist of multiple controllers physically distributed in the system. These controllers will not conflict with the "logically centralized" principle through communicating and cooperating with each other efficiently. Also the future SDN protocol implementations should focus on bringing hardware abstraction for managing embedded cloud paradigm [138]. In the future work, we will optimize the functionality of SDN and NFV tools and softwares to be in the performance criteria of large-scale mobile networks that supports diverse use cases. We will also validate the proposed SDN architectures for Cloud RAN. The Artificial Intelligence is the key driver for bringing knowledge-based decisions in the 5G networks. This necessitates the integration of such intelligence in the future SDN controllers and NFV tools

that should re-configure the networking devices and computing functions to better satisfy the user needs in terms of QoS and QoE.

List of Abbreviations

ACP Automatic Cell Planning
AFP Automatic Frequency Planning
ANR Automatic Neighbor Relation
API Application Programming Interface
ABS Almost Blanked Subframe
AD-SAL Application-driven Service Abstraction Layer
AS Access Stratum
BBU Base Band Unit
CAPEX Capital Expenditure
CoMP coordinated multiple point transmission
C-RAN Cloud Radio Access Networks
CCO Coverage and Capacity Optimization
CS Coordinated Scheduling
CB Coordinated Beamforming
CIO Cell Individual Offset
CBTC Communications-based train control
CCTV Closed-circuit television
CQI Channel Quality Indicator
CMI Connection Management and Instantiation
CPin Cyclic Prefix insertion
CPout Cyclic Prefix Out
CPRI Common Public Radio Interface
DL Downlink
DPS Dynamic Point Selection
DBMS Database Management System
DRAN Distributed Radio Access Network
DRB Data Radio Bearer
ERTMS European Rail Traffic Management System
EPC Evolved Packet Core
eICIC Enhanced Inter-Cell Interference Coordination
FIB Forwarding Information Base
FPGA Field-Programmable Gate Arrays
FEC Forward Error Correction
GUI Graphical User Interface
GTP GPRS Tunneling Protocol
GSM Global System for Mobile Communication
HTTP Hypertext Transfer Protocol

ICIC Inter-Cell Interference Coordination
IaaS Infrastructure-as-a-service
ICMP Internet Control Message Protocol
IM Infra Modem
ITS Intelligent Transportation System
KPI Key Performance Indicators
LTE Long Term Evolution
LLR Low Level Radio
MAC Medium Access Control
MVNO Mobile Virtual Network Operator
MIMO multiple-input multiple-output
MRO Mobility Robustness Optimization
MD-SAL Module-driven Service Abstraction Layer
MME Mobility Management Entity
MSF Measurement Supporting Function
NFV Network Function Virtualization
NMS Network Management Systems
NIC Network Interface Card
NB North Bound
NCL Neighbor Cell List
NAS Non-Access Stratum
NGMN Next Generation Mobile Networks
OAM operations, administration and maintenance
OPEX Operational Expenditure
ONF Open Networking Foundation
ODL OpenDaylight
ONOS Open Networking Operating System
OS Operating System
OVS Open vSwitch
OVSDB Open vSwitch Database Management Protocol
OAI OpenAirInterface
OF OpenFlow Protocol
OF-CONFIG OpenFlow Management and Configuration protocol
PBB Provider Backbone Bridging
PaaS Platform-as-a-Service
PRB Physical Resource Block
PMR Private Mobile Radio
PIS Passenger Information Systems
PDCP Packet Data Convergence Protocol
PGW Packet Data Network Gateway
QoE Quality of Experience
QoS Quality of Service
QAM Quadrature Amplitude Modulation
RRC Radio Resource Control
RRH Remote Radio Head
RU Remote Unit
RAN Radio Access Network

RNTP Relative Narrowband Transmit Power
RNS Radio Network Subsystem
RNF Radio Net Flow
RNFA Radio Net Flow Agent
RF Radio Frequency
RLC Radio Link Control
RBC Radio Bearer Control
RRM Radio Resource Management
RAC Radio Admission Control
RPC Resource Pooling Control
SGW Serving Gateway
SDN Software Defined Networking
SaaS Software-as-a-service
SAL Service Abstraction Layer
SINR Signal to Noise Ratio
SCTP Stream Control Transmission Protocol
SON Self Organizing Network
SDR Software Defined Radio
SB South Bound
SQL Structure Query Language
TDD Time Division Duplex
TLS Transport Layer Security
TAP Tracking Area Planning
TAI Tracking Area Identifier
TAO Tracking Area Optimization
TAU Tracking Area Updates
TTI Transmission Time Interval
TETRA Terrestrial Trunked Radio
TCP Transmission Control Protocol
TE Traffic Engineering
UMTS Universal Mobile Telecommunications System
UL Uplink
UHF Ultra High Frequency
UDP User Datagram Protocol
UM User Modem
VLAN Virtual LAN
VNF Virtual Network Function
VTN Virtual Tenant Network
VM Virtual Machine
WLAN Wireless Local Area Network
WiMAX Worldwide Interoperability for Microwave Access
ZMQ ZeroMQ

List of Demonstrations

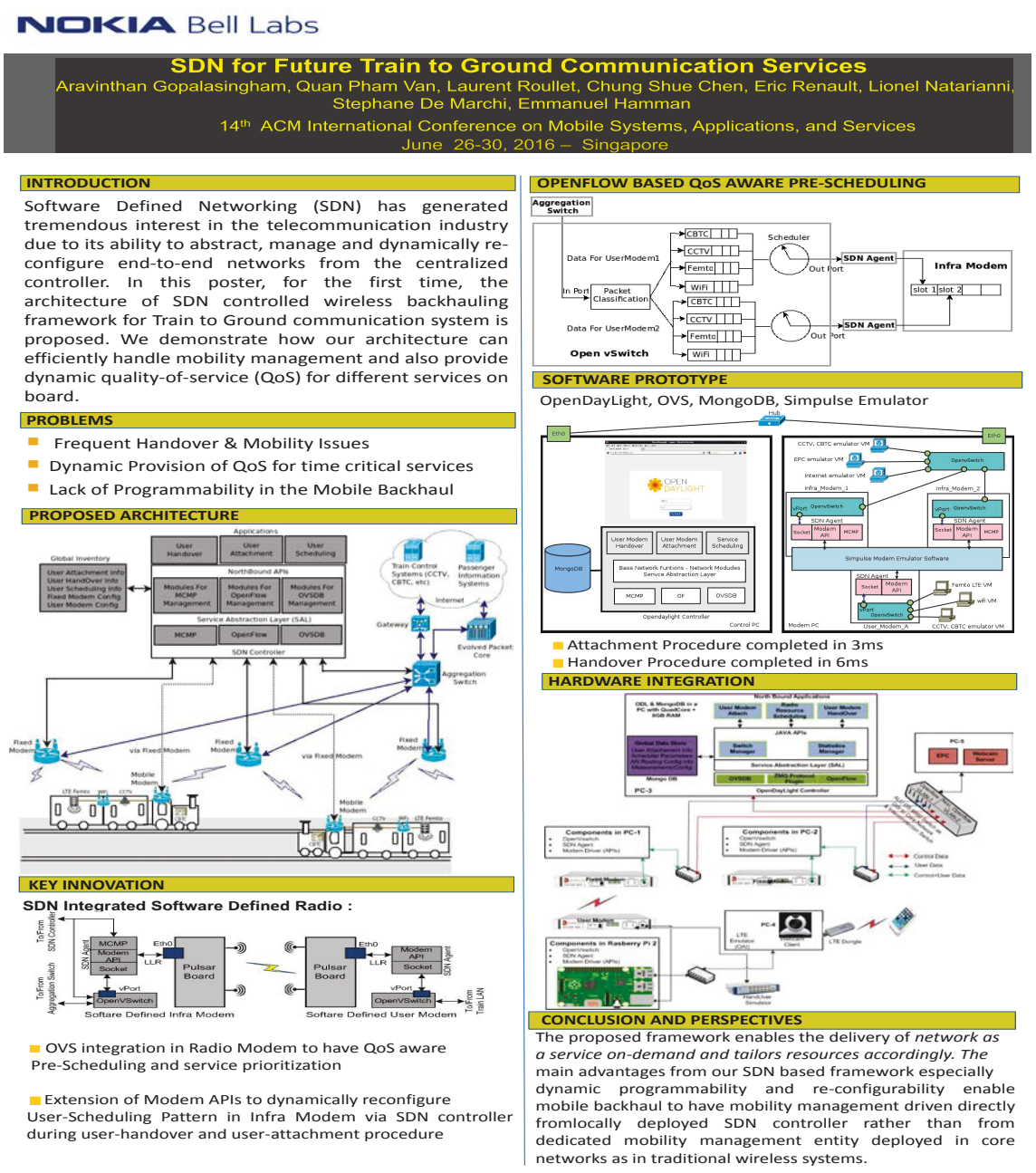


Figure 5.1 – Demo Poster: ACM MobiSys, June 26th-30th, 2016, Singapore

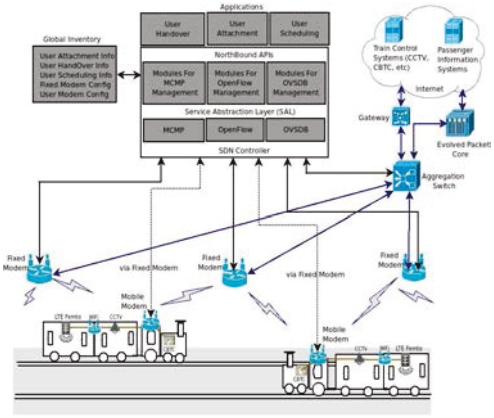
Software Defined Mobile Backhauling For Future Train to Ground Communication Services

MOTIVATION

Software Defined Networking (SDN) has generated tremendous interest in the telecommunication industry due to its ability to abstract, manage and dynamically re-configure end-to-end networks from the centralized controller. In this poster, for the first time, the architecture of SDN controlled wireless backhauling framework for Train to Ground communication system is proposed. We demonstrate how our architecture can efficiently handle mobility management and also provide dynamic quality-of-service (QoS) for different services on board.

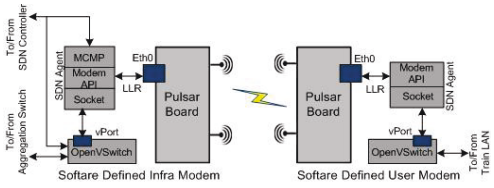
MOBILE BACKHAUL WITH SDN CONTROL

Figure.1: General System Architecture



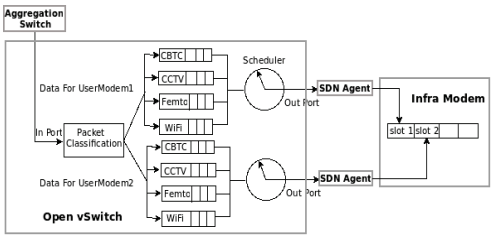
KEY INNOVATION

Figure.2 : Software Defined Radio



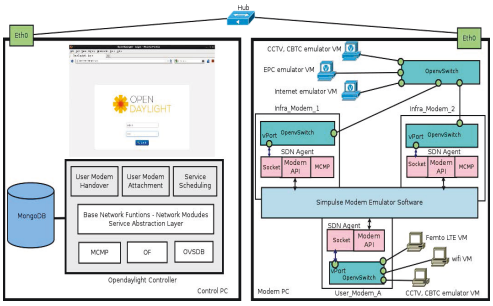
- OVS integration in Radio Modem to have QoS aware Pre-Scheduling and service prioritization as shown in Figure.3
- Extension of Modem APIs to dynamically reconfigure User Scheduling Pattern in Infra Modem via SDN controller during user-handover and user-attachment

Figure.3: Open Flow for QoS aware Pre-Scheduling



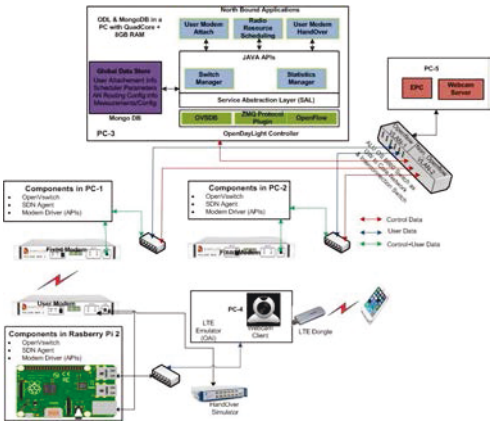
SOFTWARE PROTOTYPE

OpenDayLight, OVS, MongoDB, Simpulse Emulator



- Attachment Procedure completed in 3ms
- Handover Procedure completed in 6ms

HARDWARE INTEGRATION



Contact :

Laurent Roullet	laurent.roullet@nokia.com
Stephane Demarchi	stephane.demarchi@simpulse.fr
Hervè Bonneville	h.bonneville@fr.merce.mee.com

SYSTEME TELECOM POUR LES TRANSPORTS URBAINS DU FUTUR

Figure 5.2 – Demo Poster: Systuf Project Event, May 20th, 2016, Villarceaux, France

BELL LABS FUTURE X DAYS
INVENTING THE NETWORK OF YOU

CLOUD RAN

Scalable and flexible cloud-driven Radio Access Network platform

Challenge

Future radio networks will have to support many network configurations (macro, metro and femto, indoor, outdoor), various types of access waveforms (3G, 4G, 5G, wifi), a variety of devices (multimedia handsets, low rate high latency sensors and actuators, low latency machines), and very different types of communications from intensive video to sporadic small messaging, most of them probably not yet known. Radio networks will therefore need to be extremely scalable and elastic to the network size, and extremely reconfigurable to services and devices.

Innovation

The Cloud RAN vision is an disruptive cloud-driven wireless network architecture designed to support such flexibility in a sustainable way by "importing" several technologies from the IT industry and adapting them to the particular case of wireless networks (in terms of throughput, latency, topology):

- Software-Defined Network (OpenDayLight) is used for abstracting wireless control by introducing "network applications and services" like the X2 proxy coordinator;
- Virtualization (Docker containers and KVM virtual machines) is used for abstracting Radio Access Network and packet core functions from specialized telecom hardware and enabling telecom micro-services approach;
- IT hardware acceleration (GPU and APU) is used for offloading intensive signal processing functions (Fast Fourier Transform) to maximize energy efficiency and computing density;
- Orchestrator (Openstack) is used to dynamically map functions on IT resources;
- Shared Ethernet network is used to replace expensive point-to-point dedicated fronthaul Comon Public Radio Interface links.

All resources are controlled by a "cloud manager".

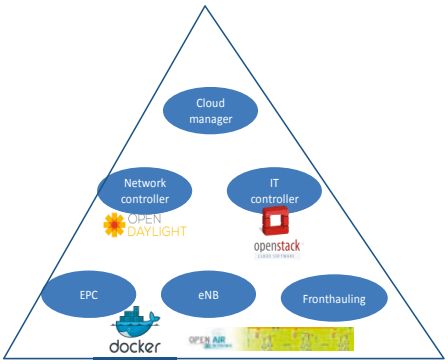


Figure 1: Cloud RAN hierarchical vision

Usecase

The Cloud RAN platform demonstrates several reference "radio as a service" use cases showing the elasticity of the solution: "RAN as a service", "EPC as a service" and "SDN as a service" demonstrated on inter and intra private cloud resources.

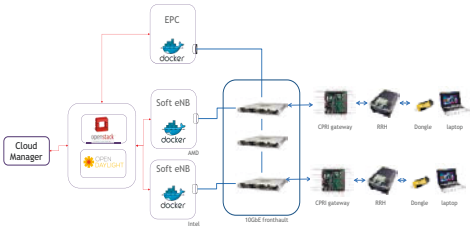


Figure 2: Cloud RAN architecture

BENEFIT: Cloud RAN is the Bell Labs technological platform used for de-risking "cloudification" of the access network on the road to 5G. It is open for internal (BU) as well as external (start up, IRT, industry) collaborations.

laurent.roullet@alcatel-lucent.com, abdelkader.outtagarts@alcatel-lucent.com **and Bell Labs Software Defined Wireless Networks team**

OUR PARTNERS

PLAN SOUVERAINTE TELECOMS (B-COM, INSTITUT MINES TELECOM, ORANGE, TDF, THALES, SYSTEMX)
XILINX – GREENFLOPS – EURECOM - OPEN SOURCE COMMUNITY

Bell Labs
Alcatel·Lucent

Figure 5.3 – Demo Poster: Bell Labs OpenDays, 2015, Villarceaux, France

NOKIA FUTURE X DAYS

Creating time by expanding the human possibility of technology

1

MOBILE NETWORK ORCHESTRATION OVER MICRO-SERVICES

"One-click" 4G and 5G deployment and operation

Challenge

Network Functions Virtualization (NFV) and software-defined networking (SDN) are dramatically changing the landscape of telecommunication industry. Cloud introduces unprecedented technological opportunities waiting to be leveraged in emerging 5G standard. Cellular infrastructure must however be (re)factored according to cloud-native design rules to maximize benefits.

Innovation

We introduce an emerging cloud-native paradigm called "micro-services" for disaggregating cellular functions (4G and 5G) in smaller components. Each component exists in the form of an image in a registry from which it is extracted and then deployed by a specialized orchestrator in a container. Containers are a fast and lightweight virtualization technology running in semi-permanent resources (Virtual or Bare-Metal Machines). Containers can be "service-chained" via SDN flows over semi-static network resources (hardware and software switches).

We therefore introduce segmentation between semi-static "infrastructure-as-a-service" and dynamic "container-as-a-service" layers.

Usecase

We demonstrate the deployment and operation of 4G (complete) and 5G (partial) systems:

- The 4G system is composed of EPC (MME, S-GW, P-GW, HSS) and eNB functions with wireless and wireline SDN-controlled backhaul.
- The 5G system is composed of a split NodeB with its SDN-controlled fronthaul.

All these functions are 'componentized' in the form of Docker containers orchestrated by SWARM and ONOS controllers and follow cloud-native paradigm (stateless, elastic, SDN-controlled). Thanks to this modern design, (re)deployments are instantaneous and highly scalable. In addition, smooth convergence and evolution between various access technologies (4G & 5G) is ensured by a common NFV & SDN control layer.

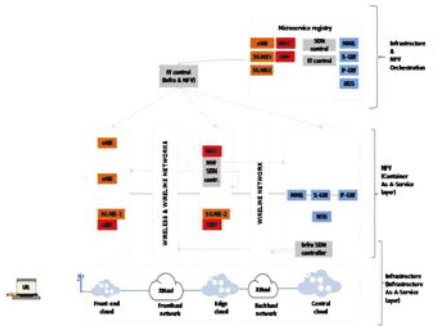


Figure 1: Mobile network micro service 4G and 5G

The platform demonstrates several "infrastructure as-a-service" use cases such as 'EPC as-a-service', 'RAN (4G and 5G) as-a-service', 'xHaul (fronthaul, backhaul) as-a-service'. Autoscaling of a stateless S-GW using KUBERNETES orchestrator is also demonstrated. Several interactive scenarios illustrate the advantages of micro-service paradigm for dynamically and instantaneously ("in 1 click") deploying and (re)configuring xHaul, RAN, EPC....

Contacts:

abdelkader.outtagarts@nokia-bell-labs.com,
laurent.roullet@nokia-bell-labs.com,
bernd.harberland@nokia-bell-labs.com

Nokia Bell Labs

OUR PARTNERS

ORANGE LABS – OPEN AIR INTERFACE SOFTWARE ALLIANCE – SIMPULSE – ANR SYSTUF

NOKIA Bell Labs

Figure 5.4 – Demo Poster: Bell Labs OpenDays, 2016, Villarceaux, France

NOKIA and Bell Labs

The thesis have taken place in the Software Defined Mobile Networks department of Nokia Bell Labs France (previously ALBLF) which is the Research and Development facility of Nokia in France. In this chapter, a brief presentation of the enterprise and its activities will be given.

General presentation of Nokia and Nokia Bell Labs

Nokia is at the forefront of global communications, providing products and innovations in IP and cloud networking, as well as ultra-broadband fixed and wireless access to service providers and their customers, enterprises and institutions throughout the world. Underpinning Nokia in driving the industrial transformation from voice telephony to high-speed digital delivery of data, video and information is Bell Labs, an integral part of Nokia, responsible for countless breakthroughs that have shaped the networking and communications industry. Nokia Bell Labs France, the French Bell labs centre and the second in size, located in Nokia Paris Saclay "Innovation City", in Nozay close to Paris. The Villarceaux "Innovation City" is the largest R and D location of Nokia in Europe, open to its ecosystems and partners, and is located in the vicinity of Paris-Saclay campus. Nokia Bell Labs France is covering research on optical components, transmission systems and optical networks, ultra-broadband wireless architectures, networking algorithms and protocols, security of communication systems, mathematics of dynamic networks, content oriented networking, Internet of Things, network energy, IP platforms and software for telecoms.

Nokia Bell Labs France is active in "poles de competitivite" System@TIC Paris-Region and Cap Digital as well as in a number of national and European collaborative projects. Several joint research laboratories were created in France with Nokia Bell Labs France contribution (the III-V lab, a laboratory with INRIA, and the LINCS). Nokia Bell Labs France is also among the leaders of the French node of "EIT ICT Labs" and is also member of the "SystemX" IRT.

Software Defined Mobile Networks department

The Software Defined Mobile Networks department belongs to the End-to-End Mobile Networks laboratory of Nokia Bell Labs. Its mission is to study and develop efficient solution for various problems in wireless systems as well as new architecture design for next generation wireless networks such as 4G+ and 5G. Research themes are Small cell systems, Self-configuration for wireless networks, Energy efficient wireless networks, "softwarization" of wireless networks, software defined networks (SDN), Network Function Virtualization (NFV), Cloud Computing for mobile networks, Machine learning and Artificial Intelligence systems. The team also has a global expertise in LTE, SDN, NFV, Embedded systems, Applied Mathematics.

Bibliography

[1] The new stack technical tutorial. [Online]. Available: https://thenewstack.io/sdn-series-part-eight-comparison-of-open-source-sdn-controllers/

[2] L. Foundation. Opendaylight. [Online]. Available: https://www.opendaylight.org

[3] S. Rao. (2015) Sdn series part six: Opendaylight, the most documented controller. [Online]. Available: http://thenewstack.io/sdn-series-part-vi-opendaylight

[4] Onos project. [Online]. Available: http://onosproject.org/

[5] A. A. Atayero, O. I. Adu, and A. A. Alatishe, "Self organizing networks for 3gpp lte," in *International Conference on Computational Science and Its Applications*. Springer, 2014, pp. 242–254.

[6] ETSI. Intelligent transport systems. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/intelligent-transport

[7] 3rd generation partnership project. http://www.3gpp.org.

[8] A. deliverable by the NGMN Alliance, in *Further Study on Critical C-RAN Technologies*, 2015.

[9] S. Hub. Opendaylight helium application developers' tutorial. [Online]. Available: http://sdnhub.org/tutorials/opendaylight-helium

[10] EugenBorcoci, "Software defined networking and architectures."

[11] ONF, "Software-defined networking: The new norm for networks," *IOSR Journal of Computer Engineering*, Dec 2013.

[12] opennetworking.org. (2014) Software-defined networking (sdn) definition. [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition

[13] Y.-J. Chang, A. Hari, P. V. Koppol, A. Martin, and T. Stathopoulos, "Scalable and elastic telecommunication services in the cloud." *Bell Labs Technical Journal*, vol. 17, no. 2, pp. 81–96, 2012.

[14] N. Vladimir, , and J. Toni, "Incremental deployment aspects of beyond 4g and 5g mobile hetnets," *International Journal of Future Generation Communication and Networking*, vol. 8, pp. 177–196, 2015.

[15] C. S. Chen and F. Baccelli, "Gibbsian method for the self-optimization of cellular networks," *EURASIP Journal on Wireless Communications and Networking*, August 2012.

[16] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "Softran: software defined radio access network." in *ACM HotSDN*. ACM, 2013, pp. 25–30.

[17] C. M. R. Institute, "White paper on "c-ran the road towards green ran"," 2011.

[18] A. Gopalasingham, Q. Pham Van, L. Roullet, C. S. Chen, E. Renault, L. Natarianni, S. D. Marchi, and E. Hamman, "Software-defined mobile backhaul for future train to ground communication services," in *IFIP Wireless and Mobile Networking Conference*, July 2016, pp. 161–167.

[19] N. Alliance. (2015) 5g white paper. [Online]. Available: https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf

[20] G. T. . Draft. (Nov. 2015) Feasibility study on new services and markets technology enablers, stage 1 (release 14). [Online]. Available: http://www.3gpp.org/ftp/specs/archive/22_series/22.891/

[21] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, 2008.

[22] Nokia network services platform. [Online]. Available: https://resources.alcatel-lucent.com/asset/187011.

[23] Extreme networks. the power of extreme networks openflow-enabled switches. [Online]. Available: http://www.extremenetworks.com/libraries/solutions/SBNetworkVisibilitywithBigSwitchBigTap_1882.pdf.

[24] Dell force10 s-series: S4810 high-performance 10/40 gbe top-of-rack switch. [Online]. Available: http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell_Force10_S4810_Spec_sheet.pdf.

[25] Hewlett-packard. openflow enabled switches. [Online]. Available: http://h17007.www1.hp.com/us/en/networking/solutions/technology/sdn/index.aspx#infrastructure.

[26] Ibm system networking rackswitch g8264. [Online]. Available: http://www-03.ibm.com/systems/networking/switches/rack/g8264/index.html.

[27] Juniper ex series ethernet switches. [Online]. Available: https://www.juniper.net/us/en/local/pdf/brochures/1500057-en.pdf.

[28] Mellanox switchx sx1016. [Online]. Available: http://www.mellanox.com/related-docs/prod_eth_switches/PB_SX1016.pdf.

[29] Pica8 1 gbe / 10 gbe open switches. [Online]. Available: http://www.pica8.com/open-switching/1-gbe-10gbe-open-switches.php.

[30] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani, "Data center network virtualization: A survey," *IEEE Commun. Surveys Tutorials*, vol. 15, no. 2, pp. 909–928, 2013.

[31] C. Sher Decusatis, A. Carranza, and C. Decusatis, "Communication within clouds: open standards and proprietary protocols for data center networking," *IEEE Commun. Mag.*, vol. 50, no. 9, pp. 26–33, 2012.

[32] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley, and L. Mathy, "Flow processing and the rise of commodity network hardware," *SIGCOMM Comput. Commun. Rev.*, 2009.

[33] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S.-H. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang, "Network virtualization in multi-tenant datacenters," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, Apr. 2014, pp. 203–216.

[34] S. Rob, G. Glen, Y. Kok-Kiong, A. Guido, C. Martin, M. Nick, and P. Guru, "Can the production network be the testbed?" in *In Proc. OSDI*, 2010, pp. 1–6.

[35] Y. Kok-Kiong, H. Te-Yuan, K. Masayoshi, Y. Yiannis, M. Nick, K. Sachin, and P. Guru, "Making use of all the networks around us: a case study in android," in *In Proc. CellNet*, 2012, pp. 19–24.

[36] D. Saurav, P. Guru, and M. Nick, "Unifying packet and circuit switched networks," in *In GLOBECOM Workshops*, 2009, pp. 19–24.

[37] H. Nikhil, H. Brandon, J. Vimalkumar, M. David, and M. Nick, "Where is the debugger for my software-defined network?" in *In Proc. HotSDN*, 2012, pp. 55–60.

[38] H. Brandon, S. Srini, M. Priya, Y. Yiannis, S. Puneet, B. Sujata, and M. Nick, "Elastictree: Saving energy in data center networks," in *In Proc. NSDI*, 2010, pp. 17–17.

[39] Switch specification, dec. 2009. version 1.0.0.

[40] Openflow switch specification, feb. 2011. version 1.1.0.

[41] Open networking foundation. openflow switch specification, dec. 2011. version 1.2.0.

[42] Open networking foundation. openflow switch specification, apr. 2013. version 1.3.2.

[43] Open networking foundation. openflow switch specification, aug. 2014. version 1.4.0.

[44] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2656877.2656890

[45] Pox project. [Online]. Available: https://openflow.stanford.edu/display/ONL/POX+Wiki

[46] Flood light. [Online]. Available: http://www.projectfloodlight.org/

[47] Ryu project. [Online]. Available: https://osrg.github.io/ryu/

[48] S. Stephen, P. Herbert, F. Marc E, B. Andy, and P. Larry, "Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors," in *In Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, EuroSys '07*, 2007, pp. 275–287.

[49] B. Eric W, "Multiple instances of the global linux namespaces," in *In Proceedings of the 2006 Ottawa Linux Symposium*, 2006.

[50] Lxc—linux containers. [Online]. Available: https://linuxcontainers.org/.

[51] What is docker? [Online]. Available: https://www.docker.com/whatisdocker/.

[52] P. Lennart, S. Kay, and L. Thorsten, in *Control centre: The systemd Linux init system.* [Online]. Available: http://www.h-online.com/open/features/Control-Centre-The-systemd-Linux-init-system-1565543.html

[53] Cloud foundry warden documentation. [Online]. Available: http://docs.cloudfoundry.org/concepts/architecture/warden.html.

[54] M. Victor *et al.*, in *Let me contain that for you: README.* [Online]. Available: https://github.com/google/lmctfy/blob/master/README.md.

[55] N. Nikaein, E. Schiller, R. Favraud, K. Katsalis, D. Stavropoulos, I. Alyafawi, Z. Zhao, T. Braun, and T. Korakis, "Network store: Exploring slicing in future 5g networks." in *MobiArch*. ACM, 2015, pp. 8–13.

[56] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization." in *SOSP*. ACM, 2003, pp. 164–177.

[57] F. Anhalt and P. Primet, "Analysis and evaluation of a XEN based virtual router," INRIA, Research Report, 2008.

[58] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of open vswitch." in *NSDI*. USENIX Association, 2015, pp. 117–130.

[59] ONF, "Openflow switch specification - version 1.3.0."

[60] N. Shelly, E. J. Jackson, T. Koponen, N. McKeown, and J. Rajahalme, "Flow caching for high entropy packet fields." in *HotSDN*. ACM, 2014, pp. 151–156.

[61] J. Pettit, J. Gross, B. Pfaff, M. Casado, and S. Crosby, "Virtual switching in an era of advanced edges." [Online]. Available: https://yuba.stanford.edu/~casado/dc-caves-10.pdf

[62] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker, "Extending networking into the virtualization layer." in *Hotnets*, 2009, pp. 1–6.

[63] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, "OpenRoads: empowering research in mobile networks," *SIGCOMM Comput. Commun. Rev.*, Jan. 2010.

[64] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise wlans with odin," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 115–120.

[65] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "OpenRadio: a programmable wireless dataplane," in *ACM HotSDN*, 2012, pp. 109–114.

[66] S. Kumar, D. Cifuentes, S. Gollakota, and D. Katabi, "Bringing cross-layer mimo to today's wireless lans," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 387–398.

[67] M. Yang, Y. Li, D. Jin, L. Su, S. Ma, and L. Zeng, "Openran: a software-defined ran architecture via virtualization," in *ACM SIGCOMM computer communication review*, vol. 43, no. 4. ACM, 2013, pp. 549–550.

[68] L. E. Li, Z. M. Mao, and J. Rexford, "Toward software-defined cellular networks," in *Software Defined Networking (EWSDN), 2012 European Workshop on*. IEEE, 2012, pp. 7–12.

[69] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "Softcell: Scalable and flexible cellular core network architecture," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 163–174.

[70] K. Pentikousis, Y. Wang, and W. Hu, "Mobileflow: Toward software-defined mobile networks," *IEEE Communications magazine*, vol. 51, no. 7, pp. 44–53, 2013.

[71] K.-K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown, "The stanford openroads deployment," in *Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization*. ACM, 2009, pp. 59–66.

[72] R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T.-Y. Huang, P. Kazemian, M. Kobayashi, J. Naous *et al.*, "Carving research slices out of your production networks with openflow," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 129–130, 2010.

[73] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, pp. 1–13, 2009.

[74] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, M. Draxler, R. Gupta, V. Mancuso, L. Roullet, and V. Sciancalepore, "CROWD: an SDN approach for DenseNets," in *Second European Workshop on Software Defined Networks (EWSDN)*, Oct. 2013, pp. 25–31.

[75] Etsi nfv working group. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv.

[76] Nox project. [Online]. Available: https://github.com/noxrepo/nox

[77] Beacon project. [Online]. Available: https://openflow.stanford.edu/display/Beacon/Home

[78] Vneio project. [Online]. Available: https://github.com/vneio/sdnc

[79] 5G – Latency: New use cases and the need for network slicing. [Online]. Available: https://knect365.com/5g-virtualisation/article/62ba763a-ad50-4b7e-96a9-ad345e282cba/5g-latency-new-use-cases-and-the-need-for-network-slicing

[80] Lte 3gpp specification. [Online]. Available: http://www.3gpp.org/technologies/keywords-acronyms/98-lte.

[81] S. H. Yeganeh and Y. Ganjali, "Beehive: Simple distributed programming in software-defined networks," in *Proceedings of the Symposium on SDN Research*. ACM, 2016, p. 4.

[82] A superfluid 5G mobile network architecture. [Online]. Available: http://www.unified-streaming.com/blog/superfluid-5g-mobile-network-architecture

[83] Nokia Government Relations Policy Paper: 5th Generation of Communication Networks. [Online] http://networks.nokia.com/system/files/document/nokia_government_relations_policy_paper_on_5g_usa_version.pdf.

[84] Xhaul: The 5g integrated fronthaul/backhaul - 5g-ppp. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2015/07/Xhaul_nutshell.pdf.

[85] J. Allen and F. Chevalier, "Report for vodafone-mobile backhaul market: Phase 2 report," *Analysys Mason, Cambridge, UK, Tech. Rep*, pp. 39 013–215, 2014.

[86] M. Jaber, M. A. Imran, R. Tafazolli, and A. Tukmanov, "5g backhaul challenges and emerging research directions: A survey," *IEEE Access*, vol. 4, pp. 1743–1766, 2016.

[87] Self-optimizing networks: The benefits of son in lte. [Online]. Available: http://www.4gamericas.org/files/2914/0759/1358/Self-Optimizing_Networks-Benefits_of_SON_in_LTE-July_2011.pdf

[88] N. Scully, S. Thiel, R. Litjens, L. Jorgueski, R. Nascimento, O. Linnell, K. Zetterberg, M. Amirijoo, C. Blondia, K. Spaey *et al.*, "D2. 1 use cases for self-organising networks," *URL http://www. fp7-socrates. eu*, 2008.

[89] L. Schmelz, J. Van Den Berg, R. Litjens, K. Zetterberg, M. Amirijoo, K. Spaey, I. Balan, N. Scully, and S. Stefanski, "Self-organisation in wireless networks use cases and their interrelation," in *Wireless World Res. Forum Meeting*, vol. 22, 2009, pp. 1–5.

[90] ——, "Self-organisation in wireless networks use cases and their interrelation," in *Wireless World Res. Forum Meeting*, vol. 22, 2009, pp. 1–5.

[91] N. Alliance *et al.*, "Ngmn recommendation on son and o&m requirements," *Next Generation Mobile Networks, White paper, December*, 2008.

[92] F. Lehser, "Use cases related to self organising network, overall description,"," *NGNM Alliance, Retrieved form the Internet at: http://www. ngmn-cooperation. com/fileadmin/content/documents/downloads/AnnexA—Informative—list—of—SON—Use—Cases—1—53. pdf*, 2007.

[93] R. Agrawal, A. Bedekar, R. Gupta, S. Kalyanasundaram, H. Kroener, and B. Natarajan, "Dynamic point selection for lte-advanced: Algorithms and performance," in *IEEE WCNC*, 2014, pp. 1392–1397.

[94] 3gpp, mobile broadband innovation path to 4g: Release 9, 10 and beyond, tr, (2010). [Online]. Available: http://lteworld.org/whitepaper/3gpp-mobile-broadband-innovation-path-4g-release-9-release-10-and-beyond-hspa-saelte-and-.

[95] Q. Li, R. Q. Hu, Y. Qian, and G. Wu, "Cooperative communications for wireless networks: techniques and applications in lte-advanced systems," *IEEE Wireless Communications*, vol. 19, no. 2, 2012.

[96] L. Ong and J. Yoakum, "An introduction to the stream control transmission protocol (SCTP)," Internet RFC 3286, May 2002.

[97] NoSQL. http://nosql-database.org.

[98] E. Commission, "Intelligent transport systems – eu-funded research for efficient, clean and safe road transport," 2010.

[99] M. of Transport, "Intelligent transport system technology conversation paper," June 2013.

[100] B. Ning, T. Tang, H. Dong, D. Wen, D. Liu, S. Gao, and J. Wang, "An introduction to parallel control and management for high-speed railway systems," *IEEE Trans. Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1473–1483, 2011. [Online]. Available: http://dx.doi.org/10.1109/TITS.2011.2159789

[101] J. Wang, H. Zhu, and N. J. Gomes, "Distributed antenna systems for mobile communications in high speed trains," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 4, pp. 675–683, 2012. [Online]. Available: http://dx.doi.org/10.1109/JSAC.2012.120502

[102] B. Ai, X. Cheng, T. Kürner, Z. Zhong, K. Guan, R. He, L. Xiong, D. W. Matolak, D. G. Michelson, and C. Briso-Rodríguez, "Challenges toward wireless communications for high-speed railway," *IEEE Trans. Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2143–2158, 2014. [Online]. Available: http://dx.doi.org/10.1109/TITS.2014.2310771

[103] S. Balázs, G. András, N. Felicián, C. János, K. Krisztián, N. Barnabás, and V. Gábor, "On qos support to ofelia and openflow," in *Proceedings of European Workshop on Software Defined Networking (EWSDN)*, 2012.

[104] P. David, G. Joao, S. Bruno, C. Luis, S. Paulo, S. Sachin, and S. Dimitri, "The queuepusher: Enabling queue management in openflow," in *Proceedings of European Workshop on Software Defined Networking (EWSDN)*, 2014.

[105] A. Bertout and E. Bernard, "Next Generation of Railways and Metros Wireless Communication Systems," *Aspect 2012,Institution of Railway Signal Engineers.*

[106] S. Itziar, C. Roberto, and P. Asier, *Wireless Technologies in the Railway:Train-to-Earth Wireless Communications.* InTech, 2012.

[107] A. Sniady and J. Soler, "Lte for railways: Impact on performance of etcs railway signaling," *IEEE Vehicular Technology Magazine*, vol. 9, no. 2, pp. 69–77, 2014.

[108] D. T. Fokum, S. Member, and V. S. Frost, "A survey on methods for broadband internet access on trains," *IEEE Communications surveys and Tutorials*, vol. 12, no. 2, 2010.

[109] Alcatel Lucent LTE for Metro Railway Operations White Paper. [Online] http://www.tmcnet.com/tmc/whitepapers/documents/whitepapers/2013/8272-alcatel-lucent-lte-metro-railway-operations.pdf.

[110] K. Masur and D. Mandoc, "Lte/sae—the future railway mobile radio system? long term visions on railway mobile radio technologies," *International Union of Railways (UIC), Technical Report*, vol. 1, 2009.

[111] G. Tingting and S. Bin, "A high-speed railway mobile communication system based on lte," in *Electronics and Information Engineering (ICEIE), 2010 International Conference On*, vol. 1. IEEE, 2010, pp. V1–414.

[112] K. Guan, Z. Zhong, and B. Ai, "Assessment of lte-r using high speed railway channel model," in *Communications and Mobile Computing (CMC), 2011 Third International Conference on*. IEEE, 2011, pp. 461–464.

[113] D. Pareit, E. Van de Velde, D. Naudts, J. Bergs, J. Keymeulen, I. De Baere, W. Van Brussel, C. Vangeneugden, P. Hauspie, G. De Vos *et al.*, "A novel network architecture for train-to-wayside communication with quality of service over heterogeneous wireless networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, p. 114, 2012.

[114] G. Barbu, "E-train-broadband communication with moving trains technical report-technology state of the art," *Int. Union Railways, Paris, France, Tech. Rep*, 2010.

[115] D. T. Fokum and V. S. Frost, "A survey on methods for broadband internet access on trains," *IEEE communications surveys & tutorials*, vol. 12, no. 2, pp. 171–185, 2010.

[116] J.-Y. Zhang, Z.-H. Tan, Z.-D. Zhong, and Y. Kong, "A multi-mode multi-band and multi-system-based access architecture for high-speed railways," in *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*. IEEE, 2010, pp. 1–5.

[117] L. Liu, C. Tao, J. Qiu, H. Chen, L. Yu, W. Dong, and Y. Yuan, "Position-based modeling for wireless channel on high-speed railway under a viaduct at 2.35 ghz," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 4, pp. 834–845, 2012.

[118] L. Tian, J. Li, Y. Huang, J. Shi, and J. Zhou, "Seamless dual-link handover scheme in broadband wireless communication systems for high-speed rail," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 4, pp. 708–718, 2012.

[119] B. Ai, R. He, Z. Zhong, K. Guan, B. Chen, P. Liu, and Y. Li, "Radio wave propagation scene partitioning for high-speed rails," *International Journal of Antennas and Propagation*, vol. 2012, 2012.

[120] R. He, Z. Zhong, B. Ai, J. Ding, W. Jiang, H. Zhang, and X. Li, "A standardized path loss model for the gsm-railway based high-speed railway communication systems," in *Vehicular Technology Conference (VTC Spring), 2014 IEEE 79th*. IEEE, 2014, pp. 1–5.

[121] Y. Dong, P. Fan, and K. B. Letaief, "High-speed railway wireless communications: Efficiency versus fairness," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 2, pp. 925–930, 2014.

[122] S. Lin, Z. Zhong, B. Ai, and C. Briso-Rodriguez, "Transmission interference improvement of railway communication via distributed antennas system," in *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*. IEEE, 2010, pp. 1–5.

[123] ZeroMQ Distributed Messaging. [Online] http://zeromq.org/.

[124] Pulsar technology platform. [Online]. Available: http://www.simpulse-dsp.com/products/technology

[125] Simpulse Software Defined Radio Modems. [Online] http://www.simpulse-dsp.com.

[126] B. Haberland, F. Derakhshan, H. Grob-Lipski, R. Klotche, W. Rehm, P. Schefczik, and M. Soellner, "Radio base stations in the cloud," in *Bell Labs Technical Journal*, vol. 18.

[127] Standard library musl. [Online]. Available: https://www.musl-libc.org

[128] BusyBox. [Online]. Available: https://busybox.net

[129] Apache Maven. [Online]. Available: https://maven.apache.org

[130] JavaScript Object Notation. [Online]. Available: http://www.json.org

[131] L. Kleinrock, *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.

[132] A. Imran and A. Zoha, "Challenges in 5G: how to empower SON with big data for enabling 5G," *IEEE Network*, vol. 28, no. 6, 2014.

[133] MongoDB White Paper: Performance Best Practices for MongoDB. [Online] http://s3.amazonaws.com/info-mongodb-com/MongoDB-Performance-Best-Practices.pdf.

[134] N. Piel. (2013) zeromq an introduction. [Online]. Available: http://nichol.as/zeromq-an-introduction

[135] N. Dmitry and S.-S. Manfred, "On micro-services architecture," *International Journal of Open Information Technologies*, 2014.

[136] Docker swarm. [Online]. Available: https://www.docker.com/products/docker-swarm

[137] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Computing*, vol. 17, no. 2, pp. 20–27, 2013.

[138] T. Laukkarinen, J. Suhonen, and M. Hännikäinen, "An embedded cloud design for internet-of-things," *International Journal of Distributed Sensor Networks*, 2013.

French Summary (Résumé)

Software-Defined Networking

Le futur Internet est considéré comme le réseau de prochaine génération (NGN) qui fournira des services intégrés, c'est-à-dire des services interactifs temps réel de type multimédias et de type données. Avec la croissance rapide du trafic sur l'Internet, la construction et la gestion des grands réseaux IP/Ethernet deviennent de plus en plus complexes. Les architectures réseaux actuelles ne répondent que partiellement aux besoins des entreprises, des opérateurs et des utilisateurs finaux. Il existe aujourd'hui de nombreux protocoles, par exemple, les protocoles de routage, Ethernet, IP, etc. utilisés pour connecter les réseaux de transport sur des distances, des vitesses et des topologies variées. Ces protocoles sont en général définis pour résoudre un problème spécifique, sans le bénéfice d'abstractions fondamentales. Cela mène à une des premières limitations dans les réseaux d'aujourd'hui : la complexité. Par exemple, pour ajouter ou déplacer des éléments de réseau, les administrateurs doivent (re)-configurer, compte tenu de la topologie, plusieurs entités matérielles/logicielles à l'aide d'outils de gestion d'éléments de réseau. En raison de cette complexité, les réseaux d'aujourd'hui sont configurés de manière relativement statique pour minimiser le risque de perturbation de service. Ainsi, la nature statique des réseaux est la deuxième limitation aujourd'hui, malgré l'environnement de serveurs dynamiques actuel, de la virtualisation et la possibilité de migration de machines virtuelles. La migration de machines virtuelles pose de nombreuses problèmes aux réseaux traditionnels, depuis l'adressage des schémas et des espaces de noms jusqu'à la notion de base d'une conception segmentée basée sur le routage. En plus d'adopter des technologies de virtualisation, de nombreuses entreprises exploitent aujourd'hui un réseau IP convergé pour le trafic conjoint de la voix, des données et de la vidéo. Alors que les réseaux existants peuvent fournir des niveaux de QoS différenciés pour différentes applications, le provisionnement de ces ressources est hautement manuel. Les administrateurs de réseaux doivent configurer l'équipement de chaque fournisseur séparément et ajuster les paramètres telle que la bande passante du réseau et la qualité de service par session et par application. En raison de sa nature statique, le réseau ne peut pas s'adapter dynamiquement à l'évolution du trafic, des applications et des demandes des utilisateurs. D'autre part, des politiques de gestion incohérentes sous les reseaux d'aujourd'hui peuvent rendu impossible la mise en place de politiques à l'échelle des reseaux. Les administrateurs de réseaux doivent configurer des milliers de dispositifs et de mécanismes ce que la complexité des réseaux rend très complexe. L'application d'un ensemble de politiques cohérentes d'accès, de sécurité, de QoS et autres reste difficile. En outre, les entreprises sont toujours à la recherche de nouvelles façons de déployer de nouvelles capacités et services en réponse rapide à l'évolution des besoins ou les demandes des utilisateurs. Toutefois, leur capacité à répondre est entravée par le cycle des équipments des vendeurs, ce pouvent prendre plusieures années. En outre, le manque de normalisation limite la capacité des opérateurs réseau à concevoir le réseau au-delà de leur propre environnement. Toutes ces limitations des réseau actuels ont conduit l'industrie à un point de

basculement. En réponse, l'industrie a créé l'architecture *Software-Defined Networking(SDN)* et développe la normalisation correspondante.

Le SDN est une architecture émergente dynamique, gérable et adaptable, pour la bande passante et la nature dynamique des applications d'aujourd'hui. L'architecture SDN découple les fonctions de contrôle réseau et de transport, et permet au réseau d'être directement programmable. Le SDN, en résumant l'infrastructure sous-jacente, fournit l'intelligence et le contrôle nécessaires pour les applications et les services réseaux. La *Open Networking Foundation (ONF)* mène la normalisation SDN et présente le modèle d'architecture SDN tel que représenté en Fig. I. Selous l'ONF, l'architecture SDN est :

- *Programmable directement :* le contrôle du réseau est directement programmable car il est découplé des fonctions de transport;
- *Agile:* le contrôle séparé du transport permet d'ajuster dynamiquement l'écoulement du trafic à l'échelle du réseau pour répondre aux besoins variés dans le temps; .
- *Géré de maniere centralisée:* les fonctions de contrôle sont (logiquement) centralisées dans les contrôleurs SDN qui maintiennent une vue globale du réseau;
- *Une configuration par logiciel :* le SDN permet aux administrateurs de réseaux de configurer, gérer, sécuriser et optimiser les ressources réseaux très rapidement, des logiciels SDN dynamiques et automatisés;
- *Un standard ouvert et neutre :* implémenté par des standards ouverts, le SDN simplifie la conception et le fonctionnement du réseau car les instructions sont fournies par les contrôleurs SDN au lieu d'avoir plusieurs éléments de réseau et protocoles spécifiques au fournisseur.

L'architecture ONF/SDN se compose de trois couches distinctes :

- *La couche application* consiste en des applications métiers liées aux utilisateurs finaux qui communiquent avec le contrôleur pour fournir différents services de communication;
- *La couche contrôle* comprend un contrôleur SDN centralisé qui maintient une vue globale du réseau. Il contrôle et supervise la fonctionnalité de transport du réseau une interface ouverte. Parmi les contrôleurs *open-source* ou peut nota NOX, Floodlight, Ryu, OpenDayLight, etc.
- *La couche physique* comprend les périphériques physiques du réseau, y compris les commutateurs Ethernet, Openvswitch et les routeurs qui fournissent la transmission.

Dans un réseau traditionnel, les éléments de réseau sont composés d'un plan de contrôle embarqué et distribué qui gère les activités de commutation, de routage et d'ingénierie de trafic. Ce plan de contrôle distribué fournit des informations utilisées pour construire une table de transfert et configurer des stratégies de gestion de paquets (par exemple, la sécurité, la mise en forme du trafic, la renvoi, etc.). Le plan de données prend une décision basée sur la table de transfert pour savoir où envoyer des trames et des paquets. Ces deux plans sont situés dans le même élément de réseau que sur la Fig. II. Contrairement aux réseaux traditionnels, le paradigme SDN découple le plan de contrôle et devient une entité externe comme le montre la Fig. II. Le plan de commande SDN est implémenté sous la forme d'un ensemble de modules logiciels avec des fonctionnalités dédiées et peut être vu comme une entité logiquement centralisée, contrôlant tous les éléments du plan de données dans le réseau. Les contrôleurs SDN utilisent différents protocoles de configuration de réseau pour interagir avec les tables de routage des routeurs, des commutateurs et des boîtiers intermédiaires compatibles SDN, apportant ainsi un niveau le programmabilité. Les protocoles de plan de commande distribués dans les réseaux traditionnels peuvent être vus soit comme un service

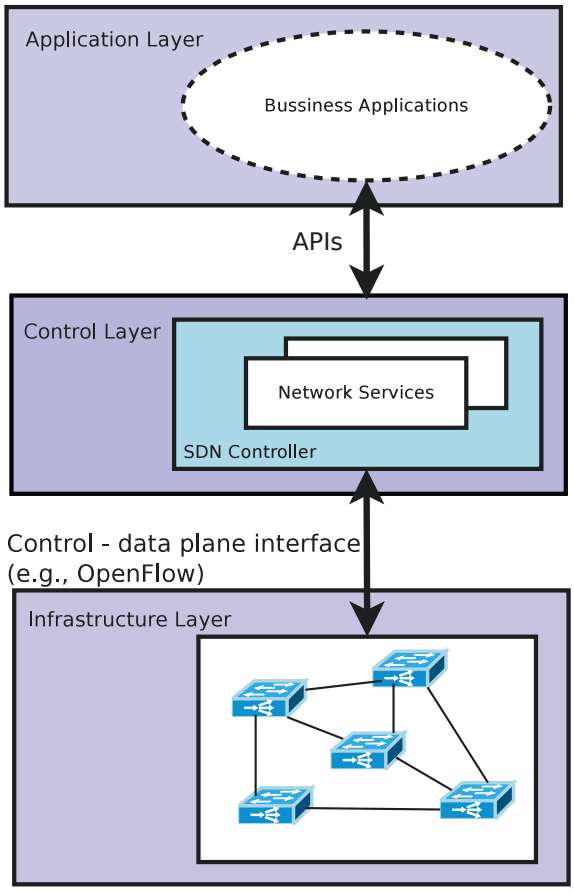


Figure I – Architecture: Software Defined Networking.

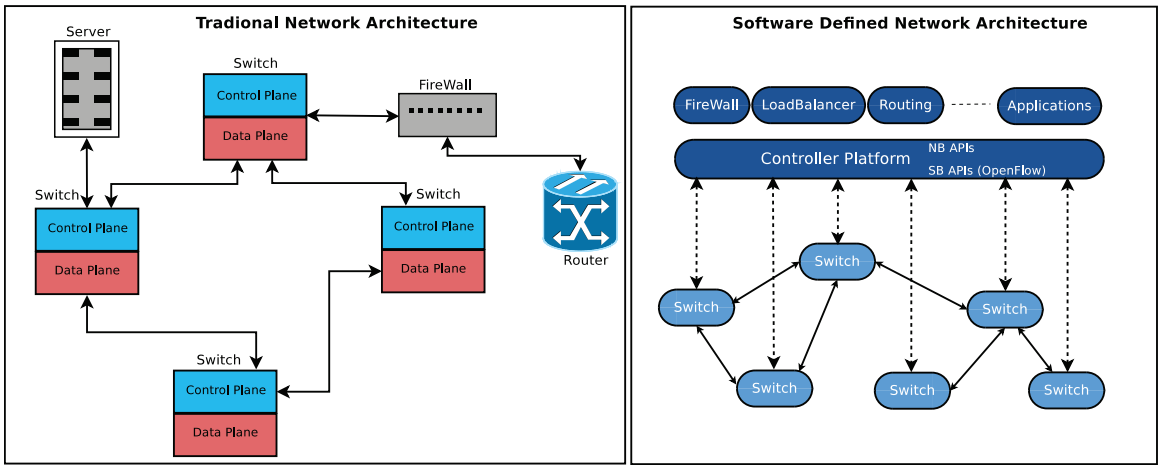


Figure II – Architecture traditionnelle vs. Software-Defined Networking (SDN).

réseau soit comme une application déployée dans le contrôleur. Toutes les applications peuvent tirer parti des mêmes informations réseaux (la vue réseau global), conduisant (sans doute) à des décisions politiques plus cohérentes et efficaces tout en réutilisant des modules logiciels de plan de contrôle. Ces applications peuvent prendre des actions, c'est-à-dire reconfigurer des éléments de réseau, à partir d'une partie quelconque du réseau. L'intégration de différentes applications devient plus simple. Par exemple, les applications d'équilibrage de charge et de routage peuvent être

combinées séquentiellement, les décisions d'équilibrage de charge ayant priorité sur les stratégies de routage.

Réseaux sans fil de prochaine génération

La croissance continue des applications et des services basés sur les systèmes haut débit mobiles exige que les opérateurs prévoient une architecture réseau capable de répondre aux exigences futures de capacité et de performance. Afin de faire face à une telle demande, certains opérateurs réseaux adoptent maintenant un paradigme de *cloud computing*, capable de faire correspondre la demande de capacité et la réduction des coûts. Le pendant ces services ont des caractéristiques de trafic et des contraintes de qualité de service (QoS) différentes par rapport aux services IT traditionnels qui doivent être pris en compte dans le processus de cloudification.

Les opérateurs et les fournisseurs de réseaux mobiles estiment que les réseaux de nouvelle génération seront construits à partir de technologies existantes mais évolutives telles que LTE, LTE-A et Wi-Fi, ainsi que des technologies révolutionnaires récentes pour avoir une expérience utilisateur gigabit avec des communications à latence pratiquement nulle. Les principales exigences pour les réseaux de nouvelle génération sont d'avoir des performances réseaux supérieures avec des cas d'utilisation différents comparés à ses prédécesseurs. Le futur réseau sans fil devra être hétérogène, avec un déploiement dense de macro et de petites cellules (HetNet). Ainsi, le déploiement dense des stations de base générera des scénarios d'interférences plus complexes dans le réseau. Les techniques communes de fréquence et de planification du temps provenant du LTE-A sont les techniques standards pour atténuer les interférences entre les cellules dans HetNets. Cependant, les solutions distribuées actuelles pour la gestion des ressources radio proposées en LTE ne sont pas évolutives pour les petits déploiements denses. En LTE-A, la *Co-Coordination multiple point transmission* (CoMP) et le réseau *Multi-Input Multi-Output* (MIMO) sont des techniques pour améliorer l'expérience utilisateur et, en général, l'utilisation du réseau. Le RAN, pour supporter de telles techniques s'appuie fortement sur la couche réseau pour la coordination entre les stations de base distribuées pendant l'émission/réception. Ceci introduit une surcharge de communication (latence) dans l'exécution de telles techniques dans la conception actuelle de RAN. Afin de répondre à ces exigences et à diverses exigences de service, le réseau doit être très flexible et facilement reconfigurable. L'évolution de SDN est introduite pour permettre la programmabilité et la flexibilité des réseaux d'accès radio (RAN). Le principal avantage d'une telle approche est d'intégrer une intelligence complète au contrôleur centralisé afin que la fonctionnalité de RAN puisse être plus facilement optimisée.

L'architecture Cloud RAN (C-RAN) peut répondre aux exigences de l'auto-optimisation, l'auto-configuration et l'auto-adaptation du réseau par SDN et par NFV. Dans le contexte de C-RAN, la virtualisation joue un rôle clé dans la facon de gérer le cycle de vie des fonctions et des services du réseau mobile dans le *cloud* central, afin d'atteindre une utilisation efficace des ressources, l'élasticité et l'équilibrage de charge. La fonction réseau virtuelle (NFV) mène également à une réduction significative du CAPEX ainsi que de OPEX l'en utilisant des techniques d'automatisation telles que l'orchestration avec SDN.

Motivation de la recherche

Bien que les principales technologies pour la prochaine génération de réseaux sans fil, à savoir la 5G, restent encore à préciser, les cas d'utilisation importants et les exigences ont été énumérés.

La compréhension commune entre les chercheurs, les opérateurs et les fournisseurs est que la 5G sera une plate-forme unifiée pour déployer des services tels que le haut débit mobile, des services de communication de type critique et *Internet of Things* (IOT). La diversité des cas d'utilisation exige certainement une nouvelle architecture réseau pour supporter un tel mélange de services. Afin de couvrir ces cas d'utilisation et exigences, le réseau d'accès radio 5G (RAN) doit être conçu pour utiliser le plus efficacement possible le spectre disponible. Le logiciel introduit dans le RAN permet de faciliter l'orchestration et la gestion des ressources radio. La séparation claire du contrôle et de la couche de données avec l'abstraction réseau de bout en bout, le concept de SDN, fermeut une solution clé pour apporter une telle flexibilité et programmabilité dans les futurs réseaux sans fil. Du point de vue RAN, le logiciel facilitera non seulement l'orchestration des ressources radio mais aussi la gestion et le partage du spectre. L'intégration de SDN dans les réseaux sans fil permet la sensibilisation au spectre, la mobilité du spectre et la mise en œuvre efficace des stratégies de partage du spectre dans les réseaux. Ainsi, l'architecture de contrôle RAN définie par le logiciel apporte une utilisation plus efficace des ressources radio. Cette approche profite également aux opérateurs en permettant la fonctionnalité de partage et de découpage RAN pour les utilisateurs primaires tels que les opérateurs de réseaux virtuels mobiles (MVNO) et les fournisseurs de contenu afin de mieux affiner leur réseau en fonction des besoins de service. Les utilisateurs finaux bénéficieront d'une expérience réseau améliorée grâce à la coordination améliorée et optimisée entre les différentes entités du réseau. Par conséquent, l'introduction de SDN aura également un impact significatif sur la chaîne de valeurs de l'industrie mobile future en réduisant à la fois les dépenses en immobilisations (CAPEX) et les dépenses opérationnelles (OPEX) en raison de ses avantages inhérents. Par conséquent, cette thèse se concentre sur la conception et la validation d'un **Software-Defined Network Access Network Architectures** qui apporte la flexibilité, la programmabilité et la gestion unifiée pour les différents cas d'utilisation et les scénarios de déploiement de la prochaine génération de réseaux sans fil.

Architecture générale et objectifs de recherche

Le réseau défini par logiciel est un aspect important de l'architecture des réseaux sans fil de prochaine génération. Le SDN permet une reconfiguration rapide pour influencer dynamiquement et optimiser les performances des réseaux mobiles. Le SDN apporte en général un ensemble d'avantages aux réseaux sans fil comme résumé ci-dessous :

- **Flexibilité :** un problème actuel pour les opérateurs des réseaux mobiles est le montant élevé des dépenses en immobilisations et les dépenses opérationnelles (CAPEX et OPEX respectivement) de leurs réseaux, indépendamment de la charge de trafic et de l'utilisation des services, et donc du gain pour les produits qu'ils vendent aux clients. Grâce à l'approche SDN, les opérateurs pourront adapter le réseau à leurs besoins en redéfinissant simplement le contrôleur et en réduisant ainsi les coûts;
- **Programmabilité :** il permet à des tiers d'acquérir des ressources réseaux à la demande satisfaisant leurs SLA individuels. En outre, la programmabilité peut améliorer la QoE perçue par l'utilisateur en personnalisant les ressources du réseau en conséquence;
- **Unifié management :** l'adoption d'un contrôle logiquement centralisé unifie les plates-formes réseaux hétérogènes et fournit une opération simplifiée du réseau sans fil: avec SDN, les opérateurs réseaux doivent uniquement contrôler un ensemble d'entités centrales (les contrôleurs) qui contrôlent l'ensemble du réseau, les technologies;

- **Développement de nouveaux services :** en modifiant le comportement des applications qui s'exécutent au-dessus du contrôleur SDN, de nombreux nouveaux services qui n'étaient pas inclus dans l'architecture conçue initialement peuvent être activés en modifiant le comportement du réseau et en adaptant ses capacités pour l'introduction de nouveaux services en quelques heures au lieu de quelques semaines.

Comme le montre la Fig. III, le sujet de recherche global de cette thèse se trouve dans deux principaux cas d'utilisation des réseaux mobiles de prochaine génération, à savoir ***Telco*** et ***Vertical***. Nous suivons l'architecture de réseau mobile 5G qui comprend ***Front End Unit***, ***Edge Cloud*** et ***Central Cloud***. Dans les cas d'utilisation de *Telco*, nous exploitons les avantages de SDN pour avoir un cadre de contrôle flexible pour les réseaux d'auto-organisé (SON) et le fractionnement du traitement dynamique des utilisateurs. Dans le cas d'utilisation verticale, nous utilisons divers avantages du protocole SDN et OpenFlow pour utiliser efficacement les ressources radio du réseaux *backhaul* sans fil dans les systèmes de communication train-sol. Le SDN dans notre architecture globale se composent de trois ensembles : les fonctionnalités de soutien SON, le contrôle de *backhaul* sans fil et contrôle pour le traitement de dynamique l'utilisateur split. L'orchestrateur est placé en haut du contrôleur SDN pour représenter le cadre d'orchestration du réseau mobile pour le déploiement des fonctions de réseau virtuelle (VNF).

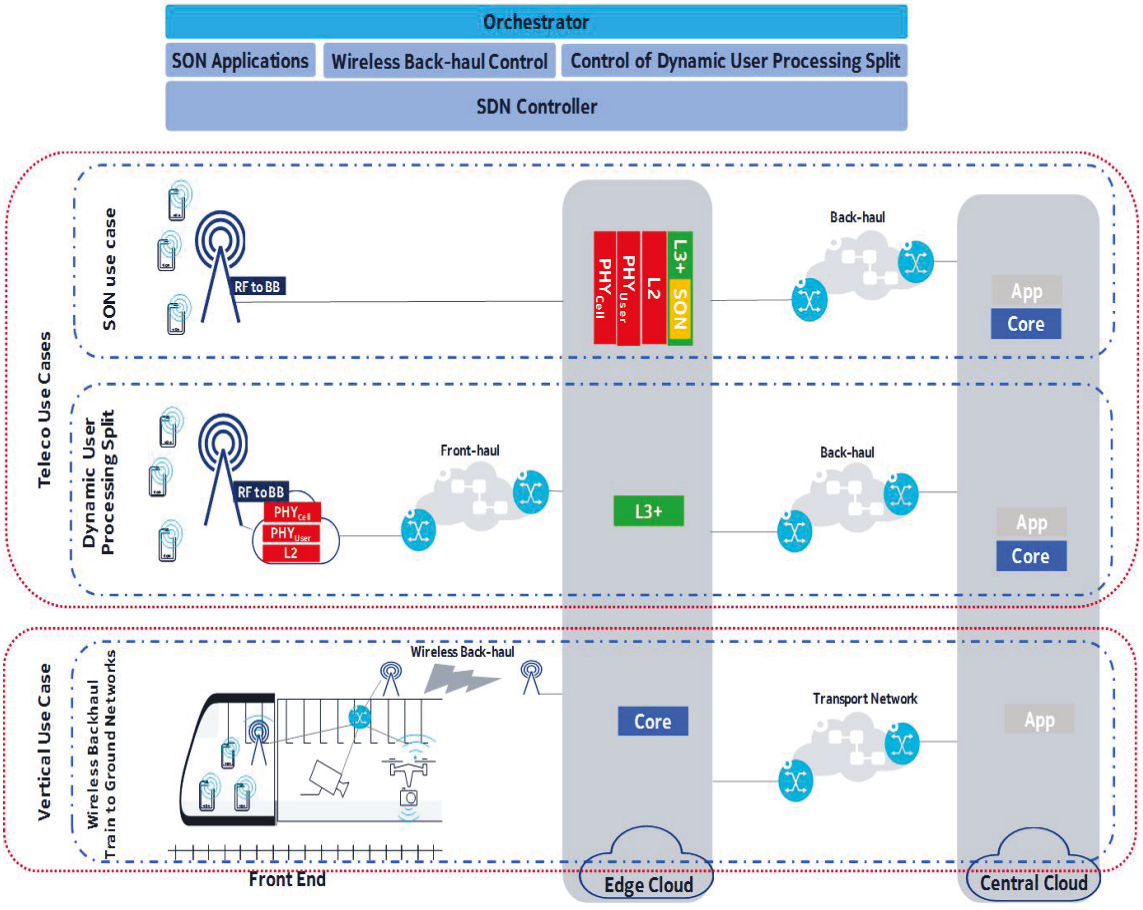


Figure III – Software-Defined Réseau sans fil: Vue globale.

- **SDN pour les SON :** L'introduction de HetNet dans 3GPP LTE/LTE-A est la solution bien connue pour augmenter la couverture et la capacité du réseau par une réutilisation spatiale à plus

haut spectre en déployant des petites cellules de faible puissance (SC) mélangér avec des cellules macro de haute puissance (MC). Il est prévu que pour atteindre les besoins de performance des applications futures, 5G sera déployé en tant que réseau hétérogène, composé de plusieurs technologies d'accès radio incluant LTE-A, Wi-Fi et 5G.

En outre, il y aura une combinaison de macro, petites et super petites cellules dans la 5G. L'augmentation de la densité du réseau et du trafic entraînera une interférence entre les cellules adjacentes et, par conséquent, il y aura une dégradation significative du débit de l'utilisateur et de la qualité du service. Les cellules adjacentes, qui se chevauchent donc, doivent coordonner leurs allocations de ressources (temps/fréquence/dimension spatiale) pour réduire les interférences nuisibles aux utilisateurs dans la zone de chevauchement. Cela entraînera généralement une complexité supplémentaire de gestion et de fonctionnement en raison de l'hétérogénéité du système et du manque d'interopérabilité flexible entre les périphériques réseaux.

La norme 3GPP SON introduite dans LTE est connue pour simplifier et réduire les opérations réseau et optimiser également le réseau d'accès radio (RAN) grâce à des mécanismes automatisés (par exemple auto-configuration, auto-optimisation, auto-guérison). Selon la norme 3GPP, il existe une interface X2AP définie entre les stations de base pour échanger des informations pour assurer la coordination entre elles. Le SON prend en charge les mécanismes de basses fréquences (par exemple l'auto-configuration au démarrage, l'auto-optimisation par jour de l'inclinaison de l'antenne) ainsi que les mécanismes à hautes fréquences (par exemple l'optimisation de la coordination des interférences distribuées *via* l'interface X2). Ce mécanisme de coordination réparti ne passe pas à l'échelle, en particulier en termes de latence. Cela entraîne de mauvaises performances, ce qui réduit considérablement la capacité globale en raison de l'inpossibilité de gérer les interférences et d'équilibrer la charge. Les techniques SON requièrent des contrôleurs dédiés déployés par des opérateurs appelés serveurs autonomiques pour réaliser une coordination centralisée dans la mise en œuvre des techniques SON. Le SDN peut être un mécanisme flexible pour adapter les solutions SON. Le contrôleur logiquement centralisé permet de prendre des décisions d'allocation de ressources radio avec une visibilité globale sur de nombreuses stations de base, ce qui est beaucoup plus optimal que la gestion distribuée. En centralisant l'intelligence de réseau et les ressources de calcul, les décisions d'allocation de ressources peuvent être ajustées en fonction du profil d'allocation de puissance dynamique et de sous-porteuse de chaque station de base. De plus, l'évolutivité est améliorée car à mesure que de nouveaux utilisateurs sont ajoutés, la capacité de calcul requise à chaque station de base reste faible car ces traitements sont centralisés dans le contrôleur RAN défini par logiciel. Dans le cadre de cette recherche, nous concevons, mettons en œuvre et validons le cadre de contrôleur SDN qui prend en charge le déploiement des solutions SON et aussi pour apporter la programmabilité dans le réseau d'accès radio.

- **SDN pour backhaul sans fil :** le réseau Xhaul mobile (*backhaul et fronthaul*) est un jalon essentiel vers la réalisation d'un réseau 5G. Les réseaux *backhaul* en 5G nécessitent plus de capacité, moins de latence, de synchronisation, de sécurité et de résilience. Les réseaux actuels de *backhaul* sont principalement construits avec des liaisons hyperfréquences (souvent appartenant en propre à l'opérateur) et des liaisons fibre/cuivre (souvent louées par l'opérateur) avec des proportions différentes selon l'opérateur et selon le pays. La 5G est en partie une évolution des technologies existantes mais est également basée sur des technologies nouvelles touchant toutes les parties du réseau. En conséquence, les défis de *back-end* 5G sont multiples : capacité supérieur 10 Gbps et latence de bout en bout inférieur à 1 ms, haute sécurité et résilience, synchronisation de temps et de fréquence, faible consommation d'énergie et faible coût. Aucune des solutions de

backhaul actuelles ne peut fournir tout ce qui précède comme une solution autonome. La fibre optique basé *backhaul* rang le meilleur candidat dans tous ces aspects, sauf peut-étre coût.

Intelligent Transportation Systems (ITS) utilise des réseaux câblés et sans fil pour gérer la communication entre les différents sous-systèmes. Le réseau de communication *train-sol* est l'un des cas d'utilisation importants des ITS. Bien qu'il existe plusieurs technologies et solutions pour fournir un backhaul sans fil dans les systèmes de communication train-sol, l'industrie ferroviaire se heurte encore à plusieurs problèmes, comme le réapprovisionnement des Paquets en raison de fréquents transferts, la difficulté de fournir des services garantis pour les services de contraintes QoS, l'infrastructure sans fil dans les tunnels, etc. L'évolution de SDN qui fournit la visibilité globale du réseaux, compléterait l'architecture *backhaul*, permettant ainsi une opération de *backhaul* entièrement coordonnée. En raison des avantages tels que l'abstraction de réseau centralisée, le contrôle fin des ressources réseau de bout en bout, la reconfigurabilité dynamique des périphériques, les réseaux de *backhaul* sans fil SDN peuvent être, tous, une solution appropriée pour les défis ci-dessus. Dans le cadre de cette recherche, nous concevons, mettons en œuvre et validons la structure de *backhaul* sans fil définie par logiciel qui peut répondre à divers défis dans les réseaux de *backhaul* sans fil actuels et futurs.

- **SDN pour le *Dynamic User Processing Split* :** les réseaux d'accès radio en nuage (C-RAN) constituent le paradigme évolutif des réseaux mobiles de la prochaine génération où les stations de base sont migrées et placées dans un emplacement central appelé bassin BBU, laissant seulement l'unité d'antenne dans le site cellulaire connecté par Ethernet ou Ethernet réseau. L'évolution de cette architecture conduit le réseau mobile vers le paradigme des nuages. La centralisation de RAN a pour but de bénéficier d'avantages tels que l'allocation des ressources à la demande, l'équilibrage des charges entre les stations de base, le partage des ressources et la faisabilité de la mise en œuvre de techniques de signalisation conjointes pour une meilleure expérience utilisateur. Bien que la centralisation de RAN apporte plusieurs avantages, il y a une latence inhérente à de telles techniques. Ceci conduit à la proposition d'une architecture partiellement centralisée, c'est-à-dire C-RAN partiel, qui permet aux opérateurs de déployer des fonctions RAN dans l'environnement de nuages distribués tels que *Front End Unit, Edge Cloud* et *Central Cloud*. Le RAN pour le service correspondant à la réalité augmentée, les véhicules autonomes et la robotique nécessite un temps d'alller retour de l'ordre de paquets micro secondes.

En général, le concept de virtualisation dans le *cloud* se caractérise par la capacité d'agréger ou de partager des ressources physiques distribuées et multiples pour atteindre le niveau de performance requis. Cela nécessite la mise en œuvre d'un cadre de contrôle qui apporte une telle dynamique dans les déploiements RAN. Le SDN est considéré comme une approche appropriée pour gérer la virtualisation, en particulier pour générer une vue globale des ressources disponibles pour la gestion, l'équilibrage de charge et la mise en réseau entre les unités de traitement distribuées. Ainsi, l'approche SDN peut profiter au C-RAN pour la mise en œuvre du processus d'allocation dynamique des ressources et de l'équilibrage de la charge du trafic entre différentes BBU. Dans cette recherche, nous proposons un cadre de contrôle pour C-RAN basé sur l'architecture SDN. L'objectif principal de ce cadre est de fournir une gestion unifiée pour le traitement dynamique des fractionné des utilisateurs en C-RAN. Ce cadre soutient également diverses caractéristiques du C-RAN tels que l'affectation flexible et dynamique des ressources, le partage des ressources, la mise en œuvre de techniques conjointes de traitement du signal.

Contribution

Les trois contributions principales vers SDN pour le RAN sont présentées ci-dessous:

SDN pour SON

L'architecture proposée correspond bien au paradigme SDN en fournissant une vue globale sur le réseaux, vue obtenue d'un controller logiquement centralisé. Puisque l'intéret principal de l'architecture vient de l'interface appelée North Bound (NB) API, qui est indépendante du constructeur et mise en place pour le support des algorithmes SON et l'optimisation, le contrôleur doit fournir une abstraction des paramétres de configurations du réseau et des indicateurs clés de performance (Key Performance Indicators-KPI). Dans ce contexte, l'application RAN utilise le controller NB API pour :

- découvrir les éléments de réseau et la topology du réseau,
- collecter les mesures du RAN et les paramétres de configurations du réseau,
- reconfigurer le RAN avec des paramétres optimaux.

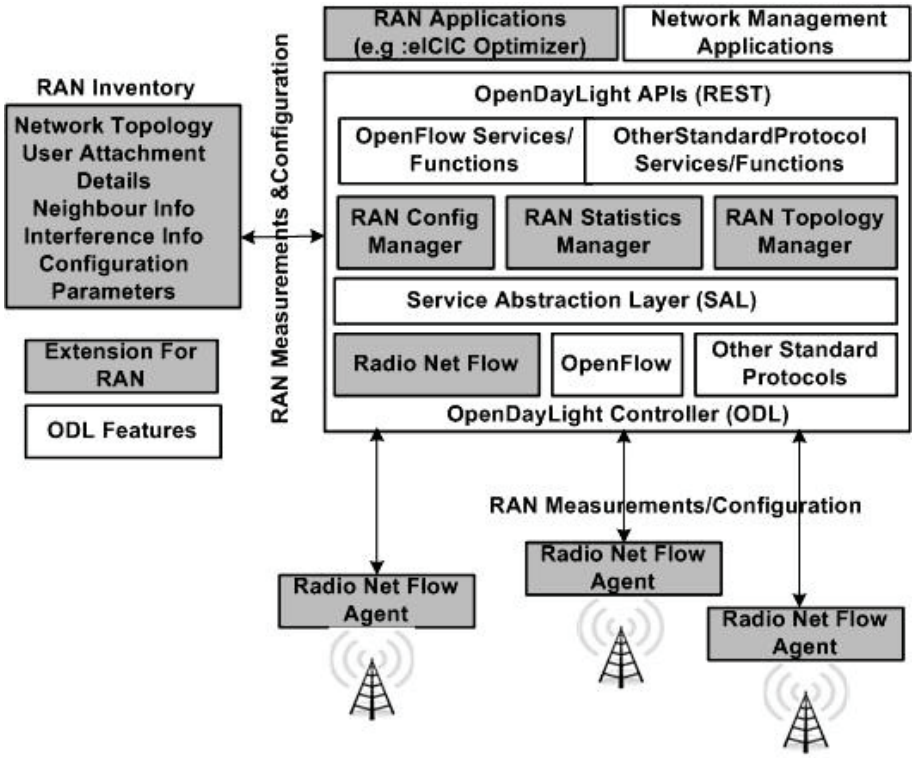


Figure IV – Architecture du *software-defined* réseau d'accès.

Le protocole Radio Net Flow (RNF) est le protocole South Bound (SB) introduit dans le contrôleur ODL en tant que protocole de configuration RAN pour (i) collecter des mesures et des configurations de réseau radio et (ii) ré-configurer les paramètres de ces éléments radio. Nous avons implémenté le RNF en utilisant SCTP car c'est le protocole de couche de transport standard dans l'architecture de réseau mobile qui facilite l'intégration de notre contrôleur SDN avec RAN standard. Nous avons développé l'agent de protocole correspondant appelé Radio Net Flow Agent (RNFA). RNFA peut être intégré dans tout eNodeB (c'est-à-dire, station de base 3GPP-LTE) en

tant qu'interface de communication avec le contrôleur SDN. Nous avons introduit trois modules de service dans le Service Abstraction Layer (SAL) d'ODL : le RAN Configuration Manager, le RAN Statistics Manager et le RAN Topology Manager. Tous ces modules de service interfèrent avec le protocole RNF SB en utilisant des bibliothèques spécifiques. Grâce au RNF, ils peuvent collecter des données du RAN et également envoyer des configurations au RAN. Etant donné que le SAL impose le stockage de données transitoires, nous avons introduit l'inventaire RAN qui est une base de données externe pour maintenir l'historique de l'état du réseau à long terme. Nous avons utilisé la base de données No-SQL pour des considérations de performance. Pendant l'exploitation, les modules de service RAN mettent à jour l'inventaire RAN. D'autre part, les applications NB accèdent à l'inventaire RAN pour la récupération d'informations et les paramètres via NB REST (compatible HTTP RESTful) API.

Notre plateforme SDN est prototypée pour supporter l'optimisation du RAN. Il existe trois cas possibles d'utilisation du SDN pour le RAN: (i) optimisation radio, (ii) optimisation de réseau, (iii) équilibrage de charge et/ou virtualisation. En ce qui concerne le RAN, nous pouvons le catégoriser en deux grandes classes en termes de nécessité de coordination : (1) RAN centralisé (C-RAN) et (2) RAN distribué (D-RAN). La plateforme SDN peut fournir un contrôle dynamique sur le réseau de transport frontal et le réseau de backhaul mobile pour allouer la capacité disponible, contrôler la latence et répondre aux exigences de QoS. De plus, l'abstraction de réseau utilisant ODL avec plusieurs protocoles SB peut être utilisée pour ajuster les ressources du réseau de transport (technique de contrôle de flux basée sur OpenFlow pour contrôler la bande passante et l'assignation de QoS) selon les besoins du réseau. Par exemple, étant donné que le contrôleur a la connaissance des éléments du RAN, il peut déclencher un transfert d'utilisateurs mobiles, en particulier des utilisateurs de bordure de cellule, vers des stations de base voisines chargées légèrement, afin d'augmenter les ressources radio et l'amélioration du débit. Il s'agit d'un scénario d'équilibrage de charge typique qui peut être facilement réalisé en utilisant le contrôleur SDN avec une récupération/réponse rapide de l'allocation de ressources dynamique pendant une panne et une reconfiguration entre différentes unités BBU.

Dans l'architecture proposée, nous avons déployé l'optimiseur 4G/5G eICIC comme une application RAN du contrôleur qui a pour tâche de coordonner la planification MAC sur les réseaux LTE. Dans le test et l'expérimentation, nous avons utilisé un simulateur de LTE basé sur Matlab de l'Institut des télécommunications de TU Wien pour les scénarios de réseau mobile LTE. Il simule des scénarios de réseaux cellulaires hétérogènes 3GPP typiques (HetNet) composés de macro et de petites cellules, d'utilisateurs mobiles, de propagation radio et de pathloss. Afin d'expérimenter la fonctionnalité de bout en bout de notre architecture, nous avons imité des eNB pondérés par la lumière qui mettent en ½uvre RNFA et sont alimentés avec des informations relatives à eNB (mesures et configuration) par Matlab en utilisant la connexion de socket User Datagram Protocol (UDP) standard. Ensuite, le eNB émulé agit comme une station de base réelle au contrôleur SDN. Le simulateur Matlab et l'environnement SDN (eNBs émulés, contrôleur SDN et MongoDB) sont implémentés dans deux machines physiquement séparées. Nous avons déployé le contrôleur SDN, MongoDB et eNB en tant qu'entités virtuelles dans une machine physique et avons utilisé une autre machine physique dédiée pour le simulateur Matlab.

Au cours du déploiement, nous avons implémenté deux techniques de virtualisation : les machines virtuelles et les conteneurs Docker, sous la même configuration matérielle des processeurs 16 cœurs et 16 Go de RAM fonctionnant avec la version Ubuntu 14.04. La motivation de la validation de notre cadre sous deux techniques de virtualisation différentes est d'analyser la performance et

le comportement du système dans les deux environnements virtuels différents, et de fournir une étude comparative de leur adéquation pour la 5G SDN/NFV.

Comme montre le tableau I, il y a trois cas d'usage possibles du SDN pour le SON : *ICIC/eICIC, CoMP et Load Balancing*.

Cas d'usage	**Description**
eICIC/ICIC (Plus généralement dynamique/rapide SON)	Different algorithmes d'optimization peuvent être intégrés comme *applications NB* dans le contrôleur SDN qui possède une vue globale du RAN.
CoMP (e.g. Ordonnancement coordonné, Selection dynamique du point)	L'ordonnancer coordonné et la selection dynamique du point peuvent être mises en oeuvre directement comme *applications NB* où Ils obtiennent les paramètres de saisie requis de l'inventaire RAN via les API REST.
Load balancing (Entre stations de base)	SDN peut déclencher *hand over* pour les utilisateurs de bordures d'une cellule pour qu'ils soient mieux servis par une station de base voisines moins chargée.

Table I – Platforme SDN pour le cas d'usage de SON .

Software-Defined Backhauling sans fil

Notre architecture est définie, en utilisant le principe SDN, pour apporter la programmabilité dans le backhauling mobile de TWC. Le principal défi est de concevoir un contrôleur qui fournit des API NB pour les applications de contrôle/reconfiguration réseau backhaul en temps réel et d'intégrer les périphériques sans fil pour fonctionner sous SDN. Notre solution se compose de différents composants, y compris le contrôleur SDN, les modems sans fil définis par logiciel et le commutateur d'agrégation de trafic supportant OpenFlow pour la connexion à la passerelle Internet et au réseau mobile. Il existe deux types de modems sans fil définis par logiciel dans notre architecture : (i) Infra Modem (IM), un modem fixe sur le sol connecté aux réseaux de fournisseurs de services, et (ii) User Modem (UM), un modem mobile placé dans le train relié au réseau terrestre via IM. La messagerie unifiée est reliée à différents nœuds de service (WiFi, LTE Femto, CCTV, CBTC, etc.) via LAN dans le train. Chaque IM peut simultanément servir plusieurs MU.

Etant donné que notre cadre peut offrir un niveau de QoS service, nous plaçons les systèmes de contrôle des trains (CCTV, CBCT, etc.) et les systèmes d'information voyageurs (PIS) dans l'Internet mondial plutôt que comme une entité distincte. En raison de l'avantage d'un débit de données haut débit/liaison descendante élevé pour les services haut débit mobiles, nous nous appuyons sur LTE en tant que technologie pour fournir la connectivité du téléphone mobile dans le train. Les passagers à bord peuvent utiliser LTE femto pour avoir une connectivité mobile sans faille ainsi que des hotspots WiFi pour l'accès à Internet. Profitant des avantages de la fourniture de QoS dans OpenFlow, notre backhaul sans fil défini par logiciel peut adopter dynamiquement la QoS pour différents flux de trafic. Cela est nécessaire pour certains services en train, par exemple le CBTC est le service le plus critique dans le train qui a besoin d'informations précises sur l'emplacement du train afin de gérer efficacement les opérations de la salle de contrôle.

La conception de contrôleur suit étroitement l'architecture de notre travail récent "Generalized SDN Plaform for RAN", où nous avons utilisé une base de données externe (DB) avec un contrôleur ODL pour stocker les mesures et les détails de configuration à utiliser par les applications NB. Le protocole de contrôle et de gestion du modem (MCMP) est le protocole South Bound (SB) que nous

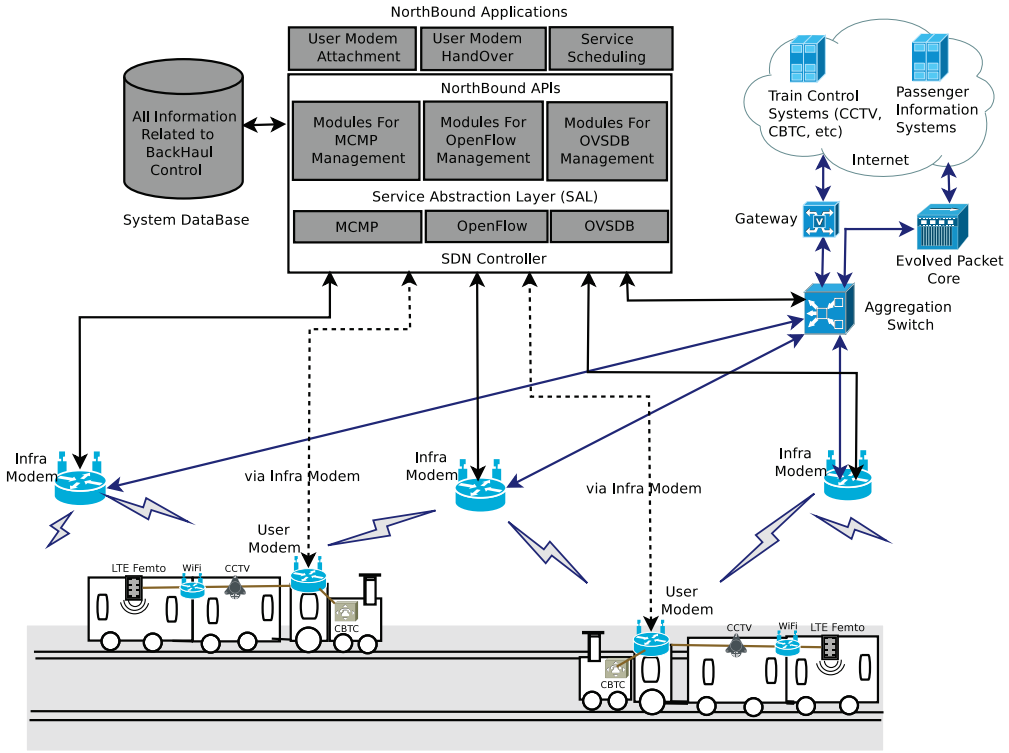


Figure V – *Software-Defined Wireless Backhauling for Train-to-Ground Systems.*

avons introduit en plus du OpenFlow standard (OF) et du Open vSwitch Database Management Protocol (OVSDB) pour gérer les messages liés aux procédures de gestion de la mobilité telles que User Modem Attachment and User Modem Handover. Ces messages proviennent des messages de contrôle radio à faible niveau (LLR-C) entre MI et UM. Le MCMP est défini à l'aide de ZeroMQ (ZMQ) en raison de la haute performance et de la capacité d'établir des connexions entre plusieurs terminaux, ce qui est nécessaire car le contrôleur doit gérer la requête/réponse de plusieurs Infra Modems en parallèle pour éviter la latence dans les décisions de contrôle. Les applications NB utilisent le MCMP pour envoyer et recevoir des messages de/pour IM pour la gestion de la mobilité. Les modules de gestion MCMP dans le contrôleur interagissent avec le DB système pour maintenir l'information relative à la mobilité à utiliser par les applications. OF et OVSDB sont utilisés par les applications pour interagir avec OVS pour établir la programmation de services QoS à la fois dans la messagerie instantanée et UM. Nous nous sommes intéressés à trois procédures principales liées à la gestion de la mobilité et la planification des services : namely UM Attachment, UM Handover, and Service Scheduling.

Nous avons développé des prototypes de logiciels ainsi que des prototypes matériels pour démontrer la validité de notre architecture dans les cas d'utilisation de l'attachement de l'utilisateur et du transfert. Nous avons montré que, grâce à l'approche de définition de réseau logicielle, nous pouvons intégrer la programmation dans le backhaul mobile des réseaux de communication train vers terre qui permet également d'avoir un contrôle strict des ressources dans le réseau de backhaul sans fil. Pour finaliser ce travail, nous avons migré notre prototype vers un cadre d'orchestration basé sur le micro-service pour démontrer la faisabilité de faire évoluer notre architecture en tant que service vers le futur cadre 5G.

As shown in Table II, there are two main use cases of SDN for wireless backhaul: Mobility Management and Service Scheduling.

Comme montre le tableau II, il y a deux cas d'usage du SDN pour le backhaul sans fil : gestion de mobilité, et ordonnancement de service.

Cas d'usage	Description
Gestion de mobilité	Les procedures d'association et du *Handover* de la partie mobile (modems dans le train) du backhaul sans fil sont gérées par les applications respectives de la NB du contrôlleur SDN.
Ordonnancement de service (Amélioration QoS)	La fonctionnalité *Priority queuing* du OpenFlow peut être utilisée pour activer un ordonnancement en avance en intégrant OpenFlow dans les équipements mobiles du réseau (modems d'utilisateurs et d'infrastructure) et en les gérant par une application d'ordonnancement de services au sein du NB du contrôlleur SDN.

Table II – Platformne SDN pour le cas d'usage du backhaul sans fil.

Software-Defined *Cloud RAN*

La motivation de cette recherche est de définir une architecture C-RAN très générique avec des modules logiciels eNodeB, dans le paradigme du cloud, fonctionnant sur des serveurs multicœurs et contrôlés et virtualisés à l'aide de SDN. L'utilisation du SDN dans le contexte C-RAN apporte un cadre de gestion unifié pour utiliser et optimiser efficacement les ressources réseau et radio qui nécessite par ailleurs des contrôleurs de réseau dédiés et des API.

Dans cette recherche, nous avons utilisé les différentes fonctions de la pile LTE pour la réalisation de l'architecture C-RAN. Dans toute l'architecture, l'objectif principal est de virtualiser le processus utilisateur (UP) avec le contrôle SDN pour différentes entités fonctionnelles dans l'architecture C-RAN telles que le fronthaul, la gestion des ressources radio et le traitement des signaux conjoints. L'étude de simulation réalisée dans l'étude précédente dans C-RAN montre que l'utilisation des ressources s'améliorerait jusqu'à un facteur 3 par la répartition de la charge de l'utilisateur (entre les cas de distribution uniformes et non uniformes). Précisément, la virtualisation dans notre approche considère la répartition de la charge correspondant aux fonctions dépendant de l'utilisateur de RAN lors de l'attribution de supports de données (DRB), c'est-à-dire les fonctions L1-UP, L2 et L3. Cette approche élimine les fonctions dépendantes de la cellule de virtualisation telles que la conversion parallèle à série (P/S), la conversion série à parallèle (S/P), la transformée de Fourier rapide (FFT), la transformée de Fourier rapide inverse (IFFT), le préfixe cyclique (CPin), les fonctions de sortie de préfixe cyclique (CPout), de mappage de ressources et de démappage.

Nous avons conclu avec la liste des architectures C-RAN définies par logiciel où l'objectif principal est de virtualiser le processus utilisateur. Bien qu'il y ait eu nombre de contrôleurs disponibles RPS comme plates-formes opensource, nous comprenons à partir de l'analyse initiale que OpendayLight contrôleur SDN de la communauté LINUX et contrôleur ONOS de ON.LAB ont plus de fonctionnalités et une vision plus globale pour les futures solutions SDN. Comme ODL était très soutenus et reconnus par les fournisseurs et opérateurs de Teleco au moment de la recherche, nous utilisons l'ODL comme référence SDN pour définir les architectures C-RAN définies par logiciel. Le SDN contrôleur utilise deux protocoles différents tels que BBU (notre protocole spécifié pour C-RAN) et Openflow Protocole. C'est la caractéristique unique du contrôleur SDN OpendayLight

d'avoir plusieurs connexions au SP et qui peut être étendu plus loin pour avoir plus de plugins de protocole de SB selon l'exigence différente.

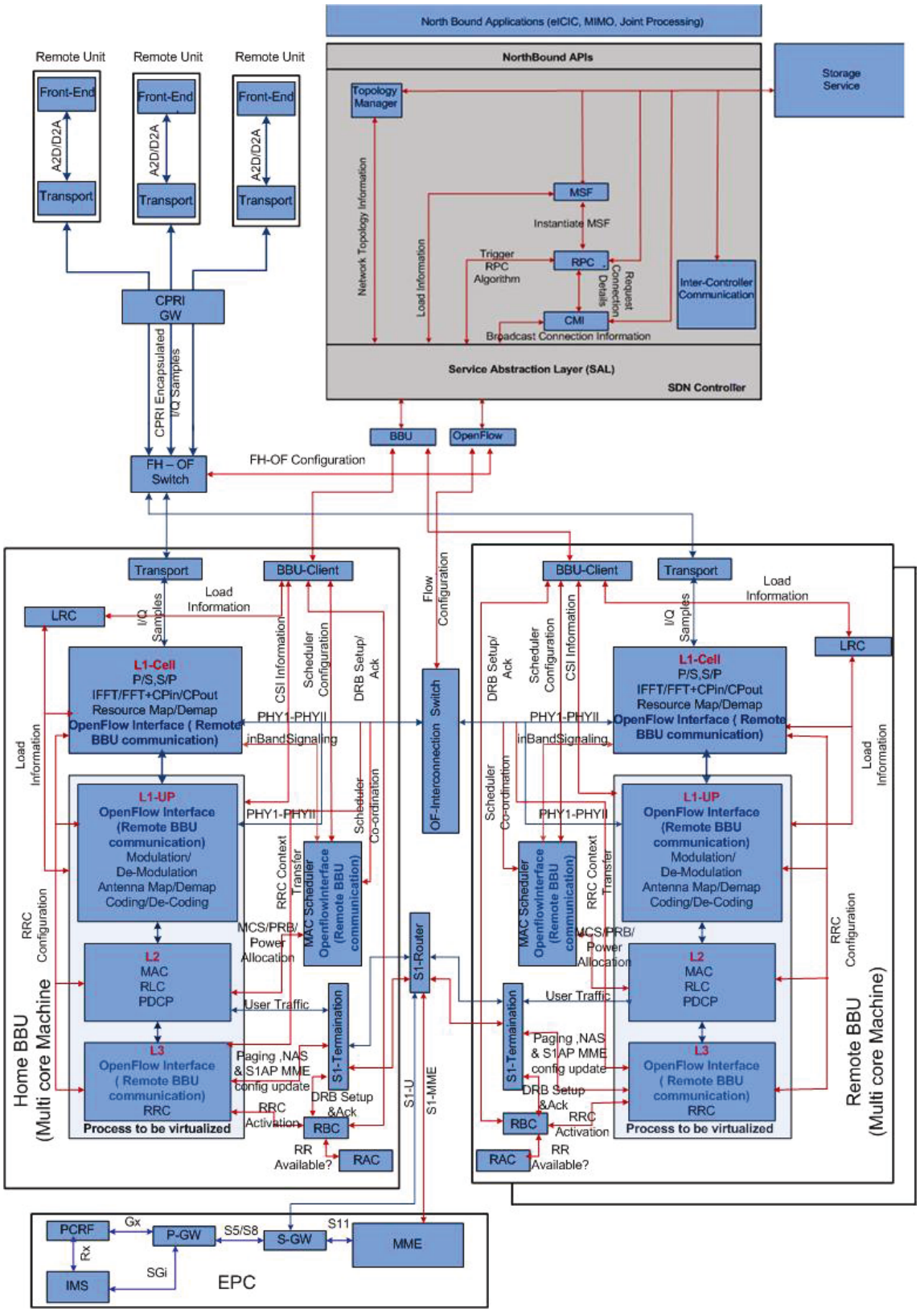


Figure VI – Software Defined Cloud RAN Architecture

Nous décrivons en détail les composants et leurs fonctionnalités que nous avons spécifiés pour les solutions C-RAN alignées SDN qui sont communes dans les quatre architectures.

***FrontHaul OpenFlow Switch* :**

Le commutateur OpenFlow établit la commutation entre les RRH et les BBU et contribue également à récupérer les liaisons de transport entre les RRH et les BBU en cas de défaillance BBU, avec le contrôle direct du contrôleur SDN central. Le commutateur OpenFlow a la possibilité d'être reprogrammé dynamiquement à partir du contrôleur SDN pour réorienter le trafic correspondant à une destination vers une autre et qui est nécessaire pour distribuer le trafic radio des RRH vers les différentes unités de traitement dans le pool BBU selon les conditions de charge.

***OF-Interconnection Switch :* :**

Le commutateur OpenFlow est utilisé pour interconnecter des BBU physiquement séparées ou des machines de traitement situées dans le pool BBU. Il est directement configuré par le contrôleur SDN centralisé pour diriger les flux selon diverses informations d'entête telles que l'adresse source et la destination de chaque flux. Le commutateur OpenFlow a la capacité de travailler à la fois sur la commutation L2 et le mode de routage L3 et est un aspect utile pour le contexte RAN où nous avons le réseau situé entre les interfaces L2 et L3.

***BBU Client* :**

BBU client est le module de communication côté client situé dans chaque BBU pour établir une communication avec le contrôleur. Il envoie et reçoit des données et des détails de configuration vers et depuis le contrôleur. Dans les quatre architectures, nous utilisons deux protocoles différents pour la communication et la configuration :

- Protocole BBU : le protocole South Band BBU, qui est un protocole SDN dédié que nous avons défini pour C-RAN. Toutes les BBU du pool MSS-BBU peuvent établir une communication avec le contrôleur en utilisant le protocole BBU. Le protocole BBU a défini des règles pour envoyer et recevoir des données à partir des BBU pour la procédure de configuration DRB, la surveillance de l'utilisation des ressources, la détection des défaillances, l'abstraction du réseau, l'allocation des ressources et la procédure de réattribution, Réception, etc.
- Protocole OpenFlow : Il peut être utilisé pour configurer le front-haid OpenFlow et le commutateur d'interconnexion, car ils ont un agent OpenFlow intégré pour la communication du contrôleur et l'exécution des commandes du contrôleur.

Voici les fonctions typiques du **BBU Client** :

- Au cours du processus d'initialisation, il envoie l'adresse MAC/IP du BBU correspondant au contrôleur SDN.
- Il lit l'état de la ressource et de la charge à partir du Local Resource Controller (LRC) à chaque intervalle de temps prédéfini et les envoie au contrôleur s'il y a un changement dans leur valeur par rapport à la précédente. Cela évite toute duplication de l'état de la ressource envoyée au contrôleur.
- BBU Client envoie des rapports de qualité d'expérience (QoE) et de Channel State Information (CSI) des fonctions UP au contrôleur SDN.
- Il reçoit des détails de connexion et d'autres informations du contrôleur SDN pendant la configuration de DRB et configure diverses fonctions (Scheduler, UPs) dans le BBU pour établir la coordination et la communication à distance.

- Dans le cas d'utilisation de la requête de configuration de DRB impliquant Radio Bearer Control (RBC), le RBC envoie la demande via BBU Client au contrôleur.
- BBU Client est le module générique qui peut être enrichi avec toute nouvelle extension fonctionnelle dans l'unité BBU dépendant des scénarios de déploiement C-RAN et des options de fractionnement.
- Local Resource Controller (LRC) est un module situé dans chaque BBU pour générer un rapport d'utilisation des ressources et le stocker dans la base de données locale qui lui est associée. Le client BBU lit cette base de données dans un intervalle de temps défini et le signale au contrôleur SDN s'il existe une variation dans le rapport d'utilisation de la ressource.

***OpenFlow Interface* :**

Il s'agit d'un module spécialisé dans chaque BBU pour supporter la communication via le commutateur d'interconnexion OpenFlow dans le pool BBU, en particulier le DRB établi à travers des BBU distribuées, c'est-à-dire pendant la communication avec le BBU distant. Il est utilisé pour compacter les données extraites de la fonction de démaillage pour discriminer les données correspond à des flux différents. Afin d'établir une communication entre le planificateur local et le planificateur distant, il doit y avoir une interface OpenFlow pour définir le message correspondant aux flux particuliers. Pendant la phase initiale de confi

guration du DRB, l'interface OF est configurée par le contrôleur avec l'identifiant DRB et l'adresse IP correspondante de ces BBU sert le DRB particulier.

***InBand Signaling for Mapping, De-Mapping et OpenFlow Interface* :**

Il y a une signalisation inband depuis le planificateur qui va à la fonction Mapping et De-Mapping ainsi qu'à la fonction OF Interface pour informer sur l'allocation du bloc de ressources physiques (PRB) pour les DRB distants.

***Software Architecture of SDN controller* :**

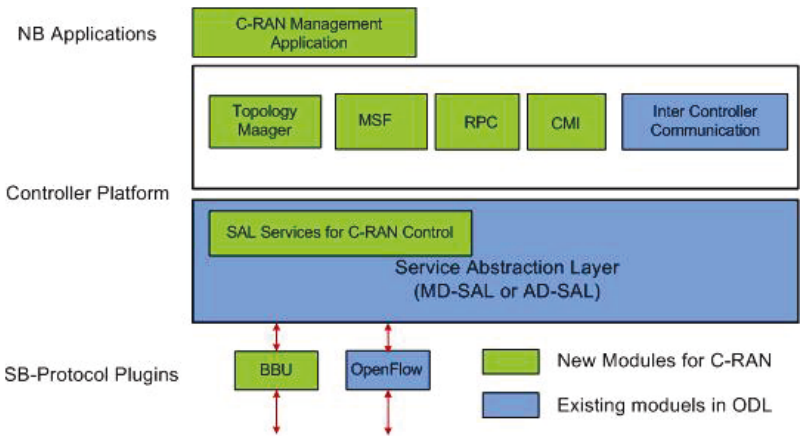


Figure VII – *SDN Controller Architecture: Cloud RAN[8].*

L'architecture générale du logiciel C-RAN est décrite dans la figure ci-dessus, où SDN joue un rôle clé dans la gestion des ressources réseau ainsi que des ressources IT. L'architecture se compose de six composants principaux : le contrôleur SDN, l'unité distante ou RRH, la passerelle CPRI, les

unités de bande de base (BBU), le réseau principal et les réseaux de transport activés OpenFlow (pour l'interconnexion frontaul et BBU). L'objectif principal de cette architecture est de réaliser la virtualisation en distribuant les fonctions de Traitement d'Utilisateur (User Processing-UP) des Data Radio Bearers (DRB) sur une ou plusieurs BBU en fonction de leur capacité et de leur charge. Une réattribution ou une migration du processus peut également être effectuée en cas de surcharge. La virtualisation des ressources de traitement peut être réalisée grâce à l'une des technologies de virtualisation disponibles telles que VM, Containers et Unikernel. L'architecture C-RAN définie par le logiciel offre une faisabilité suffisante pour réaliser la virtualisation, l'allocation des ressources à la demande, l'équilibrage de la charge, le traitement des signaux, la détection et la gestion des défaillances des ressources.

Cas d'usage	**Description**
Dynamic User Processing Split (Virtualization)	Un contrôlleur SDN ayant une vue globale et des ressources informatiques dans l'infrastructure C-RAN peut contrôler l'emplacement de DRBs dans les BBUs appropriés.
Failover Recovery	Une application de surveillance dans le NB d'un contrôlleur SDN est capable de détecter et déclencher une procedure de reprise de DRBs affectés pendant des défaillances matérielles ou logicielles.
Contrôle Fronthaul	Un contrôlleur SDN utilisant le protocole OpenFlow peut mettre en oeuvre un algorithme de routage pour proteger le trafic radio au sein du réseau de transport fronthaul. Cela est utile pour des scénarios où les fournisseurs de services telco ou non-telco partagent la même infrastructure du réseau de transport.
ICIC (eICIC, CoMP)	La possibilité d'une gestion et d'une coordination conjointe des resources radio depuis un contrôlleur SDN permet la réalisation des algorithmes ICIC en forme d'applications au sein du NB du contrôlleur SDN.

Table III – Platforme SDN pour le cas d'usage du C-RAN.

Alors que l'objectif principale de l'architecture est de mettre en oeuvre *dynamic user processing split (virtualization)* en utilisant une approche SDN, elle peut être également appliquée en d'autres cas d'usage comme *Failover Recovery*, Contrôle Fronthaul et ICIC, comme indiqué dans le tableau III.

Résultats et conclusion

Dans cette thèse, nous avons proposé des architectures définies par logiciel pour trois cas d'utilisation différents des réseaux sans fil de prochaine génération tels que les réseaux auto-organisateurs, le fractionnement dynamique des utilisateurs et le *backhaul* sans fil. Dans cette thèse, notre objectif principal était de relever les défis liés à l'expansion de la capacité, l'amélioration de la qualité de service, la souplesse et la réprogrammation du système. Dans la première partie de notre recherche, nous avons présenté deux implémentations de l'architecture RAN définie par logiciel pour les réseaux d'auto-organisation, l'une utilisant des machines virtuelles et l'autre utilisant les *Docker containers* récemment introduits. L'architecture est décrite à l'aide d'un modèle analytique qui est également utilisé pour dériver le temps de service des deux options de déploiement telles que VM et Docker. Les mesures et les calculs ont montré que l'architecture basée sur le Docker

fournit un meilleur taux de service que l'architecture basée sur des machines virtuelles. Ceci est utile pour le développement futur de réseaux mobiles définis par logiciel. Bien que notre plateforme SDN soit principalement développée pour supporter la gestion de SON, elle peut être étendue pour supporter différents cas d'utilisation dans *Cloud RAN* et la future 5G pour l'allocation dynamique des ressources et l'équilibrage de la charge du trafic entre les différentes unités de base. De plus, nous nous attendons également à ce que la méthode d'analyse présentée ici soit également utile pour étudier d'autres systèmes similaires tels que *Cloud RAN*.

Dans la seconde partie de notre recherche, nous avons mis l'accent sur la programmation dans les réseaux de *backhaul* sans fil, en particulier dans le réseau de communication train-sol qui souvent ne parvenait pas à s'adapter aux exigences de QoS strictes pour les dispositifs sensibles au temps des systèms embarqués. Bien qu'il existe diverses solutions pour fournir des services de communication à haute vitesse pour les systèmes de transport, ils ne peuvent offrir une QoS garantie pour certains services dans des conditions de circulation dynamiques. Le cadre proposé permet la fourniture du réseau en tant que service à la demande et adapte les ressources en conséquence. Les principaux avantages de notre *framework SDN* sont de fournir une programmabilité dynamique et une reconfigurabilité pour le *backhaul* sans fil pour que la gestion de la mobilité soit directement pilotée par un contrôleur SDN déployé localement plutôt que par une entité de gestion de mobilité déployée dans les réseaux centraux comme dans les systèmes sans fil traditionnels. Prenant les avantages des protocoles OpenFlow et OVSDB, nous sommes en mesure de gérer efficacement les rares ressources radio en utilisant la pré-programmation *via* OpenvSwitches intégrée avec les modems radio. Nous avons implémenté notre architecture dans un logiciel ainsi que dans des prototypes matériels et des scénarios validés liés à la gestion de la mobilité. Dans le prototype basé sur le matériel, les modems peuvent faire une connectivité de *backhaul* mobile avec un débit de données allant jusqu'à 10 Mb/s complètement contrôlés par le contrôleur SDN. Nous avons démontré la procédure complète d'attachement de l'utilisateur avec LTE-Femto soutenant la transmission vidéo en direct dans la liaison descendante. Dans la partie finale de la thèse, nous avons analysé les différentes fonctions et protocoles de l'architecture LTE comme une étape préliminaire vers la compréhension des avantages ainsi que les problèmes que le *cloud RAN* peut traiter. Nous avons résumé les architectures, les avantages et les défis de l'architecture *cloud RAN* dans l'état de l'art. En dernier lieu, nous avons positionné l'approche réseau définie par logiciel pour le RAN en nuage et répertorié les architectures et les avantages possibles. Nous avons également discuté de ces architectures en détail avec diagrammes de séquence décrivant la virtualisation des processus utilisateur, la gestion des surcharges et la récupération des ressources en cas de défaillance.

Bien que nous ayons démontré et validé les architectures possibles et les solutions SDN/NFV pour les futurs réseaux d'accès radio, l'examen d'une telle approche dans les déploiements pratiques de réseaux à grande échelle nécessite des améliorations en termes de performances et d'évolutivité. Comme les technologies SDN/NFV sont essentiellement développées dans le cadre des réseaux de transport, la mise en œuvre d'une telle solution dans les réseaux mobiles nécessite une optimisation supplémentaire de leurs fonctionnalités en utilisant des méthodologies de génie logiciel modernes. Il y a quelques études se concentrant sur l'évolutivité du SDN filaire, mais elles sont difficile à appliqué dans le scénario sans fil, à mesure que la taille d'un réseau augmente, plus de paquets sont envoyés au contrôleur. Il ne fait aucun doute que le contrôleur peut à peine gérer toutes ces demandes entrantes. Il puet simplement améliorer les performances du seul contrôleur centralisé, sans concevoir à partir du niveau de l'architecture, et ne peut intégrer s'adapter à l'échelle lorsque le réseaux devient de plus en plus dense. Par conséquent, SDWN peut intégrer plusieurs contrôleurs physique distribués dans le système. Ces contrôleurs ne seront pas en conflit

avec le «logiquement centralisé» en communiquant et coopérant efficacement entre eux. Dans les travaux futurs, nous optimiserons la fonctionnalité des outils et des logiciels SDN et NFV pour qu'ils correspondent aux critères de performance des réseaux mobiles à grande échelle qui prennent en charge divers cas d'utilisation. Nous validons également les architectures SDN proposées pour *Cloud RAN*. L'Intelligence Artificielle (AI) est le moteur clé pour apporter des décisions fondées sur la connaissance dans les réseaux 5G. Cela nécessite l'intégration d'une telle intelligence dans les futurs contrôleurs SDN et les outils NFV qui devraient reconfigurer les dispositifs de réseau et les fonctions IT pour mieux satisfaire les besoins des utilisateurs en termes de QoS et QoE.

SDN Based Service Oriented Control Approach
For Future Radio Access Networks

Software-Defined Networking (SDN) has emerged as a new intelligent architecture for network programmability. The primary idea behind SDN is to move the control-plane outside the switches and enable external control of data-plane through a logical software entity called controller. Such approach benefits mobile network management by brining complete intelligence to the logically centralized controller. Network Function Virtualization (NFV) is the process of relocating or migrating network functions from dedicated hardware to generic servers. SDN and NFV are two closely related technologies that are often used together. The traditional mobile network architecture due to its strongest coupling between control and data planes along with limitations in scalability and flexibility requires the usage of cloud computing along with the recent revolutionary approaches in networking such as SDN and NFV to have an architecture that deploys on demand "Network-as-a-Service" for users. The global research focus of this thesis falls in to two main use cases of next generation mobile networks such as **Telco** and **Vertical**. In the telco use cases, we exploit the advantages of SDN to have flexible control framework for both Self-Organizing Networks (SON) and dynamic user processing split. In vertical use case, we apply various advantages of SDN and OpenFlow protocol to efficiently utilize the scare radio resources of wireless backhaul network in the train-to-ground communication system. Our SDN framework in general can be an efficient and alternative solution for RAN management i.e. Radio Optimization, Network Optimization, Mobility Management and Load Balancing can be achieved with such framework. Through analysis and experimentation of SDN frameworks for RAN, we shows that the proposed solutions can bring set of advantages to wireless networks such as flexibility, programmability, unified management, and enables new services.

Keywords: SDN, NFV, Radio Optimization, Mobility Management, Load Balancing, Flexibility, Programmability, Unified Management.

Approche de contrôle orientée services basée sur SDN
Pour les futurs réseaux mobile

Le SDN (Software-Defined Networking) emerge comme une nouvelle architecture pour la programmation des réseaux. A l'origine, l'idée du SDN est de déplacer le plan de contrôle à l'extérieur des équipments, et de permettre ainsi un contrôle déporté de l'ensemble depuis une entité logicielle logique nommée « contrôleur ». Le principal avantage d'une telle approche est de centraliser donc toute l'intelligence de gestion du réseau dans le contrôleur, qui s'appuie pour cela sur des protocoles standard et assure par ce biais la reprogrammation de la totalité de la partie du réseau sous son contrôle. L'évolution technologique vers le SDN est toujours en cours dans des scénarios de déploiement programmable et fléxible des réseaux mobiles. Le NFV (*Network Function Virtualization*) est le processus de déplacement ou de migration des fonctions réseau d'un équipment dédié de réseau vers des serveurs génériques dans le Cloud. Les SDN et NFV sont deux technologies étroitement liées qui sont souvent utilisées ensemble. Le couplage fort entre les plans de contrôle et de données, ainsi que les limitations en matière de passage à l'échelle et de fléxibilité, font que la virtualisation des réseaux mobiles actuels nécessite non seulement l'utilisation du Cloud Computing mais aussi les récentes innovations telles que SDN et NFV pour pouvoir permettre un déploiement à la demande des services réseaux (*Network-as-a -Service*) aux utilisateurs. Les lignes de recherche globales de cette thèse s'inscrivent dans deux principaux cas d'utilisation. Ces cas d'utilisation, bien qu'appelés de la « prochaine génération de réseaux mobiles », sont le « Telco » et le « Vertical », qui apparaissent ici couplés, les deux étant traditionnellement comlétement séparés. Dans les cas d'utilisation de "télécommunications", nous exploitons les avantages de SDN pour avoir un cadre de contrôle flexible pour les réseaux d'auto-organisation (SON) et la division de traitement dynamique des utilisateurs. Dans le cas d'utilisation de « verticale », nous appliquons divers avantages du protocole SDN et OpenFlow pour utiliser efficacement les ressources radio du réseau de backhaul dans le système de communication train-sol. Notre cadre d'étude du SDN, en général, peut être une solution efficace et alternative pour la gestion RAN (*Radio Access Network*), c'est-à-dire pour des objectives comme l'optimisation des ressources radio, l'optimisation du réseau, la gestion de la mobilité et l'équilibrage de la charge, peuvent être atteint avec ce cadre. Grâce à l'analyse et l'expérimentation concrète des SDN et NFV pour le RAN, nous montrons que les solutions proposées dans ce travail peuvent apporter un faisceau d'avantages évidents aux réseaux mobiles tels que la flexibilité, la programmabilité, la gestion unifiée et la mise en œuvre de nouveaux services.

Mots-clés : SDN, NFV, Optimisation radio, gestion de la mobilité, équilibrage de charge, flexibilité, programmabilité, gestion unifiée.