ACKNOWLEGEMENTS

I would like to thank my thesis supervisor at Polytechnic of Turin,CURRI VITTORIO, for his consistent support and guidance during the running of this project.

Chang Lyu, Turin, April 2021

ABSTRACT

With the increase of users and the popularization of the internet, the needs of users have changed, business needs have become more and more complex, and conventional network architecture has become increasingly unable to meet the needs

of both users and businesses. International research organizations and individuals have launched the research on the new network architecture. SDN also emerged at this time. As a new network innovation architecture scheme, SDN has burned a research boom in the industry since its birth. Software-defined networking (SDN) separates the control plane from the data forwarding plane, which is presented as a new kind of architecture of networking control and management. SDN is designed to separate the control plane from the data forwarding plane, support the centralized network state control, which shows the transplant view from the underlying network devices to the upper layer applications. And SND has flexible programming capability, and the emergence of SDN has effectively improved the automated control and management of networking.

This article, firstly mainly introduce and summarize the functionality and significance of SDN architecture, and its most important part -Open Daylight, which serves as the core controller of SDN.

Then specifically introduce YANG modeling language, which is extensively used in SDN architecture, we mainly summarize syntax, semantic, and functions of YANG modeling language, and also pave the way for the subsequent introduction of SDN-based network protocols.

Secondly, we mainly summarize the two types of SDN-based network protocols, NETCONF and RESTCONF. Both NETCONF and RESTCONF are also based on YANG modeling language. We summarize the operation methods and functionality of NETCONF and RESTCONF at length. On the other hand, the emergence of NETCONF has greatly simplified the process of a network configuration for clients. In like manner, RESTCONF has also advantages such as flexibility, simple application, scalability, and others. In this article, we researched RESTCONF in detail, focusing on its resource access, information communication, and CRUD operation mechanisms.

Finally, we mainly introduce one of the significant southbound protocols based on SDN architecture , which is as known as OpenFlow protocol. OpenFlow protocol is a currently really popular protocol applied in SND , which enables SDN to implement separation between the control plane and the data forwarding plane. As this part , we mainly introduce its functionalities including OpenFlow controller and OpenFlow switch . And OpenFlow switch that is as an important component consist of three parts:OpenFlow table , secure channel , OpenFlow protocol. We describe an detail about how a packet goes through a OpenFlow switch , and the logic sequence of flow table in OpenFlow switch.

This article is basically a literature review, main purpose is to summarize the current hot topic in the fast-expanding field of optical networking.

CONTENTS

1

INTRODUCTION

1.1 RESEARCH CONTEXT

In recent years, along with the rapid development of the services such as big data, mobile networking and e-commerce , as well as the development of emerging novel technologies such as virtualization and cloud computing, the conventional networking technologies and architecture has become increasingly deficient for the requirement of rapid configuration, on-demand invocation as well as the requirement of automatic load balancing. Under the conventional networking architecture, the control logic plane and data forwarding plane of Networking are tightly coupled to network equipment. Thus, in order to build the specified network topology, the network administrator need to deploy each network node one by one through the configuration interface of network devices. There is always the over-loaded workload to configure for network device produced by the different suppliers, or even different models of network device under the same supplier, since the administrator always need to use the different configuration methods, which undoubtedly complicates the configuration procedure of network equipment. Furthermore, once the network node occur errors, the speed at which the network administrator is able to respond will be really limited under the conventional networking architecture.

Therefore, Stanford University proposed a type of new networking organization architecture, which is called Software Defined Networking-SDN. The original intention of SDN is to decouple between the control plane and data forwarding plane, in order to achieve the separation and independence of networking management control and data forwarding, so as to simplify networking management. Through the powerful programming capabilities of the control plane of SDN architecture, SDN is able to greatly improve the automated management capabilities of Networking, so as to better adapt to the current needs of networking business.

From the early 1990s, people proposed Active Network with a programming interface (network API); to the beginning of the 21st century, the IETF proposed a framework for separating the control plane from the forwarding plane; and then 2005, Stanford University researchers developed Open Flow framework with the centralized control

logic, SDN architecture is becoming increasingly matured and developed.
SDN is a programmable network paradigm, which is able to independently program network devices through application software to control the overall behavior of the networking. The architecture of SDN is composed of two parts: network controller and network equipment. The network controller integrates the control right of network equipment behavior and the management right of network resources, while the network equipment is responsible for providing the execution unit in the network system, Compose the network topology and undertake the task of data forwarding. The network controller is deployed in a server which is independent of the network equipment , and remotely controls the network equipment through the southbound interface, which achieve the powerful functionalities of automation and interaction.

1.2 PURPOSE

SDN provide a more automated networking architecture with its programmability, separation of the control plane and the data forwarding plane, and centralized control logic. The purpose of this article is to introduce and summarize the specific structure of SDN networking architecture as well as its controller OpenDaylight, and to analyze two types of relevant protocols used in SDN, which are NETCONF and RESTCONF respectively. And I also summarized YANG modeling language, which is extensively used in NETCONF and RESTCONF as well as other SDN-based protocols.

1.3 THESIS OUTLINE

The rest of this article is organized as follows:

Chapter 2 quickly and comprehensively introduces the research background and significance of SDN architecture. I mainly introduce the structure and functions of SDN architecture as well as its advantages.

Chapter 3 first simply introduces the background of OpenDaylight as SND controller. I mainly introduce the structure of OpenDaylight, which consist of northbound interface layer, control platform layer, southbound interface layer.

Chapter 4 mainly introduces Yang modeling language used in NETCONF protocol and RESTCONF protocol, and why use YANG modeling language. And I mainly introduce format, syntax, semantics of YANG as well as the actual example of NETCONF with instantiation of XML format.

Chapter 5 simply introduces the background of NETCONF, why NETCONF. Then I mainly summarize function and structure of NETCONF as well as its operation methods.

Chapter 6 first introduces the interdependency between NETCONF, RESTCONF, YANG. Then I mainly summarize the operation methods and functionality of RESTCONF protocol.

Chapter 7 mainly introduce one of the most popular southbound protocols in SDN, it is OpenFlow protocol. Then I introduce the significances and functionalites of OpenFlow ,and what an OpenFlow system consist of.

Chapter 8 is full literature references .

2

NETWORK CONTROL ARCHITECTURE SDN

2.1 INTRODUCTION OF SDN

Traditional network management architecture has been developing more than half a century, which has gathered a large amount of the effort and the wisdom of countless engineers. However , due to the inherent defects , it has become more and more complicated and weak in terms of many different scenarios .Thus ,a new type of network control architecture was born called SDN (software-defined networking) , which is a new approach to network management that enable dynamic , programmatically efficient network configuration in order to prove networking performance , make it more like cloud computing than traditional management [1] .

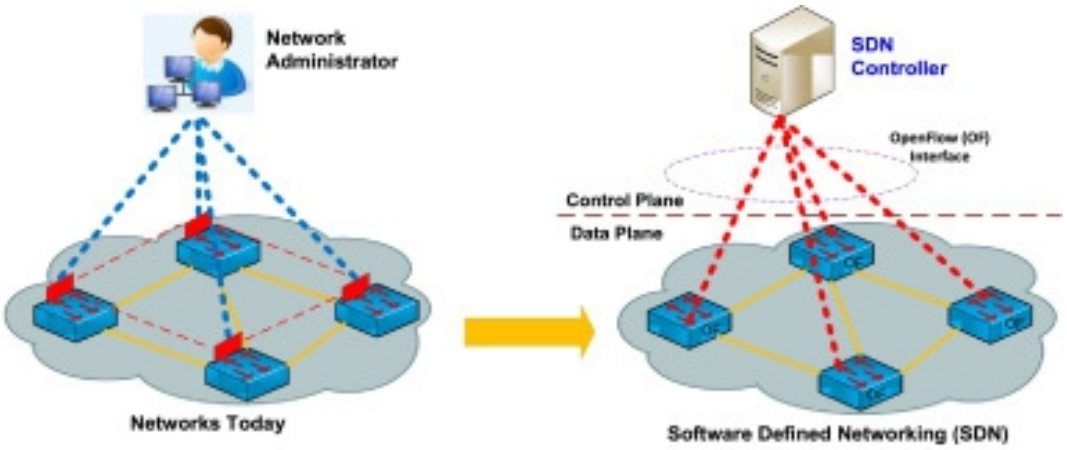

Figure 1-CONTRAST BETWEEN TRADITION ONE AND SDN

Although SDN just emerged for a short time currently , it has shown a very strong vitality. Traditional network management still has the high advantages in terms of security, reliability, maintainability and capability [2]. But along with the development of relevant SDN-based equipment, these advantages of traditional network management architecture will be bound to come with weaker and weaker. Instead, the advantages of SDN architecture will be highlighted with stronger and stronger, and it will also occupy the territory of traditional network management architecture gradually.

2.2 CONCEPT OF SDN

In the existing traditional network control architecture, the control and forwarding of traffic depends on the realization of the network equipment, and the equipment integrates the operating system and special hardware that are tightly coupled with the business characteristics. These operating systems and special hardware are developed and designed by each manufacturer[3].

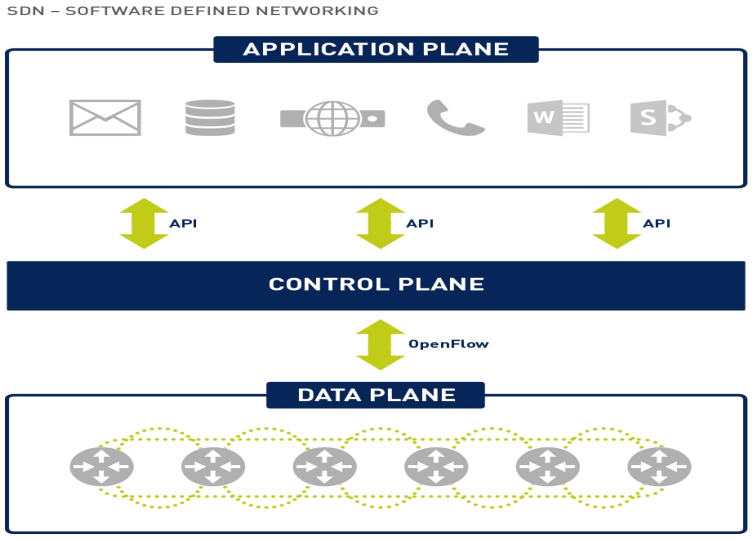

Figure 2 SEPARATELY LAYER OF SDN

SDN is a new type of network architecture. Its design concept is to separate the control plane of network from the data forwarding plane, so as to achieve programmable control of the underlying hardware through the software platform in the centralized controller, and achieve flexibility in network resources on-demand deployment [4].

In an SDN network, network devices are only responsible for simple data forwarding and can use general-purpose hardware, while the operating system that was originally responsible for control will be refined into an independent network operating system, which is responsible for adapting to different business characteristics[5]. The network operating system is compatible with business characteristic, and communication between hardware devices can be achieve via programming[6].

2.3 SDN ARCHITECTURE

2.3.1 BASIC STRUCTURE REVIEW

SDN uses a centralized control plane and a distributed forwarding plane, these two planes are separated from each other[7]. The control plane centrally control he network devices on the forwarding plane via taking advantage of the control-forwarding communication interface, and provide a flexible programmability. The network with the above characteristics should be considered as a generalized SDN, as can be seen in Figure -3 below[8]:

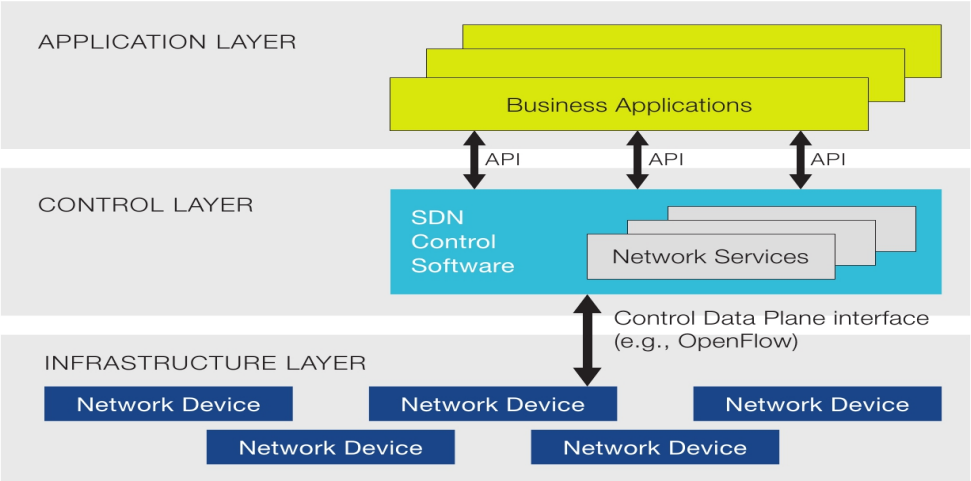

Figure 3 BASIC STURCTURE REVIWE

In the SDN architecture, the control plane centrally control the network devices via the control-forwarding communication interface. And this part of the signaling traffic keep going on between the controller and the network devices, which is independent of the data traffic generated by the communication between the terminals . The network devices generate the forwarding table through receiving the control signaling, and determine the processing of the data traffic based on this forwarding table . So it is no longer necessary to use the complex distributed network protocols to forward data, as shown in following Figure- 4[9]:

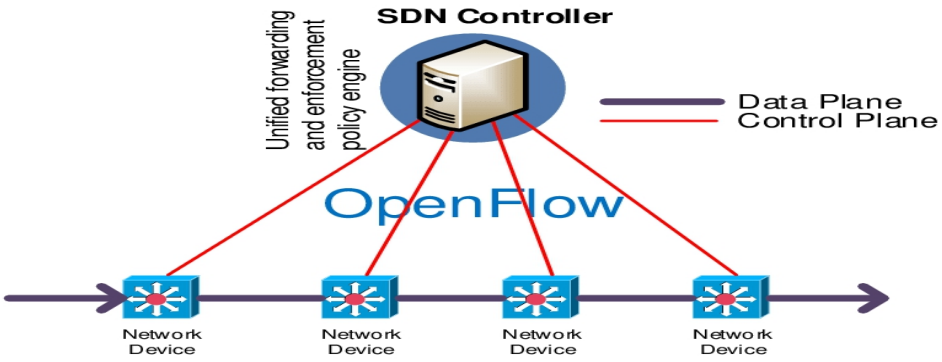

Figure 4 SDN CONTROLLER

SDN is not a kind of specific protocol , but a kind of network architecture which contain the diverse communication interface protocols . For example , the southbound interfaces such as Openflow , Netconf is used to achieve the interaction between the SDN controller and the SDN switch . The northbound API is used to achieve the interaction between the business application and the SDN controller. In this way , the SND-based network architecture will be more systematic and has better perception and management capability hereby promoting the development of the network in a new direction .

2.3.2 OFN-DEFINED SDN ARCHITECTURE

As all shown in Figure - 5 below[10]. Architecture based on ONF consist of four planes , namely data plane , control plane , application plane as well as control and manage plane on the right side[11] . Different protocols are used for interaction between each plane.

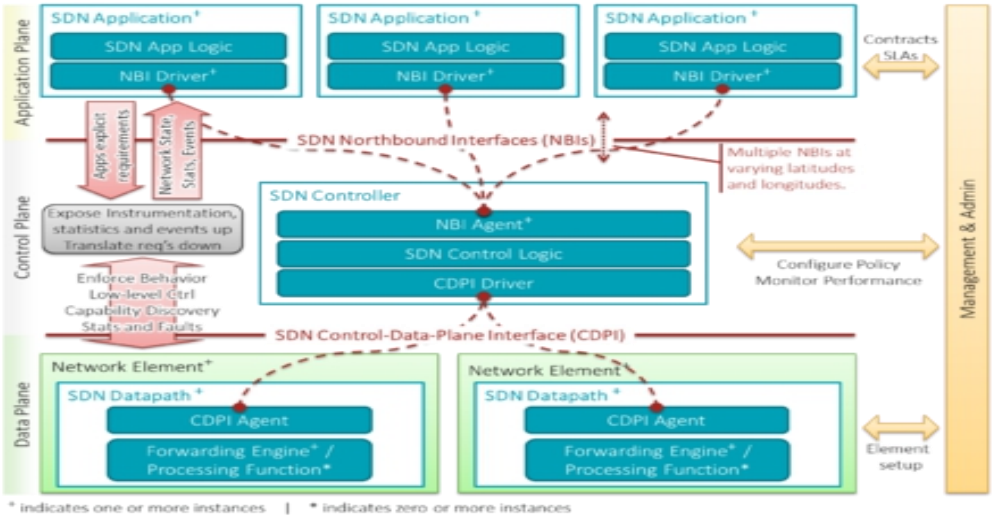

Figure 5 OFN-DEFINED SDN ARCHITECTURE

2.3.2.1 DATA PLANE

As shown in Figure - 5 above about the part of data plane , it consists of signal or multiple network element , and each network element is made up of single or multiple SDN data path . Each SDN data path is a logical network device , it has no control capability and is only used for data forwarding and data processing . It represents logically complete or partial physical resources . A signal SDN data path contains three parts : control data plane interface agent(CDPI) , forwarding engine table , processing function .

2.3.2.2 CONTROL PLANE

As shown in Figure-5 above about the part of control plane, this is so-called SDN controller . SDN controller is a logically centralized entity, which is mainly responsible for two tasks. One of the two tasks is to convert the request of application layer to SDN data path, and another is to provide a abstraction model(states , events) . A signal SDN controller contains three parts, which are respectively “northbound interface agent”, ”SDN control logic ”and “control plane data interface driver ”. SDN controller is only required complete logically , thus it can be made up of multiple controller instances , or a hierarchical controller cluster .Geographically speaking , it can be said that a controller cluster are allocated in a location , or it also can be said that multiple controller instances are distributed in different locations .

2.3.2.3 APPLICATION PLANE

As shown in Figure- 5 above about the part of application plane . It is composed of several SDN applications, and SDN applications are the applications that users pay attention to. It can interact with the SDN controller through the northbound interface, that is, these applications can submit the requested network behavior to the controller in a programmable manner. An SDN application can contain multiple northbound interface drivers (using a variety of different northbound APIs). At the same time, the SDN application can abstract and encapsulate its own functions to provide a northbound proxy interface. The encapsulated interface forms a more advanced northbound interface.

2.3.2.4 MANAGEMENT PLANE

As shown in Figure - 5 above. It is responsible for a series of static tasks, which are more suitable for implementation outside the application, control, and data plane, such as configuring network elements, specifying the controller of the SDN Datapath, and defining the SDN controller and the scope of the SDN application control[12].

2.3.2.5 SDN CONTROL DATA PLANE INTERFACE (CDPI)

SDN CDPI is the interface between the control plane and the data plane. The main functions it provides include: control of all forwarding behaviors, device performance inquiries, statistical reports, and event notifications. A very important value of SDN is reflected in the implementation of CDPI. It should be an open and vendor-independent interface[13].

2.3.2.6 NOTRHBOUND INTERFACE (NBI)

SDN NBI is a series of interfaces between the application plane and the control plane. It is mainly responsible for providing the abstract network views, and enable application to directly control the behavior of network, which includes getting abstraction of the network and functions from the different layers. And this interface should be also a open and vendor-independent interface[14].

2.3.3 SDN PROGRAMMABILITY

2.3.3.2 ACTIVE NETWORK

Basic concept of the active network is to break the mode which the network data can be only transmitted passively, and allow the network nodes to perform the calculation required by user in the data of user. For example, a user of the active network can send a customized compression program to the active nodes in network, and request the nodes to execute the same operation as soon as the corresponding packets are received. DARPA active network architecture can be divided into three main levels respectively: active application (AA), execution environment (EE) , and node operating system (Node OS) , as cab be seen in following Figure - 6 .

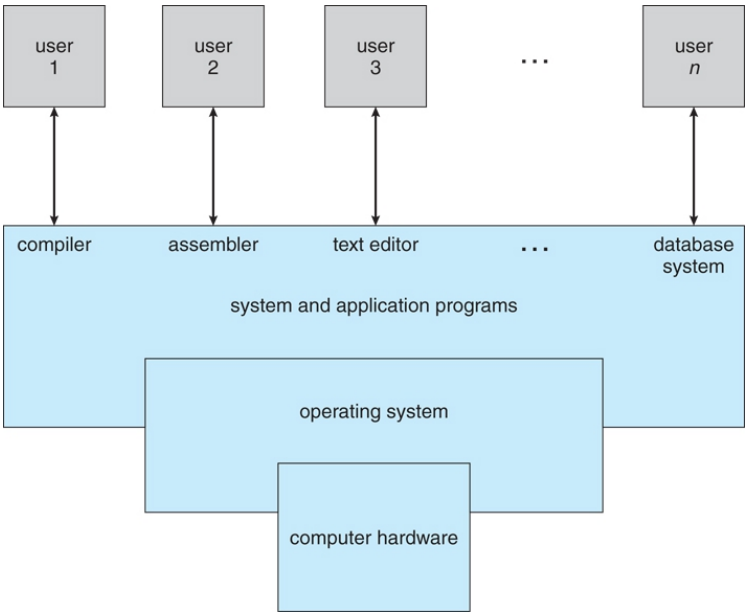

Figure 6 SDN PROGRAMMABILITY

1.3.3.2.1 ACTIVE APPLICATION (AA)

Active application is a program code of a protocol, it is loaded on the active node through active grouping. And the grouping is forwarded and calculated in the active node to complete a certain communication function.

1.3.3.2.2 EXECUTION ENVIRONMENT (EE)

Execution environment is a kind of user-level operating system in Node OS. It is able to support to the execution of multiple AA at the same time, and it is responsible to separate AA from each other. Execution environment (EE) provide AA a callable programming interface. An active network node is able to have multiple execution environment (EE), and each of execution environment (EE) is able to implement a specific function.

1.3.3.2.3 NODE OPERATING SYSTEM (Node OS)

Node OS (Node OS) is similar to the kernel of a general operating system. It is located at the underlying layer of active network nodes, and controls the use of hardware resources of active network nodes. Therefore, EE runs in Node OS, and a Node OS can support multiple EE concurrently, and coordinate the use of resources (memory area, CPU cycle, link bandwidth, etc.) available in the node by EE.

2.3.3.3 DATA TYPE OF ACTIVE NETWORK

The executable code of the node is encapsulated in the data packet in an in-band manner. This model uses data packets to carry codes to add new functions to the network, and at the same time uses cache to improve the efficiency of code distribution, and the programmable router defines a series of operation behaviors based on the packet header of the data packet. The picture shows a format of the active network encapsulation message, which contains IP message, executable program code and user data. The switching device will forward the message according to the original source and destination address. It can be seen that each message or even each message carries a piece of executable code. When the message reaches the switch or router, the code in the message will be distributed to the executable environment of each switch, and then Control the behavior of the switch or modify packets[15].

2.3.4 PROGRAMMABLE SDN

(1) SDN programmable provides a powerful programming interface for developers, so that the network has a good programming ability[16]. For developers of upper-level applications, the programming interface of SDN is mainly reflected in the northbound interface. The northbound interface provides a series of rich API. Developers can design their own applications on this basis without worrying about the underlying hardware details[17]. Like the current programming on the x86 system, there is no need to care about the specific details of the underlying registers, drivers, etc. The SDN southbound interface is used to establish a two-way session between the controller and the forwarding device[18]. Through different southbound interface protocols (OPENFLOW , NETCONF , etc), the SDN controller can be compatible with different hardware devices, and at the same time, the logic of the upper layer application can be implemented in the device. The east-west interface of SDN is mainly used for communication between controllers within the controller cluster to enhance the reliability and scalability of the entire control plane.

(2) Programmability is reflected in many levels, from bottom to top, chip programmable (such as P4, POF), FIB programmable (such as OpenFlow), RIB programmable (such as BGP, PCEP), and device OS Programming, equipment configuration programmable (such as CLI, NETCONF/YANG, OVSDB), controller programmable and business programmable (such as GBP, NEMO)[19].

3

Open Daylight OPERATING SYSTEM

3.1 ORIGIN OF Open Daylight

With the popularity of the Internet as well as the increasing numbers of users ,so the network is overwhelmed. The development of mobile terminals is seemingly a substantial trend, smart phones are constantly being updated, various mobile phone software is emerging one after another, and mobile phones control the Internet at anytime and anywhere, resulting in increasing demand for traffic, and over-loaded networks cannot meet user needs. The network system is huge, the structure is bloated, and it is not flexible enough to adapt to the emerging new business needs, and the service quality cannot be guaranteed. T

The network system is complex, and network operations need to be integrated and coordinated with other IT operations, which makes network deployment difficult. Network updates are troublesome, too many hands-on operations, and network administrators are always over-loaded and laborious to operate or configure the multiple network devices at the same time. Meanwhile improvements can no longer solve the existing network problems, network reform is imperative, so this is the vital reason why SDN came into being, moreover Open Daylight (ODL), which is our leading role today, is served as the crucial part of controller of SDN.

SDN is a new network architecture proposed by the clean slate research group of Stanford University. The traditional network uses distributed strategy work, and the device formulates the forwarding strategy. In the SDN architecture, the device does not run any protocol, and the forwarding table is issued by the controller to the device to realize the separation of the data platform and the control platform. The core idea of SDN is to separate control and forwarding, apply software to network control, and play a leading role, instead of controlling the network by a fixed-mode protocol. The purpose of SDN is to improve the controllability and programmability of the network, and can flexibly provide services of different quality levels according to user needs.

3.2 INTRODUCTION OF Open Daylight (ODL)

ODL is a network controller framework that was born under the drive of network equipment manufacturers in order to counter ONF's idea of weakening and opening network equipment. Although to a certain extent it is a product of the interests of the equipment manufacturers, ODL has developed rapidly in recent years due to its modular, expandable, and upgradeable characteristics, and has become a vital fundament to develop their respective commercial controllers by the large network equipment manufacturers such as Cisco, Huawei, H3C, NEC, etc[20].

The network controller is the core of SDN network. The characteristics of SDN network: centralized management, separation of the forwarding plane and control plane, and network programmability should all be reflected in the network controller. Considering that the network controller is a set of control and management rights of network equipment, the controller itself represents the two characteristics of centralized management and the separation of control and forwarding planes[21]. In order to make the network programmable, the network controller should include the northbound interface facing the upper application software.

Therefore, at the northernmost end of the ODL framework is a collection of northbound interfaces. The devices controlled by the network controller may come from different network equipment manufacturers, so the controller often needs to use a variety of southbound protocols to complete the communication with the network equipment, the protocols include Open Flow, OVSDB, NETCONF, SNMP etc. For each southbound protocol, the controller should contain a protocol module specifically used to analyze and decode protocol messages[22]. We call these protocol modules the southbound interface. At the southernmost end of the ODL framework, there is a collection of southbound interfaces.

3.3 THE STRUCTURE OF Open Daylight (ODL)

Open Daylight is an open source, modular SDN platform that combines three popular engineering technologies: network and software engineering, software-defined networking, model-driven software engineering and network management. It can be used as a controller for networks of any size. The network services running on Open Daylight allow different hardware devices to access in the environment of multi-vendor network. These benefits are from Open Daylight's interface design methods which is able to compatible with a variety of different southbound protocols.

Its sophisticated design architecture allows users to control applications, protocols and functional plug-ins: At the same time, applications can communicate with internal services through the northbound interface.

Open Daylight uses OSGI[23] framework to solve the problems of deployment, dynamic expansion, and isolation by operating architecture; its architecture design is the application layer, control layer, and data layer from top to bottom, as shown in the figure. The application accesses the internal service program of ODL controller through the REST API[24]; the control layer is the core of the entire ODL project, including the basic network service functions and the service abstraction layer (SAL); the data layer mainly completes the data forwarding function.

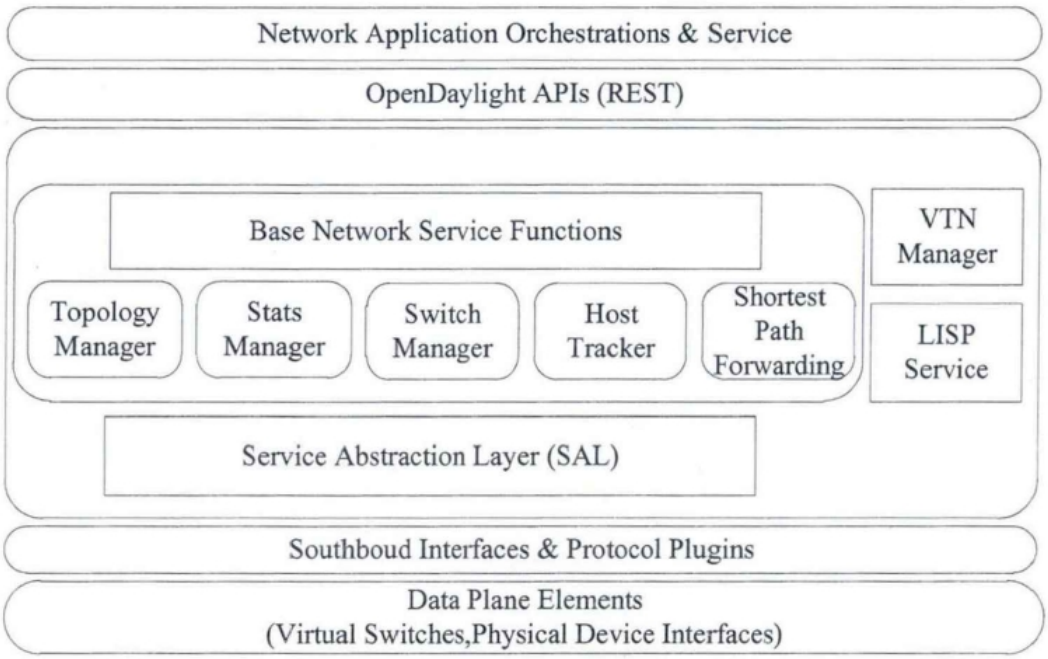

Figure 7 Open Daylight framework

3.3.1 THE SPECIFIC COMPONENTS OF ODL

3.3.1.1 THE STRUCTURE OF ODL

Open Daylight controller architecture is shown in Figure 7 above，Open Daylight can be divided into northbound interface layer (including user interface and network service), control platform layer (ODL core control function), southbound interface layer (including service abstraction layer-SAL and protocol standards such as OpenFlow)[25]. Open Daylight southbound adopts a relatively mature and unified OpenFlow protocol, and the southbound supports multiple protocols through plugins, including OpenFlow 1.0, 1.3, BGP-LS, NETCONF, etc. These modules are dynamically mounted to the Service Abstraction Layer (SAL). The SAL provides services for the upper layer and encapsulates the calls from the upper layer into suitable the protocol format of the underlying network devices. The controller needs to obtain information about the functions and reachability of the underlying devices, which is stored in the Topology Manager. Other components, including ARP handler, Host Tracker, Device Manager and Switch Manager, it generates topology data for Topology Manager.

3.3.1.2 OPEN DAYLIGHT API (REST)

The controller exposes the northbound API to the application program, and provides the OSGI frame and the WEB-based bidirectional REST interface for the northbound API[26]. The OSGI framework provides the applications that run in the same address space with the controller, while the WEB-based REST API runs in a different address space as the controller application. All business logic and algorithms are run in applications. These applications collect network resources through the Open Daylight controller, run their own algorithm analysis, and finally manage and control the network through the controller[27].

3.3.1.3 SERVICE ABSTRACTION LAYER OF ODL (SAL)

The core position of the Open Daylight controller is the Service Abstraction Layer (SAL)[28]. SAL is the core of the modular design of the controller. It supports multiple southbound protocols and provides the consistency services for modules and northbound APPs. When the northbound applications communicate with the southbound protocols, SAL serves as a role of bridge for controller communicating

between the northbound interfaces and the southbound interfaces. The role of the bridge to the south and north, by allowing the southbound protocols module to be dynamically linked into the SAL, SAL allows related apps to dynamically access the protocol modules that have been linked to the SAL through the northbound interface, and then communicate with the network resources in the southbound interfaces .

SAL embeds various protocol module bundles into SAL in the form of plug-ins, and inserts them into OSGi framework through SAL, so as to realize communication between northbound business APP and southbound protocol modules in OSGi framework. Open Daylight completes the collection, management and control of SDN network resources through the services registered in SAL. For example, Topology Manager, this service is used for topology discovery and management services. When the network is running, the topology service module can promptly discover network resources such as newly added nodes and links in the topology, thereby constructing a global network view in the SDN controller. It is convenient for other component services to better use the network resource view for optimal overall planning and management.

Open Daylight provides the unified northbound APIs for the application layer, and the unified extension what application are supported[29]. The northbound interface is currently mainly implemented by Restful tech. Since it has not yet formed a unified standard, the application layer development of the northbound API will be the focus in future research. The control plane mainly includes basic network services and some additional network services. These additional services can be installed and loaded in the form of plug-ins, which increases the flexibility of Open Daylight.

3.4 THE TECHNICAL ESSENCE OF ODL

YANG is a model language used to support NETCONF-based devices. In OpenDaylight, it is used to describe the data structure provided by MD-SAL [30](Model-derived Service Abstract Layer) in Controller and Config generator. MD-SAL is a type-safe level as can be seen in Figure-8 below[31], which can decrease the development difficulties caused by the differences of various equipment components, and provide a unified callable service for the upper layer ， the Config generator in ODL allows to provide service configuration at runtime . Therefore, for the development of each service, we firstly define the corresponding YANG data model internally, register the service into MD-SAL, and develop an open Restful API interface which is able to be called by the application layer through taking a use of Restconf.

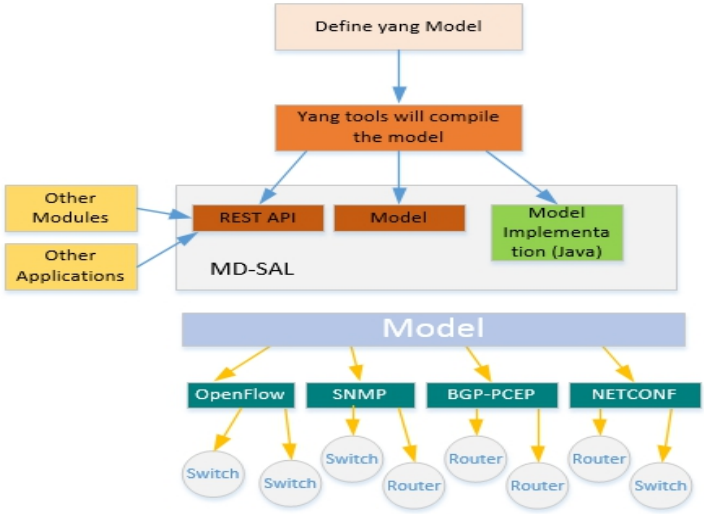

Figure 8 Model-Driven Service Abstraction Layer

OSGi is a dynamic Java-oriented model system. The OSGi-based service platform provides services to Java. These services support transplantable of the products on multiple platforms. At the same time, OSGi technology allows applications to use standardized primitives consisted of components with refined, reusable and collaborative, and those components can be assembled into an application and deployment. As an open source SDN framework, Open Daylight facilitates the realization of SDN open features and easy implementation of various service applications in an open source environment, using the OSGi component (bundle) integration method[32]. In addition, the dynamic loading feature of OSGi enables developers to dynamically load the developed services and applications while controller is running through corresponding commands, which is convenient for OpenDaylight to provide a unified open and runnable platform.

Maven is an excellent tool for structuring that be able to help us to automatically construct the project[33]. Moreover Maven can implement automatically the plenty of diverse functions via the command line, such as cleaning, compiling , testing , report-generating , packaging and deployment . Meanwhile Maven is a cross-platform tool, which means that the operation of constructing projects can be implemented by using the same instructions in the different operating systems by Maven. Maven is a software engineering management tool based on the project model (POM). Therefore, in the application or service component bundle developed in ODL, pom.xml is used to describe the various information needed to construct the component bundle, which is in order to safely constructing the service component.

Based on the above consideration and analysis of OSGi and Maven, the development of application service packages in S-PTN adopts the OSGi-based Maven project form

in ODL. For the development of each application or service component package, the following steps are required :

- 1) Use ODL's own maven prototype to generate package code framework;
- 2) Pom.xml configures the plug-in called maven-bundle-plugin to realize the content of the MANIFEST.MF file configured by maven according to the user, which is the initial running location when the newly developed bundle is loaded;

- 3) Configure pom.xml, specify Maven to package a project into a bundle, specify various development information in the bundle, including dependent libraries or packages, imported plug-ins, generated plug-ins, etc., The bundle which have finished the new development service insert into the controller server.

3.5 ONOS SYSTEM

OVERVIEW ONOS

ONOS is an open-network operating system launched and led by the operators in 2014.The purpose is to better apply SDN to the based-operator network, which is the first solution of controller proposed to aim at based-operator network in the whole industry. It also attracts the extensive attention from substantial amount of operators.
The two type of prototype architectures proposed by ONOS are shown as figure below:

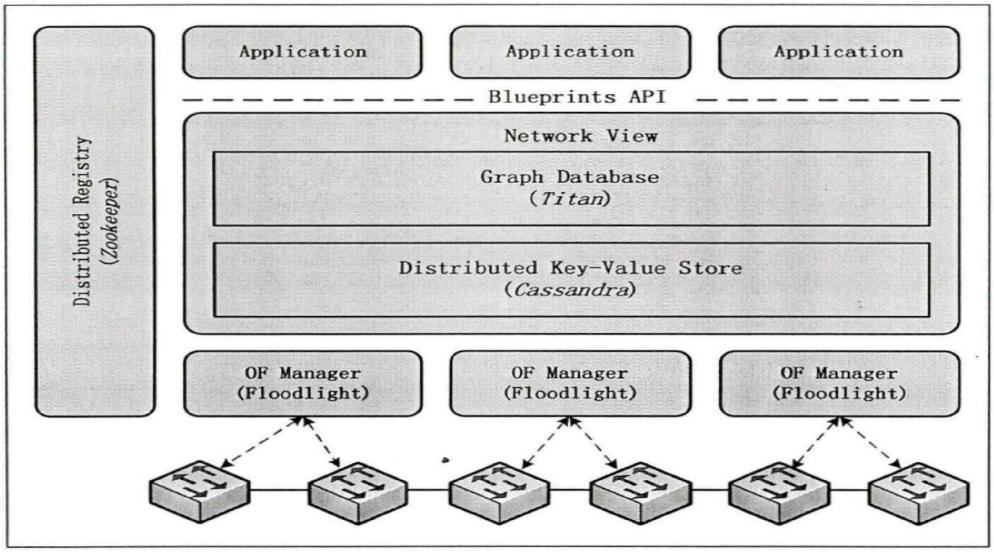

Figure 8-1 Prototype architecture 1 of ONOS

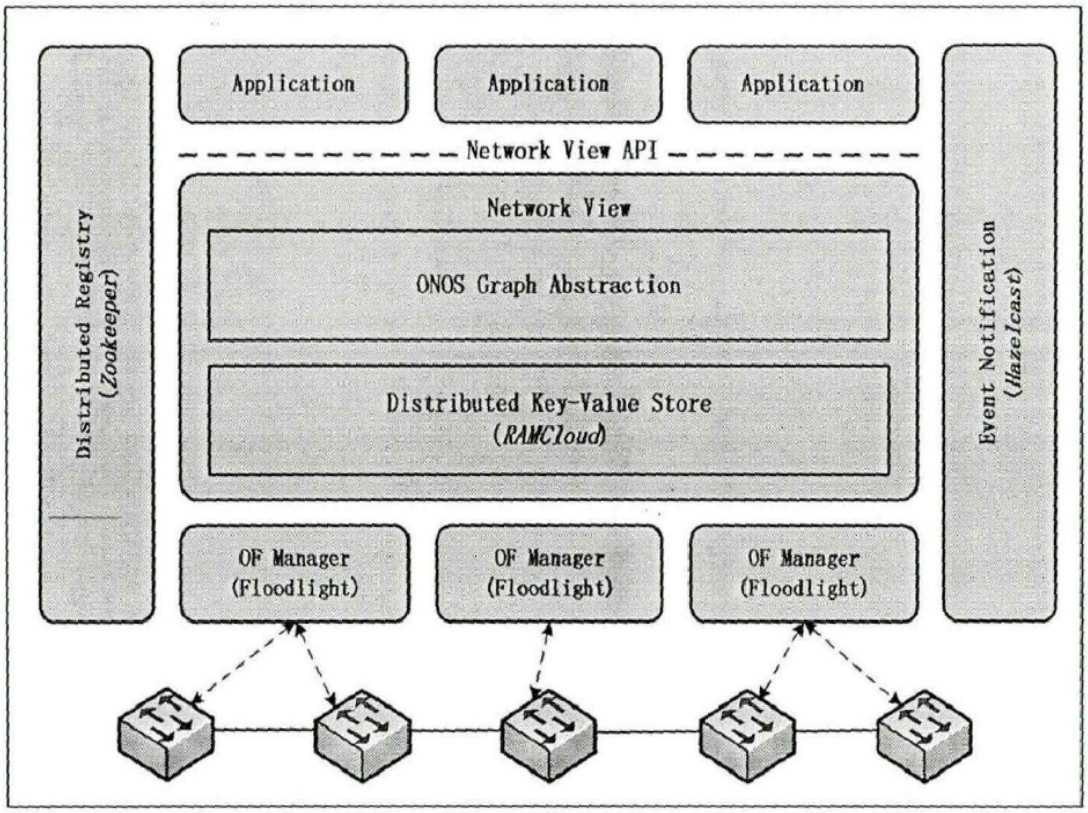

Figure 8-2 Prototype architecture 2 of ONOS

The prototype architecture 1 of ONOS as shown in Figure 1 is based on the first version of the modularized system architecture developed, which has a global network view and is used to achieve distributed deployment through ZooKeeper in which It has a good scalability and good fault-tolerance. The type 2 of prototype architecture shown in the Figure 2 adds an event notification structure on the basis of the type 1 of prototype architecture, which improves the performance of the system. It provides a northbound abstraction layer and interface to make application development easier, and also provides a southbound abstraction layer and interface in order to match OpenFlow networking and traditional networking, helping the based-vendor network to migrate from traditional equipment to OpenFlow equipment.

ONOS has high availability, great performance, scalability and provides a network abstraction at length, which meets the requirements of based-vendor network.

In addition to the above-described controller launched by open source community, many suppliers have launched their own research and development of controllers, which basically uses its own proprietary chip or interface protocols, whereas for a lot of operators, these controllers are not controllable and usable. Therefore,the research,design and implementation of a high-performance controller has a important significance for the operators, which can be able to apply and deploy in data-center network. so that, in the era of SDN, operators can occupy a leading

position in this competitive market.

4

YANG MODELLING LANGUAGE

4.1 INTRODUCTION OF YANG MODELIING LANGUAGE

NETCONF/YANG provides a standardized way to programmatically update and modify the configuration of a network device. YANG is the modelling language that describes the configuration changes. Whereas NETCONF is the protocol that applies the changes to the relevant datastore (i.e: running, saved etc) upon the device.

And we have to take a notice here that YANG modeling language present the data in a device through a abstract way which is only a kind of structuring, YANG actually do not fill in actual configuration data in a network device, instead XML or JSON generate the actual configuration data. In other word, YANG forms a kind of structuring which properly describe XML format for NETCONF protocol[34].

The full name of YANG call “Yet Another Next Generation”, which is a data modelling language, providing a standardized way to model the operational and configuration data of a network device. YANG, being a language is being protocol independent, can then be converted into any encoding format, such as XML, JASON. YANG is mainly used for the operation in the content layer of NETCONF protocol.

The YANG data modeling language provides descriptions of a network's node and their interactions. Each YANG module defines a hierarchical structure data which can be used for NETCONF-based operations -- including configuration, state data, Remote Procedure Calls (rpc) and notifications. Modules can import data from other external modules and include data from sub-modules.

YANG was developed by the IETF NETCONF Data Modeling Language Working Group (NETMOD) to be easily read by humans and as of this writing, Cisco, Juniper and Ericsson all support NETCONF and YANG. The YANG specification is published as RFC 6020 [35]and YANG types as RFC 6021[36].

4.1.1 THE CRUCIAL YANG CAPABILITIES :

- 1 Human readable, easy to learn representation
- 2 Hierarchical configuration data models
- 3 Reusable types and groupings (structured types)
- 4 Extensibility through augmentation mechanisms
- 5 Supports the definition of operations (RPCs)
- 6 Formal constraints for configuration validation
- 7 Data modularity through modules and submodules
- 8 Versioning rules and development support

4.2 YANG DATA STRUCTURING

4.2.1 YANG MODULES

YANG data model consist of data modules [YANG files]. Each module defines a single data model , which will be identified by namespace URI ,module is able to “include”,” import” and “argument” other modules or submodules[37].

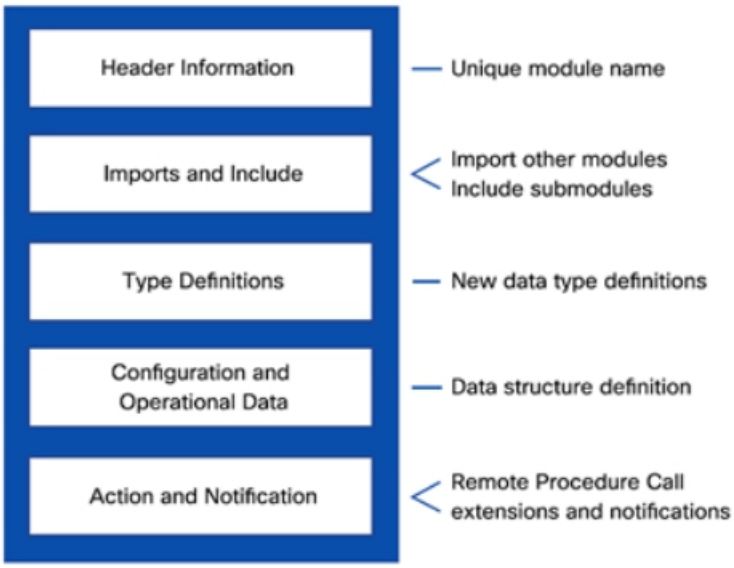

Figure 9 YANG OF HIERARCHY STRUCTURING

4.2.2 YANG OF HIERARCHY STRUCTURING

YANG models present a hierarchical structure of data, which represent as a 'tree' structuring as shown at Figure 9[38] . Each node has a name and a value or a set of child nodes as a characteristic of "tree" structuring[39]. There are four types of nodes, and the node types is at the following display respectively:

1 Leaf: A leaf contain simply a single key/value and there is no child node:

```
YANG Example:

        leaf host-name {
                type string;
                    description "hostname for this system";
            }
```

```
NETCONF XML Example

 <host-name>my example.com</host-name>
```

We defined a leaf node called host-name, which include its description, and

type is string. Then the following shows that this YANG leaf node is instantiated by XML and get a actual value.

2 Lea f-list: A leaf-list node is more similar as leaf node, but leaf-list node present an “array” of leaf node , which contains multiple key/values with the same type :

```
YANG Example:
    Leaf-list domain-search {
          Type string ;
          description  “List of domain names to search” ;
    }
```

```
NETCONF XML Example:

    <domain-search>hight.exampple.com</domain-search>
    <domain-search>low.example.com</domain-search>
    <domain-search>everywhere.example.com</domain-search>
```

As you can see, there is a leaf-list called “domain-search”, type is string with its description. But leaf-list node can be given multiple actual value with same type in XML format.

3 Container: A grouping of other nodes, but there is no “Value”.

```
YANG Example:
        container system {
            container login {
                leaf message {
                    type string;
                    description
                        “Message given at start of login session” ;
                }
            }
        }
```

```
NETCONF XML Example:
      <system>
         <login>
            <message>Good morning</message>
         </login
      </system>
```

As you can see there is a container called “system”, and this container encapsulate

another container called “login” as well as a leaf node called “message”, and its type is all about string .

4 List: Multiple records containing at least one “key” .A list node can contain multiple child nodes , which is able to include the arbitrary hierarchy of the statement of leaf/leaf-list/container.

YANG Example:

```
list user {
    key “name”;
    leaf name {
        type string;
    }
    leaf full-name {
        type string;
    }
    leaf class {
        type string;
    }
}
```

NETCONF XML Example:

```
<user>
    <name>Glocks</name>
    <full-name>Goldie lock</full-name>
    <class>introducer</class>
</user>
<user>
    <name>snowey</name>
    <full-name>Snow White</full-name>
    <class>free-loader</class>
</user>
<user>
    <name>rzell</name>
    <full-name>Rapun Zell</full-name>
    <class>tower</class>
</user>
```

There is a list node called “user” with a key called “name”, it is pretty clear
That this list “user” encapsulates three leaf nodes called “name”, “full-name” and “class” with all string type respectively. They are given the actual value in XML or

JASON format respectively.

Leaf and leaf-list nodes includes the “type” statement to identify the data type for valid data for that node. YANG defines a set of built-in types and provides the “typedef” statement for defining a derived type from a base type, which can be either a built-in type or another derived type.

4.2.3 YANG LANGUAGE KEY WORDS

- yang-version : Identifies the YANG language specification that the module will conform to. We can ensure our module conforms to YANG 1.1 which is defined in RFC 7950.

- namespace : This is an XML namespace which must be identified uniquely for the module. You can use a URL, URN, URI or any other unique identifier here. The namespace specified that here must match the namespace on any XML objects which conform to our YANG model.

- prefix : A short and unique string in order to identify the module. This prefix may be used in other YANG modules which are able to import definitions contained in this one.

- organization : A string which is identifying the entity responsible for the module.

- contact : Contact details for the entity responsible for the module.

- description : A description of the module.

- revision : Used for version control. Each edit to a YANG module will add a new.

- revision statement which is detailing the changes in sub-statements.

4.3 YANG STATEMENTS

STATE DATA

Netconf need to distinguish between configuration data and state data ,so “config false” should be added as a identifier of state data

```
list    interface {
   key   “name”  ;

   leaf    name {
         type string;
   }
   leaf    speed {
         type enumeration {
               enum 10m;
               enum 100m;
               eunm auto;
         }
    }
    leaf    observed-speed {
         type    unit32;
         config    false;
    }
}
```

BUILD-IN TYPE

Just like we have seen before ,we have used the type of “string” for so many times ,but it is only one of build-in types that YANG offer to us . I will show you all of build-in type data in YANG modelling language below .

```
+--------------------+-------------------------------------+
| Name               | Description                         |
+--------------------+-------------------------------------+
| binary             | Any binary data                     |
| bits               | A set of bits or flags              |
| boolean            | "true" or "false"                   |
| decimal64          | 64-bit signed decimal number        |
| empty              | A leaf that does not have any value |
| enumeration        | Enumerated strings                  |
| identityref        | A reference to an abstract identity |
| instance-identifier| References a data tree node         |
| int8               | 8-bit signed integer                |
| int16              | 16-bit signed integer               |
| int32              | 32-bit signed integer               |
| int64              | 64-bit signed integer               |
| leafref            | A reference to a leaf instance      |
| string             | Human-readable string               |
| uint8              | 8-bit unsigned integer              |
| uint16             | 16-bit unsigned integer             |
| uint32             | 32-bit unsigned integer             |
| uint64             | 64-bit unsigned integer             |
| union              | Choice of member types              |
+--------------------+-------------------------------------+
```

Figure 10 BUILT-IN TYPES IN YANG

DERIVED TYPE

Within a practical modelling ,maybe it is not sufficint for the restricted types above ,so YANG allow the user to define the type of data they want via derived keyword “typedef”.

```
Yang Example;
      typedef percent {
          type unit8 {
                range   “0···100”  ;
          }
          Description   “Percentage”  ;
      }
      leaf completed {
            type percent;
      }
NETCONF XML Example:

      <completed>20</completed>
```

- As you can see above, the user defined a new type data which is called percent , based on unit8 (0-255 unsigned integer) ,and restricted furthermore the range of value from 0 to 100 .

- Then the user defined leaf completed, which invoked type percent.

- Then 20 is given as a actual value after YANG-completed was instantiated by XML format, which conform to the type of data “percent”.

- And when interact with NETCONF, if the given actual value do not conform to the description of typedef “percent”(such as :the given value is more then 100 or less than 0) ,assignment request will be rejected by NETCONF because of failed pass of YANG check .

RESUABLE NIDE GROUPS

Grouping is not actually a kind of data type, its meaning is just to be invoked flexibly during programming. So a grouping has no actually value. A grouping can be imported in the current module or by other modules.

As we can see below, there is a grouping called target where contain two leaf nodes

called “address” and “port” respectively. Then there is also a container node defined, which is called ‘peer’ and contain another container called ‘destination’. Grouping is here imported through keyword “uses”. Finally, we can directly give the actual values to “address” and “port” through <peer> after <peer> was instantiated in XML schema.

YANG Example:

```
grouping target {
   leaf address {
        type inet：ip-address;
        description ” Target IP address” ;
   }
   leaf port {
         type inet : port-number;
         description “Target port number” ;
   }
}
container peer {
      container destination {
            uses target;
      }
}
```

NETCONF XML Example:

```
<peer>
   <destination>
        <address>192.0.2.1</address>
         <port>830</port>
   </destination>
 </peer>
```

EXTENDING DATA MODELS(ARGUEMENT)

This is a very useful function, YANG allow to insert the new nodes in data module. For example, the vendor are going to insert the new node which is about their own special arguments in a public YANG data , so the keyword “argument” will satisfy and implement this requirement . And as well can see the image below, there is a condition after the statement ‘when’, if this condition is satisfied , the new argument will be insert ,vice versa .

YANG Example:

```
argument /system/login/user {
      when  “class !=  ‘wheel’ ” ;
      leaf  uid {
            type uint16 {
                  range  “1000 . . 30000” ;
             }
      }
}
```

NETCONF XML Example 1;

```
<user>
    <name>alicew</name>
    <full-name>Alice N. wonderland</full-name>
    <class>drop-out</class>
    <other:uid>1024</other:uid>
</user>
```

NETCONF XML Example2:

```
<user>
      <name>jerryr<name>
      <full-name>Jerry K. Roma</full-name>
      <class>wheel</class>
</user>
```

The example 1 showed us , class != wheel ,so the new nodes “ uid ” is inserted .

The example 2 showed us , class = wheel , so nothing is inserted

Of course the type of data of new node would have been defined in advance in YANG model .

RPC DEFINITION

Of course YANG is able to define <rpc> of NETCONF ， including the arguement of input and output in <rpc> .

YANG Example:

```
rpc activate-software-image {
    input {
        leaf  image-name {
            type string;
        }
    }
    output {
        leaf  status {
            type string;
        }
    }
}
```

NETCONF XML Example:

```
<rpc message-id=" 101"
    xmlns- "urn:ietf:params:xml:na:netconf:base:1.0" >
  <activate-software-image xmlns- "http://acme.example.com/system" >
    <image-name>acmefw-2.3</image-name>
  </activate-software-image>
</rpc>
```

```
<rpc-reply>message-id- "101"
            xmlns- "urn:ietf:params:xml:ns:netconf:base:1.0" >
  <status xmlns- "http://acme.example.com/system" >
    The image acmefw-2.3 is being imstalled.
  </status>
</rpc-reply>
```

As we can see the image above ,a type of <rpc> called "activate-software-image" includes nodes "input" and "output" respectively.

The client need to identify the "image-name"as input when client is going to sand <rpc> request to server ,then server need to identify the result of "status"as output when server is going to send back <rpc-relpy> .

NOTIFICATION DEFINITION

As a type of mechanism of notification . When a characteristic event occur , server

will send <notification> to the specified client which has established a connection with NETCONF as well as that has subscribed notification .

```
YANG Example:

    notification link-failure {
        description  "A link failure has been detected" ;
        leaf if-name {
            type leafref {
                path  "/interface/name" ;
            }
        }
        leaf if-admin-status {
            type admin-status;
        }
        Leaf if-oper-status {
            Type oper-status;
        }
    }
```

```
NETCONF Example:
  <notification
        xmlns- "urn:ietf:params:netconf:capatility:notification:1.0" >
    <eventTime>2007-09-01T10:00:00Z</eventTime>
    <link-failure xmlns- "http://acme.example.com/system" >
        <if-name>50-1/2/3.0</if-name>
        <if-admin-status>up</ if-admin-status >
        <if-oper-status>down</if-oper-status>
    </link-failure>
  </notification>
```

As the example has shown , YANG modeling language can define <notification> .
In this way , when either the admin status of the interface (shutdown / not shutdown) or the link status (up/down) changes ,<notification> can be sent to the client with the current status information .

5

NETCONF PROTOCOL

5.1 WHAT IS THE NETCONF

NETCONF is a type of network management configuration protocol published and standardized by IETF, And this sort of protocol (NETCONF) is an XML-based network management protocol which provides a programmability to configure and manage different network devices[40]. NETCONF was defined in RFC 4742 by the Internet Engineering Task Force (IETF) and revised in RFC 6241-6244[41][42][43].

Along with the consist heat of SND, a ten-old protocol usher into its time which is called NETCONF. Originally, the main purpose of NETCONF is to make up for the shortcomings of the SNMP, and hoping to replace the SNMP protocol. And NETCONF has officially replaced the SNMP at present.

And this is a contrast diagram below as shown at Figure11[44]:

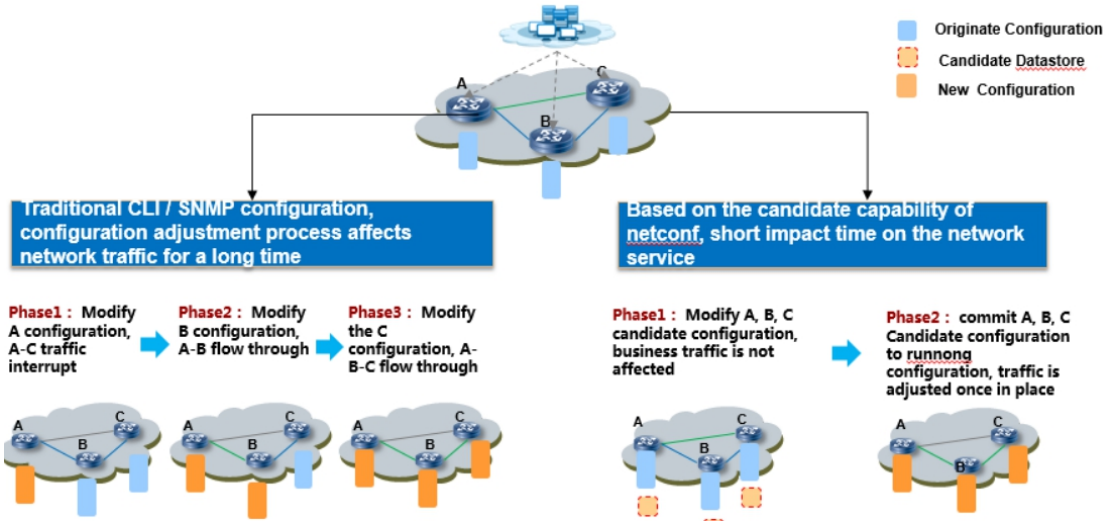

Figure 11 CONTRAST BETWEEN DNMP AND NETCONF

As all we can see, the configuration method form NETCONF is evidently more efficient then the configuration method from SNMP.

NETCONF provides a standard framework and a set of standard Remote Procedure

Call (RPC) methods through which network administrators and application developers can manage configurations of network devices and obtain network device status fast[45].
As well-known the NETCONF is based on YANG and XML data structuring ,so the NETCONF packets are in XML format and the NETCONF protocol has a powerful filtering capability. Each data field has a fixed element name and position. Therefore, the devices of the same vendor can use the same access mode and result performance mode. The devices of different vendors can implement the same effect by XML mapping. This feature facilitates third-party software development and NMS software customization in the multi-vendor, multi-device environment. With the help of such NMS software, NETCONF simplifies device configuration and improves device configuration efficiency.

NETCONF defines a series of configuration datastore (as RFC6241 Configuration Model), including running, startup, and candidate. NETCONF API supports only the running datastore operation.

5.2 NETCONF LAYERING

NETCONF is a client/service communication mode, and use a hierarchical protocol framework[46].

NETCONF can be partitioned into four layers conceptually, which is:

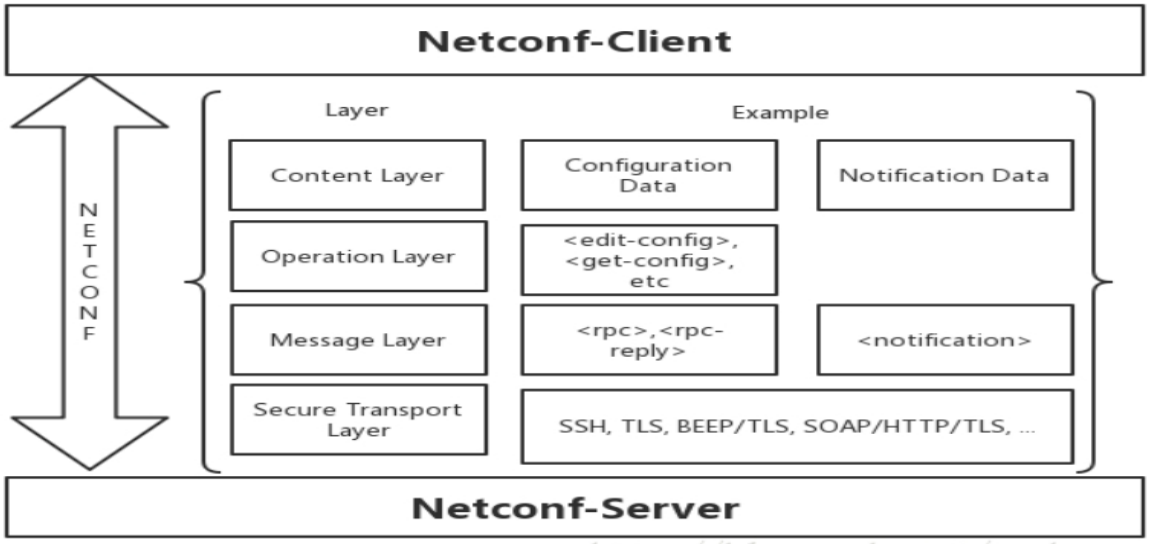

Figure 12 FOUR LAYER OF NETCONF PROTOCOL

Let me introduce this structuring similar with OSI 7 layering. And we can see from the graph above that NETCONF is partitioned into 4 layer, from bottom to top there are Secure Transport Layer, Message Layer, Operation Layer and Content Layer

respectively. And we have to build up a concept before i introduce these four layer, which is that NETCONF always think it is divided into two main types for the network configuration data ,namely configuration data and status data. And state data usually refer to the inherent attribute of the server and the running data of network devices, this type of data can only be checked, but on change.

Configuration state refer to the data configured somehow by client to server .The standard mentions that the <running/> library is used to save the currently effective configuration; <candidate/> is used to save the data that can be submitted as effective; and the <startup/> is used to save the configuration data at startup

5.2.1 CONTENT LAYER :

This layer contain configuration data and status data , it describes configuration data involved in network management. Most configuration do not have standard data model of NETCONF, thus the devices of different vendor may use different configuration data. However content layer as the essence of NETCONF present a open and standard feature. Open feature is saying it did not impose any restriction for data structure in content layer, standardized feature is saying that it needs YANG modelling language to model data in content layer.

Content layer defines data model such as YANG language, so data model defined by YANG can all be seen as content layer.

There are three main feature in content layer:

1 Data separation of configuration ,status and statistic .

1 Support for transaction mechanism and rollback operation .’

3 Good scalability.

5.2.2 OPERATION LAYER :

Operation layer defines the RPC method .it defines a series of basic protocol operation used in RPC, and these operations compose the basic capabilities of NETCONF.
There are some methods defined by operation layer.

<GET>: it is used for inquiry of status data.

<GET-CONFIG>: it is used for inquiry of configuration data , and it can define different datastore by <source/>.

<EDIT-CONFIG>: it is used for modifying the content in specified datastore,and support the following several operations:

Merge: combination operation, this is default operation.

Replace: replacement operation ,it will execute replacement if object has existed, or create if object do not exists.

Delete: deletion operation , it will report “data-exists”if object has existed.

Remove : removing operation , it will execute removing if object has existed, or ignore.

<COPY-CONFIG>: copy the all data from a datastore to another .

```
<rpc message-id=” 101”
    Xmlns=” urn:ietf:params:xml:ns:netconf:base:1.0” >
    <copy-config>
        <target>
            <running/>
        </target>
        <source>
            <url>hrrps://urse:password@example.com/cfg/new.txt</url>
        </source>
    </copy-cinfig>
</rpc>
```

<DELETE-CONFIG>: delete a datastore , but <running/> can not be deleted.

```
<rpc message-id=” 101”
    xmlns=” urn:ietf:params:xml:ns:netconf:base:1.0” >
    <delete-config>
        <target>
            <startup/>
        </target>
    </delete-config>
</rpc>
```

<LOCK>:acquire the lock of the specified datastore. After a client obtains the lock of the specified database, the rest of the clients cannot obtain the lock of the database or modify it before releasing the lock. The same client cannot repeatedly apply for a lock before releasing the lock.

```
<rpc message-id=" 101"
      xmln=" urn:ietf:params:xml:ns:netconf;base:1.0" >
      <lock>
          <target>
              <running>
          </target>
      </lock>
</rpc>
```

<UNLOCK>: release the lock of the specified datastore , client can only release the lock it held, not the lock of other clients.

```
<rpc message-id=" 101"
      xmlns= urn:ietf:params:xml:ns:netconf;base:1.0>
      <unlock>
          <target>
              <running/>
          </target>
      </unlock>
</rpc>
```

<CLOSE-SESSION>:Close the netconf session, netconf-server will release the lock held by the client and the resources allocated to it, and close the connection with the client. All operations received after <close-session> will be ignored.
<KILL-SESSION>: forcibly close the netconf session.

5.2.3 MASSAGE LAYER

Message layer provides the RPC mechanism for request and reply message ,and it support RPC-based encoding.

Message layer defines main three message elements :<hello> , <rpc>and<reply-rpc> , <notification>.

1 <HELLO> :

The main function of this element is to build up a connection of secure , connection-oriented session ,and the exchange of capabilities between the client and the server at the beginning of session.

The client and sever will send <hello> message with each other as soon as the session is established ,and their capabilities will contained in <hello> message .
Usually, <hello> message is sent to each other between client and server with a successful negotiation of NETCONF version, we think session is establish successfully.
There are some common capabilities in message layer :

(1) XPath Capability

This capability means that the client can use XPath expressions as filter conditions in the filter.
Capability Identifier:

urn:ietf:params:netconf:capability:xpath:1:0

(2) Writable-Running Capability

This capability means that server support the modification operation in <running/> library.

Capability identifier:

urn:itef:params:netconf:capability:writable-running:1.0

(3) Candidate Configuration Capability:

This capability means that server hold a <candidate > datastore , and it can submit the configuration of this <candidate > datastore as well as making it work, as updating the <running/> datastore.

Capability identifier :

urn:ietf:params:netconf:capability:caditate:1.0

(4) Rollback-on-Error Capability

This capability means that server can execute a rollback operation when there is a error in the configuration data sent by the client.

Capability identifier:

urn:ietf:params:netconf:rollback-on-error:1.0

(5) Validate Capability

This capability means that server can authenticate the correctness of the configuration data sent by the client.

Capability identifier:

urn:ietf:params:netconf:capability:validate:1.1

(6) Distinct startup Capability:

This capability means that server has a <startup> datastore used for saving startup configuration.

Capability identifier :

urn:ietf:params:netconf:capability:startup:1.0

2 < RPC > , <REPLY- RPC > :

<rpc> :

The message sent by NETCONF-client to NETCONF-server , which is used for requesting certain specific operation to server. And <rpc> contain a forcible attribute “message -id”, which is used for matching between the client and the server.

<reply-rpc>:

1 The response message sent by NETCONF-server to NETCONF-client ,and this message cannot be sent forwardly , it can be sent only after receiving the message from the client as well as carrying the same “message-id”with the corresponding received <rpc> request message.

2 <ok> and <error> is two default elements defined in <rpc-reply>. <ok> presents the successful operation of <rpc> ,and <error> presents the failed operation of <rpc>.
3 The whole procedure of this layer: The RPC layer provides a simple, transport-independent RPC request and response mechanism. The NETCONF client uses the <rpc> element to encapsulate RPC request information and sends the RPC request information to the server using a secure, connection-oriented session. The server uses the <rpc-reply> element to encapsulate RPC response information (content at the operation and content layers) and sends the RPC response information to the client. Huawei switch functions as the NETCONF server to receive NETCONF requests from the NETCONF client.

Normally, the <rpc-reply> element encapsulates data required by the client or a configuration success message. If the client sends an incorrect request or the server fails to process a request from the client, the server encapsulates the <rpc-error> element containing detailed error information in the <rpc-reply> element and sends the <rpc-reply> element to the client.

3 <NOTIFICATION>:

urn:ieft:params:netconf:capability:notification:1.0
The server which support <notification> function need to notify the capabilities when exchange the capability with the client.

5.2.4 SECURE TRANSPORT LAYER ：

1 It establishes connection-oriented session and ensure data transmission security.
The transport layer provides a communication channel between the client and the server.

2 The security of transport layer is represented at which NETCONF packets can transmit on the particular transport protocols such as SSH[47], TLS, etc, which support the attributes of authentication, encryption and integrity check.

3 The transport protocol is connection-oriented. A permanent link is established between the NETCONF client and server. After the permanent link id established, data is transmitted reliably and sequentially.

4 The transport protocol provides a mechanism which can distinguish the session type for NETCONF.

5 The transport layer is based on the transport protocols which provide user data

integrity, security encryption authentication.

5.2.5 THE OVER FRAMEWORK OF NETCONF

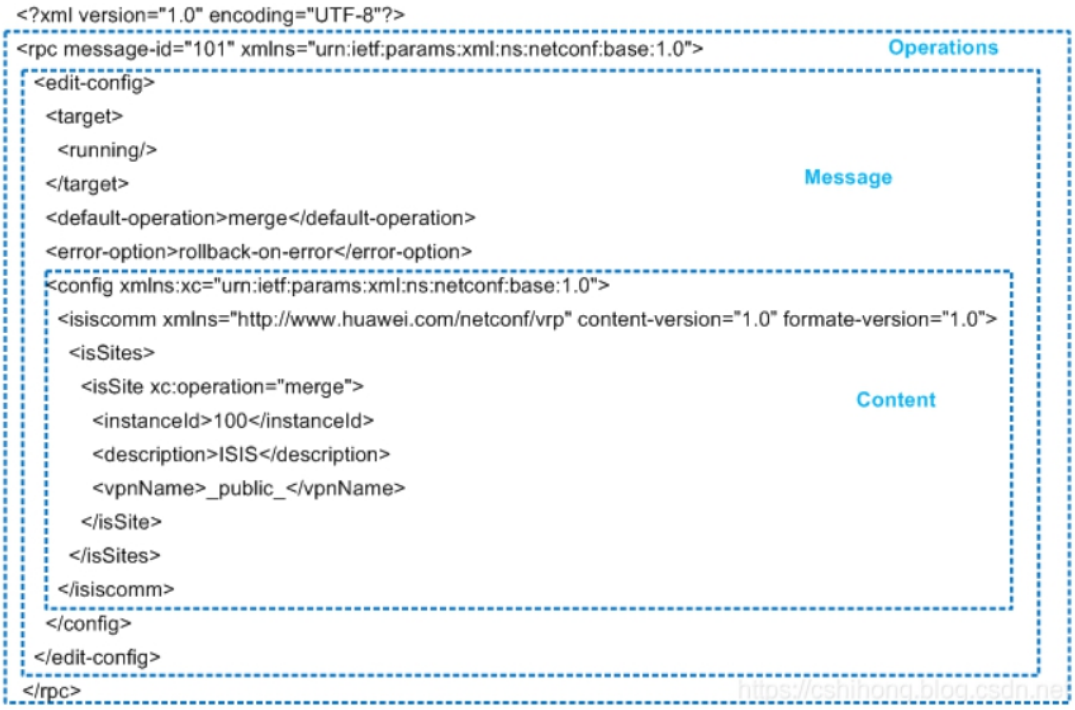

Figure 12 OVERALL XML DATA CONTENT OF NETCONF PROTOCOL

5.3 A HYPOTHETIC WORKFLOW OF NETWORKING

Assuming that a user want to configure an IP address for device interface through the client after SSH connection, authentication and authorization are completed .
1 First step, it is to initiate NETCONF session establishment on the client and to notify the capabilities through <hello> message.

2 Second step, after the capabilities negotiation succeed ,the client need to request to lock <running/> datastore through <lock> element , so that its operations will not be affected by other clients .

3 Third step, it is to copy the data in <running/> configuration datastore to <candidate/> configuration datastore through <copy-config> element to ensure that the configuration are latest.

4 Fourth step, it is to edit configuration in <candidate> configuration datastore.

5 Fifth step, data migration. configuration data in <candidate/> datastore will be migrate to <running/> datastore.

6 Sixth step, <running/> datastore should be unlocked . and finally terminate NETCONF session and tear down SSH connection.

Actually , under SDN architecture , the controller of SDN that centralizes the control rights of the devices always serve as the client of NETCONF , then the network devices always serve as the server of SDN which is used for receiving various instructions and for executing the forwarding function . Also under SDN architecture , the network devices is the provider of the network execution units .The instructions that are issued by the controller will be presented on the network devices by the form of the changes of configuration data . during running of the network devices, the network devices will set the diverse corresponding attributes of the network resource according to the configuration data validated on the network devices[48].

6

RESTCONF

6.1 TECHNICAL RESEARCH OF RESTCONF

In order to achieve SDN network architecture on the basis of the stable running of the existing networking, the key requirement is to support SDN network architecture without getting a major change of the existing network devices . At present, SDN network has become a hotspot and popular research topic. As the southbound interface has got relatively complete technical standards , most of the current research focuses on the northbound interface , particularly about the research of RESTful interface .

6.2 INTRODUCTION OF RESTCONF

The RESTCONF protocol is based on the HTTP protocol. It provides a programmable interface to access data through YANG, and to operate data storage through NETCONF. The framework and meta-model using Restful API do not necessarily need to be mapped in the NETCONF protocol, but it needs to be compatible with NETCONF protocol. RESTCONF provides the function information of the YANG module supported by the server, which can be used by the client . The protocol operation and data storage content of this specified URI can be foreseen based on the YANG module. We should pay attention that the client can optionally use YANG Modules and predictable URI. They can be completely ignored without causing any loss to the functionality of the protocol.

The RESTCONF protocol provides a simplified interface that follows the REST architectural style, and realizes the control access to the data layer through the YANG data language model. YANG language defines syntax, data storage content, operation data and customizable protocol operations, and defines notification event semantics. In YANG language, all RESTCONF content is divided into three types: data resources, operation resources, and event stream resources. A RESTful API uses HTTP and REST design principles to simplify NETCONF data storage and provide YANG operations including Create, Retrieve, Update, Delete, etc.

6.2.1 INTERDEPENDENCY : RESTCON, NETCONF, YANG[49]

- RESTCONF is formerly YANG-API , now called RESTCONF
  -old: draft-bierman-netconf-yang-api-01.
  -new: draft-bierman-netconf-restconf-02
- HTTP/REST API for CRUDX operation on YANG-defined resources.
- RESTCONF integrated with NETCONF datastores , access control , operations and notifications.
- Simple WEB client developer API for NM.

6.2.2 RESTCONF RESOUCE MODEL

- API: API framework.
- Datastore: NETCONF datastore.
- Data: YANG data definition statement.
  -container or list , not leaf , leaf-list or anyxml.

- Operation: YANG rpc statement.
- Patch: PATCH method using YANG Patch.
- Patch status: YANG patch status .
- Stream: notification stream.

6.3 OPERATION OF RESTCONF

RESTRCONF protocol use HTTP method to distinguish CRUD operation on the specific resource, including OPTIONS , HEAD , GET , PUT , PATCH , DELETE etc [50]. During the development of the northbound interface of the S-PTN controller , which mainly involved in the two types of services , Tunnel and Eline , and management and control of the network resources are also involved . Therefore this article is mainly focused on the operations of RESTCONF, such as GET , PUT , POST , DELETE etc . The each RESTCONF request have to begin with " /restconf " in URI,and RSTCONF will monitor HTTP requests at 8080 port. RESTCONF allows access to the data storage of controller.

6.3.1 PUT

The users are able to query and retrieve resource data and resource state through making use of PUT operation, supporting all types of resources except for operation resources. And operation request need to specify URI containing the query node. The query parameters supported by GET method are shown in Table 1 below.

NAME	DESCRIPTION
config	Request of the configuration data or non-configuration data
depth	Control of the depth of retrieve request
format	Request of the content type which is to be XML or JSON
select	Specifying of the nested resources in the target resources

TABLE 1 - PARAMETERS OF GET METHOD

The server will not return data to users without access rights. If the user does not have

any access rights, "403 Forbidden" message will be returned to users. If the user has only partial access rights, the returned information will omit the unauthorized part and only return the user authorized information. If request of the information query is successful, the server will return a response message with a status of "200 OK" to the user. The encoding format for obtaining resources can be XML or JSON. When obtaining resources by default, any nested resources will also be obtained at the same time. The client starts obtain from the top-layer API resource, and the entry point URI used is "/.well-known/restconf"

It should be pointed out that the Body part of the request information is the YANG data content. This YANG data defines specifically the format, fields and other the requested information content. Generally, the information nesting is complicated. The returned information will be correspondingly returned the content queried by the server. the default settings will be used while the server is not able to query or retrieve the requested related information .

6.3.2 POST

There are many reasons to cause sending of POST requests by users. Such as shown in Table 2 , the server will make the decisions on dealing with request according to type of the target media . The operation request must contain the target resource URI identified by the following resource types :

TYPE	DESCRIPTION
Datastore	Create the data resources in top-layer
Data	Create configuration data sub resource
Operation	Invoke the protocol

TABLE 2 – RESOURCE TYPES OF POST

If the target resource type is Datastore or Data, the POST method will be used to request information to create the resource. If the POST method is successful, it will return a status line with "204 No Content" and there is no response message content. If the user is not authorized to create the target resource, the server will return a response message with "403 Forbidden" to the client.

If the target resource type is an Operation resource, the POST method will be treated as a request to invoke the operation. The body part of the message will be treated as an input parameter of the operation. When successful, if there is a response message content, it will return a response status with "200 OK", if not, it will return a response status with "204 No Content". If the user is not authorized to call the target operation, the user will receive an error response with a response status of "403 Forbidden".

6.3.3 PUT

The user uses the PUT method to replace the target resource. Generally, the user uses the PUT method to modify various configuration data in the server. The request information must contain the specific URI representing the identifier of target to be replaced. If the replacement is successful, the server will return a "200 OK" response status to the user without body content.

6.3.4 DELETE

The DELETE operation is used to delete the specified resource. If DELETE method delete resource successfully, the server will return a "200 OK" response status with no response message content. If the user is not authorized to delete the specified target resource, the server will return a "403 Forbidden" error response message to the client.

6.3.5 MESSAGE

The URI request needs to specify the operation, location address and related parameters of the target to access the resource, including field information such as Method, Entry, Resource, and Query. Figure 3 shows the structure definition of the URI representing the resource in RFC3896.

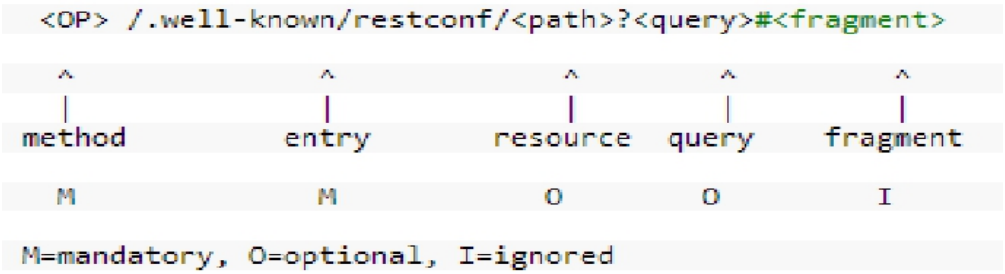

Figure 13– URI STRUCTURE

Among them, it is necessary for Method and Entry .The Method and Entry fields are required to distinguish the user's operations on the specified target resource as well as access nodes. A complete URI need to be clear on the operation of accessing resources and to access nodes. Resource and Query are optional. The Resource path expression indicates the resource acquired by the operation. If the domain does not exist, the target resource is the API itself, which is represented by "application/YANG.api". Query represents a collection of RESTCONF message related parameters, including key-value pairs such as "name=value". A series of specific parameter sets have been defined here, and the server can additionally select parameters which is not defined in this article.

The user does not know the final URI path structure of the resource. On the contrary, the existing resource can be found through the GET method. When a user creates a new resource, it will return a "Location" header message indicating the path of the new resource. Once the resource is established, the user can only use the path identifier to access the resource.

Due to the unpredictability of data storage content changes, the response sent by the RESTCONF server is generally not cached. The server needs to add a "Cache-Control" field to the header of the no response message to identify whether the response needs to be cached. To prevent the server from not supporting the "Cache-Control" field, the response message header also needs to add the "Pragma" field. The user needs to track the "ETag" and "Last-Modified" header fields returned by the server instead of using the HTTP caching mechanism.

6.4 RESOURCE

The resources in the RESTCONF protocol are identified by the path in the request

URI, and each operation is for the target resource. RESTCONF resources can be accessed through a series of defined URIs. Each URI represents a specific resource and its operation, and is unique. The RESTCONF protocol runs on all layers of resources. Starting from the top-level API resource, each resource represents a controllable component. A resource can be expressed as a series of data and a collection of operations allowed by the data. The resource includes its child nodes and various domains.

Each resource has its own media resource identifier, which is indicated by the "Content-Type" field in the HTTP response message. A resource may not contain or contain multiple embedded resources. When the upper-level resources exist, the lower-level resources can be added, deleted, and modified separately

6.4.1 API RESOURCE

API resources include the following sub-type resources: datastore, modules, operations. Table 4 shows the description of each type of resource.

Underlying resource	Description
Data Store	Point to the data storage resource
Modules	URI of YANG module functions
Operations	The specified operations of data module

TABLE 3 – API RESOURCE TYPE

API resources include the state of RESTCONF style and access points. The API resource is top-layer of the resource and has a media type of "application/vnd.yang.api+xml " or "application/vnd. yang.api+json". It can be accessed through the associated URI "/.well-known/restconf". Only one field needs

to be complete by server, namely the "version" field of RESTCONF protocol version number. Each retrieval of this field must return a server-wide response; it is assigned by the server when the server starts.

1) /.well-known/restconf/datastore. This resource is necessary and used to indicate the running configuration data storage and all available non-configuration data. The data can be updated or edited directly, and it cannot be created or deleted by user .

2) /.well-known/restconf/modules. This mandatory resource contains the identifier defined by the YANG data model supported by the server. The server must maintain a latest modified timestamp for the resource, and return a "Last-Modified" header when the resource is updated with the GET or HEAD method. At the same time, the server should maintain an entity tag for the resource, and return an "Etag" header when the resource is updated with the GET and HEAD methods. The operations type resource provides specific protocol operations of the data model supported by the access server. Any specific operation of the data model defined by the server through the YANG model can be used as a child node of this resource.

3) /.well-known/restconf/operations. This optional resource can access the specific protocol operations of the data model supported by the server. If there is no notification of the specific operation of the data model, the server can ignore this resource. Any specific operation of the data model defined by the server publishing in the YANG module can be used as a child node of this resource.

6.4.2 DATASTORE RESOURCE

Data Store resources are like the root of a tree, representing the most underlying data resources. The server must keep this resource as the latest modified timestamp, and when this resource is retrieved with the GET or HEAD method, it returns the "Last-Modified" timestamp. Changing the configuration data resource only affects the data storage of this time stamp. The server should keep the entity tag of this resource and return the "Etag" header file when this resource is checked out of the GET or HEAD method. If you want to change the configuration data resource in any data store, the resource entity flag must be changed to a new unused value before the change.

6.4.3 DATA RESOURCE

A Data resource represents a YANG data node, which is a child node of a data storage resource. YANG container and list data node types are considered to represent data resources. Other YANG data nodes are considered resources in the parent domain. A data resource can be retrieved with the GET method. Data resources are accessed via the "{+restconf}/data" entry point. This sub-tree is used to retrieve and edit data resources.[]

A Data resource can be retrieved using the GET method to obtain the non-configuration data resource in the target data resource of the configuration data resource. The "config" query parameter is used to choose between the two. Through the "depth" query parameter, you can control the depth of the subtree in the return update operation. According to the target resource and specific operation, the client can change the configuration data resource through some modification operations.

7

OPENFLOW

7.1 BACKGROUND OF OPENFLOW PROTOCOL

The Clean Slate project of Stanford University is the origin of OpenFlow. The redesigning of conventional internet architecture is considered as ultimate goal by The Clean Slate project, aiming to completely change the existing network architecture.

In 2006, Stanford University student Martin Casado led Ethane, a project on network security and management. Ethane's overall idea is to try to use a centralized controller to enable network administrators to flexibly define and apply network flow-based security control strategies, thereby achieving security control of the entire network communication. The Ethane project inspired Martin and his mentor, Professor Nick Mc Keown, who believe that if the design of Ethane is more generalized, namely the two functional modules as data plane and control plane of the conventional network equipment are separated, and various network equipment can be managed and

configurated by the standardized interfaces through centralized controller, then this can provide more possibilities for the use and management of network resources, which makes it easier to promote the development and innovation of the network. On this basis, OpenFlow came into being.

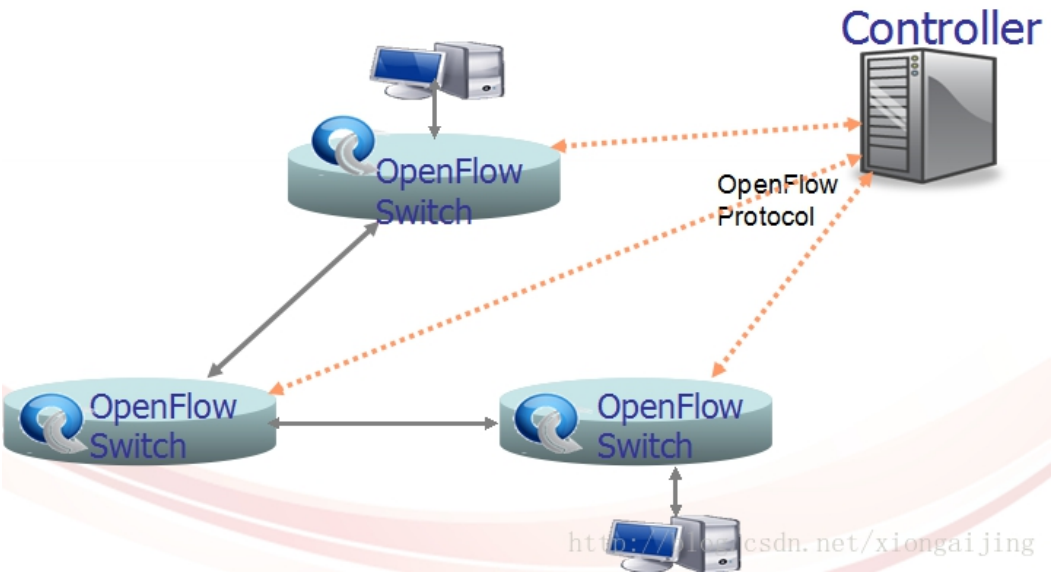

Figure 14 SDN Architecture Based On OpenFlow Protocol

The OpenFlow Switch and the OpenFlow Controller jointly complete the forwarding and control processes of the message segment that was originally completely controlled by the conventional switch/router, thereby achieving the purpose of separating control plane from data forwarding plane. Data forwarding is completed by the controller through the specified interface operation to control the flow table in the OpenFlow switch.

On this basis, support for remotely centralized control is one of the most critical features. There is no need to directly program and code in the corresponding single switch when changing flow entries, and the real-time status of the network can be learned through remote control. Under such a unified control mechanism, the network will be truly intelligent and controllable.

Under this standard, operators and even ordinary users can define the special rules in line with their needs in the normal operation of the network, allowing traffic in rules to go on the appropriate path in accordance with the requirements. As if a physical network is divided into a number of different virtual networks, running simultaneously without interference. Therefore, it is easy to test various new protocols. In the past, when processing, you need to consider the specific topology and the sequence of nodes. Now you only need to define different flow table items to change the traffic operation strategy at will. This is able to solve the problem of traffic

mobility

In this point, OpenFlow protocol transforms the hardware-defined Internet with the traditional physical fixed features into the software-defined Internet with the dynamic variable features, which is now SDN.

7.2 SDN BASED ON OPENFLOW PROTOCL

In the traditional Internet, control logic plane and data forwarding plane are tightly coupled with network equipment, which complicates the management on network control plane. And this also causes that the update and development of new technologies of control plane is difficult to be deployed on the existing networks, which cause a limited flexibility and scalability for the rapid development of networking.

The control and forwarding separation architecture of the network proposes to deploy high-level strategies on proprietary equipment, and network equipment performs data forwarding under the guidance of high-level strategies, reducing many complex functions carried by network equipment, and improving the flexibility and operability for implementation and deployment of the new network technologies and new protocols.

The TCP/IP architecture and the software design based on open-source application layer have enabled the rapid development of networking, but proprietary hardware devices and operating systems have made the network basically closed. Through SDN, the control functions of traditional distributed network equipment can be migrated to controllable computing devices. And the upper-layer network services and applications are used to abstract the underlying network infrastructure. The automated control function of networking is ultimately achieved by the mode of open programmable software.

OpenFlow realized the idea of software-defined network of SND, which represents the prototype and deployment example of SDN technology. OpenFlow refers specifically to a communication protocol between the control plane and the data plane in the entire SDN architecture. Figure describes the logical view of the SDN architecture, which is mainly divided into the infrastructure layer, the control layer, and the application layer. The infrastructure layer represents the underlying forwarding devices of the network, and includes a specific forwarding plane abstraction, namely the matching field design of the flow table in the OpenFlow switch. The middle control layer centrally maintains the network status, The middle control layer centrally maintains the network status, and obtains the underlying infrastructure information through the interface between control plane and data plane,

namely, the southbound interface OpenFlow protocol. and meanwhile provides an expandable northbound interface for the application layer. According to various application needs of the network, the northbound interface of the control layer are callable to implement applications with different functions . Using this software model, network administrators can configure, manage, and optimize the underlying network resources through dynamic SDN applications, thereby achieving a controllable and flexible networking.

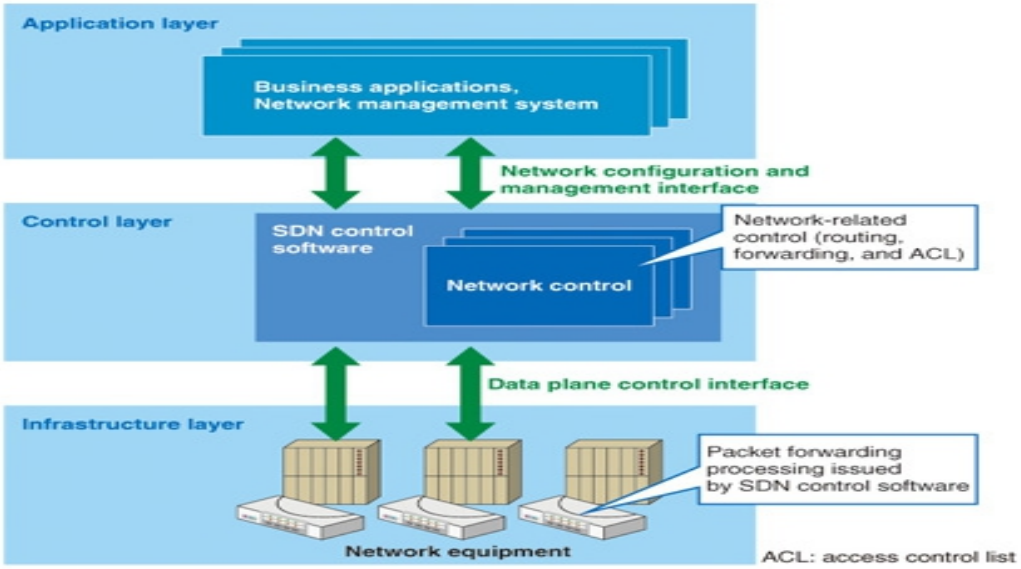

Figure 15 SDN architecture based on OpenFlow protocol

In this three-tier SDN architecture, the operation and maintenance of the network can be performed only through software updates, and the upgrade of network functions can be realized as well. The network configuration will be directly deployed in the form of applications and network services, and the network administrator does not need to then configure each single hardware device or wait for suppliers of device to release new hardware, thereby shortening the network deployment cycle. At the same time, SDN simplifies the functions of the underlying network equipment and only needs to focus on data forwarding, which reduces network complexity and effectively reduces network construction costs. In the end, network equipment changes from closed to open. On the other hand, the nodes of data forwarding in conventional network architecture can only be realized through local states and distributed algorithms, so it is difficult to achieve optimal performance. SDN realizes centralized control through software, so that the network has a centralized coordination point, so as to achieve optimal performance through software methods, thereby accelerating the network innovation cycle [51].

7.3 OPENFLOW PROTOCOL

Using the OpenFlow network platform, network research can be carried out without affecting existing network protocols and services. OpenFlow protocol implements and provides an SDN solution. By providing network researchers with a set of network resources including network links, network equipment, and network terminals, enable network researchers to control their own SDN according to their wishes.

OpenFlow switch is responsible for the function of data forwarding, and the main technical details are made up of these three parts including Flow Table, Secure Channel, and OpenFlow Protocol , as be shown as figure below:

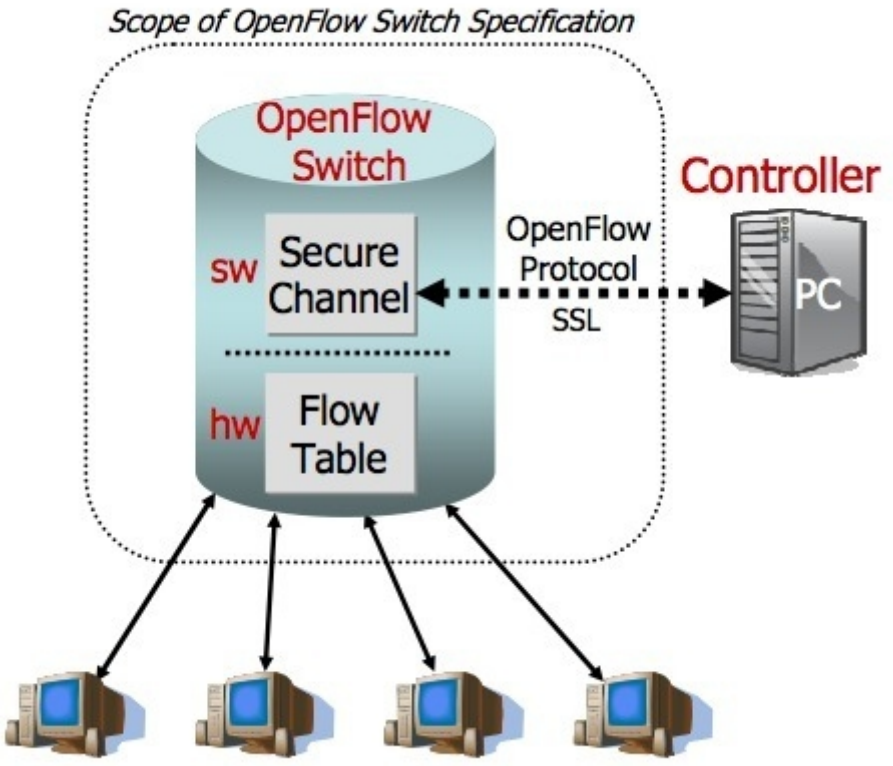

Figure 16 ideal model of OpenFlow protocol

The secure channel is an interface connecting between the OpenFlow controller and the OpenFlow switch. The controller using OpenFlow protocol is able to configure and manage OpenFlow switch through the secure channel in accordance with the specified format of OpenFlow protocol.

The processing units in each single OpenFlow switch consist of the flow tables. The

flow table is similar as the routing table of a router, the each flow table consist of many flow table entries representing the forwarding regulations. The packets entering switch can get access to the corresponding operations through the flow tables. The structure of flow table entries is shown as the figure below:

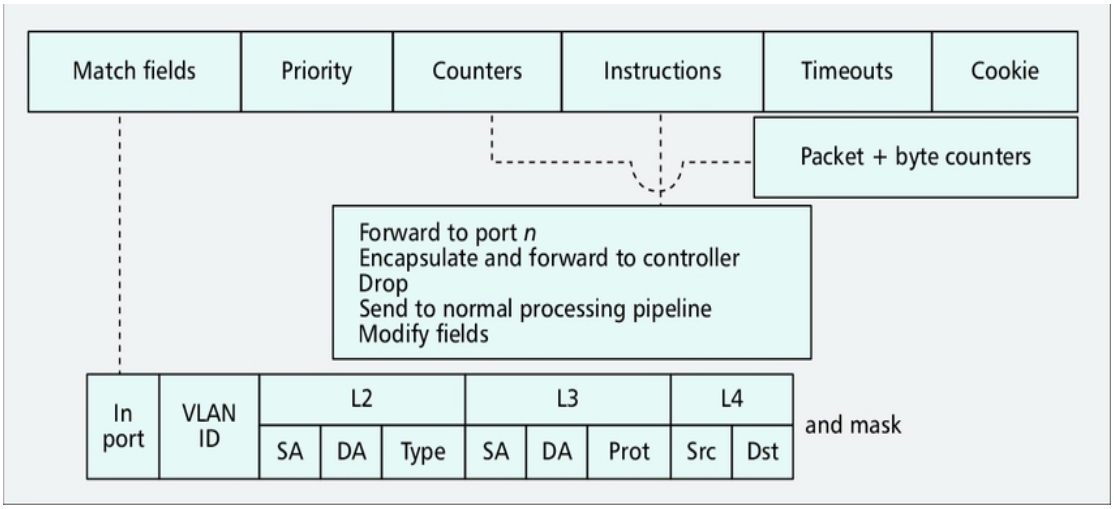

Figure 17 Structure of Flow Table Entries

The flow entry is mainly composed of match fields, priority, counters, instructions, timeout and Cookie[52]. Among them, the match field and the priority jointly define a unique flow entry in the flow table. The structure of the match field contains many matching items, covering most of the identifiers of the link layer, network layer, and transport layer. Fields such as VLAN, MPLS, and IPv6 are gradually extended to the OpenFlow standard[52]. There will no longer distinguish between routers and switches in the OpenFlow network, are collectively referred to as OpenFlow switch, which is caused by the mode of flow match and data forwarding in OpenFlow switch. The priority indicates the execution order of the flow entry when there is a conflict in the flow table; the instruction indicates the next operation step of data packet matching up with flow table entry; the counter is used to count the basic data of the data flow.

In order to improve the query efficiency of the data flow, the current flow table query obtains the corresponding operation through the multi-level flow table and pipeline mode. The OpenFlow pipeline of each OpenFlow switch contains multi-level flow tables, and each flow table contains multiple flow table entries. OpenFlow pipeline processing defines how packets interact with flow tables. An OpenFlow switch must have at least one flow table. The pipeline structure of the switch is shown in the figure below:

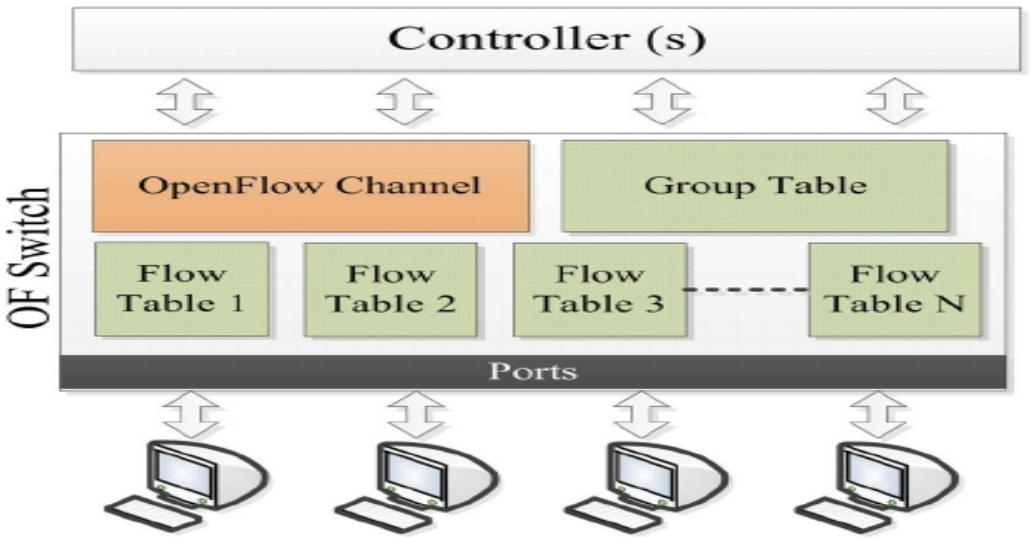

Figure 18 Multi-Level Flow Tables

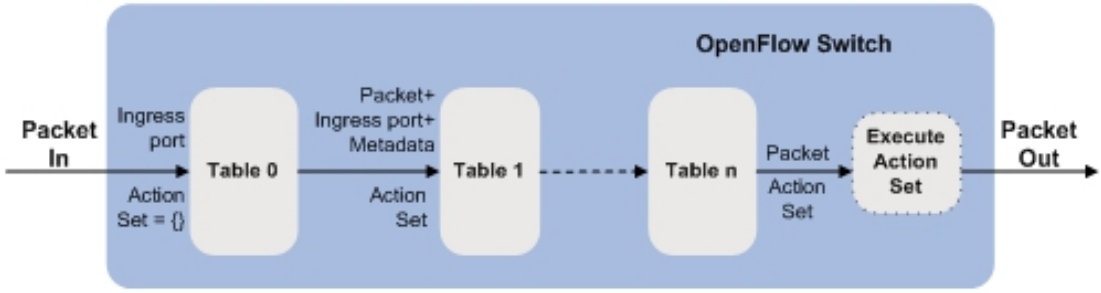

Figure 19 Pipeline Diagram Of OpenFlow Switch

When the data packet is input to the switch, the data packet searches for a matching flow table entry. When a matching flow entry is found, the forwarding behavior specified in the flow entry is executed. These instructions will explicitly point the packet to another flow table, and then perform the same operation. The flow table entry can only point the packet to the flow table whose flow table number is greater than the flow table number where it is located, that is, the processing of the pipeline can only proceed forward, not backward. Obviously, the flow entry of the last flow table cannot contain jump instructions. If the matching flow entry does not point the packet to other flow tables, the pipeline processing ends here. When the pipeline processing ends, the data packet is processed and forwarded as usual according to its combined behavior set. When a packet does not match the flow entry in the flow table, this is table-miss. The behavior of table-miss is based on the configuration of the flow table. A table-miss entry will indicate how to deal with mismatched packets: such as discarding them, or passing them to other flow tables or sending them through the control channel of packet-in messages. To the controller[52]. The above is the most

basic forwarding process of a multi-level Open Flow switch. This processing flow is shown in Figure

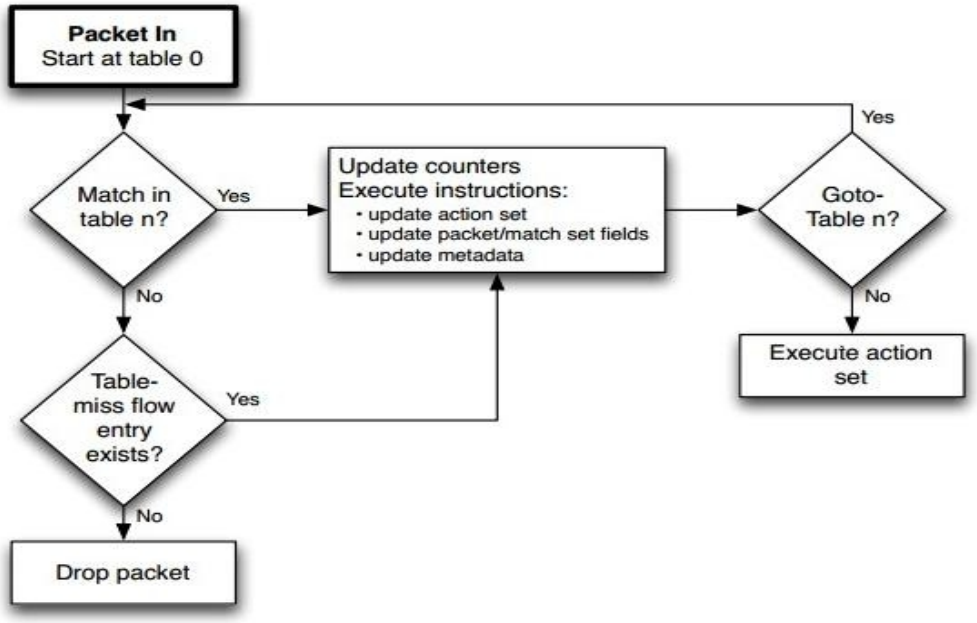

Figure 3: Flowchart detailing packet flow through an OpenFlow switch.

Figure 20 Packet Processing In Flow Table

7.4 MESSAGE CATEGORIES

The OpenFlow specification defines how an OpenFlow switch can establish connection, communication and related message types with the Controller

a) Controller/Switch messages refer to messages initiated by the Controller, received and processed by the Switch, which mainly include messages such as Features, Configuration, Modify-State, Read-State, Packet-out, Barrier, and Role-Request. These messages are mainly used by the Controller to perform operations such as state query and configuration modification of the Switch.
b) Asynchronous (Asynchronous) messages are messages sent by the Switch to the Controller to notify certain asynchronous events that occur on the Switch, including Packet-in, Flow-Removed, Port-status, and Error. For example, when a rule is deleted due to a timeout, the Switch will automatically send a Flow-Removed message to notify the Controller to facilitate the Controller to take corresponding actions, such as resetting relevant rules.
c) Symmetric messages. As the name suggests, these are two-way symmetric

messages. They are mainly used to establish a connection and detect whether the other party is online, etc., including Hello, Echo, and Experimenter.
The following figure shows a typical message exchange process between OpenFlow and Switch. For security and high availability considerations, the OpenFlow specification also stipulates how to encrypt the channel between Controller and Switch, and how to establish multiple connections, etc. (Main connection and auxiliary connection).

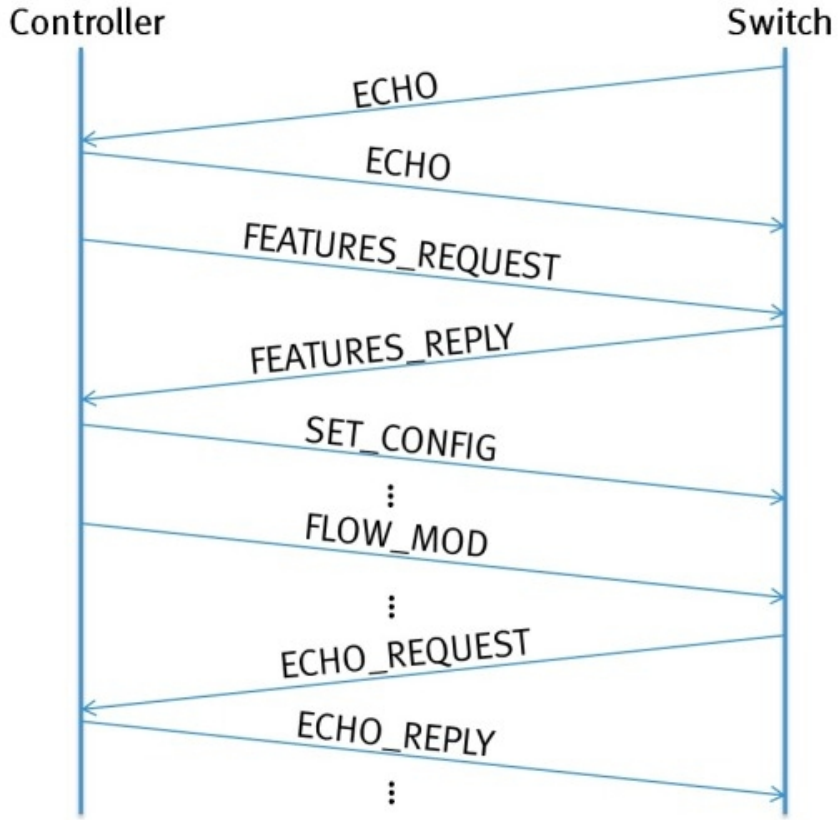

Figure 21 Message Exchange Process

At present, there are mainly two versions of Open Flow switches deployed in Linux systems based on software. The user-based software Open Flow switches are simple to operate and easy to modify, but the performance is poor; the kernel-based software Open Flow switches are faster, While providing virtualization function, but the actual modification and operation process is more complicated. In terms of hardware, the hardware-accelerated wire-speed Open Flow switch implemented by Stanford University is based on Net FPGA, and many well-known network hardware manufacturers such as NEC, HP, Huawei and other companies have successively introduced hardware switches that support Open Flow, and their implementation principles are different. There are differences, but they can be roughly divided into several implementation schemes: CPU, ASIC, and NP.

8

REFERENCE

[1] Park J , Yoon W . SDN-based heterogeneous network architecture with Multi-Controllers[C]// 2020 22nd International Conference on Advanced Communication Technology (ICACT). 2020.

[2] He, Chen , and X. Qiu . "SEA: SDN-Based Evolution Architecture for 5G Network." (2017).

[3] He, C. , & Qiu, X. . (2017). Sea: sdn-based evolution architecture for 5g network.

[4] Haleplidis E , Denazis S , Koufopavlou O . Future SDN-Based Network Architectures[M]// Handbook of Research on Redesigning the Future of Internet Architectures. 2015.

[5] J. Medved, A. Tkacik, R. Varga, et al.. Open Daylight: Towards a Model-Driven SDN Controller architecture[C]// A World of Wireless, Mobile and Multimedia Networks (Wo WMo M), 2014 IEEE 15th International Symposium on. IEEE, 2014:1-6

[6] Jain S , Kumar A , Mandal S, etc. B4: experience with a globally-deployed software defined wan[J]. Acm Sigcomm Computer Communication Review, 2013, 43(4):3-14.

[7] Lei T, Lu Z, Wen X, etc. SWAN: An SDN based campus WLAN framework[C]. International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic System. 2014:1-5

.

[8] SND Online: https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf

[9] SND Online: https://www.researchgate.net/figure/A-schematic-overview-of-SDN-implemented-

with-OpenFlow_fig1_288890479

[10] OFN-defined Online: https://en.wikipedia.org/wiki/Software-defined_networking

[11] Foundation O N. Software-Defined Networking: The New Norm for Networks[J].2012.

[12] Carthern C, Wilson W, Bedwell R, et al. Management Plane. In: Cisco Network. Apress, Berkeley, CA, 2015.

[13] Caldarola L, Choukir A, Cuda D, et al. Towards a real application-aware network[C]// International Joint Conference on E-Business and Telecommunications. IEEE, 2016:5-12.

[14] Kathiravelu P, Veiga L. Software-Defined Simulations for Continuous Development of Cloud and Data Center Networks[M]// On the Move to Meaningful Internet Systems: OTM 2016 Conferences. Springer International Publishing, 2016:3-23.

[15] S. Merugu, S. Bhattacharjee, E. Zegura and K. Calvert, "Bowman: a node OS for active networks," Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), Tel Aviv, Israel, 2000, pp. 1127-1136 vol.3, doi: 10.1109/INFCOM.2000.832473.

[16] S. Scott-Hayward, G. O'Callaghan and S. Sezer, "Sdn Security: A Survey," 2013 IEEE SDN for Future Networks and Services (SDN4FNS), Trento, 2013, pp. 1-7, doi: 10.1109/SDN4FNS.2013.6702553.

[17] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca and J. Folgueira, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges," in IEEE Communications Magazine, vol. 55, no. 5, pp. 80-87, May 2017, doi: 10.1109/MCOM.2017.1600935.

[18] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey and G. Wang, "Meridian: an SDN platform for cloud network services," in IEEE Communications Magazine, vol. 51, no. 2, pp. 120-127, February 2013, doi: 10.1109/MCOM.2013.6461196.

[19] R. L. Smeliansky, "SDN for network security," 2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), Moscow, 2014, pp. 1-5, doi: 10.1109/MoNeTeC.2014.6995602.

[20] Medved J, Varga R, Tkacik A, et al. Open Daylight: Towards a Model-Driven SDN Controller architecture[C]// IEEE, International Symposium on A World of Wireless, Mobile and Multimedia Networks. IEEE, 2014:1-6.

[21] Balus F, Bitar N, Ogaki K, et al. Federated SDN-based Controllers for NVO3[J].

2012.

[22] J. Medved, R. Varga, A. Tkacik and K. Gray, "OpenDaylight: Towards a Model-Driven SDN Controller architecture," Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, Sydney, NSW, 2014, pp. 1-6, doi: 10.1109/WoWMoM.2014.6918985.

.

[23] Alliance O. Osgi Service Platform, Release 4.2[M]. IOS Press, Inc. 2003

[24] Lelli F, Pautasso C. Design and Evaluation of a RESTful API for Controlling and Monitoring Heterogeneous Devices[J]. 2011.

[25] Z. K. Khattak, M. Awais and A. Iqbal, "Performance evaluation of OpenDaylight SDN controller," 2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), Hsinchu, 2014, pp. 671-676, doi: 10.1109/PADSW.2014.7097868.

[26] R. K. Arbettu, R. Khondoker, K. Bayarou and F. Weber, "Security analysis of OpenDaylight, ONOS, Rosemary and Ryu SDN controllers," 2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks), Montreal, QC, 2016, pp. 37-44, doi: 10.1109/NETWKS.2016.7751150.

[27] D. Suh, S. Jang, S. Han, S. Pack, T. Kim and J. Kwak, "On performance of OpenDaylight clustering," 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, 2016, pp. 407-410, doi: 10.1109/NETSOFT.2016.7502476.

[28] T. Kim, S. Choi, J. Myung and C. Lim, "Load balancing on distributed datastore in opendaylight SDN controller cluster," 2017 IEEE Conference on Network Softwarization (NetSoft), Bologna, 2017, pp. 1-3, doi: 10.1109/NETSOFT.2017.8004238.

[29] A. Vu and Y. Kim, "An implementation of hierarchical service function chaining using OpenDaylight platform," 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, 2016, pp. 411-416, doi: 10.1109/NETSOFT.2016.7502477.

[30] C. D. Cajas and D. O. Budanov, "SDN Applications and Plugins in the OpenDaylight Controller," 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg and Moscow, Russia, 2020, pp. 9-13, doi: 10.1109/EIConRus49466.2020.9039054.

[31]OpenDaylight Online: https://subscription.packtpub.com/book/networking_and_servers/9781782174523/6/ch06lvl1sec37/creating-md-sal-applications

[32] Y. Xiaohua and H. Canhui, "Design and Implementation of OpenDayLight Manager Application," 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Chengdu, China, 2020, pp. 977-982, doi: 10.1109/CISP-BMEI51763.2020.9263553.

[33] Introduction of Maven Online : http://webserver2.tecgraf.pucrio.br/~ismael/Cursos/Senac_MTSW/aulas/Modulo2_TecnicasDesnvolvimentoAgeis/2-Maven/articles/Maven2_Intro_jw-1205.pdf

[34] &Lt M B , Com&Gt M . The YANG 1.1 Data Modeling Language[J]. Annali Di Radiologia Diagnostica, 2016, 40(3):243-252.

[35] RFC 6020 Online : https://tools.ietf.org/html/rfc6020

[36] RFC 6021 Online: https://tools.ietf.org/html/rfc6021

[37] Kang M, Kang E Y, Hwang D Y, etc. Formal Modeling and Verification of SDN-OpenFlow[C]. IEEE Sixth International Conference and Testing, Verification and Validation. 2013: 481-482.

[38] Hierarchy structure of YANG Online: http://afifi.name/yang-data-modeling-language/

[39] Hui Xu and Debao Xiao, "Data modeling for NETCONF-based network management: XML schema or YANG," 2008 11th IEEE International Conference on Communication Technology, Hangzhou, 2008, pp. 561-564, doi: 10.1109/ICCT.2008.4716122.

[40] Enns R. Network Configuration Protocol (NETCONF)[J]. Rfc, 2011, 9(1):33-8.

[41] M. Wasserman, Using the NETCONF Protocol over Secure Shell (SSH), Internet Engineering Task Force (IETF), RFC 6242(Internet Standard), June. 2011. [EB/OL]. https://tools.ietf.org/html/rfc6242

[42] A. Bierman, With-defaults Capability for NETCONF, Internet Engineering Task Force (IETF), RFC 6243(Internet Standard), June. 2011. [EB/OL]. https://tools.ietf.org/html/rfc6243

[43] P. Shafer, An Architecture for Network Management Using NETCONF and YANG, Internet Engineering Task Force (IETF), RFC 6244(Internet Standard), June. 2011. [EB/OL]. https://tools.ietf.org/html/rfc6244

[44] Contrast between NETCONF and SNMP Online: https://blog.csdn.net/whatday/article/details/103418732

[45] J. Schönwälder, M. Björklund and P. Shafer, "Network configuration management using NETCONF and YANG," in IEEE Communications Magazine, vol. 48, no. 9, pp. 166-173, Sept. 2010, doi: 10.1109/MCOM.2010.5560601.

[46] M. Dallaglio, N. Sambo, F. Cugini and P. Castoldi, "Control and management of transponders with NETCONF and YANG," in IEEE/OSA Journal of Optical Communications and Networking, vol. 9, no. 3, pp. B43-B52, March 2017, doi: 10.1364/JOCN.9.000B43.

[47] Ylonen T. The Secure Shell (SSH) Protocol Architecture[C]// Rfc. 2005.

[48] Hallé S, Deca R, Cherkaoui O, et al. Automated Validation of Service Configuration on Network Devices[C]// MMNS. 2004:176-188.

[49] Interdependency between RESCTCONF, NETCONF and YANG.Online: https://facebook.azureedge.net/agenda/88/slides/slides-88-netconf-3.pdf

[50] M. Granderath and J. Schönwälder, "A Resource Efficient Implementation of the RESTCONF Protocol for OpenWrt Systems," NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 2020, pp. 1-6, doi: 10.1109/NOMS47738.2020.9110458.

[51] Open Flow Switch Specification Version 1.0.0[S], Open Network Foundation, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf

[52] Open Flow Switch Specification Version 1.3.1[S], Open Network Foundation,https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf