

Huawei SX700 Switches SDN Technology White Paper

Issue 01
Date 2016-02-15

Copyright © Huawei Technologies Co., Ltd. 2016. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <http://e.huawei.com>

About This Document

Overview

Traditional networks use the distributed control architecture. Each forwarding device (switch or router) provides a network-wide view for proactively forwarding packets. The traditional network architecture, however, cannot support rapid development of new services, such as cloud computing and Big Data services. Software-Defined Networking (SDN) is a new technology developed to overcome the shortfalls of the traditional network architecture, and is winning wide support in the industry. The OpenFlow protocol is an important part of an SDN network. Various network apps use the SDN controller platform to dynamically control network forwarding devices using the OpenFlow protocol, which makes networks programmable.






Intended Audience

This document is intended for:

- I Technical support engineers
- I Maintenance engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
 DANGER	Indicates a hazard with a high level or medium level of risk which, if not avoided, could result in death or serious injury.
 WARNING	Indicates a hazard with a low level of risk which, if not avoided, could result in minor or moderate injury.
 CAUTION	Indicates a potentially hazardous situation that, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results.
 TIP	Provides a tip that may help you solve a problem or save time.
 NOTE	Provides additional information to emphasize or supplement important points in the main text.

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Contents

About This Document	ii
1 SDN Overview	1
1.1 Traditional Network Overview	1
1.1.1 Traditional Distributed Network Architecture.....	1
1.1.2 Management, Control, and Data Planes	2
1.2 Limitations of Traditional Networks.....	2
1.2.2 Complex Traditional Network Protocols	3
1.2.3 Inflexible Traffic Path Adjustment.....	3
1.2.4 Slow Upgrade of New Services	4
1.3 Emergence of SDN	4
1.3.1 SDN Is a Network Control Architecture Featuring Centralized Control.....	4
1.3.2 The SDN Controller Is Neither an NMS Nor a Planning Tool.....	5
1.3.3 SDN Benefits	5
2 OpenFlow Working Mechanism	7
2.1 OpenFlow Overview	7
2.2 OpenFlow Channel	9
2.2.1 Controller-to-Switch	9
2.2.2 Asynchronous.....	9
2.2.3 Symmetric	10
2.2.4 Example of OpenFlow Channel Establishment	10
2.3 Table-based Packet Forwarding Mechanism of an OpenFlow Switch.....	11
2.3.1 Pipeline Processing	11
2.3.2 Flow Table Structure	12
2.3.3 Matching	12
2.3.4 Table-Miss Processing	13
2.3.5 Flow Entry Removal	13
2.3.6 Meter Table	14
2.3.7 Counters	14
2.3.8 Instructions.....	15
2.3.9 Action Set.....	16
3 Example of OpenFlow-based SDN Forwarding Mechanism	17
3.1 Introduction to the OpenDaylight Controller	17

3.2 Physical Network Connections	17
3.3 Network Topology Discovery	18
3.4 Ping Procedure	19
4 SDN Apps.....	21
4.1 XMPP-based Service Chain App	21
4.1.1 Introduction to Service Chain App.....	21
4.1.2 Implementation of Service Chain App.....	22
4.2 OpenFlow-based Lync App.....	22
4.2.1 Introduction to Lync App	22
4.2.2 Implementation Framework and Process	23
5 References.....	25
6 Appendix: OpenFlow Certification	26
A Acronyms and Abbreviations.....	27

1 SDN Overview

1.1 Traditional Network Overview

1.1.1 Traditional Distributed Network Architecture

Current communications networks, from the global Internet to enterprise private networks, are all IP-based networks. IP technology has become the core of today's communications network infrastructure. IP networks transmit various services, including data, video, and voice services. People can carry out online shopping, entertainment, social networking, and financial activities on the Internet.

IP technology has become the core of today's communications network infrastructure due to its simplicity. A host requires only a globally unique IP address to communicate with another host, without the need to know the physical location or detailed network information about the host. The simplicity of IP technology makes construction of the global Internet possible

Another characteristic of IP technology is its distributed control architecture. Switches and routers forward packets based on routing information (in this document, forwarding devices and forwarders refer to both switches and routers unless otherwise specified). In a management domain, routes are learned and calculated using an Interior Gateway Protocol (IGP), such as Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS). Across management domains, routes are synchronized using the Border Gateway Protocol (BGP). A switch uses IGP or BGP to collect network status information from its neighbors, sets up a complete network topology, calculates the shortest path between two nodes, and automatically generates routing information. If a fault occurs on a node or network link, fault information is sent layer by layer to all relevant routers or switches on the network. Upon receiving the information, the routers and switches converge to the new network topology and recalculate routes for traffic forwarding.

Routers use IGP to communicate with each other to obtain the same and complete network topology information, then independently calculate the routing table entries for packet forwarding. Route calculation is distributed and decentralized. If a router fails on a network, other routers recalculate routes to maximize communication and connection capabilities of the entire network.

1.1.2 Management, Control, and Data Planes

The management plane is responsible for management and maintenance of network devices and services. It allows users to perform operations, including management of service configurations, policies, devices, alarms, and performance, as well as fault location. Operations on the management plane are performed in terms of network elements (NEs). On the management plane, users can manage power modules, fans, temperature, hardware cards, and indicators of NEs, as well as configure NE services, including security data, service data, and network protocols.

The control plane is responsible for network control. It monitors network status in real time, and adjusts network data and behavior based on network status changes such as topology changes. This ensures that a network can work normally and provide services properly. The control plane uses protocols such as IGP and BGP.

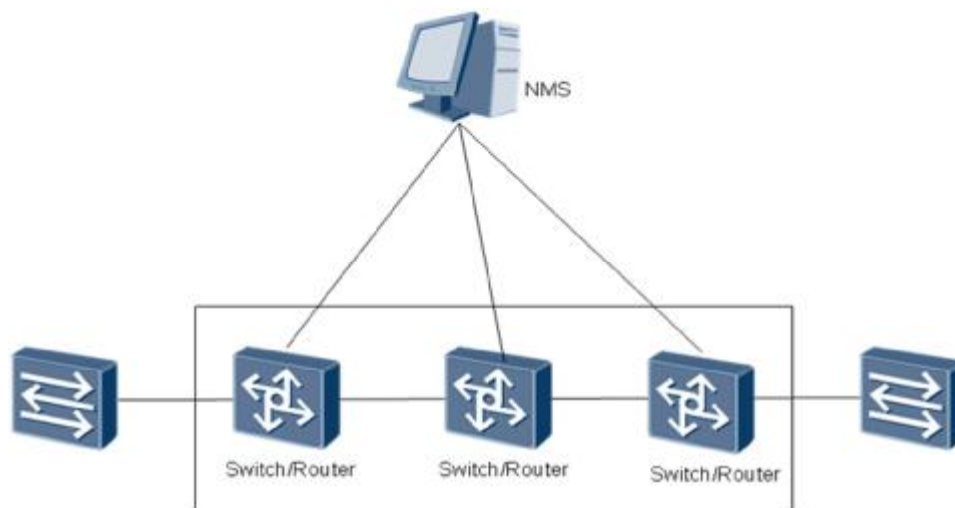
On the data plane, also known as the user plane, devices forward and process data based on instructions generated by the control plane.

Since the management plane does not monitor or control real-time network status, it can be offline for several hours or even days. If the control plane of a network runs properly when the management plane is offline, the network can work normally and provide continuous services, and services will not fail. The control plane is always online to control a network in real time.

1.2 Limitations of Traditional Networks

As previously described, a traditional network uses the distributed control architecture shown in Figure 1-1.

Figure 1-1 Traditional network architecture



A centralized network management system (NMS) server is usually deployed on a traditional network to function as the management plane. The control and data planes are distributed and run independently on each router. If the network status changes after the NMS delivers network service configurations to routers, the distributed control plane will automatically advertise the network status changes on the network. It will then automatically recalculate routes based on the new network status and update the forwarding table on the forwarding plane to restore affected user services. This distributed IP network control architecture improves network robustness, but presents limitations as communications networks develop.

1.2.2 Complex Traditional Network Protocols

There are many traditional distributed control protocols, mostly defined by the Internet Engineering Task Force (IETF), including IGP, BGP, Multiprotocol Label Switching (MPLS), multicast protocols, and IPv6 protocols. Over 30 years of development, the IETF has developed multiple protocol suites. Over 2000 protocols are directly related to network devices, and this number increases every year. Having a large number of protocols leads to complicated network maintenance and a high demand for maintenance personnel skills.

Generally, vendors use proprietary protocols in addition to these standard protocols, increasing device operations and maintenance (O&M) complexity. Over ten thousands operation commands are required for operating routers of mainstream vendors. In addition, different vendors have different operating interfaces, which makes network O&M more difficult.

1.2.3 Inflexible Traffic Path Adjustment

When a router or switch on a traditional network with the distributed control architecture collects network topology information, routing protocols running on the router or switch calculate routes based on the collected information, then generate a routing table for guiding packet forwarding. This distributed routing calculation requires that all routers in the same management domain use the same routing algorithm and no route loops occur. For example, the IGP routing protocol uses the Shortest Path Forwarding (SPF) algorithm. All devices calculate routes based on learned network topology information, including a Cost field. The routing protocol uses the Cost field to calculate the shortest path to a destination and specifies the next hop of the shortest path for forwarding traffic. This technology prevents loops on a network.

One problem with this distributed routing calculation is that traffic is transmitted only over the shortest path. If the shortest path is congested, the system cannot automatically route traffic along longer idle paths.

Multiprotocol label switching traffic engineering (MPLS TE) is defined on traditional networks to address this problem. However, MPLS TE generally uses the complex and low-scalability Resource Reservation Protocol (RSVP). RSVP can only be used to pre-plan traffic path, and cannot automatically create traffic engineering tunnels in real time based on traffic status. Therefore, traditional traffic engineering cannot adjust traffic in real time.

1.2.4 Slow Upgrade of New Services

Before new services are deployed on a network, for example, a large Layer 2 network is deployed in a data center, requirements should be discussed and standards should be defined. The IETF generally discusses standards for one to two years before reaching a consensus and launching the standards. After that, vendors require about one year to implement the standards. If the standards require that application-specific integrated circuit (ASIC) cards be upgraded, the implementation period is longer. Finally, it takes six months to one year to deploy and upgrade these services on the live network. The whole process from requirement raising to network deployment takes 3 to 5 years. In contrast, new service deployment is far more agile in the IT field. For example, the popular online WeChat red envelope service in China was released about two months after requirement analysis.

Networks serve services. As new services develop rapidly, such as cloud computing, Big Data, and 4K video services, traditional networks cannot sufficiently support the development of these services due to slow service upgrade.

1.3 Emergence of SDN

1.3.1 SDN Is a Network Control Architecture Featuring Centralized Control

SDN was first introduced in 2006 by the Clean Slate program, a Stanford University research program funded by the GENI project. The research team led by the Stanford Professor Nick McKeown proposed the concept of OpenFlow, which would allow networks to be programmed and modified flexibly just like software. After that, SDN is introduced accordingly.

The core of the SDN architecture is an SDN controller (controller and network controller in this document both refer to the SDN controller) that separates the forwarding plane from the control plane and implements centralized network control. An SDN controller, like the brain of a network, manages and controls all forwarding devices.

Figure 1-2 SDN architecture

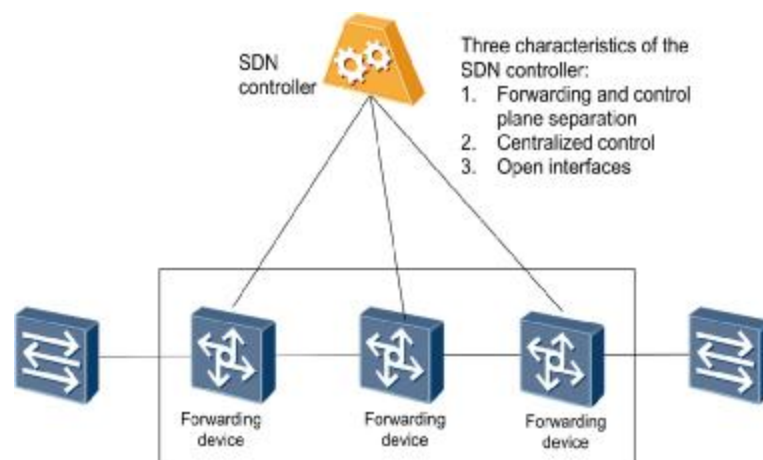


Figure 1-2 shows three basic characteristics of the SDN architecture: forwarding and control plane separation, centralized control, and open interfaces. When networks are centrally controlled and the control plane is centralized on the SDN controller, multiple distributed control protocols such as MPLS, multicast protocols, and the Multiple Border Gateway Protocol (MBGP) are no longer required. The SDN controller uses a variety of internal control programs to calculate MPLS routes, multicast routes, and service routes for each forwarding device. In this way, algorithm software on the SDN controller is used instead of traditional protocols. The centralized control technology simplifies network deployment by eliminating the need to deploy a large number of protocols, simplifying network structure. It facilitates network management and maintenance, reduces requirements on maintenance personnel skills, and lowers network O&M costs.

When centralized network control is implemented and open programmable interfaces are enabled, you can implement new services by modifying or replacing control programs, or adding new software programs to the SDN controller. This reduces the timeframe for new service deployment from 3 to 5 years to a few months or even weeks.

An SDN-based network uses an SDN controller that is not required on traditional networks. Therefore, a protocol is required to separate an SDN controller from forwarding devices and enable communication between them. Currently, OpenFlow is the most important control protocol. Other control protocols, including the Path Computation Element Communication Protocol (PCEP), Border Gateway Protocol (BGP), and Network Configuration Protocol (NETCONF), can also be used by the SDN controller to deliver control instructions to forwarders.

1.3.2 The SDN Controller Is Neither an NMS Nor a Planning Tool

People have various understandings of SDN since its emergence. Some people consider SDN as a network management system (NMS). They consider that SDN can be implemented as long as the automatic service provisioning function is implemented in real time on a centralized NMS, and northbound interfaces are open and programmable if the centralized NMS provides open northbound interfaces. An SDN controller provides functions, including forwarding and control plane separation, centralized control, and openness and programmability. However, a centralized NMS differs fundamentally in that it cannot achieve forwarding and control plane separation and centralized control.

Some people consider an SDN controller to be a planning tool. However, a planning tool and a network controller solve different problems. A planning tool is used to add nodes and links on a network during network expansion to meet requirements for future growth of network traffic. An SDN controller is a feedback control system running on an existing network. It ensures that routes converge upon network status changes so that services are provided normally.

1.3.3 SDN Benefits

1.3.3.1 Accelerating Network Service Innovation

The programmability and openness of SDN accelerate new service development and innovation. New services can be deployed on a network through modification or enhancement of SDN software. Useful new services can be reserved while unproductive services can be taken offline quickly.

1.3.3.2 Simplifying Networks

SDN simplifies networks and eliminates deployment of many IETF protocols, lowering learning costs and O&M costs. This is achieved by centralizing network control and separating the forwarding and control planes. Networks are centrally controlled by an SDN controller, so many protocols such as RSVP, Label Distribution Protocol (LDP), MBGP, and Protocol Independent Multicast (PIM) are no longer needed. Path computation and setup on an internal network are implemented on the SDN controller. The SDN controller directly delivers the calculated network paths to forwarders.

The LDP and MBGP protocols that transfers Layer 2 and Layer 3 virtual private network (L2VPN) labels respectively are no longer required. The SDN controller calculates VPN information on physical entities (PEs) and sends the information to forwarders. Similarly, intra-domain multicast protocols are also not needed. Multicast paths can be set up after the controller calculates a point-to-multipoint (P2MP) tunnel and deploys the tunnel on relevant forwarders.

1.3.3.3 Service Automation

An SDN controller uniformly manages all resources on a network in the global view, and provides southbound interfaces with powerful functions (for example, supporting the OpenFlow protocol) to control forwarding devices. In this way, network service automation is enabled and no other systems are required for service configuration resolution.

1.3.3.4 Network Path and Traffic Optimization

The shortest path first (SPF) algorithm is commonly used to implement network path and traffic optimization on traditional networks. As a result, network traffic congestion may occur on the shortest path, while other paths are idle. Under the SDN architecture, an SDN controller intelligently adjusts network paths based on network traffic status to maximize the network throughput.

Traffic engineering technologies are used on traditional networks to address this problem. For example, MPLS TE uses RSVP as the control protocol for controlling MPLS TE tunnels. Since RSVP requires that each forwarding device maintain tunnel status, MPLS TE tunnels cannot be widely deployed.

2 OpenFlow Working Mechanism

2.1 OpenFlow Overview

In December 2009, the Clean Slate program at Standard University released v1.0 of the OpenFlow protocol, which is one of the program achievements. The protocol supports only one level of flow tables and eleven flow entries in each flow table. It can implement Layer 2 forwarding and IPv4 packet forwarding.

OpenFlow v1.1 was released in February 2011. In this version, the forwarding plane consists of multi-level flow tables. MPLS, virtual local area network (VLAN), multicast protocols, and the equal-cost multi-path routing (ECMP) protocol are supported. In March 2011, the Open Networking Foundation (ONF) was founded, with a purpose to promote the SDN architecture with the OpenFlow protocol as the core.

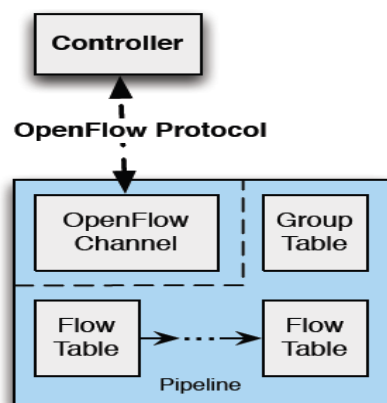
OpenFlow v1.2 was released in December 2011. It supports IPv6 and uses multiple controllers to improve reliability of sessions between a controller and a switch.

OpenFlow v1.3.0 was released in April 2012. It defines the meter table used for quality of service (QoS) control, and supports IEEE 802.1ah, that is, supporting the Provider Backbone Bridge (PBB) technology.

OpenFlow v1.3.2 was released in April 2013. It supports 39 flow entries in each flow table.

OpenFlow v1.4 was released in October 2013. It supports 41 flow entries in each flow table. In addition, some other functions are supported. For example, when a flow table on a switch is full, old flow entries are deleted according to the preset settings without the controller's intervention. OpenFlow actions are executed as one group. If the execution of one action fails, automatic rollback is implemented.

Figure 2-1 OpenFlow switch



As shown in Figure 2-1, an OpenFlow switch consists of a group table and multiple flow tables, which perform packet lookup and forwarding, and a secure channel used to interact with an external controller. The OpenFlow switch uses the OpenFlow protocol to communicate with the controller. Using the OpenFlow protocol, the switch can add, update, and delete flow entries in a flow table by sending requests to the controller and receiving replies from the controller, or by being controlled by the controller without sending requests to the controller. Each flow table in the switch contains a set of flow entries, and each flow entry consists of packet match fields, counters, and a set of instructions.

Packet matching starts from the flow table with the smallest number. Based on the orchestration of the controller, a packet forwarding task can be performed using one flow table or multiple flow tables. In a flow table, flow entries match packets in priority order, with the first matching entry in each table being used. If a packet matches a flow entry in a flow table, the instructions associated with the flow entry are executed. If no flow entry is matched, depending on the configuration of the table-miss flow entry, the switch will forward the packet to the controller using the OpenFlow protocol, discard the packet, or continue to the next flow table.

Instructions in a flow table fall into two types: specific action instructions for processing packets and pipeline processing instructions for changing packet handling processes. The action instructions describe packet forwarding, packet modification, and group table operations. The pipeline processing instructions allow packets to be sent to subsequent flow tables for further processing and allow metadata to be transmitted between flow tables for processing packets. Pipeline processing stops when the instructions associated with a matching flow entry specify the next table. At this point, the packet is modified or forwarded.

A flow entry can specify a port to which a packet is forwarded. This port is usually a physical port on a switch, but may also be a logical port or a reserved port. A logical port generally refers to a link aggregation (Trunk) port, tunnel interface, or a loopback interface. A reserved port specifies forwarding actions, such as sending a packet to the controller, flooding the packet to all physical ports, or processing a packet using a traditional switch rather than an OpenFlow switch.

During packet forwarding, the packet may be directed to a group table. Generally, group tables are used to specify actions for multicast or broadcast flooding, or other complex forwarding actions like ECMP, Fast Re-Route (FRR), and link aggregation.

2.2 OpenFlow Channel

An OpenFlow channel is the interface that connects each OpenFlow switch to a controller. Through this interface, the controller configures and manages the switch. The OpenFlow channel is encrypted using Transport Layer Security (TLS) and can also run directly over Transmission Control Protocol (TCP). The OpenFlow protocol supports three types of message: controller-to-switch, asynchronous, and symmetric, each of which has sub-types. Controller-to-switch messages are initiated by the controller and used to directly manage the switch or inspect the switch status. Asynchronous messages are initiated by the switch and used to update network events of the controller and change the switch status. Symmetric messages are initiated by the switch or controller.

2.2.1 Controller-to-Switch

Controller-to-Switch messages are initiated by the controller, and may or may not require a response from the switch.

Features: The controller obtains functions and features of a switch by sending a features request. The switch must respond with a features reply that specifies functional features of the switch. This is commonly performed upon establishment of the OpenFlow channel.

Configuration: The controller is able to configure and query configuration parameters of a switch. The switch only responds to queries from the controller.

Modify-State: Modify-State messages are sent by the controller to add, delete, and modify flow entries in OpenFlow flow tables and to set port attributes of a switch.

Read-State: Read-State messages are used by the controller to collect various information from a switch such as current configuration, statistics, and functional features.

Packet-out: Packet-out messages are used by the controller to send packets from a specified port on the switch. Packet-out messages must contain a complete packet, or a buffer ID specifying a packet stored in the switch.

Barrier: Barrier request/reply messages are used by the controller to ensure message dependencies of several messages.

Role-Request: When a switch is connected to multiple controllers, Role-Request messages are used by the controllers to set roles for their OpenFlow channels (active channel or auxiliary channel) or query configured roles.

Asynchronous-Configuration: Asynchronous-Configuration messages are used by the controller to set an additional filter on the asynchronous messages that it wants to receive.

2.2.2 Asynchronous

Asynchronous messages are sent proactively by a switch to inform a controller of packet arrival, switch status changes, or errors.

Packet-in: Transfers the control of a packet to the controller. A packet-in event is sent to the controller every time a table-miss flow entry is triggered during packets forwarding. Other processing actions such as time to live (TTL) checking also trigger packet-in events.

Flow-Removed: Informs the controller of the removal of a flow entry from a flow table. A Flow-Removed message is generated when the controller sends a request for deleting a flow entry or flow processing by a switch times out.

Port-status: Informs the controller of port change. The switch is expected to send Port-status messages to controllers when port configuration or port status changes. Port-status events include port configuration changes, for example, a port is shut down by a user.

Error: Notifies the controller of problems.

2.2.3 Symmetric

Symmetric messages are sent without requests in either direction.

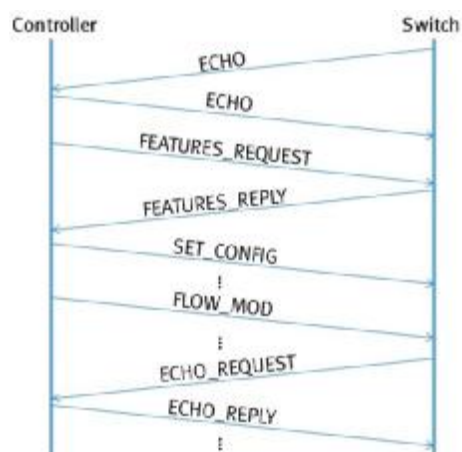
Hello: Hello messages are exchanged between the switch and controller upon connection startup.

Echo: Echo request/reply messages can be sent by either the switch or the controller. An echo request message must be responded to with an echo reply message. The messages are mainly used to check whether the connection between the controller and switch is activated, or to measure latency and bandwidth.

Experimenter: Experimenter messages provide additional functions for the OpenFlow switch. This is a staging area for features meant for future OpenFlow revisions.

2.2.4 Example of OpenFlow Channel Establishment

Figure 2-2 Channel establishment between an OpenFlow switch and a controller



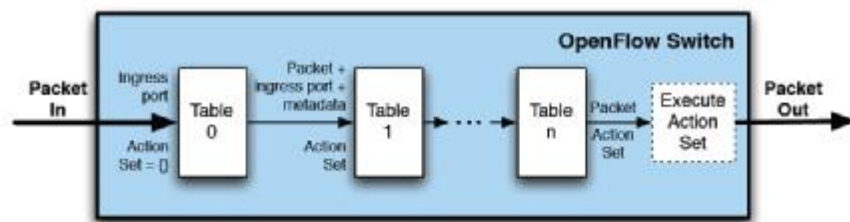
2.3 Table-based Packet Forwarding Mechanism of an OpenFlow Switch

2.3.1 Pipeline Processing

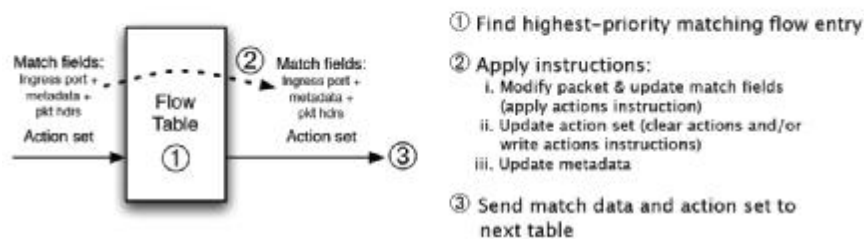
OpenFlow-compliant switches are divided into two types: OpenFlow-only and OpenFlow-hybrid. OpenFlow-only switches support only OpenFlow operations, and all packets are processed by the OpenFlow pipeline. OpenFlow-hybrid switches support both OpenFlow operations and normal Ethernet switching operations, such as traditional L2 Ethernet switching, VLAN isolation, L3 routing (over IPv4 and IPv6), access control lists (ACL), and QoS processing. OpenFlow-hybrid switches provide a flow classification mechanism outside of OpenFlow to route traffic to either the OpenFlow pipeline or the normal pipeline. For example, an OpenFlow-hybrid switch determines which pipeline to process a packet based on the VLAN tag or inbound port of the packet.

The OpenFlow pipeline of an OpenFlow switch contains multiple flow tables, each of which contains multiple flow entries. An OpenFlow switch must have at least one flow table.

Figure 2-3 Processing pipeline of an OpenFlow switch



(a) Packets are matched against multiple tables in the pipeline



- I The flow tables of an OpenFlow switch are numbered sequentially, starting from 0. Pipeline processing always starts from the first flow table: a packet is first matched against flow entries of flow table 0. Other flow tables may be used depending on the matching result in the first flow table.
- I If a packet matches a flow entry, the instruction set of the flow entry is executed. These instructions may explicitly direct the packet to another flow table, where the packet matches against flow entries. A flow entry can only direct a packet to a flow table with a larger number than its own flow table number. In other words, pipeline processing can only go forward and not backward. Therefore, the flow entries in the last flow table of the pipeline cannot include the Goto instruction. If the matching flow entry does not direct the packet to another flow table, pipeline processing stops at this table. When pipeline processing stops, the packet is processed by the action set associated with the flow table and usually forwarded.

If a packet does not match any flow entry in a flow table, a table-miss event occurs. How table-miss events are processed depends on the flow table configuration. A table-miss flow entry in the flow table can specify how to process unmatched packets. The options include discarding them, passing them to another flow table, or sending them to the controller over the control channel after encapsulation in packet-in messages.

2.3.2 Flow Table Structure

A flow table consists of flow entries, and each flow entry comprises components shown in Figure 2-4.

Figure 2-4 Components of a flow entry in an OpenFlow flow table



Match fields: used to match packets. Match fields include the ingress port, packet header, and optionally include metadata specified by a previous flow entry.

Priority: matching precedence of a flow entry.

Counters: number of matched packets.

Instruction: used to modify the action set or pipeline processing.

Timeouts: maximum aging time of a flow on a switch.

Cookie: opaque data value set by the controller. The controller can use the cookie field to filter flow statistics, flow modification, and flow deletion. This field is not used when packets are processed.

2.3.3 Matching

Figure 2-5 Packet processing through an OpenFlow switch

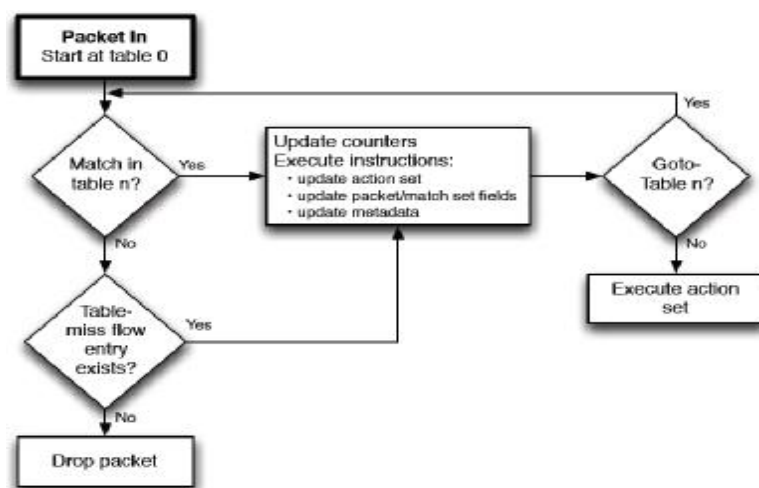


Figure 2-5 shows how an OpenFlow switch processes a packet. On receipt of a packet, an OpenFlow switch looks up the first flow table, and may look up other flow tables based on the actions defined in the matching flow entry in the first flow table.

- | Packet match fields used for table lookups depend on the packet type, and usually include various packet header fields, such as Ethernet source address or IPv4 destination address. In addition to packet headers, packets can also be matched against the inbound port and metadata fields. Metadata can be used to transit information between tables. The packet match fields represent the packet in its current state. If actions applied in the previous table using the Apply-Actions changed the packet headers, these changes are reflected in the packet match fields.
- | A packet matches a flow entry if the values in the packet match fields used for the table lookup match those defined in the flow entry. If a flow entry field contains the **ANY** field (field omitted), it is deemed to match all values in the packet header. If the switch supports arbitrary bitmask on specific match fields, these masks can precisely specify matches.
- | If a packet matches multiple flow entries in a flow table, only the matching flow entry with the highest priority is selected. The counters associated with the selected flow entry must be updated and the instruction set associated with the selected flow entry must be executed. If there are multiple matching flow entries tied for the highest priority, the selected flow entry is undetermined.
- | If the switch configuration contains the OFPC_FRAG_REASM flag, IP fragments must be reassembled before pipeline processing.

2.3.4 Table-Miss Processing

Each flow table must support a table-miss flow entry for processing table misses. The table-miss flow entry specifies how to process unmatched packets, and may, for example, send the packets to the controller, discard the packets, or direct the packets to a subsequent flow table.

- | The table-miss flow entry is identified by its match field and priority. It serves as a wildcard, matching all match fields, and has the lowest priority (0).
- | The table-miss flow entry has many similarities to other normal flow entries: it does not exist in a flow table by default, the controller may add or remove it at any time, and it may time out. The table-miss flow entry matches packets unmatched by other flow entries in a flow table. The table-miss flow entry instructions are applied to packets matching the table-miss flow entry. If the table-miss flow entry is generated and packets are directly sent to the controller using the CONTROLLER port, the reason field in the packet-in message must be specified as table-miss.
- | If the table-miss flow entry does not exist, packets that do not match flow entries are discarded by default. You can configure the switch to change the default operations and specify other actions.

2.3.5 Flow Entry Removal

Flow entries can be removed from flow tables in two ways: at the request of the controller or through the switch flow expiry mechanism.

- | The switch flow expiry mechanism is run on a switch independently of the controller based on the status and configuration of flow entries. Each flow entry has an `idle_timeout` field and a `hard_timeout` field. If neither of the field values is 0, the switch must focus on the expiry of the flow entry, as it may need to remove the flow entry later. If the `hard_timeout` field is non-zero, the flow entry will be removed after the specified number of seconds regardless of how many packets it has matched. If the `idle_timeout` field is non-zero, the flow entry will be removed if it does not match any packets within the specified number of seconds.

- I The controller may actively remove flow entries from flow tables by sending messages.
- I When a flow entry is removed either by the controller or the flow expiry mechanism, the switch must check the `OFPPF_SEND_FLOW_REM` flag of the flow entry. If the flag is set to 1, the switch must send a flow removal message to the controller. Each flow removal message contains a complete description of the flow entry, the reason for removal (expiry or deletion by the controller), the flow entry duration at the time of removal, and the flow statistics at the time of removal.

2.3.6 Meter Table

A meter table implements flow-based measurement, enables OpenFlow switches to implement simple QoS operations such as rate limiting, and can be combined with port-based queues to implement DiffServ. Meters measure and control packet rate, and are attached directly to flow entries (as opposed to queues which are attached to ports). A flow entry can specify a meter in its instruction set to measure and control the rate of the flow entry to which it is attached.

Figure 2-6 Components of an OpenFlow meter table

Meter Identifier	Meter Bands	Counters
------------------	-------------	----------

A meter table contains the following components:

Meter identifier: a 32-bit unsigned integer that uniquely identifies a meter.

Meter bands: an unordered list of meter bands, where each meter band specifies the band rate and the way to process packets.

Counters: updated when packets are processed by a meter.

2.3.7 Counters

Counters are maintained for each flow table, flow entry, port, queue, group, group bucket, meter, and meter band. Figure 2-7 describes some counters defined by the OpenFlow protocol. A switch does not need to support all counters, but needs to support those marked "Required".

Figure 2-7 List of some counters defined by the OpenFlow protocol

Counter	Bits	
Per Flow Table		
Reference Count (active entries)	32	<i>Required</i>
Packet Lookups	64	<i>Optional</i>
Packet Matches	64	<i>Optional</i>
Per Flow Entry		
Received Packets	64	<i>Optional</i>
Received Bytes	64	<i>Optional</i>
Duration (seconds)	32	<i>Required</i>
Duration (nanoseconds)	32	<i>Optional</i>
Per Port		
Received Packets	64	<i>Required</i>
Transmitted Packets	64	<i>Required</i>
Received Bytes	64	<i>Optional</i>
Transmitted Bytes	64	<i>Optional</i>
Receive Drops	64	<i>Optional</i>
Transmit Drops	64	<i>Optional</i>
Receive Errors	64	<i>Optional</i>
Transmit Errors	64	<i>Optional</i>
Receive Frame Alignment Errors	64	<i>Optional</i>
Receive Overrun Errors	64	<i>Optional</i>
Receive CRC Errors	64	<i>Optional</i>
Collisions	64	<i>Optional</i>
Duration (seconds)	32	<i>Required</i>
Duration (nanoseconds)	32	<i>Optional</i>

2.3.8 Instructions

Each flow entry contains a set of instructions that are executed when a packet matches the flow entry. These instructions result in changes to the packet, action set, and/or pipeline processing.

- I A switch does not need to support all instructions, but needs to support those marked "Required". The controller can also query a switch about which optional instructions are supported by the switch. The following lists some instructions:
 - (Optional) Meter meter_id: Directs a packet to a specified meter. As a result of metering, the packet may be discarded.
 - (Optional) Apply-Actions action(s): Applies the specific action or actions immediately without changing the Action Set. This instruction can be used to modify packets between two flow tables or to execute multiple actions of the same type.
 - (Optional) Clear-Actions: immediately clears all actions in the action set.
- I The instruction set associated with each flow entry contains a maximum of one instruction of each type.
- I If a switch is unable to execute instructions associated with a flow entry, it can reject the flow entry and return an unsupported flow error.

2.3.9 Action Set

An action set is associated with each packet. The set is empty by default. A flow entry can use the Write-Actions or Clear-Actions instruction to modify the matching action set. The action set is transferred between flow tables. When the instruction set of a flow entry does not contain the Goto-Table instruction, pipeline processing stops and the actions in the action set of the packet are executed.

- I An action set contains a maximum of one action of each type. When multiple actions of the same type are required, for example, pushing multiple MPLS labels or popping multiple MPLS labels, the Apply-Actions instruction should be used.
- I Actions in an action set are applied in the order specified in the following order, regardless of the order that they were added to the set. A switch also supports action execution in an arbitrary order through the action list of the Apply-Actions instruction.
 - copy TTL inwards: apply the copy TTL inwards action to a packet.
 - pop: apply the all label pop actions to a packet.
 - push-MPLS: apply the MPLS label push action to a packet.
 - push-PBB: apply the PBB label push action to a packet.
 - Push-VLAN: apply the VLAN label push action to a packet.
 - copy TTL outwards: apply the copy TTL outwards action to a packet.
 - decrement TTL: apply the decrement TTL action to a packet.
 - set: apply all set-field actions to a packet.
 - qos: apply all QoS actions to a packet, such as se_queue.
 - output: If no group action is specified, apply the output action to forward the packet through the specified port.
- I The output action in the action set is executed last. If both the output action and the group action are specified in an action set, the output action is ignored and the group action takes precedence. If no output action and no group action are specified in an action set, the packet is discarded.

3 Example of OpenFlow-based SDN Forwarding Mechanism

3.1 Introduction to the OpenDaylight Controller

The OpenDaylight open source project was founded on April 8, 2013. Participants include traditional network device vendors Cisco, Huawei, and Juniper; traditional IT software and hardware device vendors IBM and Microsoft; emerging network device vendors Arista and Big Switch,; and emerging IT software vendors VMware, Red Hat, and Citrix. The scope of the project demonstrates the industry opportunities SDN carries in terms of research, development, and innovation. The OpenDaylight open source project cooperates with the Linux Foundation and aims to develop the OpenDaylight platform to be the core component (controller) of the SDN architecture. OpenDaylight simplifies network operations, extends hardware lifecycle, and supports innovation of new SDN services and capabilities.

3.2 Physical Network Connections

Figure 3-1 Physical network connections under the OpenDaylight architecture

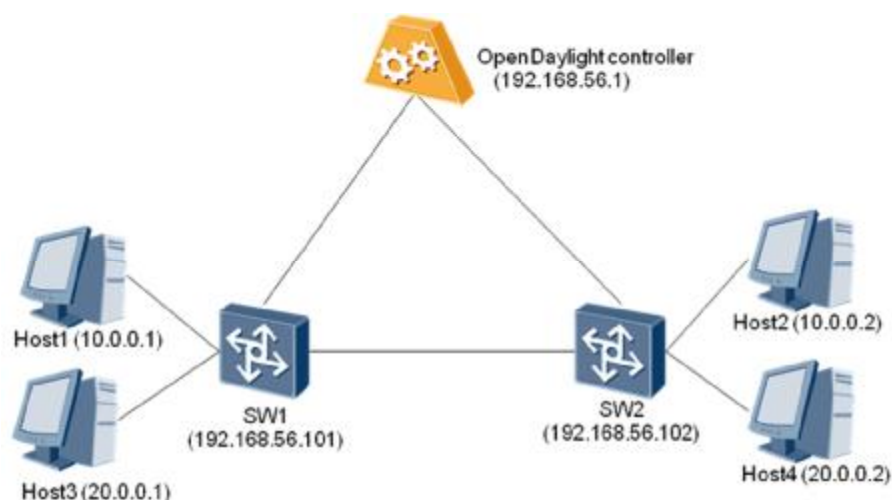
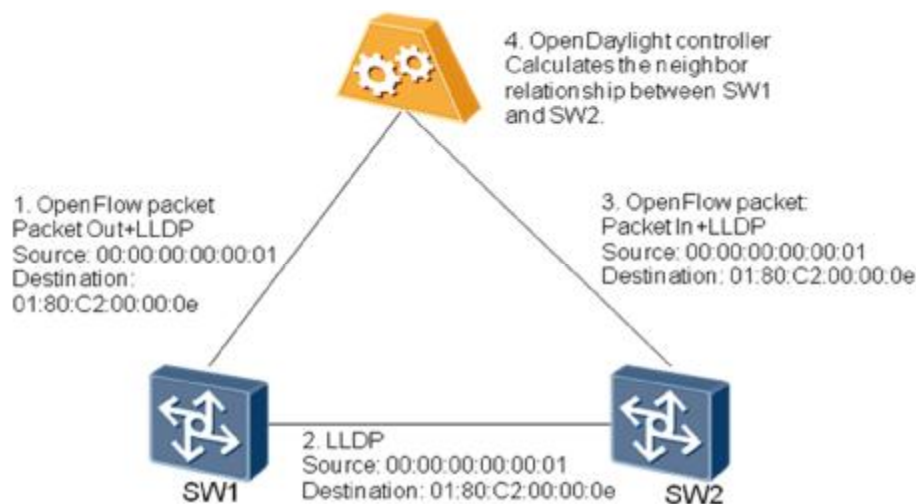


Figure 3-1 shows one OpenDaylight controller and two switches deployed on a network. Each switch is connected to two hosts that belong to two network segments. The controller uses the OpenFlow protocol to communicate with the switches.

3.3 Network Topology Discovery

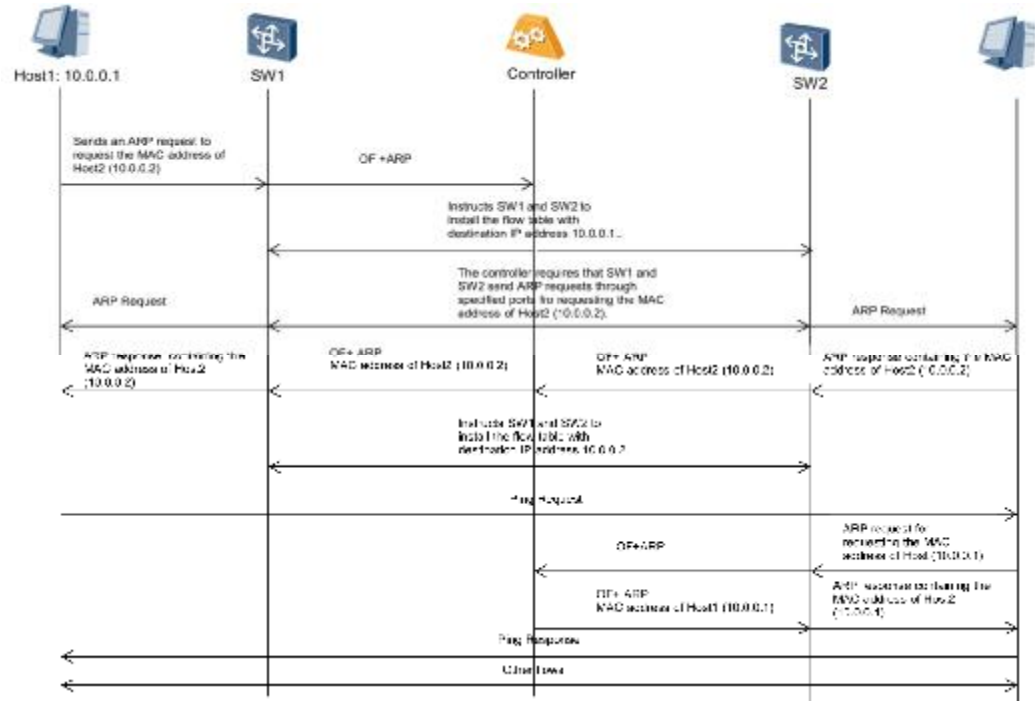
Figure 3-2 Network topology discovery under the OpenDaylight architecture



After switches are registered, the controller discovers or updates the network topology. When a new switch SW1 accesses the network, the controller uses the Packet Out instruction of the OpenFlow protocol to require that SW1 send a Link Layer Discovery Protocol (LLDP, defined in IEEE802.1ab) packet out through all its ports to detect links. The source MAC address of the LLDP packet is allocated by the controller. In this example, the MAC address of the LLDP packet is 00:00:00:00:00:01. The controller allocates such a MAC address for each switch as the switch identifier. The destination MAC address of the LLDP packet is a multicast address. When the neighboring switch SW2 receives the LLDP packet, it cannot identify the packet. It sends the LLDP packet to the controller using the OpenFlow protocol. The controller calculates the network topology between switches by exchanging LLDP packets with switches. Forwarding flow tables are generated and network visualization are achieved based on the network topology. (Note: Hosts connected to SW2 also receive the LLDP packet but do not process it.)

3.4 Ping Procedure

Figure 3-3 Ping procedure



On an SDN network, hosts at end points do not know that they have connected to the SDN network. If a host wants to send a data packet to another host, the Address Resolution Protocol (ARP) is still required to map IP addresses to MAC addresses. However, the SDN-based processing mechanism is different from the MAC address learning and flooding mechanism of Layer 2 Ethernet switches. An example of the SDN-based processing procedure is as follows:

1. When the source host Host1 (10.0.0.1) sends an ARP request to Host2 (10.0.0.2), SW1 does not know how to forward the packet. It sends the packet to the controller using the OpenFlow protocol.
2. The controller discovers that the ARP packet is sent by Host1 (10.0.0.1) and obtains the location of Host1 (the OpenFlow packet specifies the switch that sends the data packet and the switch port that the packet is sent from). At this time, the controller calculates the network topology and obtains the forwarding routes from all network nodes to Host1 (10.0.0.1). It then sends the generated flow tables to all switches using Flow Modify messages of the OpenFlow protocol.
3. After receiving the ARP request, the controller requires that all switches send ARP requests through non-switch interconnection ports within network segment 10.0.0.0/8 (only these ports are connected to hosts or traditional networks) to request the MAC address of Host2 (10.0.0.2). The controller does not directly forward the received ARP request; instead, it changes the source IP address of the ARP request to 10.0.0.254 (gateway IP address configured on the controller previously) before sending it.
4. Only Host2 (10.0.0.2) responds to the ARP request, and sends an ARP Response packet to SW2. SW2 does not know how to process the packet and it encapsulates it into an OpenFlow packet and sends the OpenFlow packet to the controller. The controller

discovers that it is an ARP Response packet and an ARP request is sent from Host1 (10.0.0.1). The controller sends the ARP Response packet to SW1 using the OpenFlow protocol and specifies the port through which SW1 forward the ARP Response packet (port connecting to Host1). SW1 forwards the ARP Response packet to Host1 through the specified port.

5. After receiving the ARP response packet from Host2, the controller obtains the location of Host2 (10.0.0.2). The controller performs routing calculation based on the network topology and obtains the forwarding routes from all network nodes to 10.0.0.2. It then sends the generated flow tables to all switches using Flow Modify messages of the OpenFlow protocol.
6. After receiving the ARP Response packet, Host1 uses ARP to map the IP address of Host2 to its MAC address. Host1 then initiates an ICMP PING Request packet with the MAC addresses of Host1 and Host2 as the source and destination MAC addresses, and the IP addresses of Host1 and Host2 as the source and destination IP addresses, respectively. Since the flow table specifying routes to Host 2 (10.0.0.2) has been successfully loaded on SW1 and SW2, the ICMP PING Request will be successfully delivered to Host2.
7. Host2 receives the ICMP PING Request with Host1 as the source host, but Host2 has not obtained the MAC address of Host1. After implementing address resolution using ARP again, SW2 sends the ARP request to the controller again that has obtained the MAC address of Host1. The controller returns a reply message to SW2 that contains the ARP request result and specifies the port through which the reply message is sent to Host2 (the port connecting to Host2).
8. After learning the ARP information, Host2 initiates an ICMP Response packet and sends the packets to SW2. SW2 matches the destination address of Host1 against entries in the forwarding table and forwards the packet to SW1. SW1 matches the destination address of Host1 against entries in the forwarding table and sends the packet to the corresponding port of Host1. In this way, bidirectional tunnels between Host1 and Host2 are set up.

4 SDN Apps

4.1 XMPP-based Service Chain App

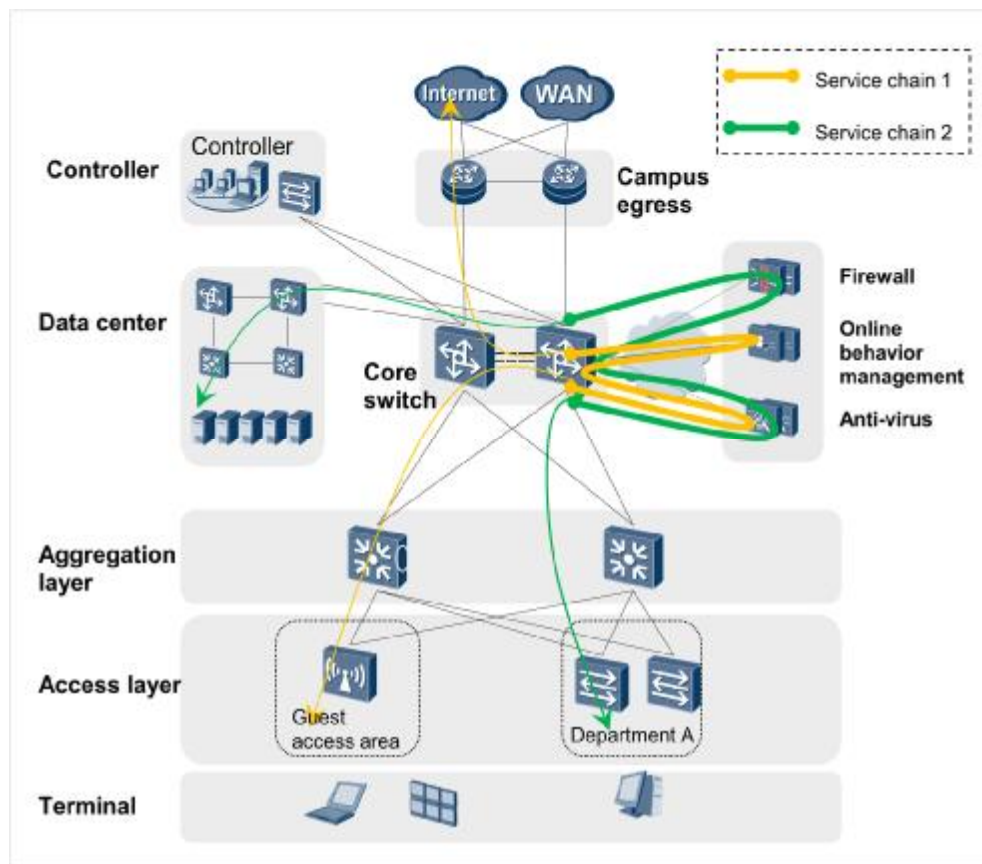
4.1.1 Introduction to Service Chain App

On a traditional campus network, value-added service devices, such as firewalls, antivirus engines (AVEs), and application security gateways (ASGs), are usually deployed at borders of important service departments, demilitarized zones (DMZs), campus egresses, and data centers. These devices are connected to the network in inline or bypass mode. This on-demand deployment mode has high costs and low efficiency.

To reduce costs and improve efficiency, these security devices are deployed in one place to form a security device pool. Traffic requiring high security is directed into this pool. Service chain app is developed based on the SDN controller. It communicates with switches using the Extensible Messaging and Presence Protocol (XMPP) and dynamically directs target service traffic into the security device pool.

4.1.2 Implementation of Service Chain App

Figure 4-1 Implementation of service chain app



Service chain app is a software application on the controller and is used for service logic configuration of service chains on the graphical user interfaces (GUIs) of the controller. Service logic specifies which type of service traffic (defined by 5-tuple ACLs) requires security control and which security control systems and devices such as intrusion prevention system (IPS) and AVE are required. Switches use 5-tuple ACLs to distinguish different types of service traffic and direct the service traffic into the security device pool over Generic Routing Encapsulation (GRE) tunnels. After processing the service traffic, security devices direct the traffic to the switches again over GRE tunnels. All forwarding paths of the service traffic are controlled by the service chain app.

4.2 OpenFlow-based Lync App

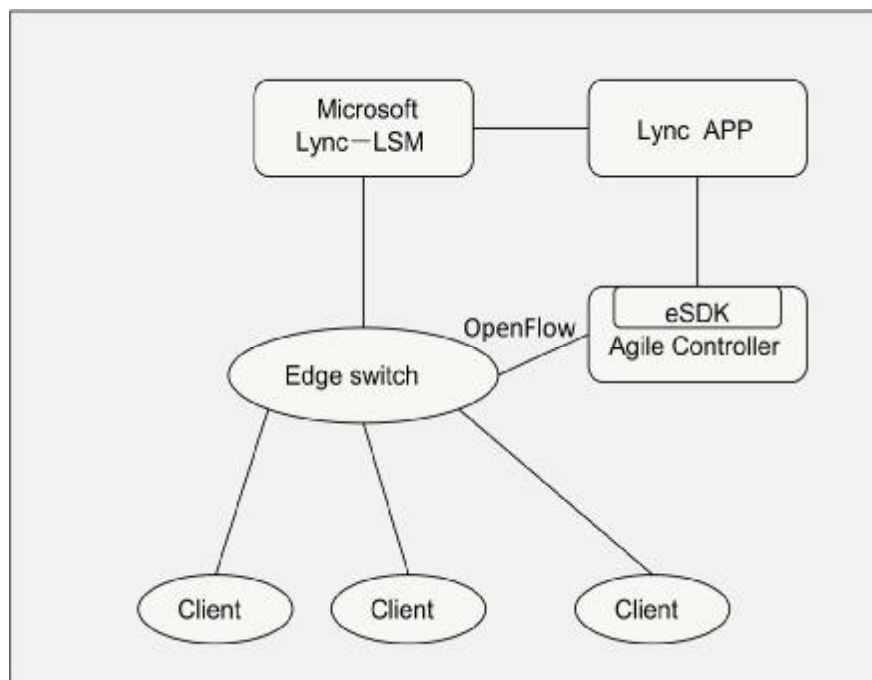
4.2.1 Introduction to Lync App

Lync is an instant messaging (IM) system released by Microsoft, which supports voice, video, and data services. The Lync app uses the application programming interface (API) provided by the SDN controller. It allows automatic deployment of network policies and end-to-end QoS policies, improving user experience.

4.2.2 Implementation Framework and Process

Service apps interconnect with Lync SDN API 2.0 provided by Microsoft and the software development kit (SDK) provided by Huawei Agile Controller to improve the priority of key Lync service data (voice and video), improving users' service experience.

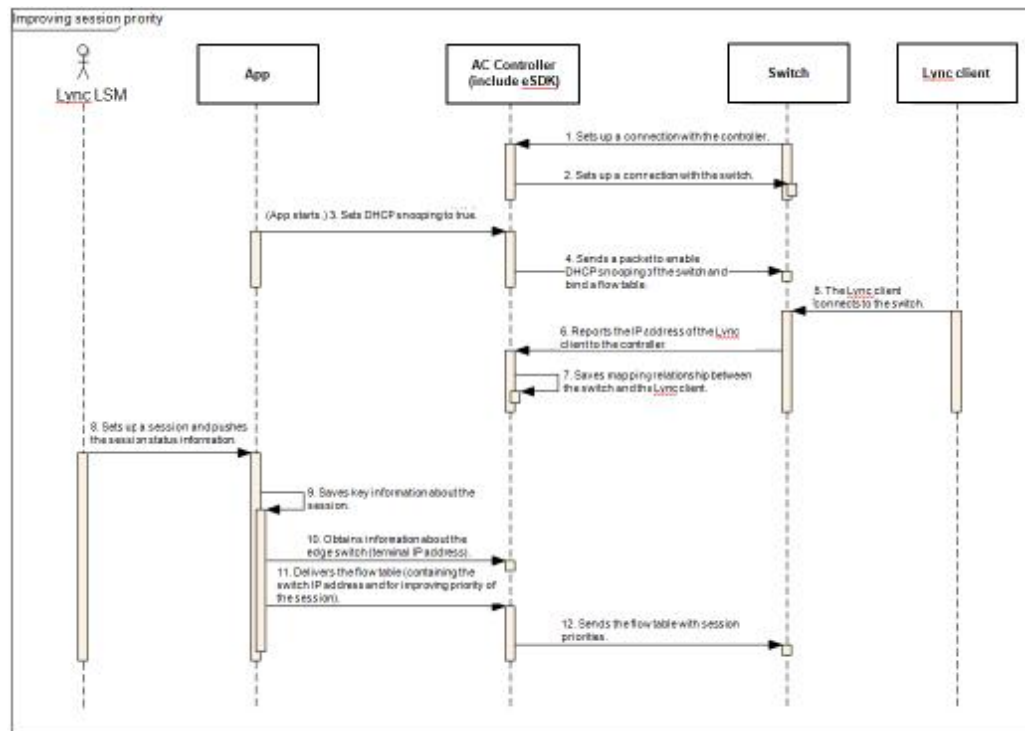
Figure 4-2 Lync app module



The Lync app uses eSDK interfaces and Agile Controller provided by Huawei to implement message subscription and flow table delivery. Lync SDN Manager (LSM) is one of the Lync SDN API components. It notifies the Lync app of instant messaging state changes, such as connections and disconnections. After receiving a message pushed by LSM, the Lync app extracts relevant message information, and encapsulates the message into an OpenFlow message. The Lync app then delivers the OpenFlow message to edge switches that users connect to. Flow tables for priority control are established for ensuring quality of audio and video sessions.

Figure 4-3 shows the complete implementation process.

Figure 4-3 Interaction process using Lync app



5 References

Standard	Document Name or Link	Remarks
openflow-spec-v1.4.0	OpenFlow Switch Specification	
OpenDaylight	https://www.opendaylight.org/	

6 Appendix: OpenFlow Certification

The global SDN Certified Testing Center (website: www.sdnectc.com), cooperating with SDN/Network Functions Virtualization (NFV) standard organizations such as the ONF, is dedicated to making testing rules, establishing a testing certification system, and research and development of testing tools. As a third-party and neutral testing institution, it provides testing and certification services around the world, driving technology and product improvement and supporting SDN/NFV deployment.

In December 2015, Huawei S12700, S9300, S7700, S6720, S5720SI, and S5720EI switches passed tests administrated by the OpenFlow testing tool OFsuite officially launched by the ONF, and obtained OpenFlow 1.3 Protocol Consistency Certificates awarded by the Global SDN Certified Testing Center. Huawei switches are highly recognized in the industry.

A Acronyms and Abbreviations

Acronym/Abbreviation	Full Name
ONF	Open Network Foundation
XMPP	Extensible Messaging and Presence Protocol
BGP	Border Gateway Protocol
PCEP	Path Computation Element Protocol
NETCONF	Network Configuration Protocol
IGP	Interior Gateway Protocol
OSPF	Open Shortest Path First
ISIS	Intermediate System to Intermediate System