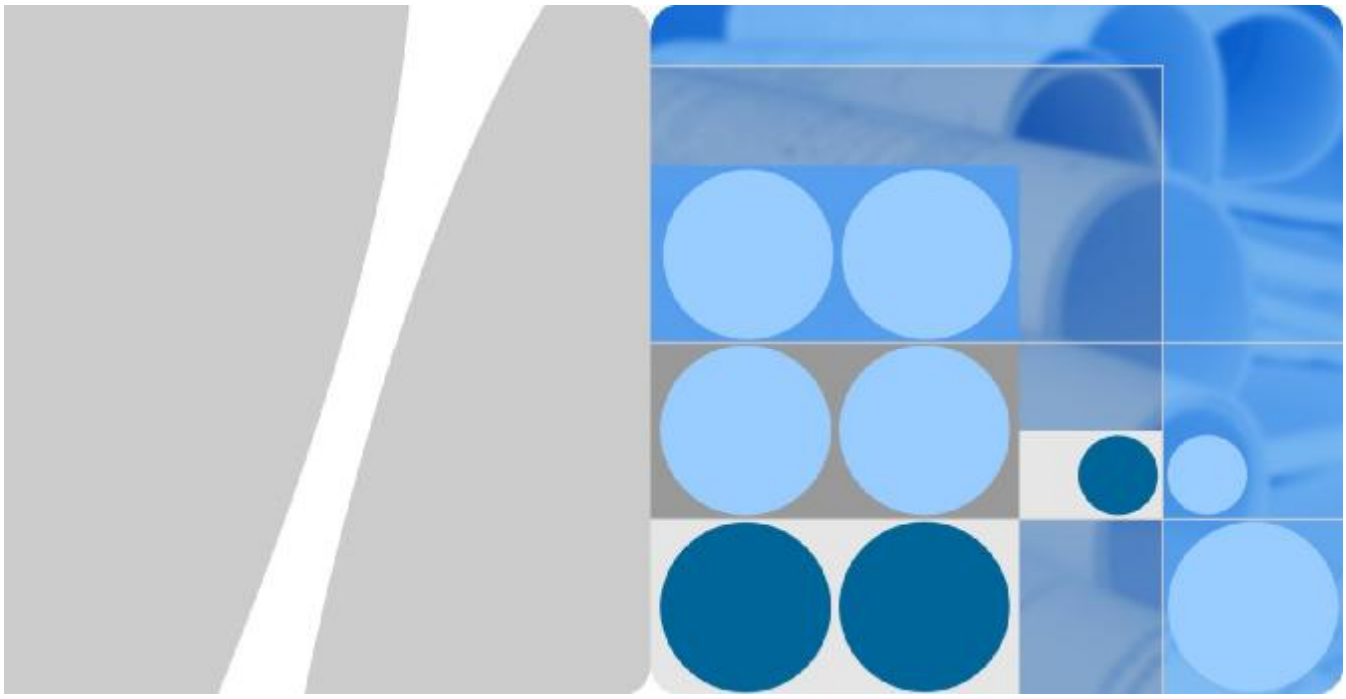


Huawei Sx700 Series Switches OpenFlow Datasheet

Issue 02

Date 2015-07-30

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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://www.huawei.com

Email: support@huawei.com

Contents

Huawei Sx700 Series Switches OpenFlow Datasheet

1.1 Challenges Facing the Network

The traditional network architecture no longer meets market requirements. Instead of creating business benefits, IT incurs additional costs for most enterprises and organizations. Faced with reduced budgets, enterprises even cut their IT investments. At the same time, enterprises strive to improve existing IT equipment and manpower utilization with the aim of enhancing their IT support capabilities and maintaining competitive advantage. The telecommunications carriers face similar challenges. They must rely on constant network capacity expansion to cope with explosive growth of mobile terminals and access bandwidth requirements. Additionally, profits are being drained by escalating equipment and maintenance costs, followed with shrinking net profits. Specifically, the live network architecture is facing the following challenges:

Increasingly complex network

Network technology to date has consisted largely of separate protocols, which are the languages used by machines to communicate. With these languages, communications devices can constitute any network topology with any distances and link speeds. To meet business and technology needs, major standard organizations of the industry, such as the IEEE, IETF, and 3GPP have developed a variety of network protocols to deliver higher bandwidth, reliability, and security. In most cases, the protocols are independent of each other, with each designed to solve a specific network problem. As the protocols accumulate, the current networks become more and more complex.

In the Open Network Summit (ONS) held in March 2014, one of the attendees showed astonishing statistics that in a small data center with only 16 rack servers, the IT personnel need to enter over 80,000 command lines to configure protocols. To add or remove a device on a network, the IT personnel need to adjust the switches, routers, firewalls, and web authentication servers, and update ACLs, VLANs, and QoS. The tools used by the IT personnel are specific to certain network devices. The IT personnel have to take the vendor switch model, software version, and network topology into account during the adjustment and update. Due to this complexity, today's networks are relatively static as the IT personnel seeks to minimize the risk of service disruption.

However, server virtualization is making waves throughput the traditional static network framework. One physical server can be virtualized into several or dozens of virtual machines (VMs). Each VM is an independent network element. Prior to virtualization, applications usually reside on an independent physical server, and the traffic model is fixed. With virtualization applied to the network, applications are distributed across multiple VMs. VMs may migrate to balance load and conserve energy. VM migration challenges the traditional networks. As VM may move from one port o another across a network, it becomes difficult to assign IP addresses and VLANs, and to divide network segments.

Most enterprises use an IP converged network to carry voice, video, and data services. To provide differentiated services, QoS must be deployed. Although the vendors claim support for a standard DiffServ QoS model, the implementation varies greatly. The IT personnel must configure each vendor's devices independently, adjust bandwidth and QoS parameters. Such a static and manual QoS configuration prevents the network from dynamically responding to changing traffic, applications, and user demands.

Policy inconsistency

To execute a network adjustment policy, the IT personnel may need to configure hundreds of devices. For example, cloud desktops are widely used by enterprises. Every time a virtual desktop is installed, the IT personnel may take hours or days to re-configure ACLs across the network. The complexity of the networks makes it difficult for the IT personnel to deploy consistent access control, application security, and QoS policies, leaving the enterprises stuck on security breaches and non-compliance with regulations.

Vendor dependence

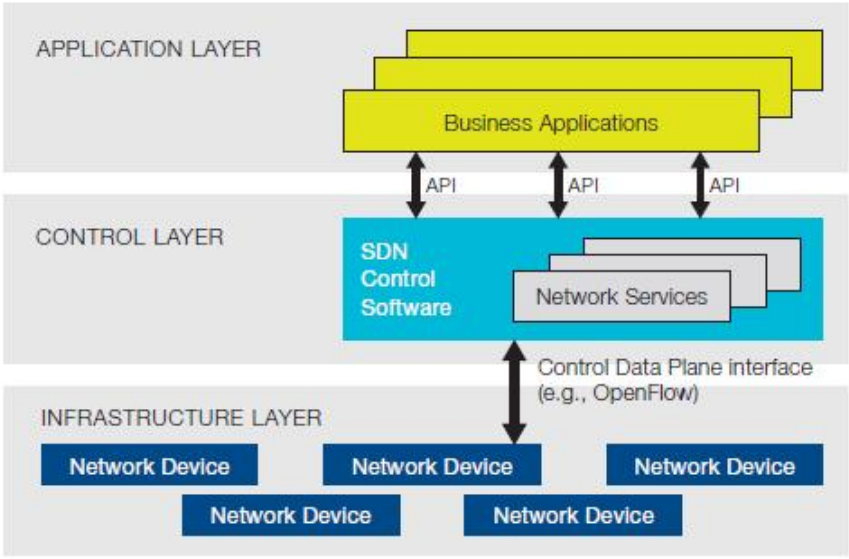
Surging popularity of Internet of Things (IoT), cloud computing, and BYOD requires that the equipment manufacturers respond quickly to emerging services. However, the product cycles always hinder their response. It takes two or three years to launch a new service by modifying the ASID and at least half a year through software updates. Lack of standard and open APIs limits the network's ability to adapt to various environments.

In conclusion, the contradiction between market demands and network capability must be resolved and calls for a more flexible network architecture. In response, Software Defined Network (SDN) is introduced.

1.2 SDN Origination and Development

The origins of SDN began in the 1990s when AT&T's GeoPlex project leveraged network APIs to implement Middleware networks. In 2008, Stanford University and UC Berkeley officially launched the SDN architecture. SDN decouples the network control plane from the forwarding plane (or data plane) and provides programmable network interfaces. On a traditional network, each switch has a complete control system to collect network status data and autonomously determine forwarding behaviors. The SDN architecture separates the control capabilities of all switches and concentrates them on common servers. The forwarding capabilities of the switches are abstracted and implemented by standard interfaces. As all control capabilities are migrated to servers, the servers possess a complete control capability view. The applications take the network as a virtual entity and do not need to care about differences among thousands of physical switches.

As shown in the following figure, the SDN architecture consists of a forwarding layer, a control layer, and an application layer.



The forwarding layer is composed of switches and routers. Stripped with the control capabilities, SDN-capable switches and routers only need to focus on high QoS, and reliable and secure data forwarding, without the need to understand thousands of protocols. Different SDN implementation modes come with different abstractions of the forwarding layer. For example, the IETF Interface to Routing System (I2RS) work group abstracts the forwarding layer into the Routing Information Base (RIB) or Forwarding Information Base (FIB). The SDN controller leverages standard I2RS interfaces and protocols to operate the RIB or FIB and then define different forwarding behaviors. For another example, the OpenFlow-based SDN abstracts the forwarding layer into a series of flow tables and actions. The SDN controller delivers flow tables and actions to the forwarding layer through the OpenFlow protocol to dynamically orchestrate packet forwarding behaviors.

The control layer is the core of SDN and consists of a group of common servers used as the controllers. These SDN controllers are integrated with network intelligence (control capabilities) and maintain a global view of the network, including the network components, topology, and device capability. The control layer provides the application layer with a complete set of open APIs for network services, which include routing, multicast, security, access control, bandwidth management, traffic engineering, QoS, forwarding device processor and storage optimization, energy control, and policy management. The application layer can use the open APIs to quickly customize network forwarding behaviors and develop various network applications. For example, based on the SDN architecture, consistent policies can be easily defined and implemented on a campus network with both wired and wireless devices.

The application layer is the embodiment of network values. Applications include the unified communication system (such as Huawei eSpace), high definition (HD) video conference system, or network performance monitoring and access policy control applications. By leveraging the standard APIs provided by the control layer, the IT personnel can easily program network behaviors and deploy network applications in real time, without the need to wait until the device manufacturers embed necessary features into closed forwarding software.

The SDN architecture makes the network be not too application-aware nor the applications too network-aware. Through forwarding layer abstraction and interface standardization, SDN simplifies the design of the forwarding plane, improves the flexibility of the control plane, and expedites service innovation at the application layer.

1.3 OpenFlow-One Implementation of SDN

OpenFlow Evolution

In December 2009, OpenFlow1.0 was released as one of the accomplishments for Stanford University's Clean Slate project. OpenFlow1.0 supports only 1-level flow tables and 11 match fields, and provides basic functions such as Layer 2 forwarding and IPv4.

OpenFlow1.1 was released in February 2011. In OpenFlow1.1, the forwarding plane consists of multi-level flow tables. OpenFlow1.1 also adds support for MPLS, VLAN, multicast, and ECMP.

In March 2011, Open Network Foundation (ONF) was established. It promotes the industry to use the OpenFlow-based SDN architecture.

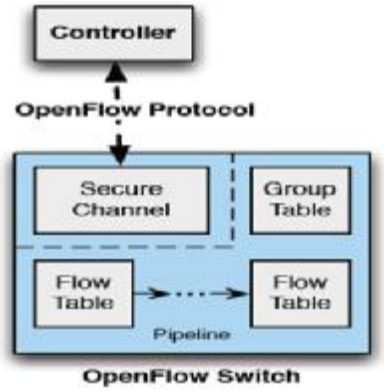
In December 2011, OpenFlow1.2 was released. It adds support for IPv6 features and uses a multiple-controller scheme to improve reliability of communications between the controllers and switches.

In April 2012, OpenFlow1.3.0 was released. It defines the meter table used for QoS control and adds support for IEEE 802.1ah, that is, the Provider Backbone Bridges (PBB).

OpenFlow 1.3.2 was released in April, 2013. It supports 39 flow match fields.

OpenFlow 1.4 released on October, 2013 supports 41 flow match fields. In addition, some new functions are added in OpenFlow 1.4. For example, when the flow table of a switch is full, the switch automatically deletes the old flow entries according to pre-configured rules, and the controller does not need to participate; a group of OpenFlow actions can be executed simultaneously. If execution of one action fails, all actions must be executed again.

OpenFlow Switch Components and Working Process



As shown in the figure, an OpenFlow switch has multiple flow tables and a group table, which are used for packet lookup and forwarding, and the Secure Channel used to interact with external controllers. The switch communicates with the controller using the OpenFlow protocol. The controller can add, update, and delete flow entries in flow tables, both reactively (in response to switch request packets) and proactively. Each flow table is composed of a set of flow entries. Each flow entry consists of the match fields, counters, and instructions.

Packet matching starts from the flow table with the minimum number. According to the controller orchestration, one flow table or a chain of flow tables can be used to complete a packet forwarding task. In a flow table, flow entries are also sequenced in descending order of priority, with the entry of the highest priority placed on the top. If a packet matches an entry,

the instructions associated with the entry are executed. If the packet matches no entry, the switch processes the packet depending on whether table-miss entry is configured. The switch may forward the packet to the controller using the OpenFlow protocol or to the next flow table, or drop the packet.

Instructions defined in the flow table are classified into action instructions containing specific actions to process packets and pipeline instructions to modify processing of packets. Action instructions describe packet forwarding, packet modification, and group table operations. Pipeline instructions send packets to subsequent tables for further processing and transfer packet processing meta data between tables. When an instruction associated with a matching flow entry does not specify a next table, table pipeline processing stops, and the packet matching the flow entry has been modified or forwarded.

Instructions in a flow entry may request packets to be forwarded to a physical port, logical port, or reserved port. A logical port usually refers to an Eth-Trunk, tunnel, or loopback port. A reserved port specifies a forwarding behavior, for example, sending the packet to the controller, flooding the packet to all physical ports, or processing the packet using non-OpenFlow methods.

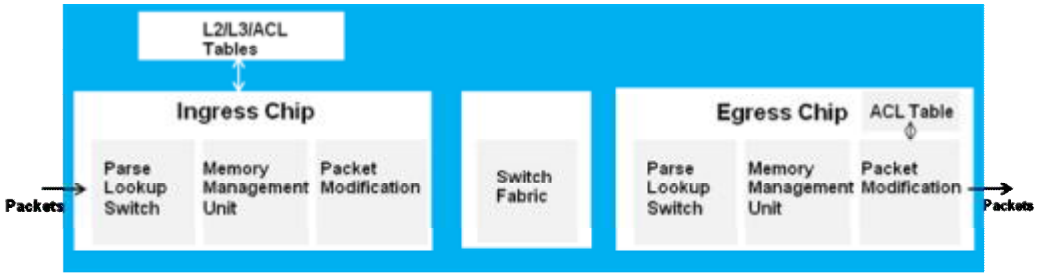
Packets may also be directed to a group table. A group table is usually used for multicast, broadcast, flooding, or other complex forwarding behaviors, for example, Equal Cost Multi-Path (ECMP), Fast Re-Route (FRR), and link aggregation. The group table can also implement output aggregation. For example, when multiple packets are forwarded to the same next hop, all associated flow entries direct to a public group table entry which will complete the next hop processing action. This aggregation changes the next hop without the need to change each flow entry.

1.4 Implementation and Characteristics

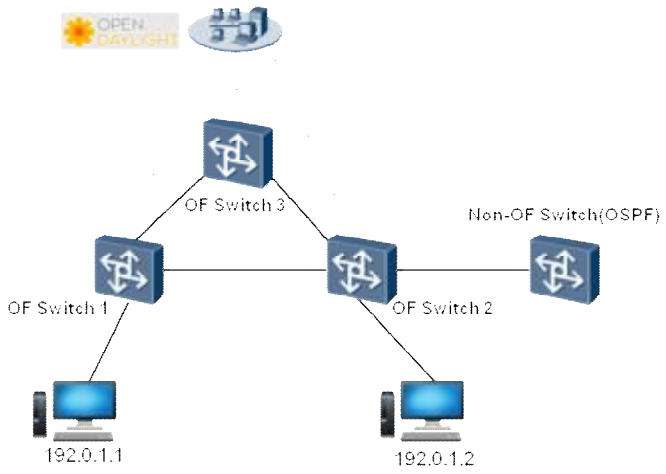
The following figure shows the packet forwarding process on a traditional switch. Packets are processed by the ingress chip, switch fabric, and then the egress chip. The ingress chip contains three units:

- Packet parsing unit (parse lookup switch): Analyzes data packet headers and searches for the corresponding Layer 2, Layer 3, or ACL entries to determine from which port to send the packets.
- Memory management unit: Caches the packets and implements priority scheduling on the packets.
- Packet modification unit: Modifies packets as required.

The switch fabric transmits packets from the ingress chip to the egress chip. The egress chip has the same units as the ingress chip.



In the OpenFlow architecture, the controller delivers a series of flow entries to the switch to determine the forwarding behaviors of the switch. The OpenFlow table structure is different from the table structure pre-defined on traditional switches. In the following figure, two PCs connect to an OpenFlow network and ping each other. Hydrogen developed by OpenDaylight is used as the controller.



PC:192.0.1.1

Switch	Table	Entry	Pri	MATCH	Instructions
OF Switch 1	0	23	0	all match	apply{acts=[out{port="ctrl", mlen="65535"
OF Switch 1	0	25	1	ipv4_dst=192.0.1.1, eth_type="0x800"	set_field{field:eth_dst=00:0d:88:4a:7d:cc}, out{port="11"}]}
OF Switch 1	0	26	1	ipv4_dst=192.0.1.2, eth_type="0x800"	apply{acts=[out{port="19"]}
OF Switch 2	0	28	0	all match	apply{acts=[out{port="ctrl" mlen="65535"
OF Switch 2	0	29	1	ipv4_dst 192.0.1.1, eth_type="0x800"	apply{acts=[out{port="19"]};
OF Switch 2	0	30	1	ipv4_dst 192.0.1.2, eth_type="0x800"	apply{acts [set_field{field:eth_dst 00:1b:21:a3:77 /e}, out{port="11"}];
OF Switch 3	0	18	0	all match	apply{acts=[out{port="ctrl" mlen="65535"
OF Switch 3	0	19	1	ipv4_dst=192.0.1.1, eth_type="0x800"	apply{acts=[out{port="15"]}
OF Switch 3	0	20	1	ipv4_dst=192.0.1.2, eth_type="0x800"	apply{acts=[out{port="15"]};

PC:192.0.1.2

The OpenFlow logic entries must be mapped to the forwarding entries supported by switches. Huawei designed the Data Path Abstract Layer (DPAL) to do the mapping.

DPAL uses service paths to abstract switch capabilities and decouples the OpenFlow protocol layer and hardware layer. Essentially, forwarding rules of both OpenFlow and ASIC switches can be abstracted into match fields and action sets. DPAL analyzes and aggregates the flow entries delivered by the controller, abstracts match fields and action sets from the entries, and matches the abstracted items against the service paths. After that, DPAL completes packet forwarding based on ASIC entries of the switches. The DPAL-based OpenFlow solution has the following advantages:

Powerful compatibility: OpenFlow standards are immature, and flow tables supported by vendors differ greatly. To ensure interoperability with various controllers, DPAL supports all match fields and action sets of multi-level flow tables, and can interoperate with controllers of various vendors, such as, OpenDaylight, RYU, and Floodlight.

Protocol consistency: OFTest is a test framework and suite used to verify OpenFlow protocol consistency. Based on OFTest1.2, Huawei developed OFTest1.3, which contains 419 test

cases. All OpenFlow switches have passed the test cases, covering IPv4, IPv6, MPLS, multicast, and QoS.

Extraordinary forwarding performance: By making full use of the forwarding capability of switch ASIC, DPAL provides flexible SDN services and also inherits high forwarding performance of traditional switches.

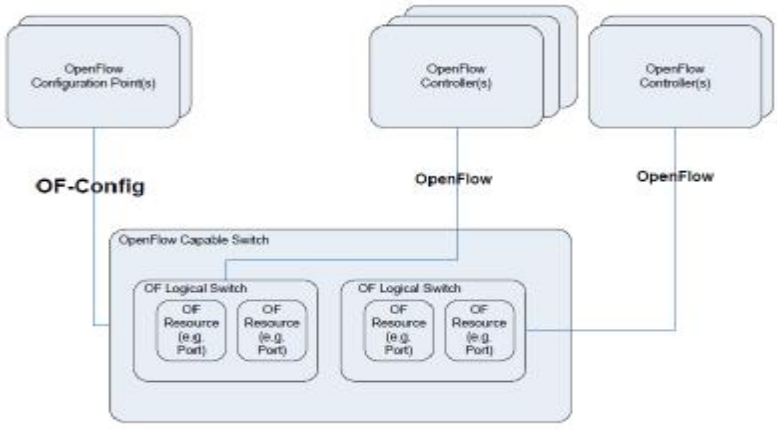
1.5 Products and Specifications

Item	Description
OpenFlow specifications	Supports multiple controllers.
	Supports Multi-level flow tables.
	Supports group tables.
	Supports meter tables.
	Complies with Open Flow 1.3 standards.
Supported products	S5720 series switches (including S5720-SI, S5720-EI and S5720-HI series)
	S6720-EI series switches
	S7700 series switches
	S9700 series switches
	S12700 series agile switches

1.6 Roadmap and Vision

To adapt to the SDN development trend and improve OpenFlow features, Huawei plans to implement the following features on switches for large-scale commercial use of OpenFlow.

OF-Config protocol:



As shown in the figure, an OpenFlow-capable switch can be virtualized into multiple OpenFlow logical switches. To enable the logical switches and controllers to work properly, the OpenFlow Configuration Points module must perform the following configurations on the switches through the OF-Config protocol:

- Configure one or more OpenFlow controller addresses.

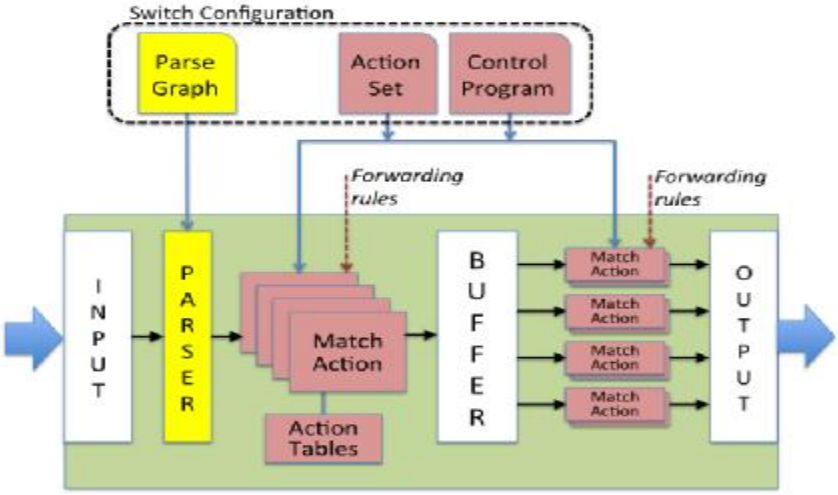
  In the actual deployment, a hybrid mode may be used, namely, the switches support both traditional forwarding mode (such as Layer 2/Layer 3/ACL forwarding) and OpenFlow forwarding. In this situation, the resources on the physical switches, such as the ports and queues must be properly configured.
- Enable or disable the logical OpenFlow ports according to service needs.
- Configure security certificates for the OpenFlow switch and controller to ensure their secure communications.
- Configure tunnels, such as IP-in-GRE, NV-GRE, and VxLAN for the OpenFlow switch according to service needs.

NDM&TTP

As defined by OpenFlow 1.3, OpenFlow switches need to support 39 match fields, including the ingress port, source MAC address, destination MAC address, source IP address, destination IP address, IP DSCP field, and protocol number. Theoretically, the SDN controller can deliver flow entries that contain combinations of any match fields. However, the OpenFlow switches, especially the ASIC switches have limited capabilities and may not communicate properly with the SDN controller. To address this problem, ONF is developing Negotiable Datapath Models (NDMs), which is called Table Type Patterns (TTPs) in OpenFlow1.x.

OpenFlow2.0&POF

The following figure shows the abstract switch model in OpenFlow2.0, with enhanced Parse Graph (highlighted in yellow).



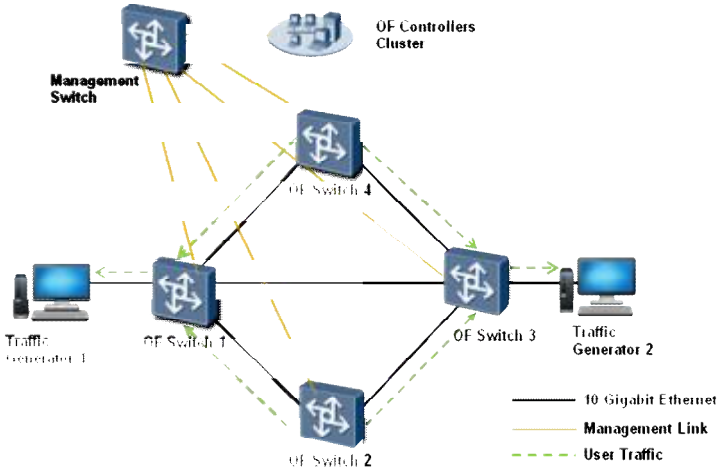
The switch needs to parse protocol packets. The protocol parser of traditional ASIC or OpenFlow1.x switches can only resolve known protocols. When the switch receives a packet of a new protocol, for example, an IPv8 packet, the protocol parser cannot parse the packet and discards the packet. OpenFlow2.0 defines a programmable protocol parser, which allows

switches to send unknown packets to the controller for analysis. The controller programs the protocol parser based on the analysis results, so that the switch can process similar packets.

Huawei further extends OpenFlow and develops Protocol Oblivious Forwarding (POF), which is based on the idea of programmable protocol parsing. POF is widely accepted in the industry and expected to be integrated into OpenFlow2.0.

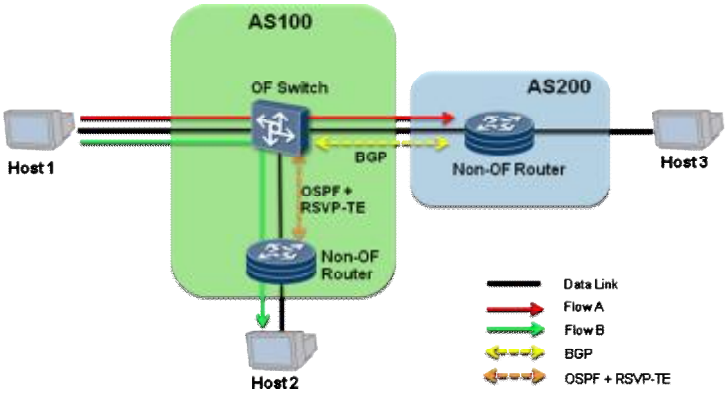
1.7 OpenFlow Applications

- **Smart TE, automatic path calibration**



As shown in the figure, all switches communicate with the OpenFlow controller through a management switch. Four OpenFlow switches constitute a ring network through which two traffic generators send data to each other. The controller displays the global network topology. When a link is overloaded, the OpenFlow controller switches traffic to a lightly loaded link.

- **Seamless interconnection with traditional IP networks**



The complete transition from traditional networks to SDNs requires a fairly long period of time, so the SDN and traditional IP networks must be interoperable. In the figure, an OpenFlow switch communicates with two non-OpenFlow routers through BGP, OSPF, and RSVP-TE. All protocol packets are processed by the controller.

- **Programmable and customizable network**

  Separation of the control plane and data plane on the SDN enables enterprises to easily customize and change their networks in response to specific needs of applications. For example, network traffic can be dynamically distributed to financial service applications according to specific standards.

For more information, visit http://e.huawei.com/en or contact your local Huawei sales office.