Next generation networks

Salvatore Andaloro

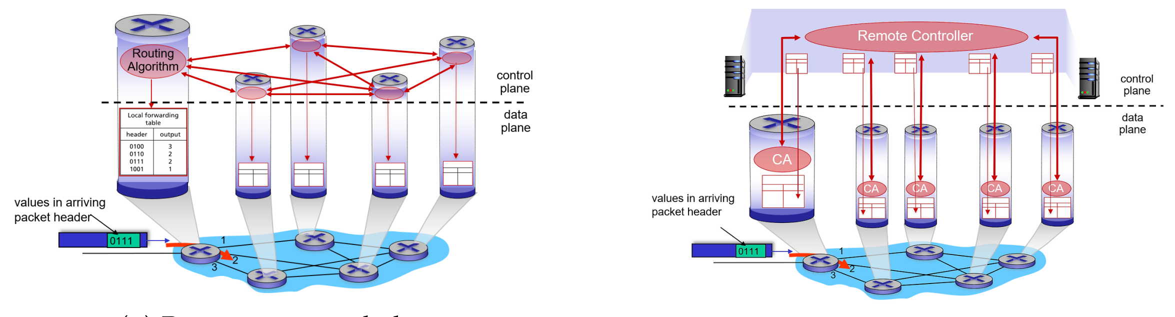
October 12, 2024

Contents

Chapter 1

Software defined networking

The network layer manages moving packets from the input to the output port of the router (forwarding - data plane) and determining the route taken by a packet from its source to its destination (routing - control plane). There are two approaches to structuring the network control plane: a decentralized approach and a centralized approach. In the decentralized structure, each router routes packets according to its own logic and runs a proprietary OS and implementation of the Internet protocols. In the centralized structure, a remote controller centrally manages all routers in the network by modifying their routing tables. This is called Software defined networking (SDN).



(a) Per-router control plane (b) SDN control plane

Figure 1.1: Control plane approaches

The advantages of SDN are the avoidance router misconfigurations, better management of traffic flow due to having full knowledge and control of the whole network and the availability of many choices for the routing software, including open-source solutions, fostering raping innovation.

SDN networking can be divided into three components:

- Data-plane switches: fast and simple switches implementing data plane forwarding in hardware and supporting the reprogramming of the routing ables using a standard API (ex. OpenFlow)
- SDN controller: maintains network state information (state of switches, network links, services) and interacts with network applications and network switches. Usually implemented as a distributed system for performance, scalability, fault tolerance, robustness (ex. OpenDaylight, ONOS)

- Network-control apps: standalone programs which implement control functions by interacting with the API provided by the SDN controller

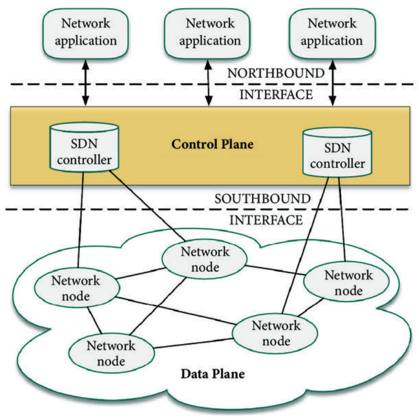


Figure 1.2: SDN components

1.1 OpenFlow protocol

OpenFlow is a communications protocol that gives access to the forwarding plane of a network switch or router over the network. OpenFlow relies on TCP (with optional encryption). OpenFlow supports three classes of messages, where each class provides a specific set of actions:

- Controller-to-switch messages:
  - features: controller queries switch features, switch replies
  - configure: controller sets switch configuration parameters
  - modify-state: add, delete, modify flow entries in the OpenFlow tables
  - packet-out: controller can send this packet out of specific switch port
- switch-to-controller (asynchronous):
  - packet-in: transfer packet to controller
  - flow-removed: inform controller that a flow table entry has been deleted
  - port status: inform controller of a change on a port

1.2 SDN example

The following is an example of a SDN control/data plane interaction:

1. S1, experiencing link failure uses OpenFlow port status message to notify controller
2. SDN controller receives OpenFlow message and updates link status info
3. Dijkstra's routing algorithm application, which has previously registered to be called when ever link status changes, is called
4. Dijkstra's routing algorithm access network graph and link state info stored by the controller and computes new routes
5. Link state routing application interacts with flow-table-computation component in SDN controller, which computes new flow tables

6. Controller uses OpenFlow to install new tables in switches that need updating

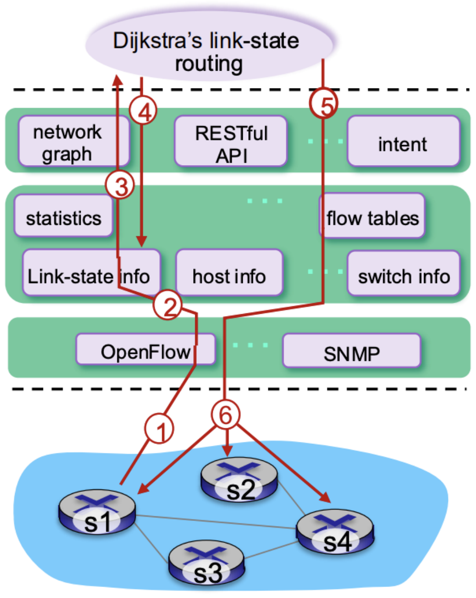


Figure 1.3: SDN example

1.3 Network management

Network management includes the deployment of the hardware, software, and human elements to control the network and satisfy performance and quality of service requirements at a reasonable cost.

Network management is made of multiple components: a *managing* server which interacts with *managed devices* (or agents) (equipment with configurable hardware and software components) through a *network management protocol* to exchange data and for configuration.

The protocol between the server and the managed devices can be implemented in different ways: using a CLI over an SSH interface (ex. bash scripts), SNMP/MIB or NETCONF/YANG.

1.3.1 SNMP

SNMP (Simple Network Management Protocol) is a protocol for organizing and modifying the configuration about managed devices. SNMP supports two types of communications: get/set request from the controller, followed by a response by the agent or trap messages, used by the agent to inform the controller of some event.

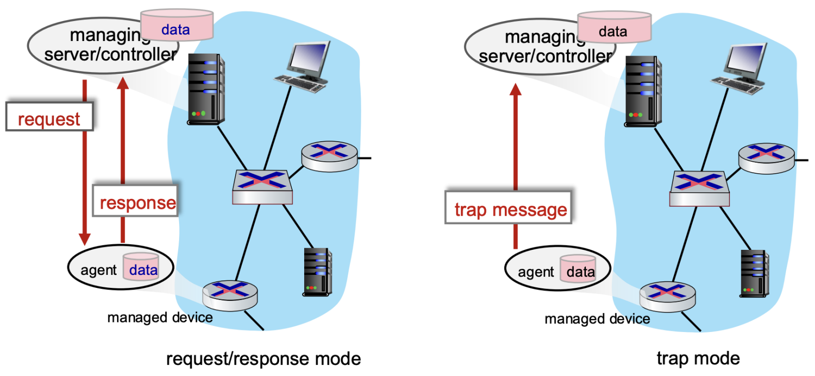
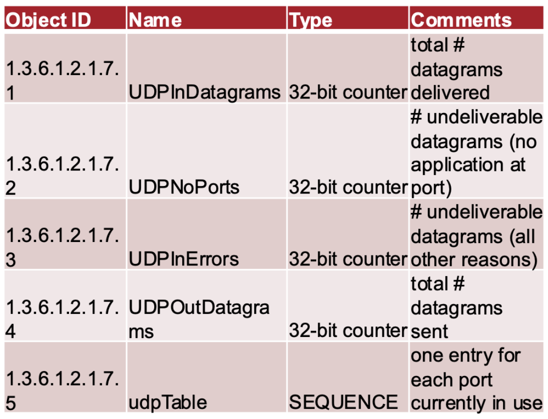


Figure 1.4: SNMP protocol

Data exchanged using the SNMP protocol is organized in variables. Related variables are grouped in management information base (MIB) modules. There are 400 MIB modules defined in RFCs and many more vendor-specific MIBs.



Object ID	Name	Type	Comments
1.3.6.1.2.1.7.1	UDPInDatagrams	32-bit counter	total # datagrams delivered
1.3.6.1.2.1.7.2	UDPNoPorts	32-bit counter	# undeliverable datagrams (no application at port)
1.3.6.1.2.1.7.3	UDPInErrors	32-bit counter	# undeliverable datagrams (all other reasons)
1.3.6.1.2.1.7.4	UDPOutDatagrams	32-bit counter	total # datagrams sent
1.3.6.1.2.1.7.5	udpTable	SEQUENCE	one entry for each port currently in use

Figure 1.5: MIB variables for the UDP protocol

1.3.2 NETCONF

The Network Configuration Protocol (NETCONF) is another network management protocol which provides mechanisms to install, manipulate, and delete the configuration of network devices. Its operations are realized on top of a simple Remote Procedure Call (RPC) layer. The NETCONF protocol uses XML data encoding for the configuration data as well as the protocol messages. Messages are exchanged over secure and reliable transport protocol (ex. TLS).

XML data exchanged over NETCONF can be generated from another data modeling language called YANG.

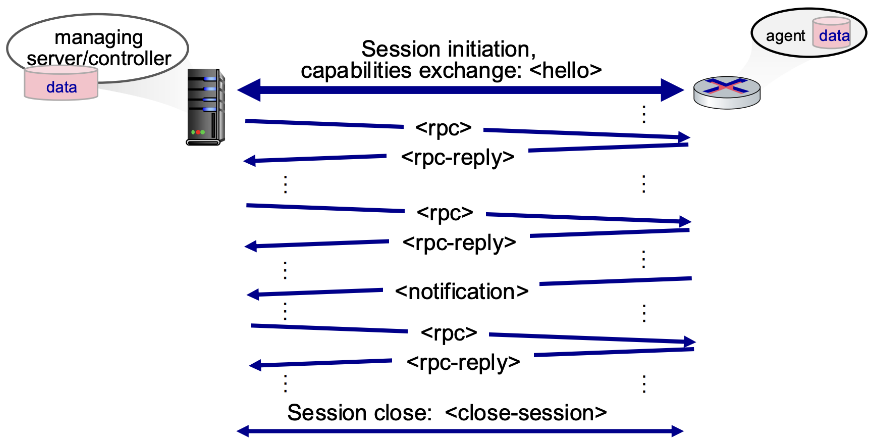

managing server/controller
data
agent
data
Session initiation, capabilities exchange: <hello>
<rpc>
<rpc-reply>
<rpc>
<rpc-reply>
<notification>
<rpc>
<rpc-reply>
Session close: <close-session>

Figure 1.6: NETCONF exchange

NETCONF	Operation Description
<get-config>	Retrieve all or part of a given configuration. A device may have multiple configurations.
<get>	Retrieve all or part of both configuration state and operational state data.
<edit-config>	Change specified (possibly running) configuration at managed device. Managed device <rpc-reply> contains <ok> or <rpcerror> with rollback.
<lock>, <unlock>	Lock (unlock) configuration datastore at managed device (to lock out NETCONF, SNMP, or CLIs commands from other sources).
<create-subscription>, <notification>	Enable event notification subscription from managed device

Figure 1.7: NETCONF operations

Chapter 2

Network virtualization

Network virtualization refers to the ability of decoupling the physical infrastructure and topology from a logical topology or infrastructure, typically by creating overlay networks.

2.1 VLAN

Virtual Local Area Networks (VLAN) are used to divide a physical network into several broadcast domains. VLAN is implemented at layer 2 (data link layer), therefore the devices responsible for VLANs are switches.

There are two ways of implementing VLANs: port-based or using tagged packets. In port-based VLANs a switch is configured to assign specific ports to specific VLANs. In VLANs using tagged packets, special fields are used by the host to define to which VLAN the frame belongs to. The following fields need to be set (part of the IEEE 802.1Q Frame):

- TPI (Tag Protocol Identifier): two bytes with fixed value (81 00) to identify frame as 802.1Q
- TCI (Tag Control Information): two VLAN tag bytes, first three bits specify frame priority, the fourth bit (CFI) is 1 for frames belonging to token ring LANs and the remaining 12 bits (VID) specify the VLAN tag (0 to 4095)

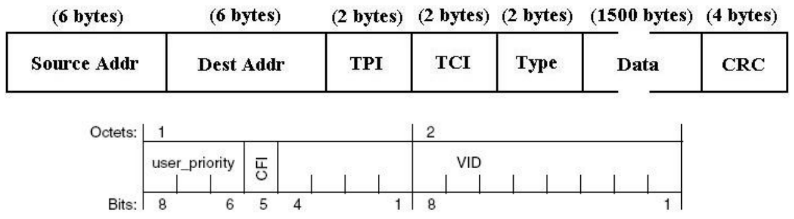


Figure 2.1: VLAN frame

2.2 VPN

A Virtual Private Network is a type of private network that uses a public network, such as the Internet to communicate. VPNs must implement four services: authentication (validating that the data was sent from the sender), access control (limiting unauthorized users from accessing the network), confidentiality (preventing the data to be read or copied as the data is being transported), data integrity (ensuring that the data has not been altered during transport). Data confidentiality is ensured by using encryption, based on the public key encryption technique. The data to be transported by the VPN is encapsulated into encrypted datagrams, which are then encapsulated into the outer datagram, visible to the public network. Datagrams are transported to the destination over a virtual point-to-point connection made through a public network called tunnel.

There are multiple VPN protocols:

- PPTP - Point-to-Point Tunneling Protocol
- L2TP - Layer 2 Tunneling Protocol
- IPsec - Internet Protocol Security
- WireGuard - carries traffic over UDP
- SOCKS

VPNs can securely connect single clients to a remote network over a public network or bridge multiple networks together (site-to-site VPN).

VPNs can be implemented in hardware inside a router (easy to use and deploy, high throughput, lack of flexibility and higher cost), on the firewall or in software (flexible, low cost, less efficient, more complicated to deploy).

2.3 VXLAN

Virtual eXtensible LAN (VXLAN) is a network virtualization technology which encapsulates OSI layer 2 Ethernet frames within layer 4 UDP datagrams, with an added VXLAN header. Compared to single-tagged IEEE 802.1Q VLANs which provide a limited number of layer-2 VLANs (4094, using a 12-bit VLAN ID), VXLAN increases scalability up to about 16 million logical networks (using a 24-bit VNID) and allows for layer-2 adjacency across IP networks.

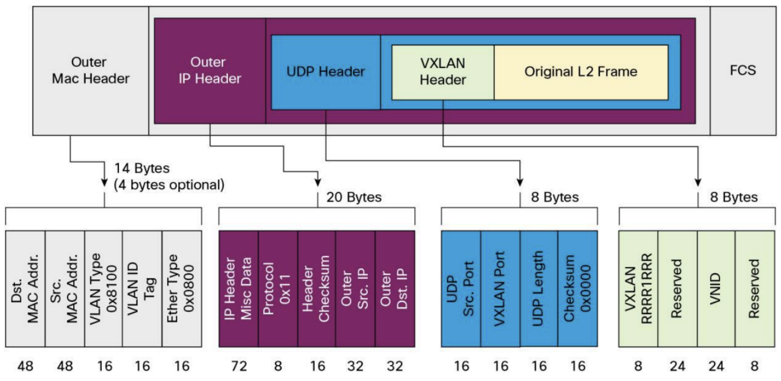


Figure 2.2: L2 frame encapsulated inside a UDP packet

VXLANs use devices called VTEP (VXLAN Tunnel Endpoint) that encapsulate and decapsulate the packets and map the devices connected to it to different VXLANs. Each VTEP has therefore two interfaces: one is to a switch connected to the devices in the

LAN, while the other is an IP interface towards the IP transport network. The IP network between the VTEPs is independent from the VXLAN network, because it simply routes encapsulated packets on the basis of the IP address in the external header, which has the starting VTEP as source IP and terminating VTEP as destination IP.

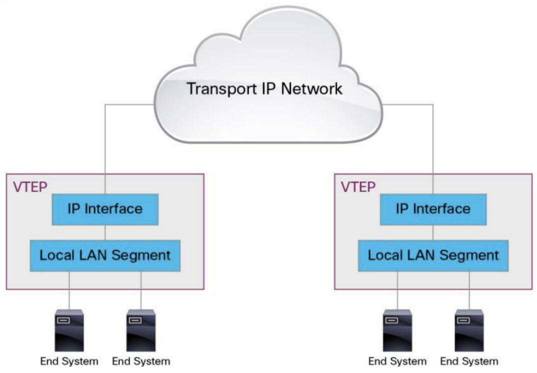


Figure 2.3: VXLAN topology

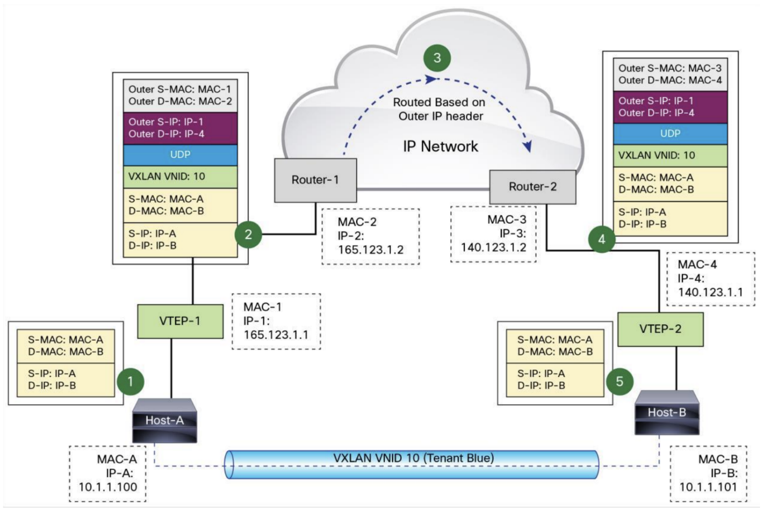


Figure 2.4: Packet travelling through a VXLAN network

Chapter 3

Software defined radios