UCF / FLR Workshop on Networking Topics
SDN concepts, Open vSwitch and Open Flow

Jorge Crichigno, Elie Kfoury
University of South Carolina
http://ce.sc.edu/cyberinfra

University of Central Florida (UCF)
Florida LambdaRail (FLR)
The Engagement and Performance Operations Center (EPOC)
Energy Sciences Network (ESnet)
University of South Carolina (USC)

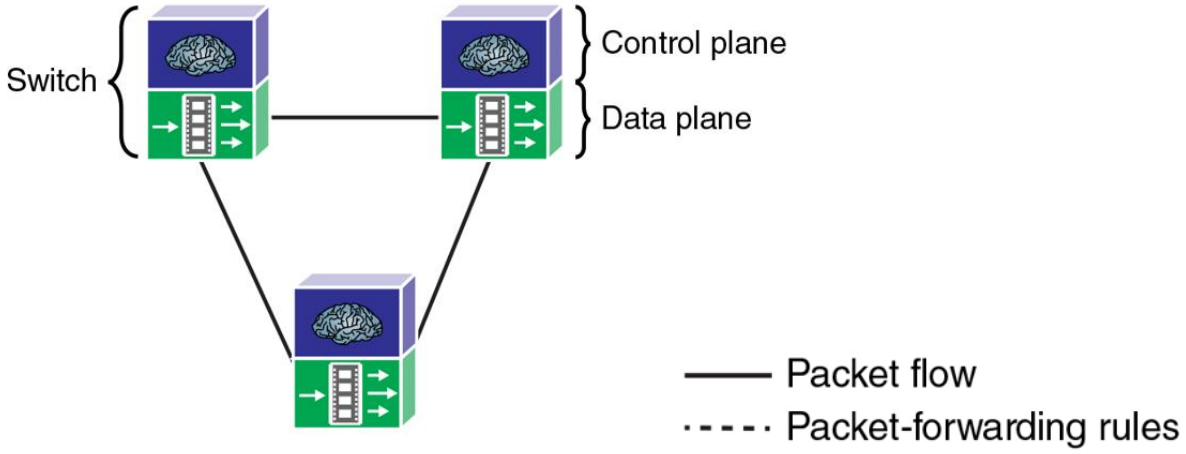
Orlando, Florida
February 17th, 2023

Agenda

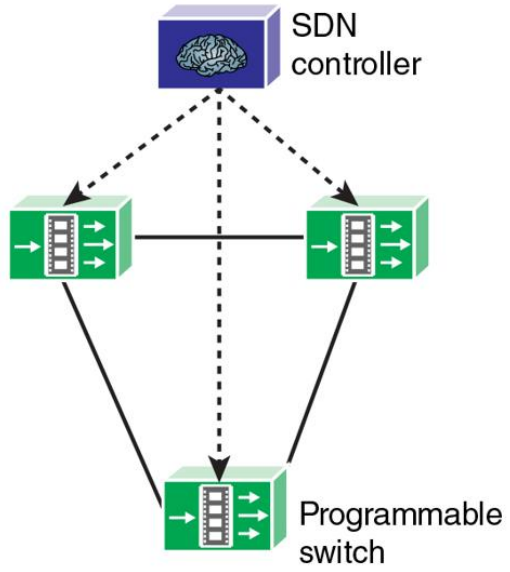
- SDN fundamentals – data and control planes
- SDN operation
- Flow / Forwarding tables
- OpenFlow protocol
- Limitations of SDN and OpenFlow

Plane Separation

- The first fundamental characteristic of SDN is the separation of planes
  - Data plane, implemented in the device
  - Control plane, implemented by a centralized controller



Traditional network

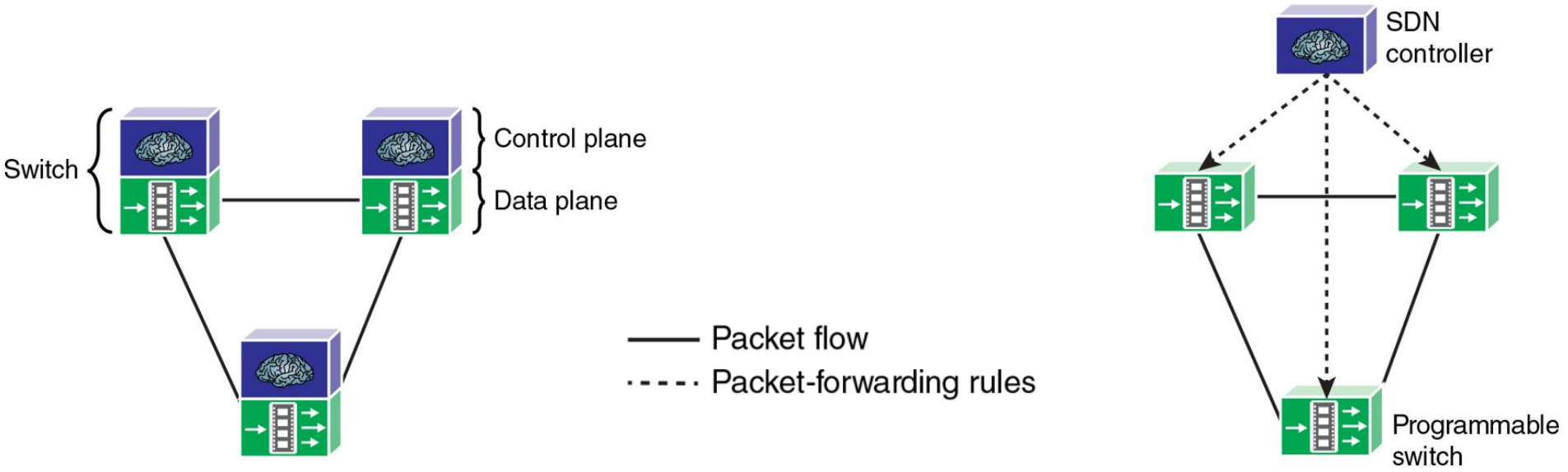


SDN network

W. Stallings, “Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud” Addison Wesley, 2017.

Plane Separation – Data Plane

- The data plane implements forwarding functionality (logic and tables for choosing how to deal with incoming packets)
  - Forwarding based on MAC address, IP address, VLAN ID, etc.
  - Dropping, replicating an incoming packet

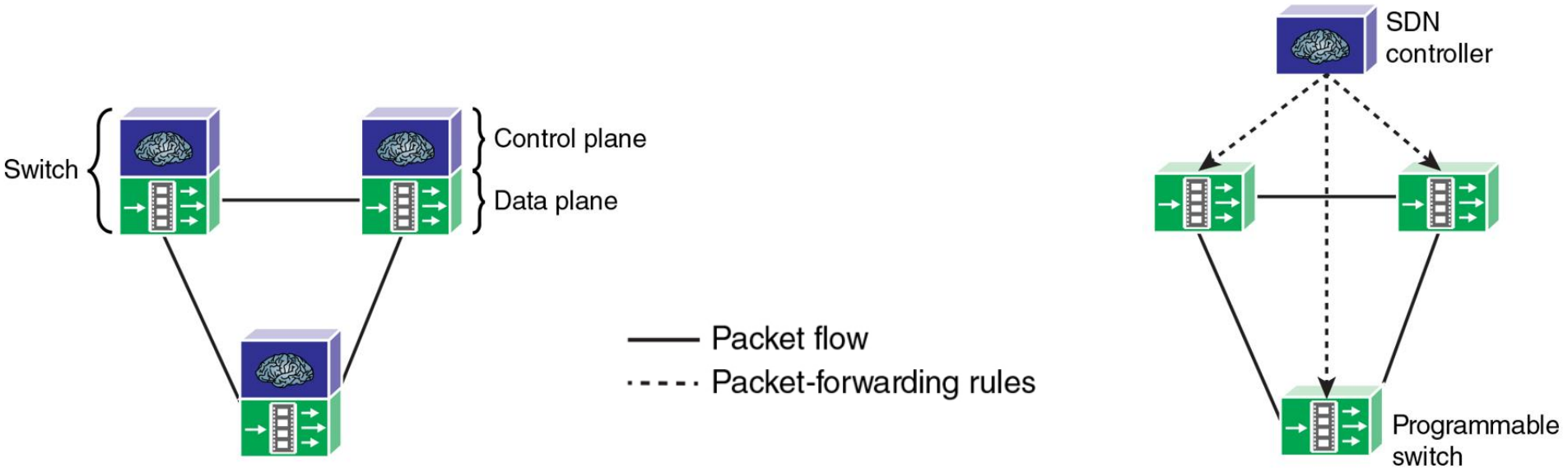


Traditional networks

W. Stallings, “Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud” Addison Wesley, 2017.

Plane Separation – Data Plane

- Special-case packets (e.g., routing advertisements) that require processing by the control plane are passed to that plane

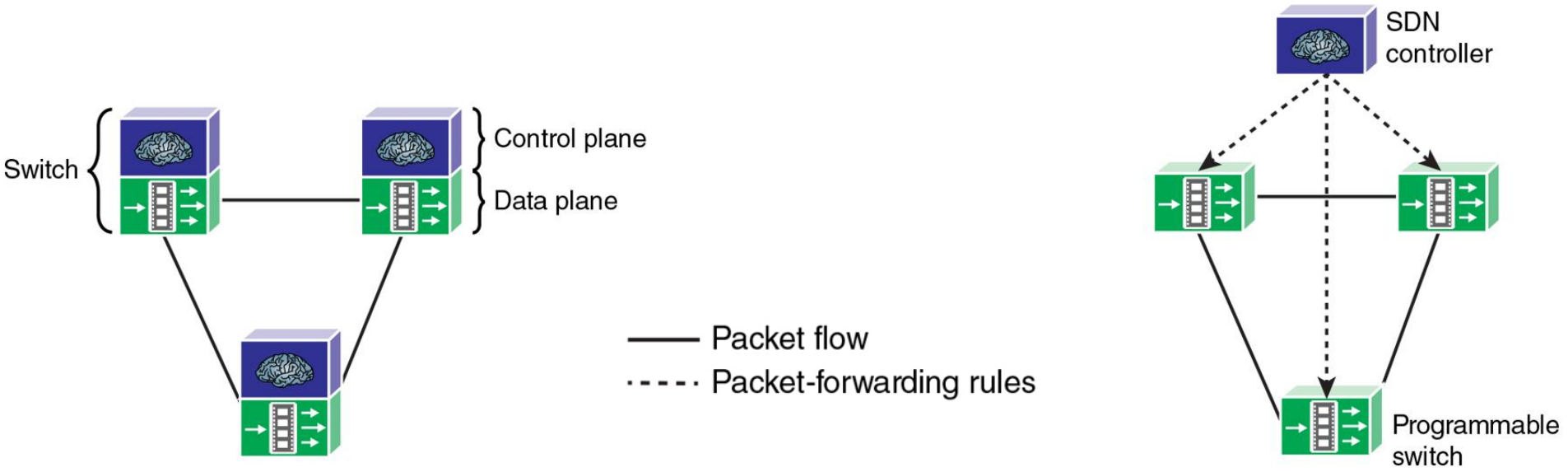


Traditional networks

W. Stallings, “Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud” Addison Wesley, 2017.

Plane Separation – Control Plane

- The control plane is moved off the switching device, onto a centralized controller
- The algorithms used to program the data plane (populate tables) reside in the control plane (e.g., OSPF, BGP)

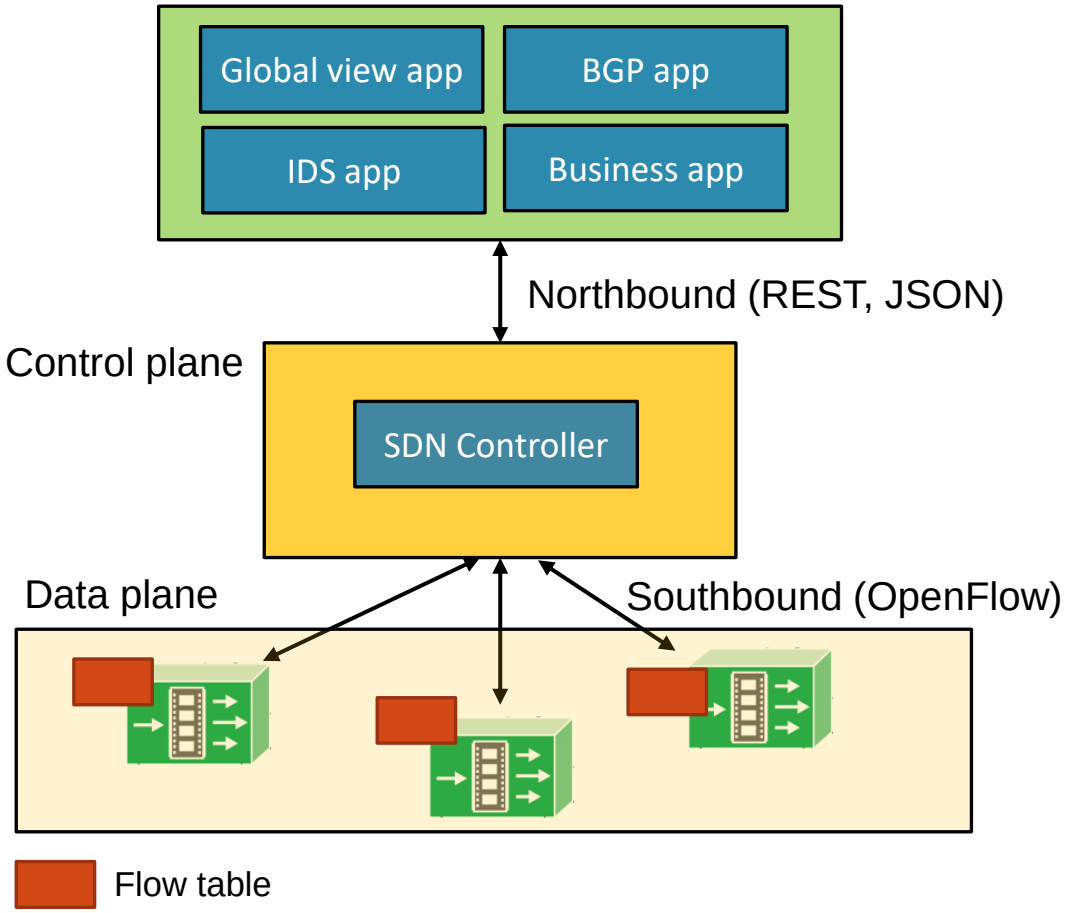


Traditional networks

W. Stallings, “Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud” Addison Wesley, 2017.

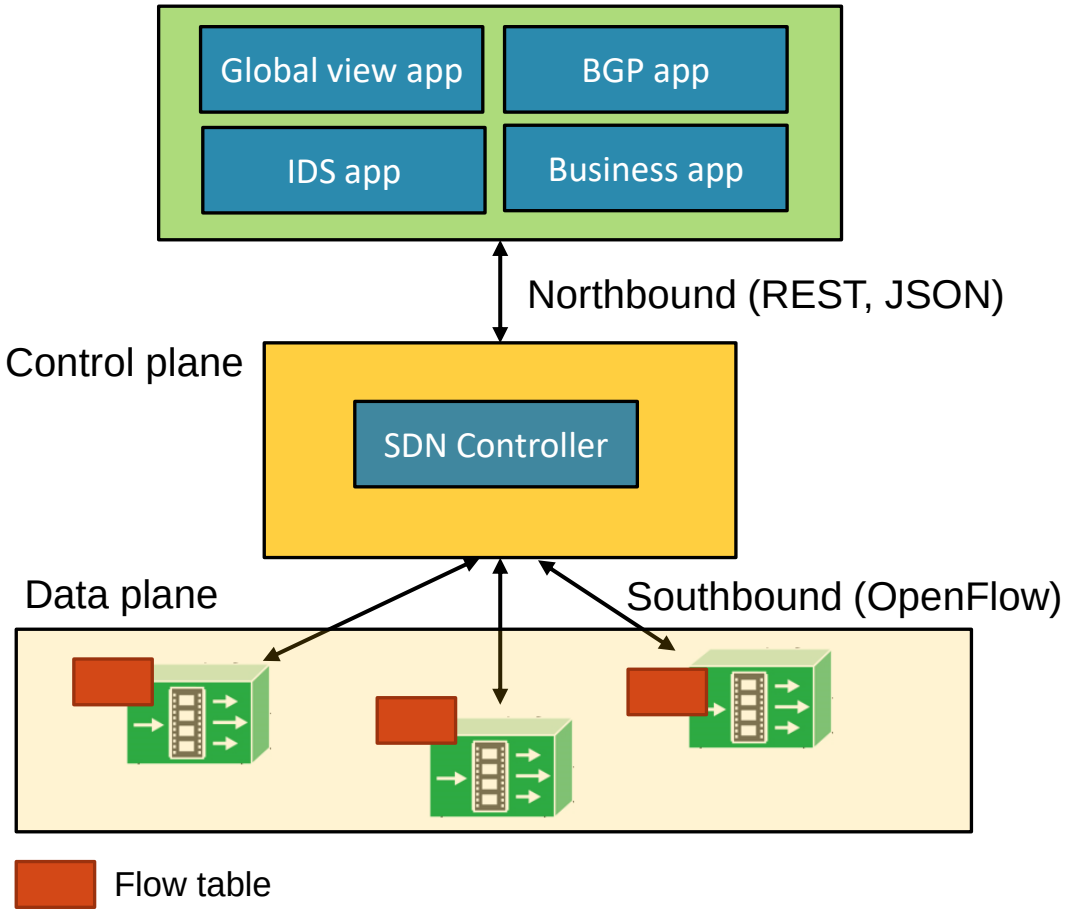
SDN Operation

- Basic components (bottom-up)
  - SDN switches (e.g., Open vSwitch)
  - Controller (e.g., ONOS controller)
  - Applications (e.g., BGP app, IDS app)



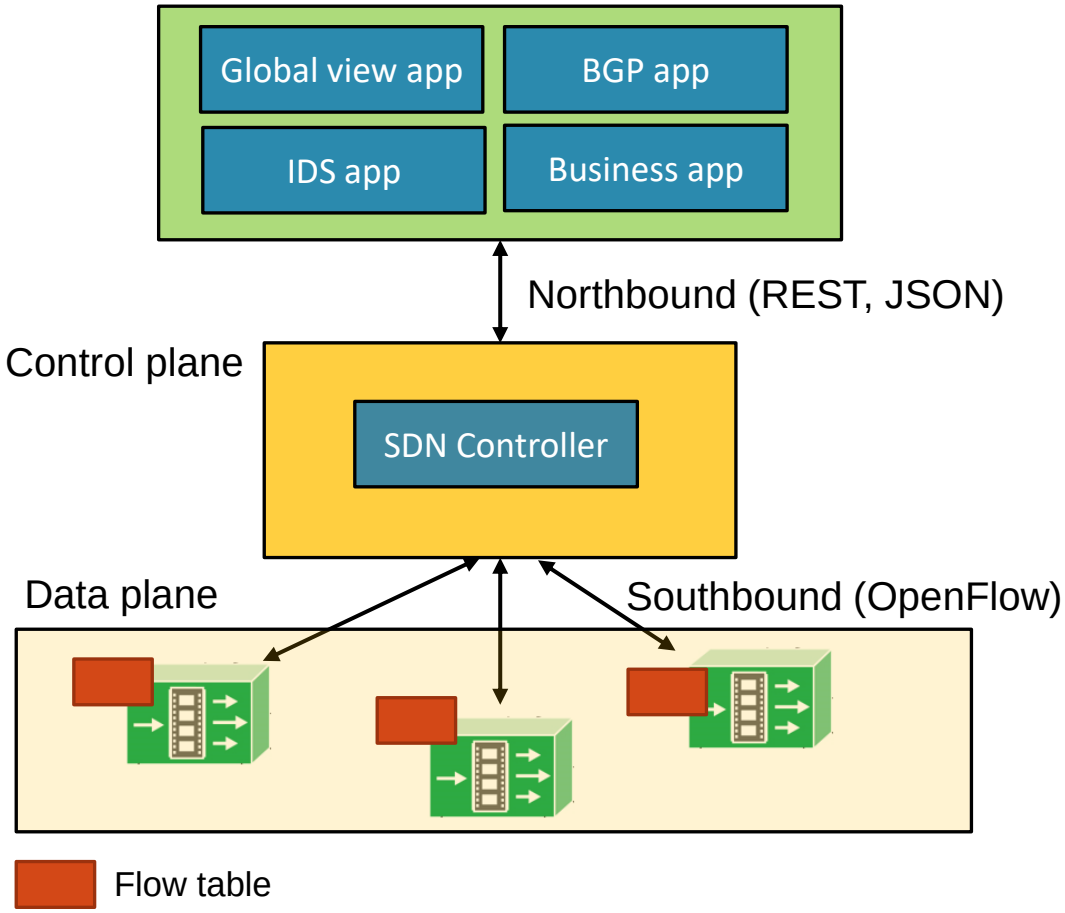
SDN Operation – Switches

- SDN devices contain forwarding functionality
- Forwarding information is stored in flow tables
- If it does not find a match, it can either drop the packet or pass it to the controller



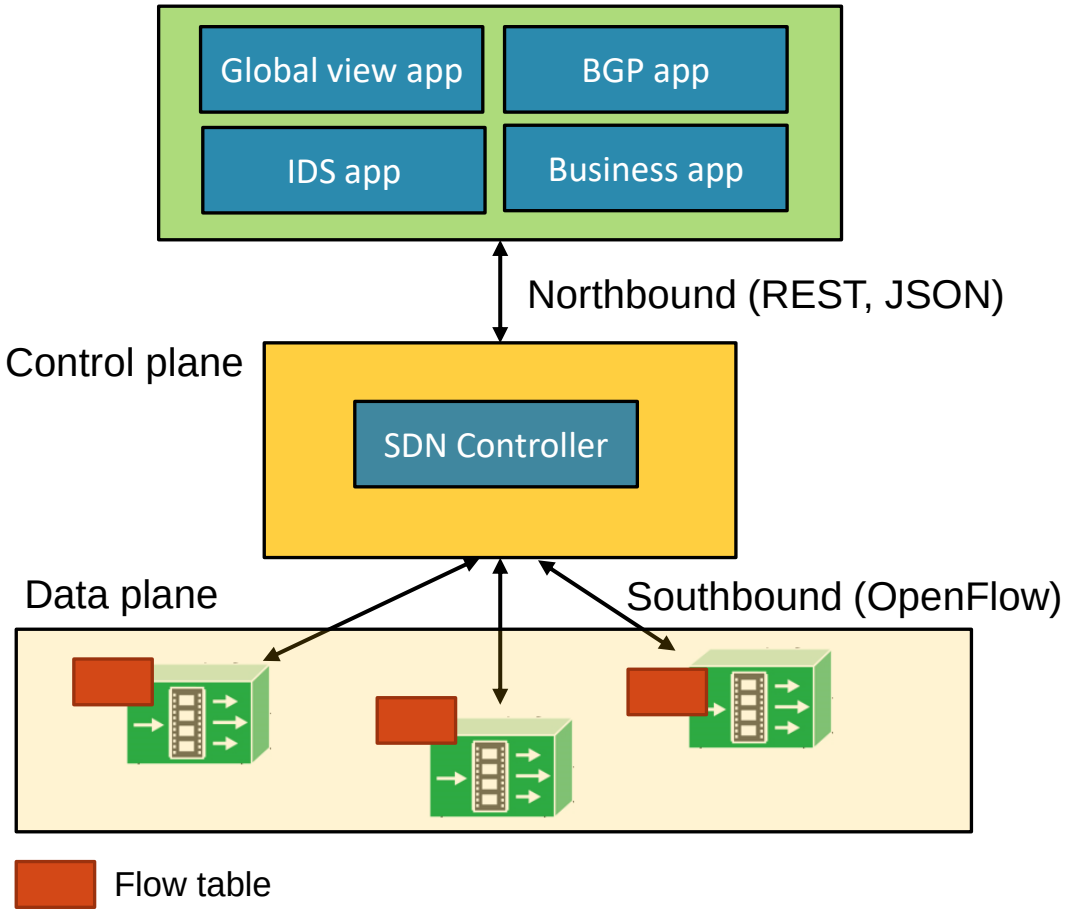
SDN Operation – Controller

- SDN controller implements control plane functionality
- It presents an abstraction of the network to the SDN applications running above
- It allows the SDN applications to define flows on devices
- It maintains a view of the entire network (global network view)



SDN Operation – Applications

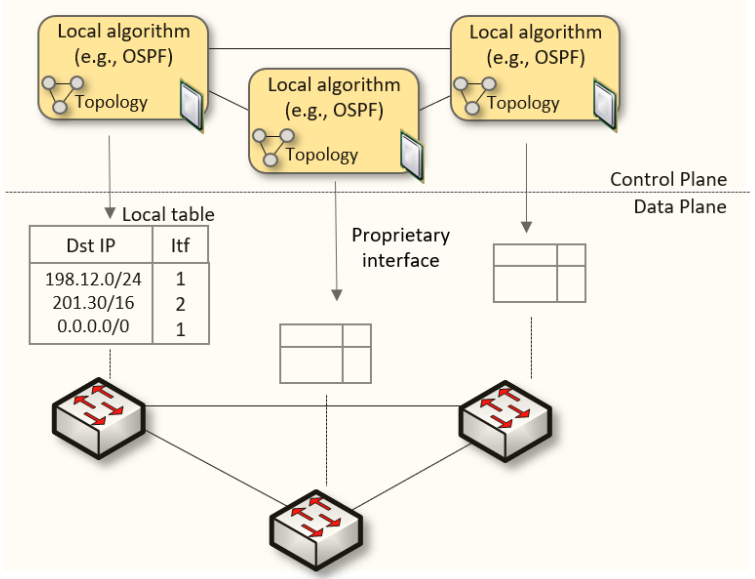
- SDN applications are built on top of the controller
- Software applications can implement forwarding, routing, overlay, multipath, access control, etc.
- Application are driven by events coming from the controller and from external inputs



Forwarding Tables

- The interface between the control and data planes has been historically proprietary
- Vendor dependence: slow product cycles of vendor equipment, standardization

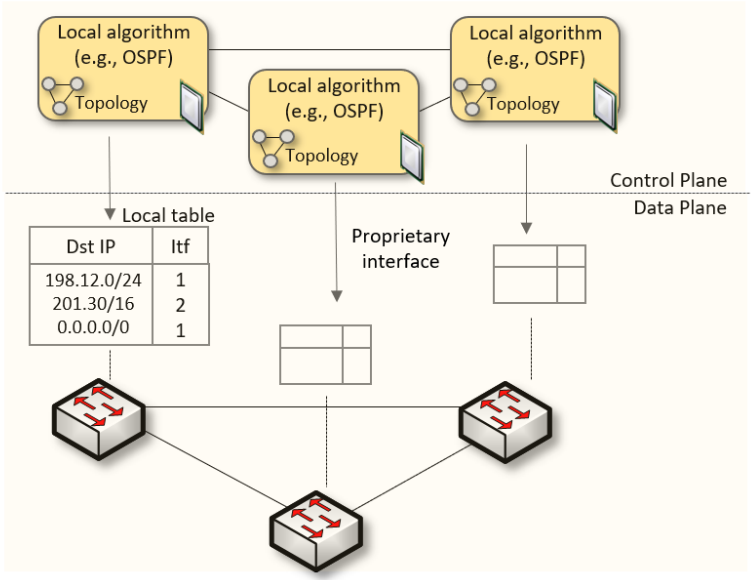
Legacy network



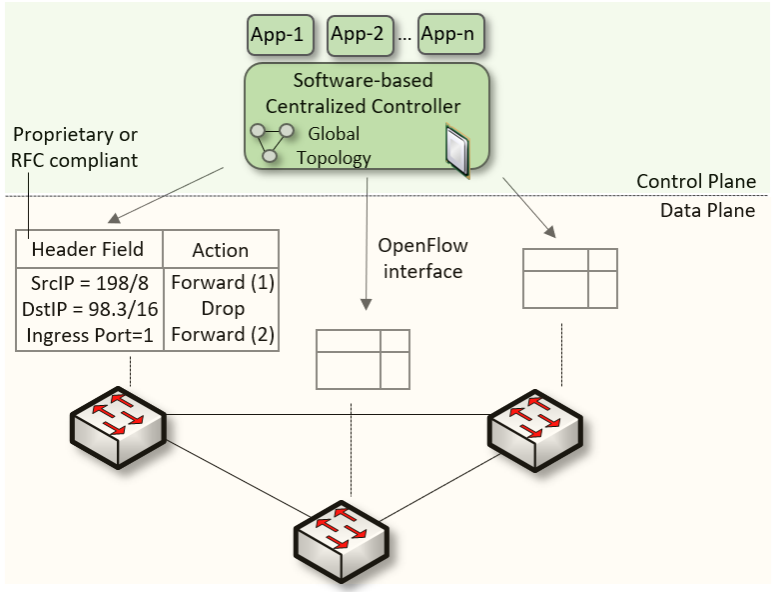
Forwarding Tables

- In SDN networks, that function is now performed by the controller
  - The controller is responsible for programming packet-matching and forwarding rules

Legacy network



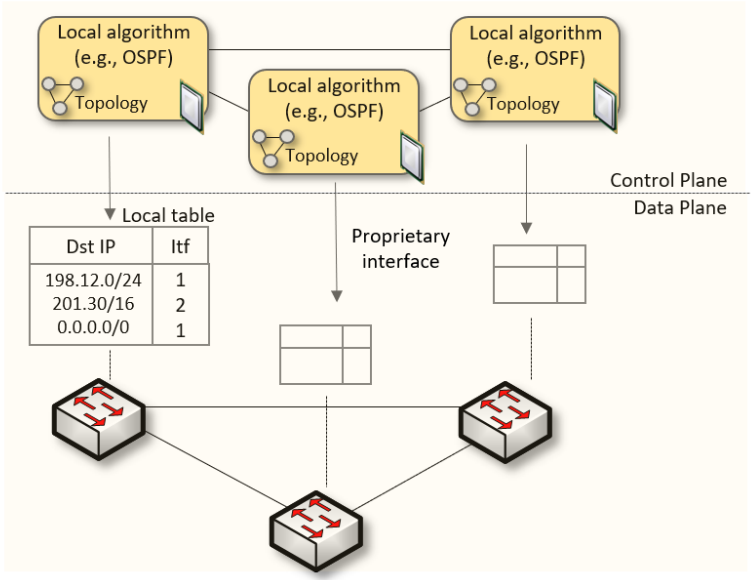
SDN network



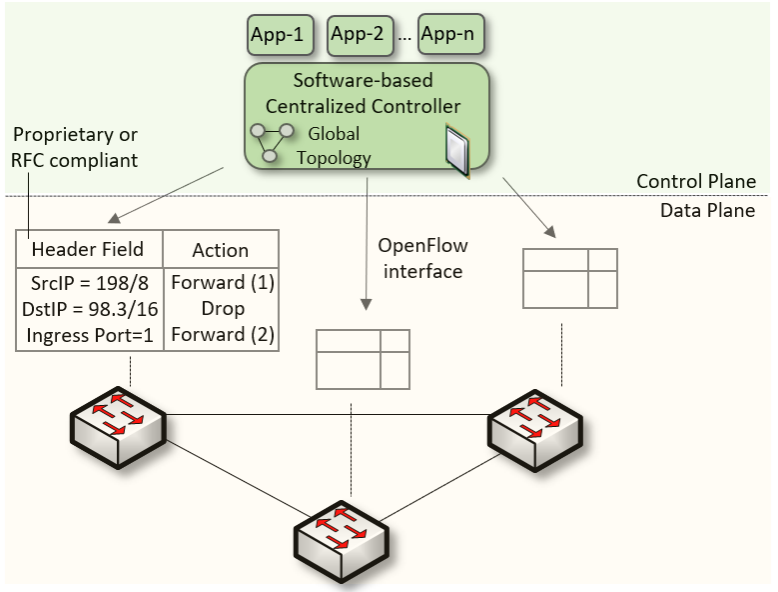
Advantages of SDN Networks vs. Legacy Networks

- Ease of network management
- “More customized” network behavior
- Possibility of experimentation and innovation (custom policies, apps can be deployed)
- Packets can be forwarded based on other fields, such as TCP port number

Legacy network



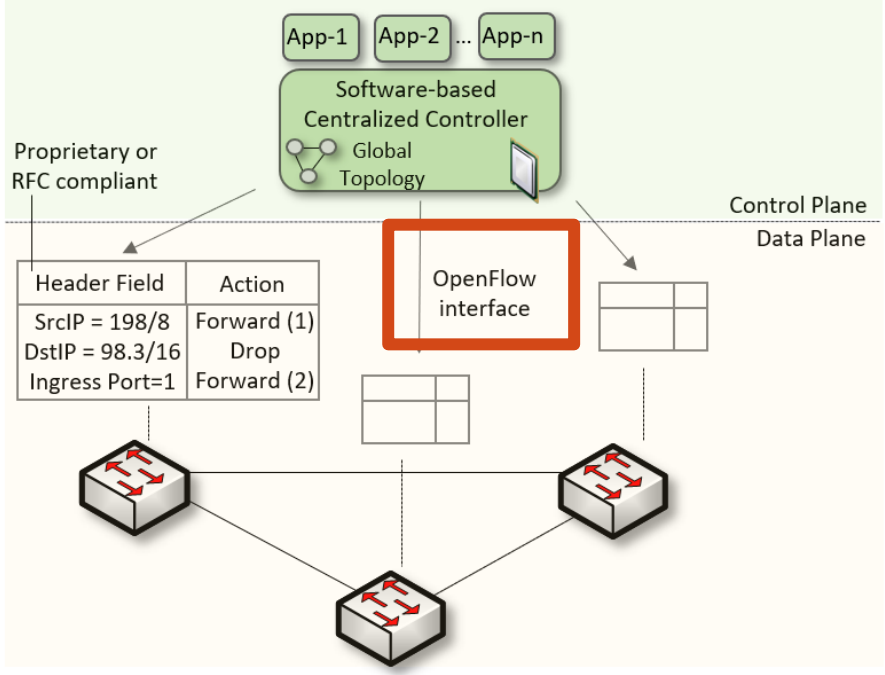
SDN network



Overview of OpenFlow

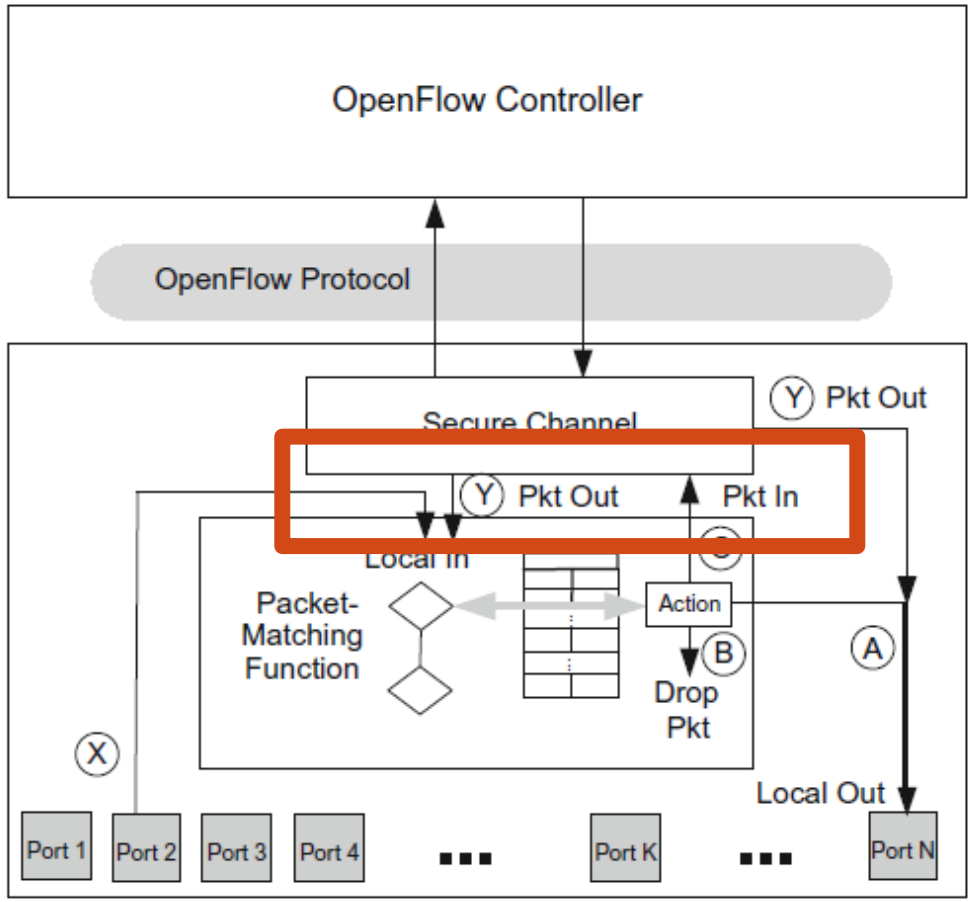
- OpenFlow is a protocol specification that describes the communication between OpenFlow switches and an OpenFlow controller
- The consortium responsible for the OpenFlow specification is the Open Networking Foundation (ONF), which was created in 2011

SDN network



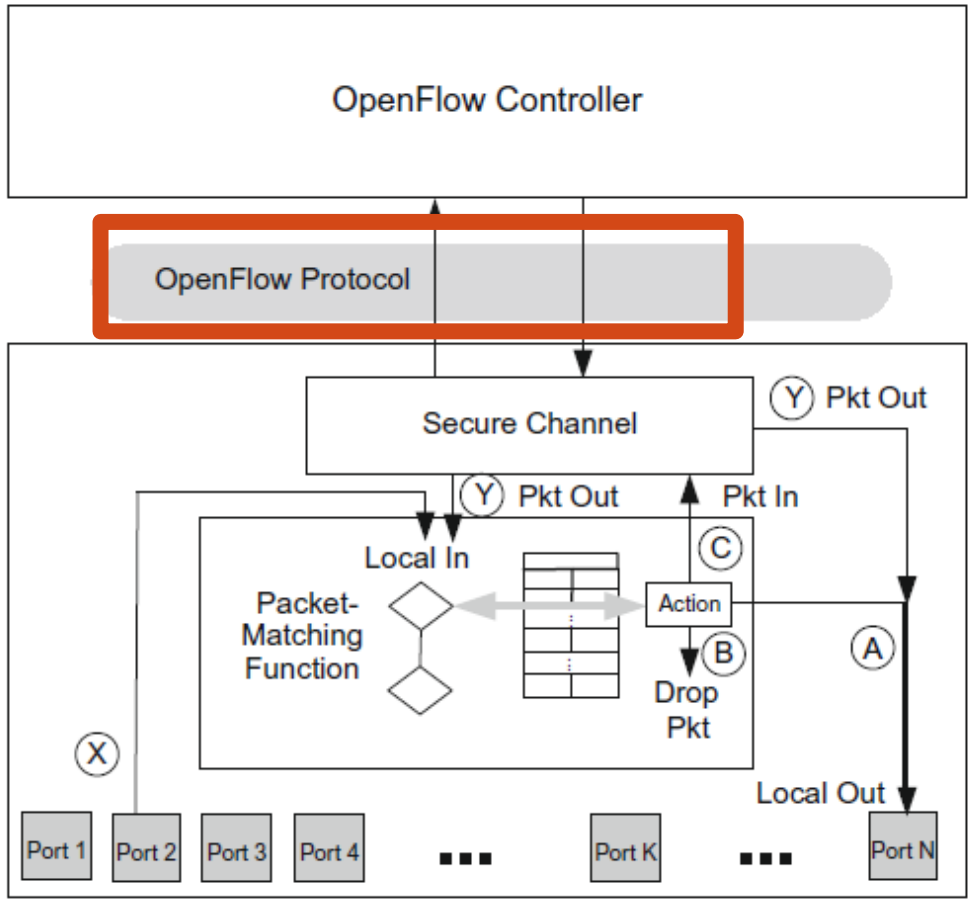
Overview of OpenFlow

- A switch receives packets on one port and forward it through another
  - Example: port 2 -> port N
- Potential actions
  - (A) Forward the packet out a local port
  - (B) Drop the packet
  - (C) Pass the packet to the controller via a PKT_IN message
- When the controller has a data packet to forward out through the switch, it uses the PACKET_OUT message (e.g., route advertisements)



Overview of OpenFlow

- The protocol consists of a set of messages from the controller to the switch and a corresponding set of messages in the opposite direction
- Basic ops: defining, modifying, and deleting flows

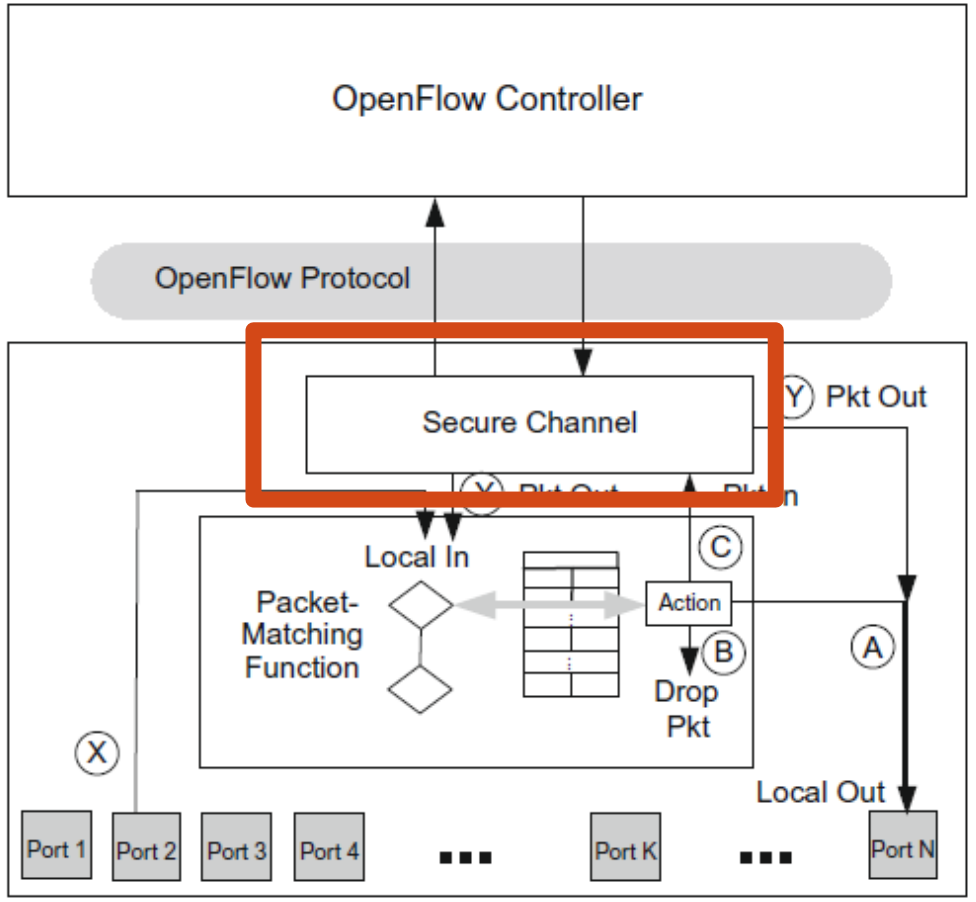


Overview of OpenFlow

- The secure channel is the path used for communications between the OpenFlow controller and the OpenFlow device
- This communication is secured by

  TLS-based encryption, though unencrypted

  TCP connections are allowed
- Connections may be in-band or out-of-band



Overview of OpenFlow – Flow Table

- The flow table lies at the core of the definition of an OpenFlow switch
- A flow table consists of flow entries
- A flow entry consists of header fields, counters, and actions associated with that entry

Flow Entry 0		Flow Entry 1		...	Flow Entry F		...	Flow Entry M	
Header Fields	Inport 12 192.32.10.1, Port 1012	Header Fields	Inport * 209.*.*.*, Port *		Header Fields	Inport 2 192.32.20.1, Port 995		Header Fields	Inport 2 192.32.30.1, Port 995
Counters	val	Counters	val		Counters	val		Counters	val
Actions	val	Actions	val		Actions	val		Actions	val

Overview of OpenFlow – Flow Table

- The header fields are used as match criteria
- The counters are used to track statistics relative to this flow, such as how many packets have been forwarded or dropped for this flow
- The actions fields prescribe what to do with a packet matching this entry

Flow Entry 0		Flow Entry 1		...	Flow Entry F		...	Flow Entry M	
Header Fields	Inport 12 192.32.10.1, Port 1012	Header Fields	Inport * 209.*.*.*, Port *		Header Fields	Inport 2 192.32.20.1, Port 995		Header Fields	Inport 2 192.32.30.1, Port 995
Counters	val	Counters	val		Counters	val		Counters	val
Actions	val	Actions	val		Actions	val		Actions	val

Overview of OpenFlow – Flow Table

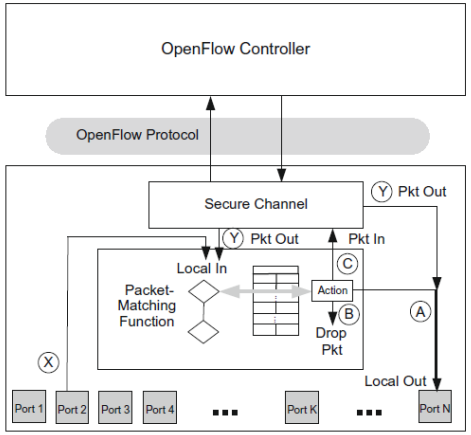
- The header fields are used as match criteria
- The counters are used to track statistics relative to this flow, such as how many packets have been forwarded or dropped for this flow
- The actions fields prescribe what to do with a packet matching this entry

Flow table, switch s1

```
root@admin:/home/sdn# ovs-ofctl dump-flows s1
 cookie=0x10000ea6f4b8e, duration=2519.646s, table=0, n_packets=280, n_bytes=11760, priority=40000,arp actions=CONTROLLER:65535
 cookie=0x100009465555a, duration=2519.646s, table=0, n_packets=0, n_bytes=0, priority=40000,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x100007a585b6f, duration=2519.644s, table=0, n_packets=0, n_bytes=0, priority=40000,dl_type=0x8942 actions=CONTROLLER:65535
 cookie=0x10000021b41dc, duration=2123.090s, table=0, n_packets=2, n_bytes=196, priority=5,ip actions=CONTROLLER:65535
 cookie=0x5f00002fa2d3c1, duration=2104.661s, table=0, n_packets=2055, n_bytes=201390, priority=10,in_port="s1-eth1",dl_src=ba:03:97:
90:39:4e,dl_dst=e2:8c:0c:de:82:db actions=output:"s1-eth2"
 cookie=0x5f000031ebed71, duration=2104.661s, table=0, n_packets=2055, n_bytes=201390, priority=10,in_port="s1-eth2",dl_src=e2:8c:0c:
de:82:db,dl_dst=ba:03:97:90:39:4e actions=output:"s1-eth1"
root@admin:/home/sdn#
```

Overview of OpenFlow

- Each message between controller and switch starts with the OpenFlow header
- The header specifies the OpenFlow version, message type, message length, and transaction ID of the message
- Three categories
  - Symmetric: sent by controller or switch w/o solicitation
  - Controller-switch: sent by controller to switch
  - Async: sent by switch to controller when there is any state change in the system



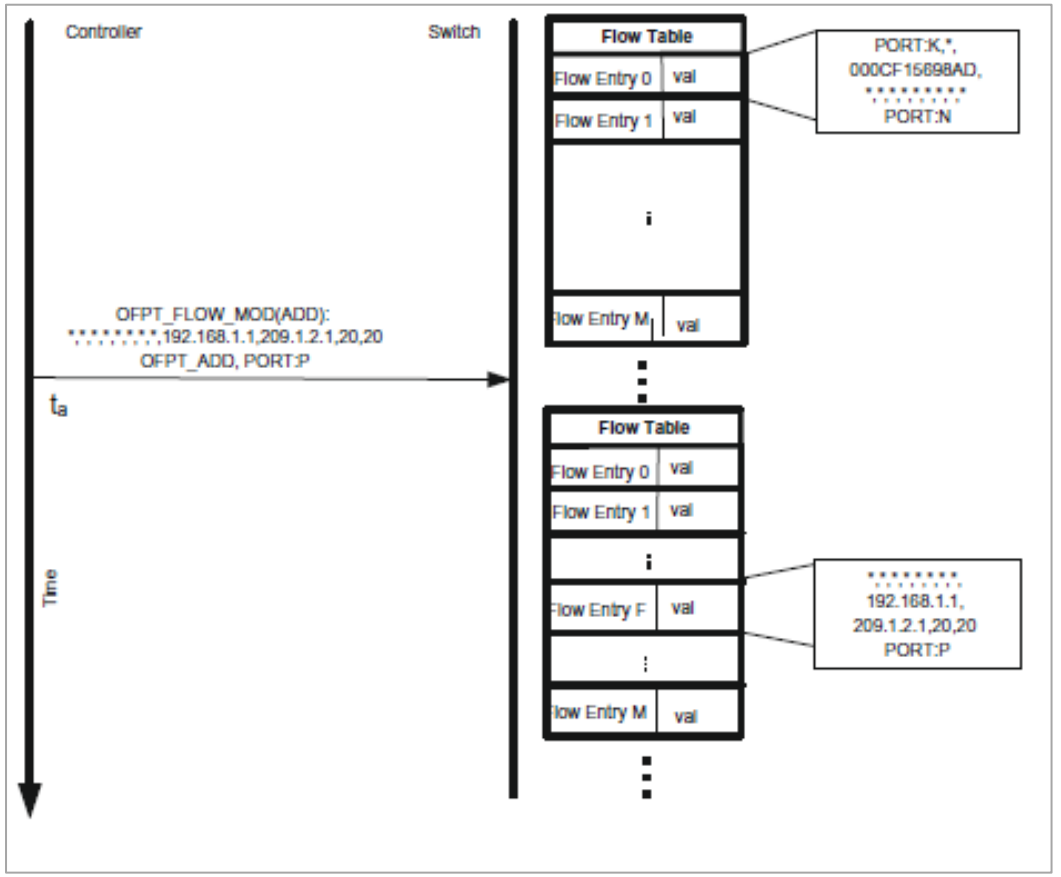
OFPT Message Types in OpenFlow 1.0

Message Type	Category	Subcategory
HELLO	Symmetric	Immutable
ECHO_REQUEST	Symmetric	Immutable
ECHO_REPLY	Symmetric	Immutable
VENDOR	Symmetric	Immutable
FEATURES_REQUEST	Controller-switch	Switch configuration
FEATURES_REPLY	Controller-switch	Switch configuration
GET_CONFIG_REQUEST	Controller-switch	Switch configuration
GET_CONFIG_REPLY	Controller-switch	Switch configuration
SET_CONFIG	Controller-switch	Switch configuration
PACKET_IN	Async	NA
FLOW_REMOVED	Async	NA
PORT_STATUS	Async	NA
ERROR	Async	NA
PACKET_OUT	Controller-switch	Cmd from controller
FLOW_MOD	Controller-switch	Cmd from controller
PORT_MOD	Controller-switch	Cmd from controller
STATS_REQUEST	Controller-switch	Statistics
STATS_REPLY	Controller-switch	Statistics
BARRIER_REQUEST	Controller-switch	Barrier
BARRIER_REPLY	Controller-switch	Barrier
QUEUE_GET_CONFIG_REQUEST	Controller-switch	Queue configuration
QUEUE_GET_CONFIG_REPLY	Controller-switch	Queue configuration

Overview of OpenFlow

Example: Controller Programming (populating) Flow Table

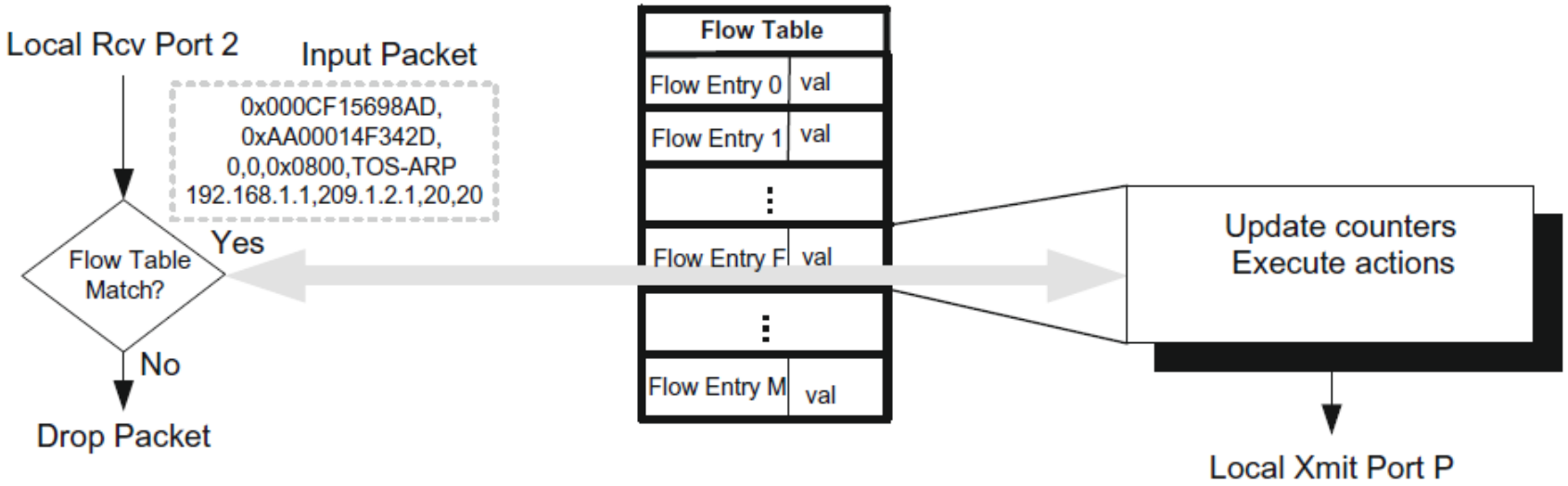
- At ta, the controller sends a FLOW_MOD (ADD) command
- A flow is added for packets entering the switch on any port
  - Source IP: 192.168.1.1
  - Destination IP: 209.1.2.1
  - Source TCP port: 20
  - Destination TCP port: 20
  - All other match fields have been wildcarded
  - The outport port is specified as P



Overview of OpenFlow

Example: Controller Programming (populating) Flow Table

- A packet arrives at the switch through port 2 with source IPv4 192.168.1.1 and destination IPv4 209.1.2.1
- The packet-matching function scans the flow table starting at flow entry 0 and finds a match in flow entry F
- Flow entry F stipulates that a matching packet should be forwarded out port P



OpenFlow Additions

- The OpenFlow interface started simple, with few protocols that could be matched against incoming packets
- Over few years, the specification has been extended with many more header fields and new protocols

Version	Date	Header fields
OpenFlow 1.0	Dec. 2009	12 (Ethernet, TCP, IPv4)
OpenFlow 1.1	Feb. 2011	15 (MPLS, …)
OpenFlow 1.2	Dec. 2011	36 (ARP, ICMP, IPv6, …)
OpenFlow 1.3	Jun. 2012	40
OpenFlow 1.4	Oct. 2013	41
OpenFlow 1.5	Mar. 2015	44

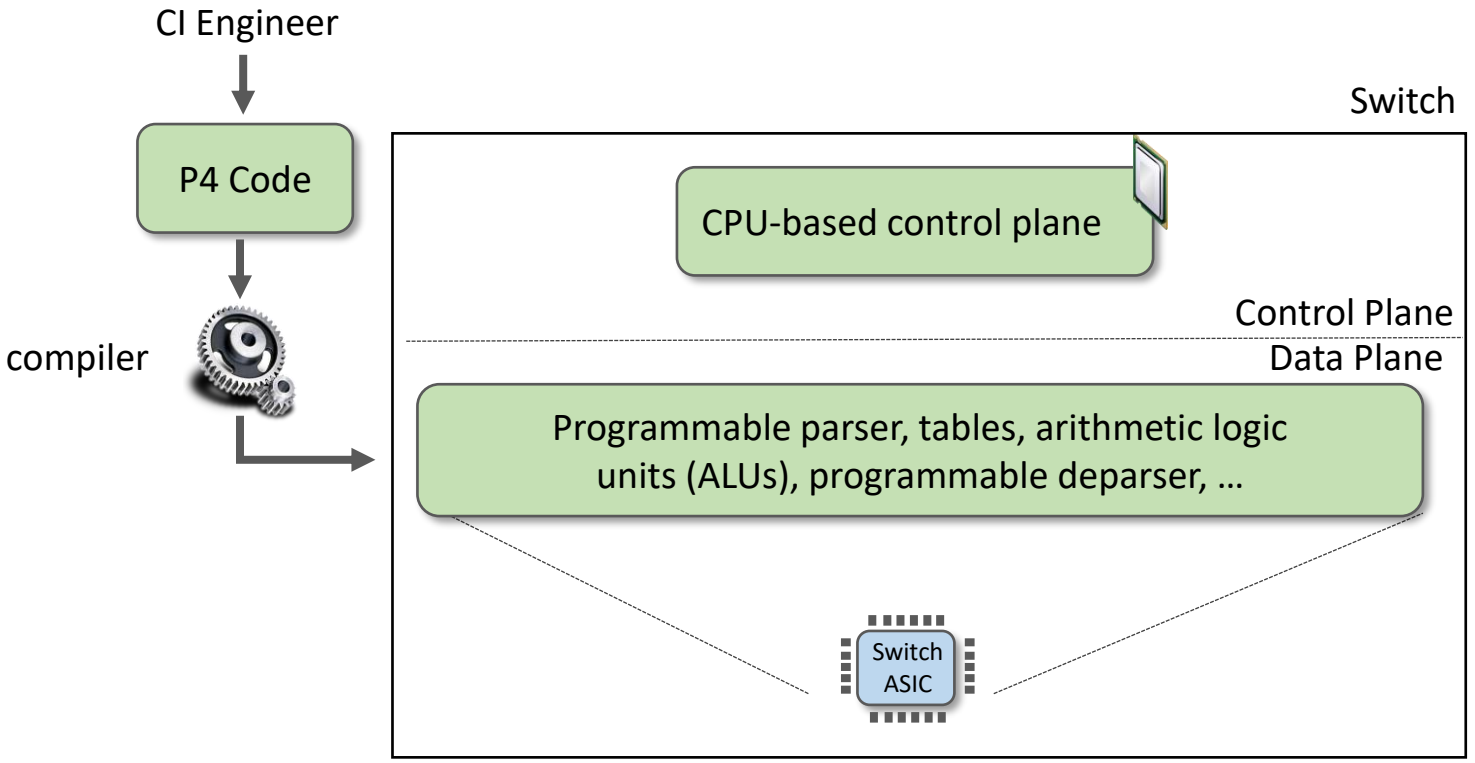
Bossart et al. “P4: Programming Protocol-Independent Packet Processors”
OpenFlow Switch Specs v1.5.1. Online https://tinyurl.com/y4j4a5eh

Limitations of SDN / OpenFlow

- SDN
  - Fixed number of header fields
  - OpenFlow repeatedly extends the specification
  - Long standardization cycles
  - Fixed protocols / header fields
  - Fixed parser
  - Devices still in control of manufacturers
  - Operators / programmers limited to functionality specified in the OpenFlow specification
  - Match+action stages are in series
- P4 switches (see p4.org)
  - Operators / programmers can define their own protocols and header fields
  - Immediate implementation
  - Customized protocols / header fields
  - Devices in control of operators / programmers
  - Match+action stages are in series or in parallel
  - Actions are composed of protocol-independent primitives (switch is not tight to specific protocols)
  - More future-proof

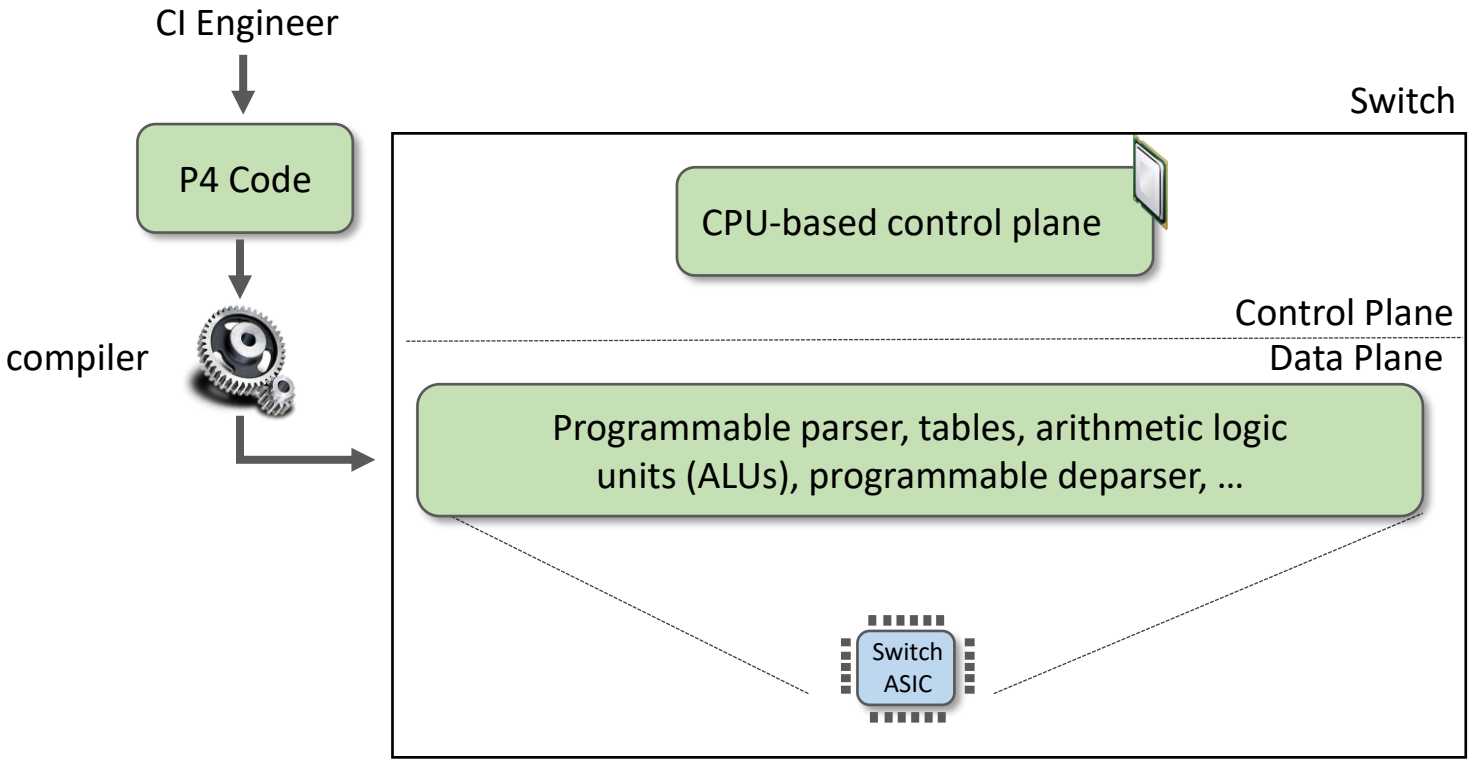
Programmable Switch ASICs

- We now have the technology that permits CI engineers to run customized functions



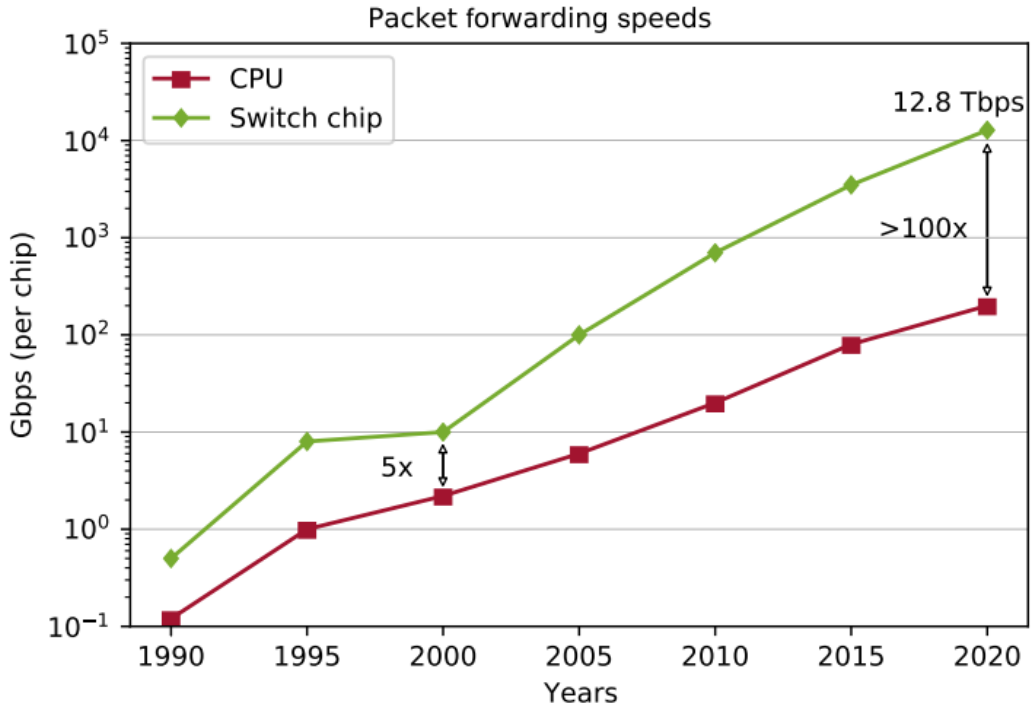
Programmable Switch ASICs

- We now have the technology that permits CI engineers to run customized functions
  - Designed for packet processing operations



Programmable Switch ASICs

- We now have the technology that permits CI engineers to run customized functions
  - Designed for packet processing operations
  - Much faster than general-purpose CPUs for processing packets



N. McKeown, "Creating an End-to-End Programming Model for Packet Forwarding," Netdev 0x14 Conference 2020, https://www.youtube.com/watch?v=fiBuao6YZl0&t=619s.

Workshop on Networking Topics

- Webpage with PowerPoint presentations:

  http://ce.sc.edu/cyberinfra/workshop_2023_feb.html

- Hands-on sessions: to access labs for the hands-on sessions, use the following link:

  https://netlab.cec.sc.edu/

- Username: email used for registration
- Password: nsf2023