High-speed Networks, Cybersecurity, and Software-defined Networking Workshop

Jorge Crichigno
University of South Carolina

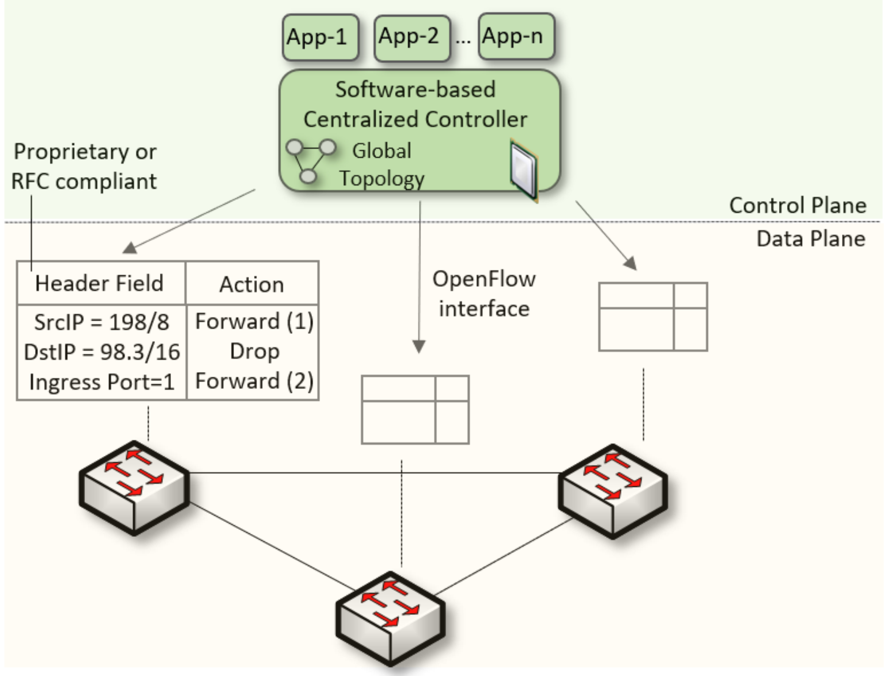
2020 Western Academy Support and Training Conference
Summer Conference
June 15 – June 19

National Science Foundation (NSF), Office of Advanced Cyberinfrastructure (OAC) and Advanced Technological Education (ATE)

Lab 6: OpenFlow

OpenFlow Overview

- 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)[1], which was created in 2011



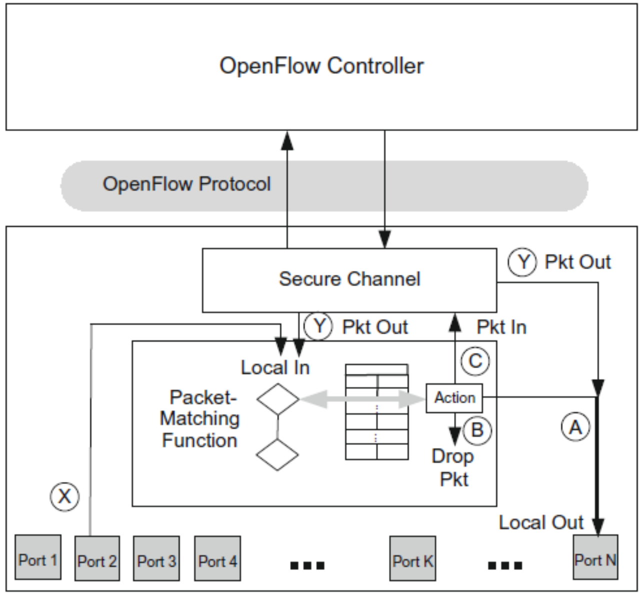
Header Field	Action
SrcIP = 198/8	Forward (1)
DstIP = 98.3/16	Drop
Ingress Port=1	Forward (2)

SDN network

[1] https://www.opennetworking.org/

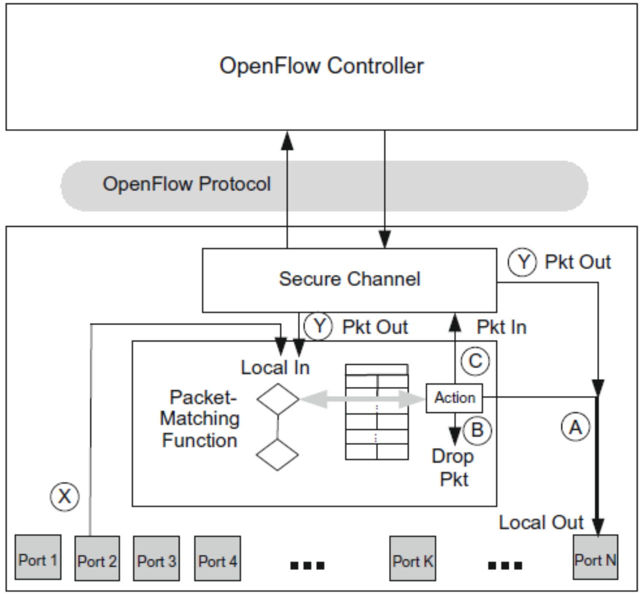
OpenFlow Switch / Controller

- The core function of a switch is to take packets arriving on one port (path X, port 2) and forward it through another port (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 OpenFlow PACKET_OUT message (e.g., routing advertisements, complex decisions)



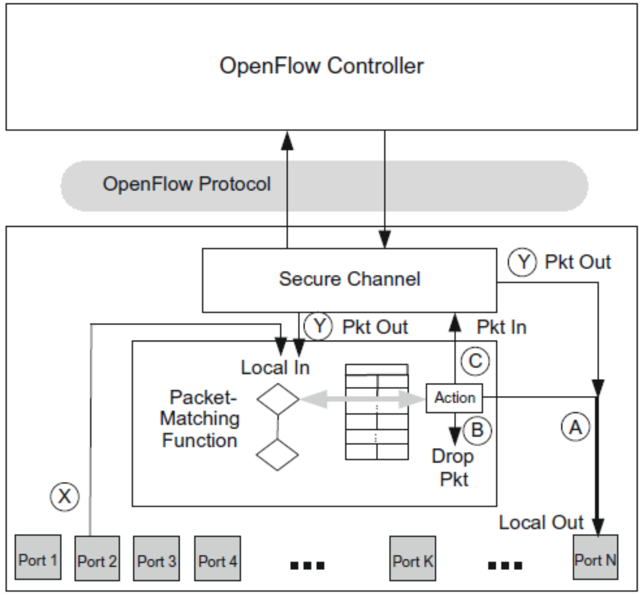
OpenFlow Protocol

- The protocol consists of a set of messages that are sent from the controller to the switch and a corresponding set of messages that are sent in the opposite direction
- The most basic operations are defining, modifying, and deleting flows
- A flow is a set of packets transferred from one network endpoint to another endpoint



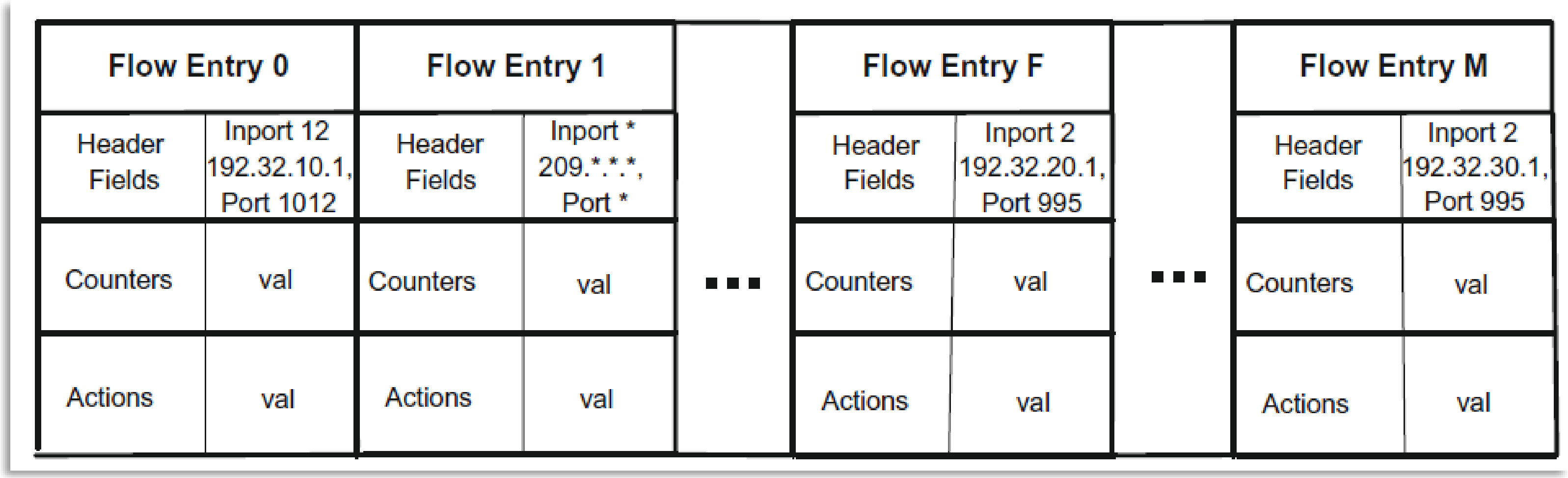
Controller-switch Secure Channel

- The secure channel is the path used for communications between the OpenFlow controller and the OpenFlow device
- Generally, this communication is secured by TLS-based encryption, though unencrypted TCP connections are allowed
- Connections may be in-band or out-of-band



Flow Table

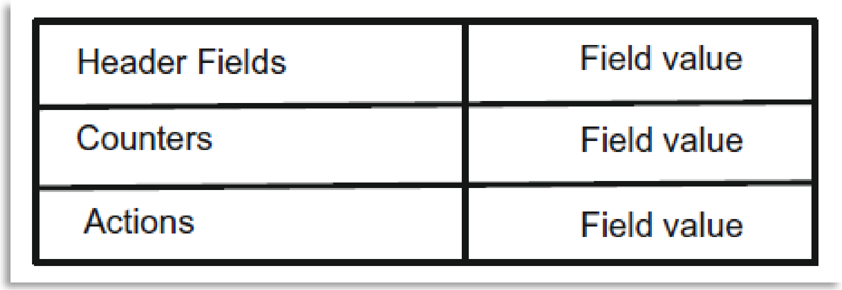
- The flow table lies at the core of the definition of an OpenFlow switch
- A flow table consists of flow entries
- A flow entry consists of header fields, counters, and actions associated with that entry



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

Flow Table

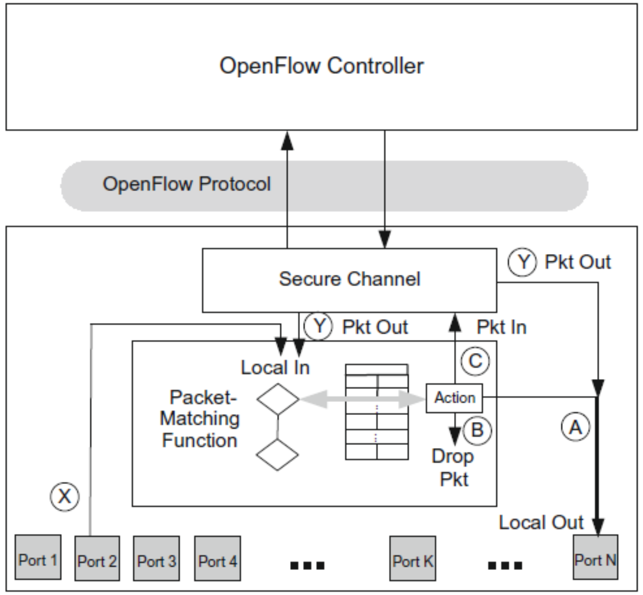
- The header fields are used as match criteria to determine whether an incoming packet matches this entry
- The counters are used to track statistics relative to this flow, such as how many packets have been forwarded or dropped for this flow
- The actions fields prescribe what to do with a packet matching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



Header Fields	Field value
Counters	Field value
Actions	Field value

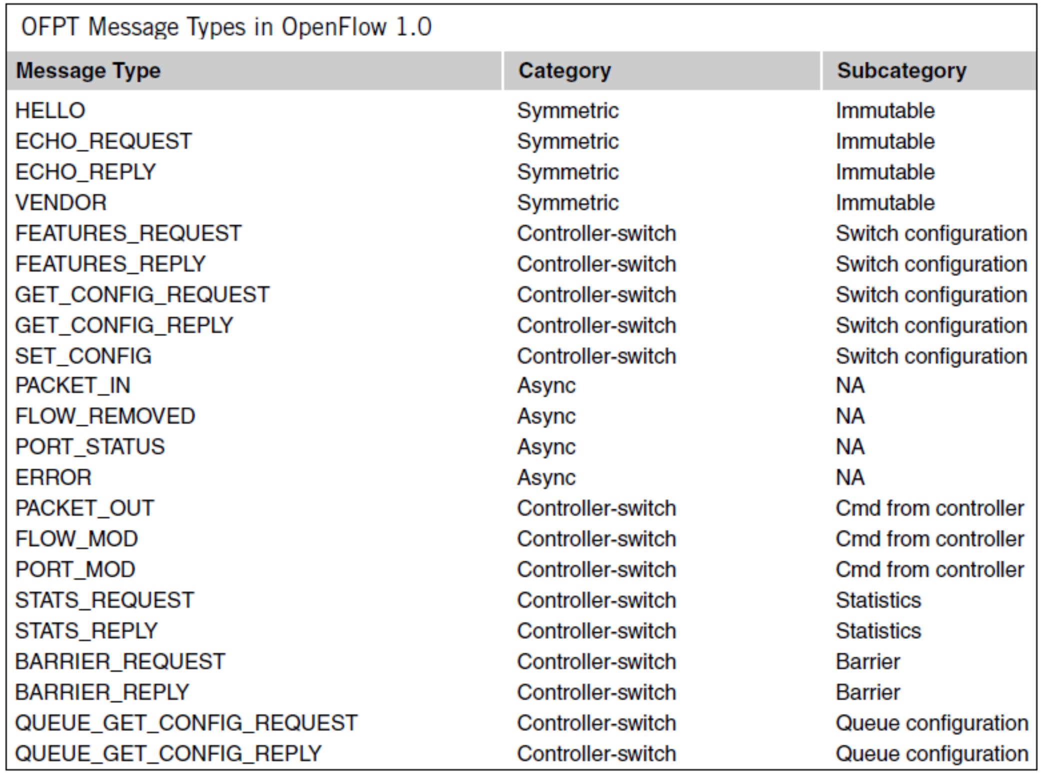
Actions and Packet Forwarding

- The required actions that must be supported by a flow entry are to either forward or drop the matched packet
- The most common case is that the output action specifies a physical port on which the packet should be forwarded



Messaging between Controller and Switch

- 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
  - Controller-switch: sent by controller to switch
  - Async: sent by switch to controller

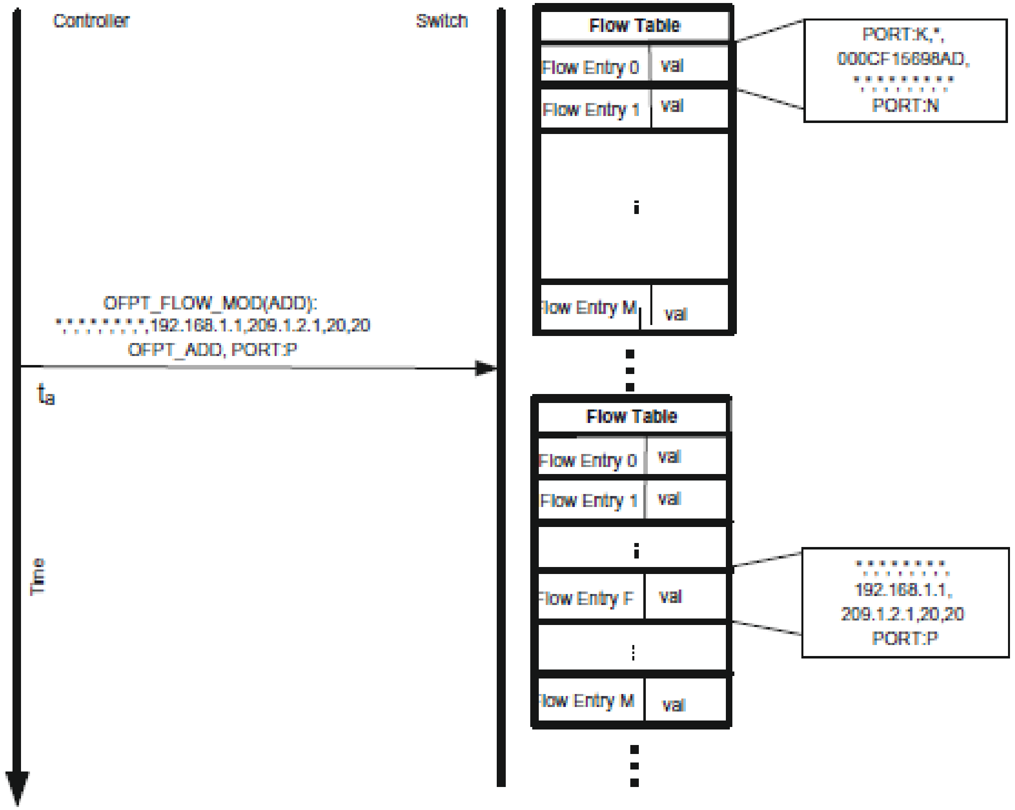


OFPT Message Types in OpenFlow 1.0

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

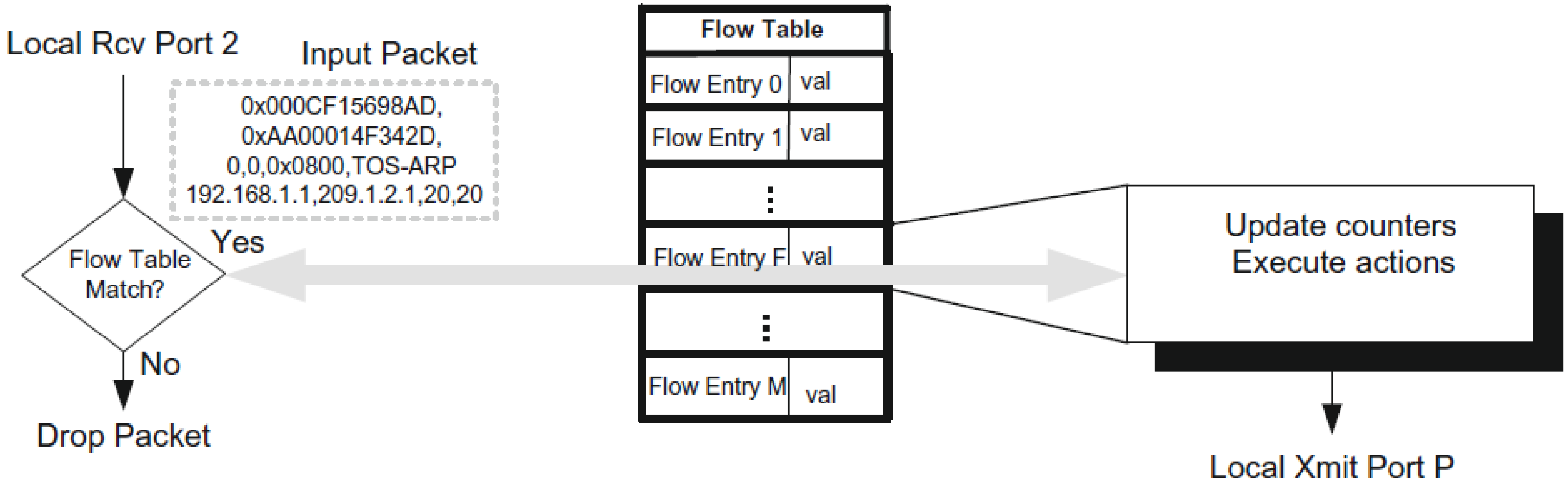
Example: Controller Programming Flow Table

- 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



Example: Basic Packet Forwarding

- 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

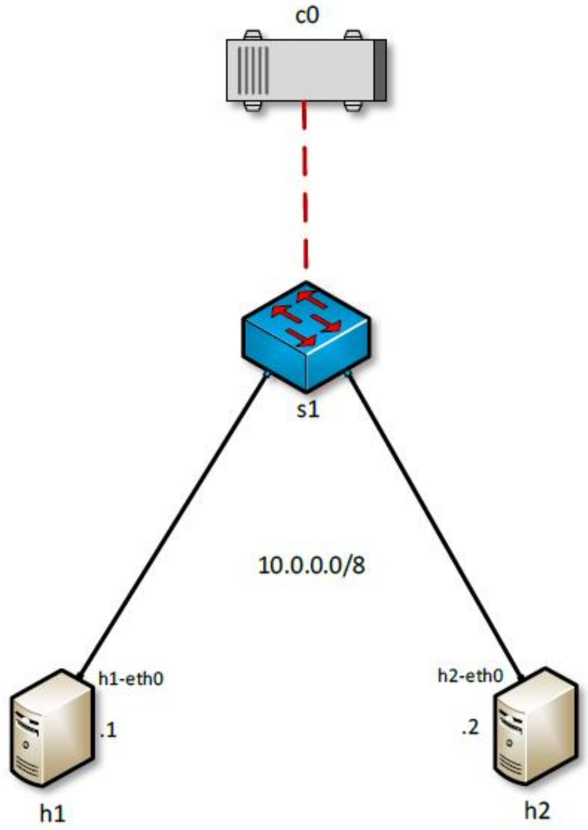
Bossart et al. "P4: Programming Protocol-Independent Packet Processors"
OpenFlow Switch Specs v1.5.1. Online https://tinyurl.com/y4j4a5eh

Weakness 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

Lab 6: OpenFlow Overview

- The topology consists of an ONOS controller, an open virtual switch (OVS) device, and hosts h1 and h2
- The OVS switch is administered using the ovs-ofctl command line utility
- The lab demonstrates how to inspect OpenFlow messages exchanged between the ONOS controller and the OVS switch


c0
s1
10.0.0.0/8
h1-eth0
.1
h1
h2-eth0
.2
h2