



PicOS Open vSwitch Configuration Guide

January, 2017

Version: 1

www.pica8.com

Pica8, Inc.
1032 Elwell Court, Suite 105
Palo Alto, CA. 94303
+1 (650) 614-5838

sales@pica8.com
support@pica8.com

Contents

OpenFlow Support Matrix	15
Introduction to Open vSwitch	16
Introduction to OpenFlow	17
OVS Web User Interface	18
Configuring Open vSwitch	19
Troubleshooting PicOS OVS	21
Examples and Topologies	22
PicOS OpenFlow Tutorials	23
OpenFlow Support Matrix	24
PicOS Support for OpenFlow 1.3	24
PicOS Support for OpenFlow 1.3.0	61
OpenFlow Messages	61
Symmetric Messages	61
Controller-to-Switch Messages	61
Asynchronous Messages	62
Multipart Messages	63
Capabilities Supported by Datapath	64
OpenFlow Ports	64
Port Descriptions	64
Port Administrative Settings	65
Port States	65
Port Features	65
OpenFlow Instructions	66
OpenFlow Action Types	67

OpenFlow Match Fields	67
OpenFlow Group Types	69
PicOS Support for OpenFlow 1.4	69
PicOS Support for OpenFlow 1.4.0	90
OpenFlow Messages	90
Symmetric Messages	90
Controller-to-Switch Messages	91
Asynchronous Messages	92
Bundle Operations	92
Multipart Messages	92
Capabilities Supported by Datapath	93
OpenFlow Ports	94
Port Descriptions	94
Port Administrative Settings	94
Port States	95
Port Features	95
OpenFlow Instructions	96
OpenFlow Action Types	96
OpenFlow Match Fields	97
OpenFlow Group Types	99
Introduction to Open vSwitch	100
Open vSwitch Components	100
Kernel Module	100
Open vSwitch Database	100
Open vSwitch Daemon	101
Introduction to OpenFlow	102
OVS Web User Interface	103
Login Interface	103
Monitoring the Switch	103
Adding a Bridge	105
Add a Port	106
Add GRE Port	107
Add Group Table	108
Add or Edit a Controller	108

Edit Flow Tables	108
Edit Lag Interface	109
Configuring Open vSwitch	110
Port Ranges	110
Get switch feature	110
Configuring ovsdb Locally	110
Configuring ovsdb Remotely	111
Add flow locally	112
Add flow remotely	112
 Basic Configuration in OVS Mode	114
Understanding OVS Components	115
 Creating Bridge and add ports	116
Creat a Bridge	116
Adding Ports to a Bridge	117
Adding the Default VLAN-ID	117
Setting the Port Link Speed	117
Add Bond Ports	118
Viewing the Bridge Settings	118
Deleting the Bridge	119
View software table flows	119
Enable STP in bridge.	120
 Connecting to a Controller	120
Modify Openflow Protocol	122
Connection Mode between Bridge and Controller	122
Example:	123
Configure Flow Table Flush once set or delete controller	124
 40G Changes to 4*10G in OVS	124
PORt SFP	127
Commands	128
OVS Mode Configuration	129
Normal (8 x 40G+40*10G)	130
Max (72x 10G)	131
40G Changes to 4*10G in OVS Mode on P-5401	134
OVS Mode Configuration	134
Normal (32 x 40G)	135
Half (16 x 40G + 64 x 10G)	136
Max (8 x 40G + 96 x 10G)	139
OVS Mode Configuration	144

Normal (32 x 40G)	144
Max (8 x 40G + 96 x 10G)	145
OVS mode Configuration	150
Normal(48*10G+4*40G)	150
Max(64*10G)	152
OVS mode Configuration	154
Normal(48*10G+4*40G)	155
Max(64*10G)	157
OVS mode Configuration	159
Normal (48*10G+6*40G)	159
Max (72*10G)	161
OVS mode Configuration	164
Max(64*10G)	166
OVS Mode Configuration	169
Normal (32 x 40G)	169
Half (16 x 40G + 64 x 10G)	170
Max (8 x 40G + 96 x 10G)	174
OVS mode Configuration	178
Normal (32 x 40G)	179
Half (16 x 40G + 64 x 10G)	180
Max (8 x 40G + 96 x 10G)	183
OVS Mode Configuration	188
Normal: 32 x 100G (or 40G)	188
SFP/max: 128 x 25G (or 10G)	189
Examples	195
100G port	195
40G Changes to 4*10G in OVS Mode on as7712_32x	197
OVS Mode Configuration	197
Normal: 32 x 100G (or 40G)	198
SFP/max: 128 x 25G (or 10G)	199
In OVS mode Configuration	204
In OVS mode Configuration	209
normal (48*10G+6*40G)	210
max (72*10G)	212
Configuring sFlow v5	214
Configuring NetFlow	215
Configuration Examples	215
Configuring IPv4 OpenFlow	216
Configure GRE Tunneling	217
GRE ON Physical PORT	217
Description	217
GRE ON LAG/LACP PORT	218
Static Lag and GRE Tunnel	218

LACP and GRE Tunnel	218
SW1 CONFIG:	219
SW2 CONFIG:	219
SEND Packets	220
Then reconfigure sw2	220
SEND Packets	221
Configuring MPLS	221
Hardware or Software based Forwarding	221
PUSH MPLS	222
SWAP MPLS Label	222
POP MPLS Label	222
PUSH MPLS Label and VLAN	223
POP One or Two MPLS Labels	224
POP One or Two MPLS Labels and PUSH VLAN	224
PUSH MPLS and POP VLAN	224
PUSH Two MPLS Labels and POP VLAN	225
PUSH Two MPLS Labels and POP Two VLANs	225
NOTICE:	225
Configuring LAG and LACP	225
LAG/LACP	225
	226
Create a Static LAG	226
Create a LACP Port	228
Create Static Flow for LAG or LACP	230
Display the LACP Information	230
LAG speed	231
Examples	231
Group Tables	233
The ALL Group	234
Example:	235
The INDIRECT Group	235
The FAST-FAILOVER Group	236
The SELECT Group	237
Example	239
Modify Bucket in a Group Table	242
Delete Group Table	242
Display the Information of Group Table	242
Priority of Arp Group	243
Example 1	243
Example 2	244
Configuring ECMP	244
Command	244
Parameters	244

Example	245
Class of Service Mapping for QoS	245
Configuring QoS Queue	246
PicOS OVS Supports qos/queue	246
WRR	246
WRED	249
Result	251
Configuring OpenFlow Meter	251
type=drop,without burst_size	252
type=drop,with burst_size	252
type=dscp_remark,without burst_size	252
type=dscp_remark,with burst_size	252
Configuring QinQ	253
Configuring OpenFlow Provider Backbone Bridge	254
Configuring Loopback	255
Loop the traffic which into switch back to ingress.	255
Loop the traffic which out from switch back to switch again.	256
Optimizing TCAM Usage	258
1, extend-group	258
2, match-mode	258
Configuring Layer 2 over GRE on Trident-2 based switches	260
Description	260
Examples	260
push one L2GRE header	261
Creating a L2GRE tunnel	261
strip L2GRE tunnel	261
configuration	261
configure two L2GRE tunnels on one physical port	262
Length of l2gre_key	263
Collaboration between nvGRE and VXLAN	264
Configuring VXLAN	265
Command	265
Description	265
Examples	265
configure a VXLAN tunnel	265
strip a VXLAN header	266
configure two VXLAN tunnels on a pair of physical port	267
collaboration between L2GRE and VXLAN	268
Configuring Multi-Table	270

Hardware OpenFlow Multi-table Limitations	270
Multi-Tables in TCAM	271
Using the Forwarding Database instead of the TCAM	272
Examples	273
Egress Flow Table	274
Multitable Resources	275
Description	275
Example	277
Configuring Network Address Translation	278
	279
Example 1: SNAT	279
Example 2: D NAT	280
Example 3: Packet-driven-flow	280
1. And te-1/1/2 receive packets with the src_ip 192.168.5.5 and src_port 1110.	282
Due to ASIC limitation, a flow can not modify l4_src_port without modifying SIP or modify l4_dst_port without modifying DIP.	282
If only modifying SIP(DIP) or SIP+L4_SRC_PORT(DIP+L4_DST_PORT), up to 2k flow can be configured. If modifying both SIP[]L4_SRC_PORT] and DIP[]L4_DST_PORT], the flow supported is 1k.	282
2. If set_dl_src is included in actions, the packets will be stamped with set_dl_src (as before). If set_dl_src is not included in actions, the packets will be stamped with the original dl_src . That is to say, keep the original source mac address.	282
ASIC Limitation	283
udp/ip, tcp/ip	283
Vlan Isolation	283
Configuring CFM	284
Monitor connectivity to a remote maintenance point on ge-1/1/1	285
Set the MPID of CFM:	285
CFM Example	286
Step 1: Basic configuration	286
Step 2: Configure cfm:	286
Step 3: Check packets	287
Configure cfm on port ge-1/1/13, if delete cfm	289
Open vSwitch Configuration File	289
OVS LLDP	289
LLDP commands	291
Enable or disable lldp on bridge	291
LLDP admin status	291
LLDP transmit parameters	291
LLDP transmit parameters on a per interface	291
Optional TLVs in lldpdu	291
show configuration	293
show LLDP remote neighbors	293

show LLDP using "ovs-appctl"	293
Examples	293
Abstract	294
	294
Example 1	294
Example 2	295
UDF L4	296
Macro UDF	296
Combinated Mode	298
Commands	298
Example	298
Enabling Radius in PicOS OVS Mode	299
Creating SSL Connection to a Controller	301
PicOS Switch	301
Controller	302
Related articles	303
Configuring L2MPLS	303
PUSH L2 MPLS	303
SWAP L2 MPLS Label	303
POP L2MPLS Label	303
PUSH L2MPLS Label and VLAN	304
Pushing One MPLS Label and Pop One VLAN	304
	304
Pushing One L2MPLS Label and Pop One VLAN	304
Pushing Two L2MPLS Headers	305
Configuring Inner VLAN	305
Inventory Database	306
Introduction	306
Switch Information	306
SFP/QSFP Information	307
Counter Information	308
Alarm Information	309
Hardware Flow Information	310
Show Power/Fan Information	311
Configuring option-match-vlan-type	312
Connecting to Manager	312
OVS LFS	314
Abstract	314
Link Fault Signaling	315
LFS Commands	315
Up Mode	315

Configure TPID in port	316
1, TPID used in Q-in-Q	316
2, TPID used in push L2MPLS	317
Broadcom Chip Limitation in OVS	319
Goto_table	319
Clear counter	321
Clear port counter	321
Clear flow counter	321
Ovs CLI Enhancements	321
List ovs Configuration	321
List System Resources Usage	322
Associate sw-flow with hw-flow	323
Display dpid in both hex and decimal	324
List Interface Details	324
Table Type Pattern	325
TTP Multicast	326
Examples	327
TTP Unicast	335
Introduction	335
Enabling TTP Mode	336
VLAN Table	336
VLAN Filter Table	336
VLAN Assignment Table	336
VLAN Translate Single Tag Table	337
Terminal MAC Table	337
L3 Table	337
Policy ACL Table	338
Check TTP flows	339
	340
List TTP System Resources Usage	340
Flow_id in TTP flows	340
Flow Handling Mode	341
Direct Flows	341
Packet-driven-flows	342
Drop counter	344
ecmp-select and lag-select group	347
ECMP Select Group	348
LAG Select Group	350
Close all Group-ranges	351
ingress-mirror-group and egress-mirror-group	351

Example 1	352
Example 2	352
Close all Group-ranges	353
Configuring Meter	353
Summary	353
Configuration	353
Ingress Meter	354
1R2C: Add a meter, the type=drop.	354
1R2C: Add a meter, the type=dscp_remark.	354
Check the meter configuration	354
Modify one meter	354
Dump meter stats	355
Delete one meter or all meters	355
Notice: If one meter applies to multiple flow entries, all the flow entries will share the meter rate.	
355	
Example:	355
2R3C:	355
Example:	356
Egress Meter	356
Application	357
Other Result:	358
2.	359
Configuration saving	359
Troubleshooting PicOS OVS	361
Verifying PicOS Mode	361
Verifying Bridge Configuration	361
Checking Flow Discrepancies	362
Displaying OVSDB	363
Debug Packet-In Messages	363
Issue 1, During add port error.	363
Examples and Topologies	366
802.1Q VLAN	366
Configure Switch-A	366
Configure Switch-B	367
ECMP	367
GRE Tunnel	367

Configure Switch-A	368
Configure Switch-B	368
MPLS Network	368
Configure Switch-A	369
Configure Switch-B	370
Configure Switch-C	370
Configure Switch-D	370
Multiple Virtual Bridges	371
SSL Connection to Controller	371
PicOS OpenFlow Tutorials	375
Basic Bridge Configuration	375
Basic Flow Configurations	375
Connection to a RYU Controller	375
Connection to OpenDaylight Controller	375
Connection to a Floodlight Controller	376
Configuration Guide for Atrium Stack on ONOS Controller	376
Basic Bridge Configuration	376
Basic Bridge Introduction	376
Power On Configuration	378
Configure Switch	380
Configure Bridge	380
Configure Port	382
Default Bridge Behavior	382
OVS Commands Reference	384
Basic Flow Configurations	384
Flow Introduction	385
Modify Default Flow	386
Uni-directional Flow	387
1-to-Many Multicasting	390
Many-to-One Aggregation	391
OVS Commands Used in this Tutorial	393
Packet Address File	394
Connection to a RYU Controller	394
RYU Introduction	394
Introduce RYU Open Flow Controller	396
Configure OVS for RYU OpenFlow Controller	397
Controller-OVS Interaction	398

RYU Simple Switch Application	400
Open Flow Message Type	402
RYU Guide OVS Commands Reference	403
RYU Controller Configuration	403
Connection to OpenDaylight Controller	410
OpenDaylight Introduction	410
Introduction to the OpenDaylight OpenFlow Controller	411
Configure OVS for OpenDaylight Open Flow Controller	412
OpenDaylight Controller-OVS Interaction	413
OpenDaylight Simple Switch Application	415
Message Type of Open Flow	416
OVS Commands Reference 04	416
Connection to a Floodlight Controller	417
Floodlight Controller Introduction	417
Floodlight Open Flow Controller	417
Changes to Floodlight	418
Test Topology	419
Configure OVS	420
Launch Floodlight	423
Floodlight REST Interface	425
curl	426
Add Flows	426
Delete Flows	426
Configuration Guide for Atrium Stack on ONOS Controller	427
ONOS	427
Atrium	427
Installation Guide	428
Distribution VM	428
Running the Distribution VM on a Server	431
Running the Distribution VM on the Switch	431
Installation Steps	431
Configuring ONOS	431
Launching ONOS for a Test	434
Launching ONOS for Deployment	435
Quagga Configuration Guide	436
Configuring Quagga BGP & Connecting to ONOS	436
Launching Quagga	438
Configuration & Launch	439
Atrium Specific Configuration	439
Managing PicOS Switches with OpenDaylight	442
Installing OpenDaylight on Linux	444
Installing OpenDaylight on Windows	446
Configuring PicOS Switch for OpenDaylight	447

The OpenFlow protocol is driven by ONF (Open Networking Foundation), a leader in software-defined networking (SDN). The OpenFlow protocol encompasses three essential components of an SDN framework:

1. A physical OpenFlow switch.
2. A virtual OpenFlow switch to manage virtual machines.
3. An OpenFlow controller, to organize all network pieces.

i The Pica8 PicOS software supports features in OpenFlow 1.0 / OpenFlow 1.1 / OpenFlow1.2 / OpenFlow 1.3.x and OpenFlow 1.4. The details of feature supports in OpenFlow1.3.0 and OpenFlow 1.4.0 please see PicOS Support for OpenFlow 1.3.0 and PicOS Support for OpenFlow 1.4.0.

The following websites provide detailed information on Open vSwitch and the OpenFlow protocol.

- **Open vSwitch:** <http://openvswitch.org>
- **OpenFlow:** <http://www.opennetworking.org/sdn-resources/openflow>

PicOS can run in two different modes:

- **OVS (Open vSwitch) mode:** In this mode, PicOS is dedicated and optimized for Openflow applications.
- **L2/L3 (Layer 2/Layer 3) mode:** In this mode, PicOS can run switching and routing protocols, as well as OpenFlow applications

In OVS mode, L2/L3 daemons are not running; the system is fully dedicated to Openflow and OVS.

In L2/L3 mode, L2/L3 daemons are running, but OVS can also be activated if CrossFlow is activated.

This chapter assumes that the PicOS OVS mode is active. Please see PicOS Mode Selection to modify the PicOS mode.

OpenFlow Support Matrix

- PicOS Support for OpenFlow 1.3
- PicOS Support for OpenFlow 1.3.0
- PicOS Support for OpenFlow 1.4
- PicOS Support for OpenFlow 1.4.0

Introduction to Open vSwitch

Introduction to OpenFlow

OVS Web User Interface

- Login Interface
- Monitoring the Switch
- Adding a Bridge
- Add a Port
- Add GRE Port
- Add Group Table
- Add or Edit a Controller
- Edit Flow Tables
- Edit Lag Interface

Configuring Open vSwitch

- Basic Configuration in OVS Mode
- Creating Bridge and add ports
- Connecting to a Controller
- 40G Changes to 4*10G in OVS
- Configuring sFlow v5
- Configuring NetFlow
- Configuring Port Mirroring
- Configuring IPv4 OpenFlow
- Configure GRE Tunneling
- Configuring MPLS
- Configuring LAG and LACP
- Creating a Group Table
- Priority of Arp Group
- Configuring ECMP
- Class of Service Mapping for QoS
- Configuring QoS Queue
- Configuring OpenFlow Meter
- Configuring QinQ
- Configuring OpenFlow Provider Backbone Bridge
- Configuring Loopback
- Optimizing TCAM Usage
- Configuring Layer 2 over GRE on Trident-2 based switches
- Configuring VXLAN
- Configuring Multi-Table
- Configuring Network Address Translation
- ASIC Limitation
- Configuring CFM
- Open vSwitch Configuration File
- OVS LLDP
- OVS Udf
- Combinated Mode

- Enabling Radius in PicOS OVS Mode
- Creating SSL Connection to a Controller
- Configuring L2MPLS
- Inventory Database
- Configuring option-match-vlan-type
- Connecting to Manager
- OVS LFS
- Configure TPID in port
- Broadcom Chip Limitation in OVS
- Goto_table
- Clear counter
- Ovs CLI Enhancements
- Table Type Pattern
- Flow Handling Mode
- Drop counter
- ecmp-select and lag-select group
- ingress-mirror-group and egress-mirror-group
- Configuring Meter
- Configuration saving

Troubleshooting PicOS OVS

- Debug while switch port cannot up

Examples and Topologies

- 802.1Q VLAN
- ECMP
- GRE Tunnel
- MPLS Network
- Multiple Virtual Bridges
- SSL Connection to Controller

PicOS OpenFlow Tutorials

- Basic Bridge Configuration
- Basic Flow Configurations
- Connection to a RYU Controller
- Connection to OpenDaylight Controller
- Connection to a Floodlight Controller
- Configuration Guide for Atrium Stack on ONOS Controller

OpenFlow Support Matrix

- PicOS Support for OpenFlow 1.3
- PicOS Support for OpenFlow 1.3.0
- PicOS Support for OpenFlow 1.4
- PicOS Support for OpenFlow 1.4.0

PicOS Support for OpenFlow 1.3

The following table contains OpenFlow 1.3 features supported by PicOS. For clarity, the feature names in this table are identical to the feature names found in OpenFlow Switch Specification Version 1.3.0.

Table 1 OpenFlow 1.3 Features Supported by PicOS

OpenFlow V1.3 Section #	Title	Features	Additional Feature Specification	R2.0 OVS	R2.0 TCA
1	Introduction	NA			
2	Switch Components	Flow tables		Y	Y
		Group table		Y	N
3	Glossary				
4	OpenFlow Ports	See Section 4.3 - 4.5		Y	Y
4.1	OpenFlow Ports	See Section 4.3 - 4.5		Y	Y
4.2	Standard Ports	See Section 4.3 - 4.5		Y	Y
4.3	Physical Ports	Ingress	OpenFlow packets are received on an ingress port, processed by the OpenFlow pipeline. The packet ingress port is a property of the packet throughout the OpenFlow pipeline and represents the OpenFlow port on which the packet was received into the OpenFlow switch	Y	Y
		Output	The OpenFlow pipeline can decide to send the packet on an output port using the output action (see 5.9), which defines how the packets goes back to the network	Y	Y
		Groups		Y	Y
		Hardware interface		N	N
		Virtual slicing of hardware interface		Y	Y

4.4	Logical Ports	Logical ports are switch defined ports that don't correspond directly to a hardware interface of the switch	Logical ports are higher level abstractions that may be defined in the switch using non-OpenFlow methods		
		LAG		Y	N
		Tunnels		Y	N
		Loopback interface		N	N
		Ingress		Y	Y
		Output		Y	Y
		Groups		Y	
		Map to various physical port		N	N
		PACKET_IN reports logical port and its underlying physical port (GRE & LAG)		N	N
4.5	Local Reserved Port	Ingress		N	N
		Output		Y	Y
		Groups		Y	N
		ALL	Only as an output port	Y	Y
		CONTROLLER	Represent the control channel with the OpenFlow controller	Y	Y
		TABLE	Represent the start of the OpenFlow pipeline	Y	Y
		IN_PORT	Used only as an output port, send the packet out its ingress port	N	N
		ANY	Cannot be used as an ingress port nor as an output port	Y	Y
		LOCAL	Represent the switch's local networking stack. Can be used as an ingress port or as an output port	Y	Y
			The local port enables remote entities to interact with the switch via the OpenFlow network, rather than via a separate control network. it can be used to implement an in-band controller connection		
		NORMAL	Non-OpenFlow pipeline used only as an output port	N	N
		FLOOD	Flooding using the normal pipeline of the switch, used only as an output port	Y	Y
			Packet out all standard ports	Y	Y

			But not to the ingress port or ports that are in OFPPS_BLOCKED state	Y	Y
5	OpenFlow Tables				
5.1	Pipeline Processing	OpenFlow-only	All packets are processed by the OpenFlow pipeline	Y	N
		OpenFlow-hybrid	OpenFlow operation and normal Ethernet switching operation	N	N
			VLAN tag to decide whether to process the packet using which pipeline	N	N
			Input port to decide whether to process the packet using which pipeline	N	N
			Allow a packet to go from the OpenFlow pipeline to the normal pipeline through the NORMAL and FLOOD reserved ports	N	Y
		Multiple flow tables		Y	N
		Sequentially numbered, starting at 0.		Y	Y
		Only go forward and not backward		Y	Y
		Last table of the pipeline can not include the Goto instruction		Y	N
		Table miss behavior configuration	Send packets to the controller	Y	Y
			Drop the packet	Y	Y
			The packet is processed by the next sequentially numbered table	N	N
			The packet is processed by L2/L3 pipelines	N	N

5.2	Flow Table	Flow table entry	Match fields	Y	Y
			Counters	Y	Y
			Instructions	Y	Y
5.3	Matching	Packet headers		Y	Y
		Ingress port		Y	Y
		Metadata fields	Used to pass information between tables	N	N
		State transition	Actions applied in a previous table using the Apply-Actions are reflected in the packet match field	N	N
		Support ANY	Matches all possible values in the header	Y	Y
		Support arbitrary bitmasks on specific match fields		Y	Y
		Select highest priority flow entry		Y	Y
		Counters associated with the selected flow entry must be updated		Y	Y
		CHECK_OVERLAP bit on flow mod messages to avoid overlapping entries		Y	N
		Multiple matching flow entries with the same highest priority		N	N
		Support OFPC_FRAG_REASM flag	IP fragments must be reassembled before pipeline processing	N	N
		Behavior when a switch receives a corrupted packet		N	N
5.4	Group Table	Group identifier	A 32 bit unsigned integer	Y	N
		Group type	To determine group semantics	Y	N

		Counters	Updated when packets are processed by a group	Y	N
		Action buckets	An ordered list of action buckets	Y	N
5.4.1	Group Types	All	Execute all buckets in the group for multicast or broadcast	Y	N
			Packet clone is dropped if a bucket directs a packet explicitly out the ingress port	Y	N
			Support output action to the OFPP_IN_PORT reserved port	Y	N
		Select	Execute one bucket in the group based on a switch-computed selection algorithm	N	N
		Indirect	Execute the one defined bucket in this group	Y	Y
		Fast failover	Execute the first live bucket which is associated with a live port/group is selected	Y	Y
	ECMP		Hashing	N	N
			Round robin	N	N
5.5	Per Table Counters	Reference count (active entries)	32 bits	N	N
		Packet Lookups	64 bits	N	N
		Packet Matches	64 bits	N	N
	Per Flow Counters	Received Packets	64 bits	N	N
		Received Bytes	64 bits	Y	Y
		Duration (seconds)	32 bits	Y	Y
		Duration (nanoseconds)	32 bits	N	N
	Per Port Counters	Received Packets	64 bits	Y	Y
		Transmitted Packets	64 bits	Y	Y
		Received Bytes	64 bits	Y	Y
		Transmitted Bytes	64 bits	Y	Y
		Receive Drops	64 bits	Y	Y
		Transmit Drops	64 bits	Y	Y
		Receive Errors	64 bits	Y	Y
		Transmit Errors	64 bits	Y	Y
		Receive Frame Alignment Errors	64 bits	Y	Y

		Receive Overrun Errors	64 bits	Y	Y
		Receive CRC Errors	64 bits	Y	Y
		Collisions	64 bits	Y	Y
	Per Queue Counters	Transmit Packets	64 bits	N	N
		Transmit Bytes	64 bits	N	N
		Transmit Overrun Errors	64 bits	N	N
	Per Group Counters	Reference Count (flow entries)	32 bits	Y	Y
		Packet Count	64 bits	N	N
		Byte Count	64 bits	Y	Y
	Per Bucket Counters	Packet Count	64 bits	N	N
		Byte Count	64 bits	N	N
5.6	Instructions	The controller can query the switch about which of the "Optional Instruction" it supports		Y	Y
		Apply-Actions action(s)	Applies the specific action(s) immediately, without any change to the Action Set	Y	Y
		Clear-Actions	Clears all the actions in the action set immediately	N	N
		Write-Actions action(s)	Merges the specified action(s) into the current action set	N	N
		Write-Metadata metadata / mask	Writes the masked metadata value into the metadata field	N	N
		Goto-Table next-table-id	Indicates the next table in the processing pipeline	Y	Y
		Clear-Actions instruction is executed before the Write-Actions instruction		N	N
		Goto-Table is executed last		Y	Y
		Reject a flow entry if it is unable to execute the instructions and return an unsupported flow error		Y	Y
5.7	Action Set	Action set is associated with each packet		Y	Y

		Set is empty by default		Y	Y
		Action set is carried between flow tables		Y	Y
		When the instruction set of a flow entry does not contain a Goto-Table instruction, pipeline processing stops and the actions in the action set of the packet are executed		Y	Y
		Action set contains a maximum of one action of each type		Y	Y
		The actions in an action set are applied in the order specified below		Y	Y
		1. copy TTL inwards		N	N
		2. pop		Y	Y
		3. push		Y	Y
		4. copy TTL outwards		N	N
		5. decrement TTL		N	N
		6. set		Y	Y
		7. qos		Y	N
		8. group	If a group action is specified, apply the actions of the relevant group bucket(s) in the order specified by this list.	Y	N
		9. output	If no group action is specified, forward the packet on the port specified by the output action. The output action in the action set is executed last.	Y	Y
			If both an output action and a group action are specified in an action set, the output action is ignored and the group action takes precedence.	Y	Y
			If no output action and no group action were specified in an action set, the packet is dropped.	Y	Y
			The execution of groups is recursive if the switch supports it; a group bucket may specify another group, in which case the execution of actions traverses all the groups specified by the group configuration.	Y	Y
5.8	Action List	Apply-Actions instruction and the Packet-out message include an action list	The actions of an action list are executed in the order specified by the list, and are applied immediately to the packet.	Y	Y
			The effect of those actions is cumulative.	Y	Y

			If the action list contains an output action, a copy of the packet is forwarded in its current state to the desired port.	Y	Y
			If the list contains a group actions, a copy of the packet in its current state is processed by the relevant group buckets.	N	N
5.9	Actions	The controller can also query the switch about which of the "Optional Action" it supports		Y	Y
		Output	Support forwarding to physical ports, switch-defined logical ports and the required reserved ports.	Y	Y
		Set-Queue	The set-queue action sets the queue ID for a packet and is used to provide basic Quality-of-Service (QoS) support.	Y	Y
		Drop		Y	Y
		Group		Y	Y
		Push-Tag/Pop-Tag	Order of header fields - Ethernet, VLAN, MPLS, ARP/IP, TCP/UDP/SCTP (IP-only).	Y	Y
		Push VLAN header	Push a new VLAN header onto the packet. The Ethertype is used as the Ethertype for the tag. Only Ethertype 0x8100 and 0x88a8 should be used.	N	N
		Pop VLAN header	Pop the outer-most VLAN header from the packet.	N	N
		Push MPLS header	Push a new MPLS shim header onto the packet. Only Ethertype 0x8847 and 0x8848 should be used.	Y	Y
		Pop MPLS header	Pop the outer-most MPLS tag or shim header from the packet.	Y	Y
		Set-Field			
		Set VLAN ID		Y	Y
		Strip VLAN ID		N	N
		Change-TTL	Modify the values of the IPv4 TTL, IPv6 Hop Limit or MPLS TTL in the packet.	N	N
			If it is supported, applied to the outermost-possible header.	N	N

		Set MPLS TTL	8 bits: New MPLS TTL, Replace the existing MPLS TTL. Only applies to packets with an existing MPLS shim header.	Y	Y
		Decrement MPLS TTL	Decrement the MPLS TTL. Only applies to packets within existing MPLS shim header.	Y	Y
		Set IP TTL	Replace the existing IPv4 TTL or IPv6 Hop Limit and update the IP checksum. Only applies to IPv4 and IPv6 packets.	N	N
		Decrement IP TTL	Decrement the IPv4 TTL or IPv6 Hop Limit field and update the IP checksum. Only applies to IPv4 and IPv6 packets.	N	N
		Copy TTL outwards	Copy the TTL from next-to-outermost to outermost header with TTL. Copy can be IP-to-IP, MPLS-to-MPLS, or IP-to-MPLS.	N	N
		Copy TTL inwards	Copy the TTL from outermost to next-to-outermost header with TTL. Copy can be IP-to-IP, MPLS-to-MPLS, or MPLS-to-IP.	N	N
5.9.1	Default Values (for Fields on Push)	Field values for all fields specified in Table 6 should be copied from existing outer headers to new outer headers	VLAN ID VLAN ID	Y	Y
		New fields listed in Table 6 without corresponding existing fields should be set to zero	VLAN priority VLAN priority	Y	Y
		Fields in new headers may be overridden by specifying a "set" action for the appropriate field(s) after the push operation	MPLS label MPLS label	Y	N
			MPLS traffic class MPLS traffic class	N	N
			MPLS TTL MPLS TTL & IP TTL	N	N
6	OpenFlow Channel	Encrypted using TLS		Y	Y
		Directly over TCP		Y	Y
6.1	OpenFlow Protocol Overview	Controller-to-switch message type	Initiated by the controller and used to directly manage or inspect the state of the switch.	Y	Y
		Asynchronous message type	Initiated by the switch and used to update the controller of network events and changes to the switch state.	Y	Y
		Symmetric message type	Initiated by either the switch or the controller and sent without solicitation.	Y	Y
6.1.1	Controller-to-Switch	Features	Request the capabilities of a switch; the switch must respond with a features reply.	Y	Y

		Configuration	Set and query configuration parameters in the switch; switch only responds to a query from the controller.	Y	Y
		Modify-State	Add, delete and modify flow/group entries in the OpenFlow tables and to set switch port properties.	Y	Y
		Read-State	Used by the controller to collect statistics from the switch.	Y	Y
		Packet-out	Used by the controller to send packets out of a specified port on the switch, and to forward packets received via Packet-in messages.	Y	Y
		Barrier	Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations.	Y	Y
6.1.2	Asynchronous	Switches send asynchronous messages to controllers to denote a packet arrival, switch state change, or error			

6.1.3	Symmetric	Sent without solicitation, in either direction			
		Hello	Exchanged between the switch and controller upon connection startup.	Y	Y
		Echo	Sent from either the switch or the controller, and must return an echo reply.	Y	Y
		Experimenter	A standard way for OpenFlow switches to offer additional functionality within the OpenFlow message type space.	Y	Y
6.2	Connection Setup	Establish communication with a controller at a user-configurable (but otherwise fixed) IP address, using a user-specified port	Traffic to and from the OpenFlow channel is not run through the OpenFlow pipeline. Therefore, the switch must identify incoming traffic as local before checking it against the flow tables.	Y	Y
			Each side of the connection must immediately send an OFPT_HELLO message with the version field set to the highest OpenFlow protocol version supported by the sender.	Y	Y
			The recipient may calculate the OpenFlow protocol version to be used as the smaller of the version number that it sent and the one that it received.	Y	Y
			If the negotiated version is not supported by the recipient, the recipient must reply with an OFPT_ERROR message with a type field of OFPET_HELLO_FAILED, a code field of OFPHFC_COMPATIBLE, and then terminate the connection.	Y	Y
			Optionally an ASCII string explaining the situation in data.	Y	Y
6.3	Multiple Controllers	Establish communication with multiple controllers	Controller fail-over.	Y	Y
			Controller load balancing.	N	N
			Switch virtualisation.	Y	Y
			Switch must connect to all controllers it is configured with, and try to maintain connection with all of them concurrently.	Y	Y
			The reply or error messages related to those command must only be sent on the controller connection associated with that command.	Y	Y
			Asynchronous messages may need to be sent to multiple controllers, the message is duplicated for each eligible controller connection and each message sent when the respective controller connection allows it.	Y	Y

		The default role of a controller is OFPCR_ROLE_EQUAL	Controller has full access to the switch and is equal to other controllers in the same role.	Y	Y
			Controller receives all the switch asynchronous messages (such as packet-in, flow-removed).	Y	Y
			The controller can send controller-to-switch commands to modify the state of the switch.	Y	Y
			The switch does not do any arbitration or resource sharing between controllers.	Y	Y
		Controller can request its role to be changed to OFPCR_ROLE_SLAVE	Controller has read-only access to the switch.	Y	Y
			Controller does not receive switch asynchronous messages, apart from Port-status messages.	Y	Y
			The controller is denied ability to send controller-to-switch commands that modify the state of the switch, OFPT_PACKET_OUT, OFPT_FLOW_MOD, OFPT_GROUP_MOD, OFPT_PORT_MOD and OFPT_TABLE_MOD.	Y	Y
			If the controller sends one of those commands, the switch must reply with an OFPT_ERROR message with a type field of OFPET_BAD_REQUEST, a code field of OFPBRC_IS_SLAVE.	Y	Y
			Other controller-to-switch messages, such as OFPT_STATS_REQUEST and OFPT_ROLE_REQUEST, should be processed normally.	Y	Y
		A controller can request its role to be changed to OFPCR_ROLE_MASTER	The switch makes sure there is only a single controller in this role.	Y	Y
			When a controller change its role to OFPCR_ROLE_MASTER, the switch change all other controllers which role is OFPCR_ROLE_MASTER to have the role OFPCR_ROLE_SLAVE.	Y	Y
			When the switch perform such role change, no message is generated to the controller which role is changed (in most case that controller is no longer reachable).	Y	Y
		A switch may be simultaneously connected to multiple controllers in Equal state, multiple controllers in Slave state, and at most one controller in Master state	Each controller may communicate its role to the switch via a OFPT_ROLE_REQUEST message, and the switch must remember the role of each controller connection. A controller may change role at any time.	Y	Y
			To detect out-of-order messages during a master/slave transition, the OFPT_ROLE_REQUEST message contains a 64-bit sequence number field, generation_id, that identifies a given mastership view.	Y	Y

		On receiving a OFPT_ROLE_REQUEST with role equal to OFPCR_ROLE_MASTER or OFPCR_ROLE_SLAVE the switch must compare the generation_id in the message against the largest generation id seen sofar	A message with a generation_id smaller than a previously seen generation id must be considered stale and discarded. The switch must respond to stale messages with an error message with type OFPET_ROLE_REQUEST_FAILED and code OFPRRFC_STALE.	Y	Y
6.4	Connection Interruption	A switch loses contact with all controllers, the switch should immediately enter either "fail secure mode" or "fail standalone mode"	In "fail secure mode", the only change to switch behavior is that packets and messages destined to the controllers are dropped.	Y	Y
			Flows should continue to expire according to their timeouts in "fail secure mode."	Y	Y
			In "fail standalone mode," the switch processes all packets using the OFPP_NORMAL port; in other words, the switch acts as a legacy Ethernet switch or router.	Y	Y
			Upon connecting to a controller again, the existing flow entries remain.	Y	Y
			The controller then has the option of deleting all flow entries, if desired.	Y	Y
		The first time a switch starts up, it will operate in either "fail secure mode" or "fail standalone mode" mode, until it successfully connects to a controller		Y	Y
6.5	Encryption	The switch and controller may communicate through a TLS connection	The TLS connection is initiated by the switch on startup to the controller, which is located by default on TCP port 6633	Y	Y
			Each switch must be user-configurable with one certificate for authenticating the controller (controller certificate) and the other for authenticating to the controller (switch certificate)	Y	Y
6.6	Message Handling	Message Delivery	Messages are guaranteed delivery, unless the connection fails entirely.	Y	Y
		Message Processing	Switches must process every message received from a controller in full, possibly generating a reply.	Y	Y
			If a switch cannot completely process a message received from a controller, it must send back an error message.	Y	Y
			Switches must send to the controller all asynchronous messages generated by internal state changes, such as flow-removed or packet-in messages.	Y	Y

	Message Ordering	Ordering can be ensured through the use of barrier messages	In the absence of barrier messages, switches may arbitrarily reorder messages to maximize performance.	N	N
			Messages must not be reordered across a barrier message and the barrier message must be processed only when all prior messages have been processed.	Y	Y
			Messages before a barrier must be fully processed before the barrier, including sending any resulting replies or errors.	Y	Y
			The barrier must then be processed and a barrier reply sent.	Y	Y
			Messages after the barrier may then begin processing.	Y	Y
6.7	Flow Table Modification Messages	OFPFC_ADD	For add requests (OFPFC_ADD) with the OFPFF_CHECK_OVERLAP flag set, the switch must first check for any overlapping flow entries in the requested table	Y	Y
			If an overlap conflict exists between an existing flow entry and the add request, the switch must refuse the addition and respond with an ofp_error_msg with OFPET_FLOW_MOD_FAILED type and OFPFMFC_OVERLAP code	Y	Y
			If a flow entry with identical match fields and priority already resides in the requested table, then that entry, including its duration, must be cleared from the table, and the newflow entry added	Y	Y
			If the OFPFF_RESET_COUNTS flag is set, the flow entry counters must be cleared, otherwise they should be copied from the replaced flow	N	N
			No flow-removed message is generated for the flow entry eliminated as part of an add request; if the controller wants a flow-removed message it should explicitly send a DELETE STRICT for the old flow prior to adding the new one	Y	Y
			If a switch cannot find any space in the requested table in which to add the incoming flow entry, the switch should send an ofp_error_msg with OFPET_FLOW_MOD_FAILED type and OFPFMFC_TABLE_FULL code.	N	N
			If a matching entry exists in the table, the instructions field of this entry is updated with the value from the request, whereas its cookie, idle_timeout, hard_timeout, flags, counters and duration fields are left unchanged.	Y	Y

		If the OFPFF_RESET_COUNTS flag is set, the flow entry counters must be cleared.	N	N
		If no flow currently residing in the requested table matches the request, no error is recorded, and no flow table modification occurs.	Y	Y
		In the strict versions, the set of match fields, all match fields, including their masks, and the priority, are strictly matched against the entry, and only an identical flow is modified or removed.	Y	Y
		If the match in a flow mod specifies an arbitrary bitmask for another field which the switch cannot support, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and OFPBMC_BAD_MASK code.	Y	Y
		If the match in a flow mod specifies values that cannot be matched, for example, a VLAN ID greater than 4095 and not one of the reserved values, or a DSCP value with one of the two higher bits set, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and OFPBMC_BAD_VALUE code.	Y	Y
		If the match in a flow mod message specifies a field that is unsupported in the table, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and OFPBMC_BAD_FIELD code.	Y	Y
		If the match in a flow mod message specifies a field more than once, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and OFPBMC_DUP_FIELD code.	Y	Y
		If the match in a flow mod message specifies a field but fail to specify its associated prerequisites, for example specifies an IPv4 address without matching the EtherType to 0x800, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and OFPBMC_BAD_PREREQ code.	Y	Y
		If the match in a flow mod specifies an arbitrary bitmask for either the datalink or network addresses which the switch cannot support, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and either OFPBMC_BAD_DL_ADDR_MASK or OFPBMC_BAD_NW_ADDR_MASK.	Y	Y
		If an action in a flow mod message references a group that is not currently defined on the switch, or is a reserved group, such as OFPG_ALL, the switch must return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_BAD_OUT_GROUP code.	Y	Y

		If an action in a flow mod message has a value that is invalid, for example a Set VLAN ID action with value greater than 4095, or a Push action with an invalid Ethertype, the switch should return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_BAD_ARGUMENT code.	Y	Y
		If an action in a flow mod message performs an operation which is inconsistent with the match, for example, a pop VLAN action with a match specifying no VLAN, or a set IPv4 address action with a match wildcarding the Ethertype, the switch may optionally reject the flow and immediately return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_MATCH_INCONSISTENT code.	Y	Y
		If any other errors occur during the processing of the flow mod message, the switch may return an ofp_error_msg with OFPET_FLOW_MOD_FAILED type and OFPFMC_UNKNOWN code.	Y	Y
	OFPFC_DELETE or OFPFC_DELETE_STRICT	If a matching entry exists in the table, it must be deleted.	Y	Y
		If the entry has the OFPFF_SEND_FLOW_REM flag set, it should generate a flow removed message.	Y	Y
		If no flow currently residing in the requested table matches the request, no error is recorded, and no flow table modification occurs.	Y	Y
		In the strict versions, the set of match fields, all match fields, including their masks, and the priority, are strictly matched against the entry, and only an identical flow is modified or removed.	Y	Y
		For non-strict modify and delete commands, all flows that match the flow mod description are modified or removed.	Y	Y
		In the non-strict versions, a match will occur when a flow entry exactly matches or is more specific than the description in the flow mod command; in the flow mod the missing match fields are wildcarded, field masks are active, and other flow mod fields such as priority are ignored.	Y	Y
		Delete commands can be optionally filtered by destination group or output port.	N	N
		If the out_port field contains a value other than OFPP_ANY, it introduces a constraint when matching.	Y	Y
		Modify and delete commands can also be filtered by cookie value.	Y	Y

		Delete commands can use the OFPTT_ALL value for table-id to indicate that matching flows are to be deleted from all flow tables.	Y	Y
		If the flow modification message specifies an invalid table-id, the switch should send an ofp_error_msg with OFPET_FLOW_MOD_FAILED type and OFPFMFC_BAD_TABLE_ID code.	Y	Y
		If the instructions requested in a flow mod message are unknown the switch must return an ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_UNKNOWN_INST code.	Y	Y
		If the instructions requested in a flow mod message are unsupported the switch must return an ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_UNSUP_INST code.	Y	Y
		If the instructions requested contain a Goto-Table and the next-table-id refers to an invalid table the switch must return an ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_BAD_TABLE_ID code.	Y	Y
		If the instructions requested contain a Write-Metadata and the metadata value or metadata mask value is unsupported then the switch must return an ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_UNSUP_METADATA or OFPBIC_UNSUP_METADATA_MASK code.	Y	Y
		If the bitmasks specified in both the datalink and network addresses are not supported then OFPBMC_BAD_DL_ADDR_MASK should be used.	Y	Y
		If any action references a port that will never be valid on a switch, the switch must return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_BAD_OUT_PORT code.	Y	Y
		If the referenced port may be valid in the future, e.g. when a linecard is added to a chassis switch, or a port is dynamically added to a software switch, the switch may either silently drop packets sent to the referenced port, or immediately return an OFPBAC_BAD_OUT_PORT error and refuse the flow mod.	Y	Y
		If an action list contain a sequence of actions that the switch can not support in the specified order, the switch should return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_UNSUPPORTED_ORDER code.	Y	Y

6.8	Flow Removal	Switch flow expiry mechanism	Is run by the switch independently of the controller and is based on the state and configuration of flow entries.	Y	Y
			A non-zero hard_timeout field causes the flow entry to be removed after the given number of seconds, regardless of how many packets it has matched.	Y	Y
			A non-zero idle_timeout field causes the flow entry to be removed when it has matched no packets in the given number of seconds.	Y	Y
			The switch must implement flow expiry and remove flow entries from the flow table when one of their timeout is exceeded.	Y	Y
			When a flow entry is removed, the switch must check the flow entry's OFPFF_SEND_FLOW_Rem flag. If this flag is set, the switch must send a flow removed message to the controller.	Y	Y
			Each flow removed message contains a complete description of the flow entry, the reason for removal (expiry or delete), the flow entry duration at the time of removal, and the flow statistics at time of removal.	Y	Y
6.9	Group Table Modification Messages	OFPGC_ADD	Groups may consist of zero or more buckets.	Y	Y
			A group may also include buckets which themselves forward to other groups if the switch supports it.	Y	Y
			The action set for each bucket must be validated using the same rules as those for flow mods (Section 6.7), with additional group-specific checks.	Y	Y
			If an action in one of the buckets is invalid or unsupported, the switch should return an ofp_error_msg with OFPET_BAD_ACTION type and code corresponding to the error.	Y	Y
			If a group entry with the specified group identifier already resides in the group table, then the switch must refuse to add the group entry and must send an ofp_error_msg with OFPET_GROUP_MOD_FAILED type and OFPGMFC_GROUP_EXISTS code.	Y	Y
			If a specified group type is invalid then the switch must refuse to add the group entry and must send an ofp_error_msg with OFPET_GROUP_MOD_FAILED type and OFPGMFC_INVALID_GROUP code.	Y	Y

		If a switch does not support unequal load sharing with select groups (buckets with weight different than 1), it must refuse to add the group entry and must send an ofp_error_msg with OFPET_GROUP_MOD_FAILED type and OFPGMFC_WEIGHT_UNSUPPORTED code.	Y	Y
		If a switch cannot add the incoming group entry due to lack of space, the switch must send an ofp_error_msg with OFPET_GROUP_MOD_FAILED type and OFPGMFC_OUT_OF_GROUPS code.	Y	Y
		If a switch cannot add the incoming group entry due to restrictions (hardware or otherwise) limiting the number of group buckets, it must refuse to add the group entry and must send an ofp_error_msg with OFPET_GROUP_MOD_FAILED type and OFPGMFC_OUT_OF_BUCKETS code.	Y	Y
		If a switch cannot add the incoming group because it does not support the proposed liveness configuration, the switch must send an ofp_error_msg with OFPET_GROUP_MOD_FAILED type and OFPGMFC_WATCH_UNSUPPORTED code.	N	N
	OFPGC MODIFY	if a group entry with the specified group identifier already resides in the group table, then that entry, including its type and action buckets, must be removed, and the new group entry added.	Y	Y
		If a group entry with the specified group identifier does not already exist then the switch must refuse the group mod and send an ofp_error_msg with OFPET_GROUP_MOD_FAILED type and OFPGMFC_UNKNOWN_GROUP code	Y	Y
	OFPGC_DELETE	If no group entry with the specified group identifier currently exists in the group table, no error is recorded, and no group table modification occurs	Y	Y
		To delete all groups with a single message, specify OFPG_ALL as the group value	Y	Y
	Groups	Groups may be chained if the switch supports it, when at least one group forward to another group, or in more complex configuration	Y	Y
		If a switch does not support groups of groups, it must send an ofp_error_msg with OFPET_GROUP_MOD_FAILED type and OFPGMFC_CHAINING_UNSUPPORTED code.	Y	Y
		A switch may support checking that no loop is created while chaining groups.	Y	Y

			If a group mod is sent such that a forwarding loop would be created, the switch must reject the group mod and must send an ofp_error_msg with OFPET_GROUP_MOD_FAILED type and OFPGMFC_LOOP code.	Y	Y
			A switch may support checking that groups forwarded to by other groups are not removed.	N	N
			If a switch cannot delete a group because it is referenced by another group, it must refuse to delete the group entry and must send an ofp_error_msg with OFPET_GROUP_MOD_FAILED type and OFPGMFC_CHAINED_GROUP code.	Y	Y
A	Appendix A The OpenFlow Protocol				
A.2.1	Port Structures	Port_no field uniquely identifies a port within a switch	Ports are numbered starting from 1.	Y	Y
		Name field is a null-terminated string containing a human-readable name for the interface		Y	Y
		Port administrative settings support the following states	The OFPPC_PORT_DOWN bit indicates that the port has been administratively brought down and should not be used by OpenFlow.	Y	Y
			The OFPPC_NO_RECV bit indicates that packets received on that port should be ignored.	Y	Y
			The OFPPC_NO_FWD bit indicates that OpenFlow should not send packets to that port.	Y	Y
			The OFPPC_NO_FWD bit indicates that OpenFlow should not send packets to that port.	Y	Y
			The OFPPFL_NO_PACKET_IN bit indicates that packets on that port that generate a table miss should never trigger a packet-in message to the controller	Y	Y
			The port config bits are set by the controller and not changed by the switch.		
			If the port config bits are changed by the switch through another administrative interface, the switch sends an OFPT_PORT_STATUS message to notify the controller of the change.	Y	Y
		State field describes the port internal state that supports the following states	OFPPS_LINK_DOWN bit indicates the physical link is not present.	Y	Y
			The OFPPS_BLOCKED bit indicates that a switch protocol outside of OpenFlow, such as 802.1D Spanning Tree, is preventing the use of that port with OFPP_FLOOD.	Y	Y

			OFPPS_LIVE indicates Live for Fast Failover Group.	N	N
			All port state bits are read-only and cannot be changed by the controller.	Y	Y
			When the port flags are changed, the switch sends an OFPT_PORT_STATUS message to notify the controller of the change.	Y	Y
		Curr, advertised, supported, and peer fields indicate link modes (speed and duplexity), link type (copper/fiber) and link features (auto negotiation and pause)		Y	Y
		Curr_speed and max_speed fields indicate the current and maximum bit rate (raw transmission speed) of the link in kbps		Y	Y
A.2.2	Queue Structures	QoS (DSCP & Q mapping?)	An OpenFlow switch provides limited Quality-of-Service support (QoS) through a simple queuing mechanism. One (or more) queues can attach to a port and be used to map flows on it. Flows mapped to a specific queue will be treated according to that queue's configuration.	Y	Y
A.2.3	Flow Match Structures	OpenFlow match is composed of a flow match header and a sequence of zero or more flow match fields	The only valid match type in this specification is OFPMT_OXM, the OpenFlow 1.1 match type OFPMT_STANDARD is deprecated.	Y	Y
			The flow match fields are described using the OpenFlow Extensible Match (OXM) format.	Y	Y
		OpenFlow specification distinguishes two types of OXM match classes	ONF member classes.		
			ONF reserved classes.		
		Flow Match Fields			
		OXM_OF_IN_PORT	/* Switch input port. */	Y	Y
		OXM_OF_IN_PHY_PORT	/* Switch physical input port. */	Y	Y
		OXM_OF_METADATA	/* Metadata passed between tables. */	N	N
		OXM_OF_ETH_DST	/* Ethernet destination address. */	Y	Y
		OXM_OF_ETH_SRC	/* Ethernet source address. */	Y	Y
		OXM_OF_ETH_TYPE	/* Ethernet frame type. */	Y	Y
		OXM_OF_VLAN_VID	/* VLAN id. */	Y	Y
		OXM_OF_VLAN_PCP	/* VLAN priority. */	Y	Y
		OXM_OF_IP_DSCP	/* IP DSCP (6 bits in ToS field). */	Y	Y

		OXM_OF_IP_ECN	/* IP ECN (2 bits in ToS field). */	N	N
		OXM_OF_IP_PROTO	/* IP protocol. */	Y	Y
		OXM_OF_IPV4_SRC	/* IPv4 source address. */	Y	Y
		OXM_OF_IPV4_DST	/* IPv4 destination address. */	Y	Y
		OXM_OF_TCP_SRC	/* TCP source port. */	Y	Y
		OXM_OF_TCP_DST	/* TCP destination port. */	Y	Y
		OXM_OF_UDP_SRC	/* UDP source port. */	Y	Y
		OXM_OF_UDP_DST	/* UDP destination port. */	Y	Y
		OXM_OF_SCTP_SRC	/* SCTP source port. */	N	N
		OXM_OF_SCTP_DST	/* SCTP destination port. */	N	N
		OXM_OF_ICMPV4_TYPE	/* ICMP type. */	N	N
		OXM_OF_ICMPV4_CODE	/* ICMP code. */	N	N
		OXM_OF_ARP_OP	/* ARP opcode. */	Y	Y
		OXM_OF_ARP_SPA	/* ARP source IPv4 address. */	Y	Y
		OXM_OF_ARP_TPA	/* ARP target IPv4 address. */	Y	Y
		OXM_OF_ARP_SHA	/* ARP source hardware address. */	Y	Y
		OXM_OF_ARP_THA	/* ARP target hardware address. */	Y	Y
		OXM_OF_IPV6_SRC	/* IPv6 source address. */	Y	Y
		OXM_OF_IPV6_DST	/* IPv6 destination address. */	Y	Y
		OXM_OF_IPV6_FLABEL	/* IPv6 Flow Label */	N	N
		OXM_OF_ICMPV6_TYPE	/* ICMPv6 type. */	N	N
		OXM_OF_ICMPV6_CODE	/* ICMPv6 code. */	N	N
		OXM_OF_IPV6_ND_TARGET	/* Target address for ND. */	N	N
		OXM_OF_IPV6_ND_SLL	/* Source link-layer for ND. */	N	N
		OXM_OF_IPV6_ND_TLL	/* Target link-layer for ND. */	N	N
		OXM_OF_MPLS_LABEL	/* MPLS label. */	Y	Y
		OXM_OF_MPLS_TC	/* MPLS TC. */	N	N
		Required match fields			

		OXM_OF_IN_PORT	Ingress port. This may be a physical or switch-defined logical port.	Y	Y
		OXM_OF_ETH_DST	Ethernet source address. Can use arbitrary bitmask.	Y	Y
		OXM_OF_ETH_SRC	Ethernet destination address. Can use arbitrary bitmask.	Y	Y
		OXM_OF_ETH_TYPE	Ethernet type of the OpenFlow packet payload, after VLAN tags.	Y	Y
		OXM_OF_IP_PROTO	IPv4 or IPv6 protocol number.	Y	Y
		OXM_OF_IPV4_SRC	IPv4 source address. Can use subnet mask or arbitrary bitmask.	Y	Y
		OXM_OF_IPV4_DST	IPv4 destination address. Can use subnet mask or arbitrary bitmask.	Y	Y
		OXM_OF_IPV6_SRC	IPv6 source address. Can use subnet mask or arbitrary bitmask.	Y	Y
		OXM_OF_IPV6_DST	IPv6 destination address. Can use subnet mask or arbitrary bitmask.	Y	Y
		OXM_OF_TCP_SRC	TCP source port.	Y	Y
		OXM_OF_TCP_DST	TCP destination port.	Y	Y
		OXM_OF_UDP_SRC	UDP source port.	Y	Y
		OXM_OF_UDP_DST	UDP destination port.	Y	Y
A.2.4	Flow Instruction Structures	See Section 5.6			
A.2.5	Action Structures	A number of actions may be associated with flows, groups or packets. The currently defined action types are			
		OFPAT_OUTPUT = 0,	/* Output to switch port. */	Y	Y
		OFPAT_COPY_TTL_OUT = 11,	/* Copy TTL "outwards" – from next-to-outermost to outermost */	N	N
		OFPAT_COPY_TTL_IN = 12,	/* Copy TTL "inwards" – from outermost to next-to-outermost */	N	N
		OFPAT_SET_MPLS_TTL = 15,	/* MPLS TTL */	Y	Y
		OFPAT_DEC_MPLS_TTL = 16,	/* Decrement MPLS TTL */	Y	Y
		OFPAT_PUSH_VLAN = 17,	/* Push a new VLAN tag */	N	N
		OFPAT_POP_VLAN = 18,	/* Pop the outer VLAN tag */	N	N
		OFPAT_PUSH_MPLS = 19,	/* Push a new MPLS tag */	Y	Y

	OFPAT_POP_MPLS = 20,	/* Pop the outer MPLS tag */	Y	Y
	OFPAT_SET_QUEUE = 21,	/* Set queue id when outputting to a port */	Y	Y
	OFPAT_GROUP = 22,	/* Apply group. */	Y	Y
	OFPAT_SET_NW_TTL = 23,	/* IP TTL. */	N	N
	OFPAT_DEC_NW_TTL = 24,	/* Decrement IP TTL. */	N	N
	OFPAT_SET_FIELD = 25,	/* Set a header field using OXM TLV format. */	Y	Y
	OFPAT_EXPERIMENTER = 0xffff		Y	Y
	The type of a set-field action can be any valid OXM header type	OMX types OFPXMT_OFB_IN_PORT and OFPXMT_OFB_METADATA are not supported, because those are not header fields.		
	OXM_OF_IN_PHY_PORT	/* Switch physical input port. */	N	N
	OXM_OF_ETH_DST	/* Ethernet destination address. */	Y	Y
	OXM_OF_ETH_SRC	/* Ethernet source address. */	Y	Y
	OXM_OF_ETH_TYPE	/* Ethernet frame type. */	Y	Y
	OXM_OF_VLAN_VID	/* VLAN id. */	Y	Y
	OXM_OF_VLAN_PCP	/* VLAN priority. */	Y	Y
	OXM_OF_IP_DSCP	/* IP DSCP (6 bits in ToS field). */	Y	Y
	OXM_OF_IP_ECN	/* IP ECN (2 bits in ToS field). */	N	N
	OXM_OF_IP_PROTO	/* IP protocol. */	N	N
	OXM_OF_IPV4_SRC	/* IPv4 source address. */	N	N
	OXM_OF_IPV4_DST	/* IPv4 destination address. */	N	N
	OXM_OF_TCP_SRC	/* TCP source port. */	N	N
	OXM_OF_TCP_DST	/* TCP destination port. */	N	N
	OXM_OF_UDP_SRC	/* UDP source port. */	N	N
	OXM_OF_UDP_DST	/* UDP destination port. */	N	N
	OXM_OF_SCTP_SRC	/* SCTP source port. */	N	N
	OXM_OF_SCTP_DST	/* SCTP destination port. */	N	N
	OXM_OF_ICMPV4_TYPE	/* ICMP type. */	N	N
	OXM_OF_ICMPV4_CODE	/* ICMP code. */	N	N
	OXM_OF_ARP_OP	/* ARP opcode. */	Y	Y

		OXM_OF_ARP_SPA	/* ARP source IPv4 address. */	Y	Y
		OXM_OF_ARP_TPA	/* ARP target IPv4 address. */	Y	Y
		OXM_OF_ARP_SHA	/* ARP source hardware address. */	Y	Y
		OXM_OF_ARP_THA	/* ARP target hardware address. */	Y	Y
		OXM_OF_IPV6_SRC	/* IPv6 source address. */	N	N
		OXM_OF_IPV6_DST	/* IPv6 destination address. */	N	N
		OXM_OF_IPV6_FLABEL	/* IPv6 Flow Label */	N	N
		OXM_OF_ICMPV6_TYPE	/* ICMPv6 type. */	N	N
		OXM_OF_ICMPV6_CODE	/* ICMPv6 code. */	N	N
		OXM_OF_IPV6_ND_TARGET	/* Target address for ND. */	N	N
		OXM_OF_IPV6_ND_SLL	/* Source link-layer for ND. */	N	N
		OXM_OF_IPV6_ND_TLL	/* Target link-layer for ND. */	N	N
		OXM_OF_MPLS_LABEL	/* MPLS label. */	Y	Y
		OXM_OF_MPLS_TC	/* MPLS TC. */	Y	Y
A.3	Controller-to-Switch Messages				
A.3.1	Handshake	Datapath_id	The datapath_id field uniquely identifies a datapath. The lower 48 bits are intended for the switch MAC address, while the top 16 bits are up to the implementer.	Y	Y
			Use datapath_id to distinguish multiple virtual switch instances on a single physical switch.	Y	Y
		Capabilities supported by the datapath	OPPC_FLOW_STATS = 1 << 0, /* Flow statistics. */	Y	Y
			OPPC_TABLE_STATS = 1 << 1, /* Table statistics. */	Y	Y
			OPPC_PORT_STATS = 1 << 2, /* Port statistics. */	Y	Y
			OPPC_GROUP_STATS = 1 << 3, /* Group statistics. */	Y	Y
			OPPC_IP_REASM = 1 << 5, /* Can reassemble IP fragments. */	N	N
			OPPC_QUEUE_STATS = 1 << 6, /* Queue statistics. */	N	N
			OPPC_PORT_BLOCKED = 1 << 8 /* Switch will block looping ports. */	Y	Y

A.3.2	Switch Configuration	Controller is able to set and query configuration parameters in the switch with the OFPT_SET_CONFIG and OFPT_GET_CONFIG_REQUEST messages, respectively		Y	Y
		OFPC_* flags	/* Handling of IP fragments. /OFPC_FRAG_NORMAL = 0, / No special handling for fragments. */	Y	Y
			OFPC_FRAG_DROP = 1 << 0, /* Drop fragments. */	N	N
			OFPC_FRAG_REASM = 1 << 1, /* Reassemble (only if OFPC_IP_REASM set). */	N	N
			OFPC_FRAG_MASK = 3,	N	N
			/* TTL processing - applicable for IP and MPLS packets /OFPC_INVALID_TTL_TO_CONTROLLER = 1 << 2, / Send packets with invalid TTL to the controller */	N	N
		Miss_send_len	Defines the number of bytes of each packet sent to the controller as a result of flow table miss when configured to generate packet-in messages.	Y	Y
			If this field equals 0, the switch must send zero bytes of the packet in the ofp_packet_in message.	Y	Y
			If the value is set to OFPML_NO_BUFFER the complete packet must be included in the message, and should not be buffered.	Y	Y
A.3.3	Flow Table Configuration	Flow tables are numbered from 0 and can take any number until OFPTT_MAX	OFPTT_MAX = 0xfe	Y	Y
		Controller can configure and query table state in the switch with the OFP_TABLE_MOD and OFPST_TABLE_STATS requests, respectively	The switch responds to a table stats request with aOFPT_STATS_REPLY message.	Y	Y
		OFP_TABLE_MOD	If the table_id is OFPTT_ALL, the configuration is applied to all tables in the switch.	Y	Y
		Config field is a bitmap that is used to configure the default behavior of unmatched packets	OFPTC_TABLE_MISS_CONTROLLER = 0, /* Send to controller. */	Y	Y
			OFPTC_TABLE_MISS_CONTINUE = 1 << 0, /* Continue to the next table in the pipeline (OpenFlow 1.0 behavior). */	Y	Y
			OFPTC_TABLE_MISS_DROP = 1 << 1, /* Drop the packet. */	Y	Y

			OFPTC_TABLE_MISS_MASK = 3	Y	Y
A.3.4	Modify State Messages	Modifications to a flow table from the controller are done with the OFPT_FLOW_MOD message		Y	Y
		Modifications to the group table from the controller are done with the OFPT_GROUP_MOD message		Y	Y
		The controller uses the OFPT_PORT_MOD message to modify the behavior of the port		Y	Y
A.3.5	Read State Messages	While the system is running, the datapath may be queried about its current state using the OFPT_STATS_REQUEST message	<pre>/* Description of this OpenFlow switch.* The request body is empty.* The reply body is struct ofp_desc_stats. /OFPST_DESC = 0, / Body of reply to OFPST_DESC request. Each entry is a NULL- terminated ASCII string. /struct ofp_desc_stats {char mfr_desc[DESC_STR_LEN]; / Manufacturer description. /char hw_desc[DESC_STR_LEN]; / Hardware description. /char sw_desc[DESC_STR_LEN]; / Software description. /char serial_num[SERIAL_NUM_LEN]; / Serial number. /char dp_desc[DESC_STR_LEN]; / Human readable description of datapath. */;</pre>	Y	Y
			<pre>/* Individual flow statistics. The request body is struct ofp_flow_stats_request. The reply body is an array of struct ofp_flow_stats. /OFPST_FLOW = 1, / Body of reply to OFPST_FLOW request. /struct ofp_flow_stats {uint16_t length; / Length of this entry. /uint8_t table_id; / ID of table flow came from. /uint8_t pad; uint32_t duration_sec; / Time flow has been alive in seconds. /uint32_t duration_nsec; / Time flow has been alive in nanoseconds beyond duration_sec. /uint16_t priority; / Priority of the entry. /uint16_t idle_timeout; / Number of seconds idle before expiration. /uint16_t hard_timeout; / Number of seconds before expiration. /uint8_t pad2[6]; / Align to 64-bits. /uint64_t cookie; / Opaque controller-issued identifier. /uint64_t packet_count; / Number of packets in flow. /uint64_t byte_count; / Number of bytes in flow. /struct ofp_match match; / Description of fields. Variable size. //struct ofp_instruction instructions[0]; / Instruction set. */;</pre>	Y	Y

		<pre>/* Aggregate flow statistics. The request body is struct ofp_aggregate_stats_request. The reply body is struct ofp_aggregate_stats_reply. /OFPST_AGGREGATE = 2, / Body of reply to OFPST_AGGREGATE request. /struct ofp_aggregate_stats_reply {uint64_t packet_count; / Number of packets in flows. /uint64_t byte_count; / Number of bytes in flows. /uint32_t flow_count; / Number of flows. /uint8_t pad[4]; / Align to 64 bits. */};</pre>	Y	Y
		<pre>/* Flow table statistics. The request body is empty. The reply body is an array of struct ofp_table_stats. /OFPST_TABLE = 3, / Body of reply to OFPST_TABLE request. /struct ofp_table_stats {uint8_t table_id; / Identifier of table. Lower numbered tables are consulted first. /uint8_t pad[7]; / Align to 64-bits. /char name[OFP_MAX_TABLE_NAME_LEN];uint64_t match; / Bitmap of (1 << OFPXMT_) that indicate the fields the table can match on. */uint64_t wildcards; / Bitmap of (1 << OFPXMT_) wildcards that are supported by the table. */uint32_t write_actions; / Bitmap of OFPAT_* that are supported by the table with OFPIT_WRITE_ACTIONS. /uint32_t apply_actions; / Bitmap of OFPAT_* that are supported by the table with OFPIT_APPLY_ACTIONS. /uint64_t write_setfields;/ Bitmap of (1 << OFPXMT_) header fields that can be set with OFPIT_WRITE_ACTIONS. */uint64_t apply_setfields;/ Bitmap of (1 << OFPXMT_) header fields that can be set with OFPIT_APPLY_ACTIONS. */uint64_t metadata_match; / Bits of metadata table can match. /uint64_t metadata_write; / Bits of metadata table can write. /uint32_t instructions; / Bitmap of OFPIT_* values supported. /uint32_t config; / Bitmap of OFPTC_* values /uint32_t max_entries; / Max number of entries supported. /uint32_t active_count; / Number of active entries. /uint64_t lookup_count; / Number of packets looked up in table. /uint64_t matched_count; / Number of packets that hit table. */};</pre>	Y	Y

		<pre>/* Port statistics. The request body is struct ofp_port_stats_request. The reply body is an array of struct ofp_port_stats. /OFPST_PORT = 4, / Body of reply to OFPST_PORT request. If a counter is unsupported, set the field to all ones. /struct ofp_port_stats {uint32_t port_no;uint8_t pad[4]; / Align to 64-bits. /uint64_t rx_packets; / Number of received packets. /uint64_t tx_packets; / Number of transmitted packets. /uint64_t rx_bytes; / Number of received bytes. /uint64_t tx_bytes; / Number of transmitted bytes. /uint64_t rx_dropped; / Number of packets dropped by RX. /uint64_t tx_dropped; / Number of packets dropped by TX. /uint64_t rx_errors; / Number of receive errors. This is a super-set of more specific receive errors and should be greater than or equal to the sum of all rx_err values. */uint64_t tx_errors; / Number of transmit errors. This is a super-set of more specific transmit errors and should be greater than or equal to the sum of all tx_err values (none currently defined.) */uint64_t rx_frame_err; / Number of frame alignment errors. /uint64_t rx_over_err; / Number of packets with RX overrun. /uint64_t rx_crc_err; / Number of CRC errors. /uint64_t collisions; / Number of collisions. */};</pre>	Y	Y
		<pre>/* Queue statistics for a port The request body is struct ofp_queue_stats_request. The reply body is an array of struct ofp_queue_stats /OFPST_QUEUE = 5, The body of the reply consists of an array of the following structure:/struct ofp_queue_stats {uint32_t port_no;uint32_t queue_id; / Queue i.d /uint64_t tx_bytes; / Number of transmitted bytes. /uint64_t tx_packets; / Number of transmitted packets. /uint64_t tx_errors; / Number of packets dropped due to overrun. */};</pre>	N	N
		<pre>/* Group features. The request body is empty. The reply body is struct ofp_group_features_stats. /OFPST_GROUP_FEATURES = 8, / Body of reply to OFPST_GROUP_FEATURES request. Group features. /struct ofp_group_features_stats {uint32_t types; / Bitmap of OFPGT_* values supported. /uint32_t capabilities; / Bitmap of OFPGFC_* capability supported. /uint32_t max_groups[4]; / Maximum number of groups for each type. /uint32_t actions[4]; / Bitmaps of OFPAT_* that are supported. */};</pre>	Y	Y

			<pre>/* Experimenter extension. The request and reply bodies begin with* struct ofp_experimenter_stats_header. The request and reply bodies are otherwise experimenter-defined. /OFPST_EXPERIMENTER = 0xffff / Body for ofp_stats_request/reply of type OFPST_EXPERIMENTER. /struct ofp_experimenter_stats_header {uint32_t experimenter; / Experimenter ID which takes the same form as in struct ofp_experimenter_header. /uint32_t exp_type; / Experimenter defined. // Experimenter-defined arbitrary additional data. */;</pre>	Y	Y
A.3.6	Queue Configuration Messages	Queue configuration takes place outside the OpenFlow protocol, either through a command line tool	CLI support	Y	Y
		The switch replies back with an ofp_queue_get_config_reply command, containing a list of configured queues	<pre>/* Queue configuration for a given port. /struct ofp_queue_get_config_reply {struct ofp_header header;uint32_t port;uint8_t pad[4];struct ofp_packet_queue queues[0]; / List of configured queues. */;</pre>	N	N
A.3.7	Packet-Out Message	When the controller wishes to send a packet out through the datapath, it uses the OFPT_PACKET_OUT message		Y	Y
A.3.8	Barrier Message	When the controller wants to ensure message dependencies have been met or wants to receive notifications for completed operations, it may use an OFPT_BARRIER_REQUEST message	Upon receipt, the switch must finish processing all previously-received messages, including sending corresponding reply or error messages, before executing any messages beyond the Barrier RequestRequest. When such processing is complete, the switch must send an OFPT_BARRIER_REPLY message with the xid of the original request.	Y	Y
A.3.9	Role Request Message	When the controller wants to change its role, it uses the OFPT_ROLE_REQUEST message and can have the following values	OFPCR_ROLE_NOCHANGE = 0, /* Don't change current role. */	Y	Y
			OFPCR_ROLE_EQUAL = 1, /* Default role, full access. */	Y	Y
			OFPCR_ROLE_MASTER = 2, /* Full access, at most one master. */	Y	Y
			OFPCR_ROLE_SLAVE = 3, /* Read-only access. */	Y	Y
A.4	Asynchronous Messages				

A.4.1	Packet-In Message	Switches that implement buffering are expected to expose, through documentation, both the amount of available buffering, and the length of time before buffers may be reused	A switch should prevent a buffer from being reused until it has been handled by the controller, or some amount of time (indicated in documentation) has passed.	Y	Y
A.4.2	Flow Removed Message	If the controller has requested to be notified when flows time out or are deleted from tables, the datapath does this with the OFPT_FLOW_REMOVED message	The reason field is one of the following: OFPRR_IDLE_TIMEOUT = 0, /* Flow idle time exceeded idle_timeout. /OFPRR_HARD_TIMEOUT = 1, / Time exceeded hard_timeout. /OFPRR_DELETE = 2, / Evicted by a DELETE flow mod. /OFPRR_GROUP_DELETE = 3, / Group was removed. */	Y	Y
A.4.3	Port Status Message	As ports are added, modified, and removed from the datapath, the controller needs to be informed with the OFPT_PORT_STATUS message	The status can be one of the following values: OFPPR_ADD = 0, /* The port was added. /OFPPR_DELETE = 1, / The port was removed. /OFPPR_MODIFY = 2, / Some attribute of the port has changed. */	Y	Y
A.4.4	Error Message	There are times that the switch needs to notify the controller of a problem. This is done with the OFPT_ERROR_MSG message	Currently defined error types are: OFPET_HELLO_FAILED = 0, /* Hello protocol failed. /OFPET_BAD_REQUEST = 1, / Request was not understood. /OFPET_BAD_ACTION = 2, / Error in action description. /OFPET_BAD_INSTRUCTION = 3, / Error in instruction list. /OFPET_BAD_MATCH = 4, / Error in match. /OFPET_FLOW_MOD_FAILED = 5, / Problem modifying flow entry. /OFPET_GROUP_MOD_FAILED = 6, / Problem modifying group entry. /OFPET_PORT_MOD_FAILED = 7, / Port mod request failed. /OFPET_TABLE_MOD_FAILED = 8, / Table mod request failed. /OFPET_QUEUE_OP_FAILED = 9, / Queue operation failed. /OFPET_SWITCH_CONFIG_FAILED = 10, / Switch config request failed. /OFPET_ROLE_REQUEST_FAILED = 11, / Controller Role request failed. /OFPET_EXPERIMENTER = 0xffff / Experimenter error messages. */	Y	Y
A.5	Symmetric Messages	See Section 6.1.3		Y	Y
B	Appendix B Release Notes				
B.6.6	Vendor Extensions	Vendors are now able to add their own extensions, while still being OpenFlow compliant. The primary way to do this is with the new OFPT_VENDOR message type		N	N

B.6.8	802.1D Spanning Tree	A switch that implements STP must set the new OFPC_STP bit in the 'capabilities' field of its OFPT_FEATURES_REPLY message.	A switch that implements STP at all must make it available on all of its physical ports, but it need not implement it on virtual ports (e.g. OFPP_LOCAL)	Y	Y
			The complete set of port configuration flags are: OFPPC_PORT_DOWN = 1 << 0, /* Port is administratively down. */ OFPPC_NO_STP = 1 << 1 , / Disable 802.1D spanning tree on port. OFPPC_NO_RECV = 1 << 2 , / Drop most packets received on port. OFPPC_NO_RECV_STP = 1 << 3 , / Drop received 802.1D STP packets. OFPPC_NO_FLOOD = 1 << 4 , / Do not include this port when flooding. OFPPC_NO_FWD = 1 << 5 , / Drop packets forwarded to port. OFPPC_NO_PACKET_IN = 1 << 6 / Do not send packet-in msgs for port. */	Y	Y
		Packets received on ports that are disabled by spanning tree must follow the normal flow table processing path		Y	Y
B.6.21	Behavior Defined When Controller Connection Lost	In the case that the switch loses contact with the controller, the default behavior must be to do nothing - to let flows timeout naturally. Other behaviors can be implemented via vendor-specific command line interface or vendor extension OpenFlow messages	Default behavior supported.	Y	Y
B.7.1	Failover	Switch can be configured with a list of controllers. If the first controller fails, it will automatically switch over to the second controller on the list		Y	Y
B.7.2	Emergency Flow Cache	The protocol and reference implementation have been extended to allow insertion and management of emergency flow entries. Emergency-specific flow entries are inactive until a switch loses connectivity from the controller	The switch invalidates all normal flow table entries and copies all emergency flows into the normal flow table. Upon connecting to a controller again, all entries in the flow cache stay active. The controller then has the option of resetting the flow cache if needed.	N	N
B.7.9	Rewrite DSCP in IP ToS header	Added Flow action to rewrite the DiffServ CodePoint bits part of the IP ToS field in the IP header.	This enables basic support for basic QoS with OpenFlow in some switches.	Y	Y
B.8.1	Slicing	OpenFlow now supports multiple queues per output port. Queues support the ability to provide minimum bandwidth guarantees	The bandwidth allocated to each queue is configurable.	Y	Y
B.9	OpenFlow version 1.1				

B.9.1	Multiple Tables	The switch now expose a pipeline with multiple tables.		Y	Y
		Flow entry have instruction to control pipeline processing		Y	Y
		Controller can choose packet traversal of tables via goto instruction		Y	Y
		Metadata field (64 bits) can be set and match in tables		N	N
		Packet actions can be merged in packet action set		N	N
		Packet action set is executed at the end of pipeline		Y	Y
		Packet actions can be applied between table stages		N	N
		Table miss can send to controller, continue to next table or drop	To controller only.	N	N
		Rudimentary table capability and configuration		N	N
B.9.2	Groups	Group indirection to represent a set of ports		Y	Y
		Group table with 4 types of groups :	All - used for multicast and flooding.	Y	Y
			Select - used for multipath.	N	N
			Indirect - simple indirection.	Y	Y
			Fast Failover - use first live port.	N	N
		Group action to direct a flow to a group		Y	Y
B.9.3	Tags: MPLS & VLAN	Support for VLAN and QinQ, adding, modifying and removing VLAN headers		N	N
		Support for MPLS, adding, modifying and removing MPLS shim headers		Y	Y
B.9.4	Virtual ports	Make port number 32 bits, enable larger number of ports	GRE & LAG	Y	Y
		Enable switch to provide virtual port as OpenFlow ports		Y	Y
		Augment packet-in to report both virtual and physical ports		N	N

B.9.5	Other changes	Remove 802.1d-specific text from spec		N	N
		Remove Emergency Flow Cache from spec		N	N
		Cookie Enhancements Proposal		N	N
		Set queue action (unbundled from output port)		N	N
		Maskable DL and NW address match fields		Y	Y
		Add TTL decrement, set and copy actions for IPv4 and MPLS		N	N
		SCTP header matching and rewriting support		N	N
		Set ECN action		N	N
		Connection interruption trigger fail secure or fail standalone mode		Y	Y
		Define message handling : no loss, may reorder if no barrier		N	N
		Rename VENDOR APIs to EXPERIMENTER APIs			
B.10	OpenFlow version 1.2				
B.10.1	Extensible match support	The Extensible set_field action reuses the OXM encoding defined for matches, and enables to rewrite any header field in a single action (EXT-13)	Deprecate most header rewrite actions.	N	N
			Introduce generic set-field action (EXT-13).	N	N
			Reuse match TLV structure (OXM) in set-field action.	N	N
B.10.2	Extensible 'set field' packet rewriting support	Rather than introduce a hard coded field in the packet-in message, the flexible OXM encoding is used to carry packet context	Reuse match TLV structure (OXM) to describe metadata in packet-in (EXT-6).	N	N
			Include the 'metadata' field in packet-in.	N	N
			Move ingress port and physical port from static field to OXM encoding.	N	N
			Allow to optionally include packet header fields in TLV structure.	N	N

B.10.3	Extensible context expression in 'packet-in'	Rather than introduce a hard coded field in the packet-in message, the flexible OXM encoding is used to carry packet context	Reuse match TLV structure (OXM) to describe metadata in packet-in (EXT-6).	Y	Y
			Include the 'metadata' field in packet-in.	Y	Y
			Move ingress port and physical port from static field to OXM encoding.	Y	Y
			Allow to optionally include packet header fields in TLV structure.	Y	Y
B.10.4	Extensible Error messages via experimenter error type	An experimenter error code has been added, enabling experimenter functionality to generate custom error messages (EXT-2). The format is identical to other experimenter APIs		N	N
B.10.5	IPv6 support added	Basic support for IPv6 match and header rewrite has been added	Added support for matching on IPv6 source address, destination address, protocol number, traffic class, ICMPv6 type, ICMPv6 code and IPv6 neighbor discovery header fields (EXT-1).	Y	Y
			Added support for matching on IPv6 flow label (EXT-36).	N	N
B.10.6	Simplified behaviour of flow-mod request	The behaviour of flow-mod request has been simplified (EXT-30)	MODIFY and MODIFY STRICT commands never insert new flows in the table.	N	N
			New flag OFPFF RESET COUNTS to control counter reset.	N	N
			Remove quirky behaviour for cookie field.	Y	Y
B.10.7	Removed packet parsing specification	The match fields are only defined logically	OpenFlow does not mandate how to parse packets.	N	N
			Parsing consistency achieved via OXM pre-requisite.	N	N
B.10.8	Controller role change mechanism	The controller role change mechanism is a simple mechanism to support multiple controllers for failover (EXT-39)	The switch only need to remember the role of each controller to help the controller election mechanism.	N	N
			Simple mechanism to support multiple controllers for failover.	Y	Y
			Switches may now connect to multiple controllers in parallel.	Y	Y

			Enable each controller to change its roles to equal, master or slave.	Y	Y
B.10.9	Other changes	Per-table metadata bitmask capabilities (EXT-34)		Y	Y
		Rudimentary group capabilities (EXT-61)		N	N
		Add hard timeout info in flow-removed messages (OFP-283)		Y	Y
		Add ability for controller to detect STP support(OFP-285)		Y	Y
		Turn off packet buffering with OFPCML NO BUFFER (EXT-45)		Y	Y
		Added ability to query all queues (EXT-15)		N	N
		Added experimenter queue property (EXT-16)		Y	Y
		Added max-rate queue property (EXT-21)		Y	Y
		Enable deleting flow in all tables (EXT-10)		Y	Y
		Enable switch to check chaining when deleting groups (EXT-12)		N	N
		Enable controller to disable buffering (EXT-45)		Y	Y
		Virtual ports renamed logical ports (EXT-78)		N	N
		New error messages (EXT-1, EXT-2, EXT-12, EXT-13, EXT-39, EXT-74 and EXT-82)		Y	Y
		Include release notes into the specification document		Y	Y
		Many other bug fixes, rewording and clarifications		Y	Y
	OpenFlow 1.3				
	Per flow meters	Flexible meter framework based on per-flow meters and meter bands.			
		Meter statistics, including per band statistics.	1 band per meter	Y	Y
		Enable to attach meters flexibly to flow entries.		Y	Y
		^Simple rate-limiter support (drop packets		Y	Y

	Per connection event filtering	Add asynchronous message filter for each controller connection		Y	Y
	Per connection event filtering	Add asynchronous message filter for each controller connection			
		Set default Iter value to match OpenFlow 1.2 behaviour		Y	Y
		Remove OFPC_INVALID_TTL_TO_CONTROLLER config flag			
	Auxiliary connections	Auxiliary connections are mostly useful to carry packet-in and packet-out messages		N	N
	MPLS BoS matching	match the Bottom of Stack bit (BoS) from the MPLS header (EXT-85). The BoS bit indicates if other MPLS shim header are in the payload of the present MPLS packet, and matching this bit can help to disambiguate case where the MPLS label is present MPLS packet, and matching this bit can help to disambiguate case where the MPLS label is reused across levels of MPLS encapsulation		N	N
	Provider Backbone Bridging tagging	Push and Pop operation to add PBB header as a tag. [^]		Y	Y
		New OXM field to match I-SID for the PBB header	PBB-MPLS-VLAN order	N	N
	Rework tag order	the final order of tags in a packet is dictated by the order of the tagging operations, each tagging operation adds its tag in the outermost position	Remove defined order of tags in packet from the specification.	N	N
			Tags are now always added in the outermost possible position.	Y	Y
			Action-list can add tags in arbitrary order.	N	N
			^Tag order is predefined for tagging in the action-set.	Y	Y
	Tunnel-ID metadata	A new OXM field that expose to the OpenFlow pipeline metadata associated with the logical port, most commonly the demultiplexing id from the encapsulation header	If the logical port perform GRE encapsulation, the tunnel-id id would map to the GRE key field from the GRE header. be able to match the GRE key in the tunnel-id match id.	N	N
	Cookies in packet-in				
	Duration for stats	Duration field was added to most statistics, including port statistics, group statistics, queue statistics and meter statistics		Y	Y

	On demand flow counters	New flow-mod flags have been added to disable packet and byte counters on a per-flow basis		N	N
--	-------------------------	--	--	---	---

PicOS Support for OpenFlow 1.3.0

This document contains OpenFlow 1.3.0 features supported by the Pica8 PicOS software. For clarity, the feature names in this table are identical to the feature names found in OpenFlow Switch Specification Version 1.3.0.

OpenFlow Messages

Each OpenFlow message begins with the OpenFlow header. The OpenFlow header has several fields, including the **type** field. The **type** field identifies the type of OpenFlow message.

The OpenFlow protocol has three message types: *symmetric*, *controller-to-switch*, *asynchronous*. Each message type has multiple sub-types.

Symmetric Messages

Symmetric messages are unsolicited messages that may be initiated by either the switch or the controller. Symmetric messages are sent without explicit solicitation, in either direction. The following table describes PicOS support for symmetric messages:

Table 1 PicOS Support for OpenFlow Symmetric Messages

Message	Support	Comments
OFPT_HELLO	Supported	
OFPT_ERROR	Supported	
OFPT_ECHO_REQUEST	Supported	
OFPT_ECHO_REPLY	Supported	
OFPT_EXPERIMENTER	Not Supported	

Controller-to-Switch Messages

Controller-to-switch messages are sent from the controller to the switch. These messages are used to directly manage the state of a switch. Controller-to-switch messages may or may not require a response from the switch.

The following table describes PicOS support for controller-to-switch messages:

Table 2 PicOS Support for OpenFlow Controller-to-Switch Messages

Message	Support	Comments
OFPT_FEATURES_REQUEST	Supported	

Message	Support	Comments
OFPT_FEATURES_REPLY See Capabilities Supported by Datapath		
OFPT_GET_CONFIG_REQUEST	Supported	
OFPT_GET_CONFIG_REPLY		
OFPT_SET_CONFIG	Supported	
OFPT_PACKET_OUT	Supported	
OFPT_FLOW_MOD	Supported	
OFPT_GROUP_MOD	Supported	
OFPT_PORT_MOD	Supported	
OFPT_TABLE_MOD	Supported	
OFPT_MULTIPART_REQUEST See Multipart Messages	Supported	
OFPT_MULTIPART_REPLY See Multipart Messages	Supported	
OFPT_BARRIER_REQUEST	Supported	
OFPT_BARRIER_REPLY		
OFPT_QUEUE_GET_CONFIG_REQUEST	Supported	
OFPT_QUEUE_GET_CONFIG_REPLY	Supported	
OFPT_ROLE_REQUEST	Supported	
OFPT_ROLE_REPLY		
OFPT_GET_ASYNC_REQUEST	Supported	
OFPT_GET_ASYNC_REPLY	Supported	
OFPT_SET_ASYNC	Supported	
OFPT_METER_MOD	Supported	

Asynchronous Messages

Asynchronous messages are sent from the switch to the controller. These messages are used to communicate network events and switch state changes to the controller. Asynchronous messages are sent without a controller explicitly requesting them from a switch.

The following table describes PicOS support for asynchronous messages:

Table 3 PicOS Support for OpenFlow Asynchronous Messages

Message	Support	Comments
OFPT_PACKET_IN	Supported	
OFPT_FLOW_REMOVED	Supported	
OFPT_PORT_STATUS	Supported	

Multipart Messages

The following table describes PicOS support for multipart messages: a single OpenFlow message cannot be larger than 64 kilobytes. Multipart messages are used to encode requests or replies that would carry a large amount of data, and would not always fit in a single OpenFlow message. The sender encodes the request or reply as a sequence of multipart messages, with a specific multipart type. The receiver re-assembles the request or reply.

Table 4 PicOS Support for OpenFlow Multipart Messages

Message	Support	Comments
OFPMP_DESC	Supported	
OFPMP_FLOW	Supported	
OFPMP_AGGREGATE		
OFPMP_TABLE	Supported	
OFPMP_PORT_STATS	Supported	
OFPMP_QUEUE	Supported	
OFPMP_GROUP	Supported	
OFPMP_GROUP_DESC	Supported	
OFPMP_GROUP_FEATURES	Not Supported	
OFPMP_METER	Supported	
OFPMP_METER_CONFIG	Supported	
OFPMP_METER_FEATURES	Supported	
OFPMP_TABLE_FEATURES	Supported	
OFPMT_PORT_DESC	Supported	
OFPMP_EXPERIMENTER	Not Supported	

Capabilities Supported by Datapath

The controller sends an **OFPT_FEATURES_REQUEST** message to the switch, once a session is established. The switch responds to the **OFPT_FEATURES_REQUEST** message with an **OFPT_FEATURES_REPLY** message. The **OFPT_FEATURES_REPLY** message has several fields, including the **capabilities** field. The **capabilities** field identifies the capabilities supported by the switch datapath.

The **capabilities** field is a combination of the following bits/flags:

Table 5 PicOS Support for OpenFlow Capabilities Supported by Datapath

Specification	Support	Comments
OFPC_FLOW_STATS	Supported	
OFPC_TABLE_STATS	Supported	
OFPC_PORT_STATS	Supported	
OFPC_GROUP_STATS	Supported	
OFPC_IP_REASM		
OFPC_QUEUE_STATS		
OFPC_PORT_BLOCKED		

OpenFlow Ports

OpenFlow ports are abstract network interfaces used for passing traffic between the OpenFlow switch and the rest of the network. An OpenFlow switch makes a number of OpenFlow ports available. The set of OpenFlow ports does not have to be identical to the set of physical network interfaces on the switch hardware.

Port Descriptions

The OpenFlow switch receives and sends packets on OpenFlow ports. The switch may define physical and logical ports, and the OpenFlow specification also defines some reserved ports.

Table 6 PicOS Support for OpenFlow Ports

Specification	Support	Comments
OFPP_MAX		
OFPP_IN_PORT	Supported	
OFPP_TABLE	Supported	
OFPP_NORMAL	Supported	
OFPP_FLOOD	Supported	

OFPP_ALL	Supported	
OFPP_CONTROLLER	Supported	
OFPP_LOCAL	Supported	
OFPP_ANY	Supported	

Port Administrative Settings

The following table describes PicOS support for administrative settings of OpenFlow ports:

Table 7 PicOS Support for OpenFlow Port Administrative Settings

Specification	Support	Comments
OFPPC_PORT_DOWN	Supported	
OFPPC_NO_STP	Supported	
OFPPC_NO_RECV	Supported	
OFPPC_NO_RECV_STP	Supported	
OFPPC_NO_FLOOD		
OFPPC_NO_FWD		
OFPPC_NO_PACKET_IN		

Port States

The following table describes PicOS support for OpenFlow port states:

Table 8 PicOS Support for OpenFlow Port States

Specification	Support	Comments
OFPPS_LINK_DOWN	Supported	
OFPPS_BLOCKED		
OFPPS_LIVE		

Port Features

The following table describes PicOS support for OpenFlow port features available in the datapath:

Table 9 PicOS Support for OpenFlow Port Features

Specification	Support	Comments
OFPPF_10MB_HD	Supported	
OFPPF_10MB_FD	Supported	
OFPPF_100MB_HD	Supported	
OFPPF_100MB_FD	Supported	
OFPPF_1GB_HD	Supported	
OFPPF_1GB_FD	Supported	
OFPPF_10GB_FD	Supported	
OFPPF_40GB_FD	Supported	
OFPPF_100GB_FD	Supported	
OFPPF_1TB_FD		
OFPPF_OTHER		
OFPPF_COPPER	Supported	
OFPPF_FIBER	Supported	
OFPPF_AUTONEG	Supported	
OFPPF_PAUSE	Supported	
OFPPF_PAUSE_ASYM		

OpenFlow Instructions

Each flow entry contains a set of instructions that are executed when a packets matches the entry. The following table details OpenFlow instructions supported by PicOS:

Table 10 PicOS Support for OpenFlow Instructions

Specification	Support	Comments
OFPIT_GOTO_TABLE	Supported	
OFPIT_WRITE_METADATA	Supported	
OFPIT_WRITE_ACTIONS	Supported	
OFPIT_APPLY_ACTIONS	Supported	
OFPIT_CLEAR_ACTIONS	Supported	
OFPIT_METER	Supported	
OFPIT_EXPERIMENTER		

OpenFlow Action Types

The following table details OpenFlow action types supported by PicOS:

Table 11 PicOS Support for OpenFlow Action Types

Specification	Support	Comments
OFPAT_OUTPUT	Supported	
OFPAT_COPY_TTL_OUT	Supported	
OFPAT_COPY_TTL_IN	Supported	
OFPAT_SET_MPLS_TTL	Supported	
OFPAT_DEC_MPLS_TTL	Not Supported	
OFPAT_PUSH_VLAN	Supported	
OFPAT_POP_VLAN	Supported	
OFPAT_PUSH_MPLS	Supported	
OFPAT_POP_MPLS	Supported	
OFPAT_SET_QUEUE	Supported	
OFPAT_GROUP	Supported	
OFPAT_SET_NW_TTL	Not Supported	
OFPAT_DEC_NW_TTL	Not Supported	
OFPAT_SET_FIELD	Supported	
OFPAT_PUSH_PBB	Supported	
OFPAT_POP_PBB	Supported	
OFPAT_EXPERIMENTER	Not Supported	

OpenFlow Match Fields

A match field may include the packet header, the ingress port, and the metadata value. A match field may use wildcards to match any value, and in some cases bitmasks. A packet is matched against a match field.

The following table details the OXM flow match field types supported by PicOS:

Table 12 PicOS Support for OpenFlow Flow Match Fields

Specification	Support	Comments
OFPXMT_OFB_IN_PORT	Supported	

Specification	Support	Comments
OFPXMT_OFB_IN_PHY_PORT	Supported	
OFPXMT_OFB_METADATA		
OFPXMT_OFB_ETH_DST	Supported	
OFPXMT_OFB_ETH_SRC	Supported	
OFPXMT_OFB_ETH_TYPE	Supported	
OFPXMT_OFB_VLAN_VID	Supported	
OFPXMT_OFB_VLAN_PCP	Supported	
OFPXMT_OFB_IP_DSCP	Supported	
OFPXMT_OFB_IP_ECN		
OFPXMT_OFB_IP_PROTO	Supported	
OFPXMT_OFB_IPV4_SRC	Supported	
OFPXMT_OFB_IPV4_DST	Supported	
OFPXMT_OFB_TCP_SRC	Supported	
OFPXMT_OFB_TCP_DST	Supported	
OFPXMT_OFB_UDP_SRC	Supported	
OFPXMT_OFB_UDP_DST	Supported	
OFPXMT_OFB_SCTP_SRC		
OFPXMT_OFB_SCTP_DST		
OFPXMT_OFB_ICMPV4_TYPE	Supported	
OFPXMT_OFB_ICMPV4_CODE	Supported	
OFPXMT_OFB_ARP_OP	Supported	
OFPXMT_OFB_ARP_SPA	Supported	
OFPXMT_OFB_ARP_TPA	Supported	
OFPXMT_OFB_ARP_SHA		
OFPXMT_OFB_ARP_THA		
OFPXMT_OFB_IPV6_SRC	Supported	
OFPXMT_OFB_IPV6_DST	Supported	
OFPXMT_OFB_IPV6_FLABEL		
OFPXMT_OFB_ICMPV6_TYPE		

Specification	Support	Comments
OFPXMT_OFB_ICMPV6_CODE		
OFPXMT_OFB_IPV6_ND_TARGET		
OFPXMT_OFB_IPV6_ND_SLL		
OFPXMT_OFB_IPV6_ND_TLL		
OFPXMT_OFB_MPLS_LABEL	Supported	
OFPXMT_OFB_MPLS_TC	Supported	
OFPXMT_OFP_MPLS_BOS		
OFPXMT_OFB_PBB_ISID	Supported	
OFPXMT_OFB_TUNNEL_ID	Supported	
OFPXMT_OFB_IPV6_EXTHDR		

OpenFlow Group Types

The following table describes PicOS support for OpenFlow group types:

Table 13 PicOS Support for OpenFlow Group Types

Specification	Support	Comments
OFGT_ALL	Supported	
OFGT_SELECT	Supported	
OFGT_INDIRECT	Supported	
OFGT_FF	Supported	

PicOS Support for OpenFlow 1.4

The following table contains OpenFlow 1.4 features supported by PicOS. For clarity, the feature names in this table are identical to the feature names found in OpenFlow Switch Specification Version 1.4.0.

Table 1 OpenFlow 1.4 Features Supported by PicOS

Pica8 OpenFlow V1.4 Compliance Matrix						
Chapter	Title	Features	Detail Feature Specification	Optional	R2.3 Support	Remarks
2	Switch Components	NA				
		Flow table			Y	
		Group table			Y	

			All, indirect, select, fast_failover group table are all supported.		
		Add/update/delete flow entries		Y	
		Match fields		Y	
		Counters		Y	
		Set of instructions		Y	
4	OpenFlow Ports	NA			
4.1	OpenFlow Ports	see 4.2-4.5		Y	
4.2	Standard Ports	See 4.2-4.5		Y	
4.3	Physical Ports	NA			
		Ingress	PicOS only supports it as matching port.	Y	
		Output		Y	
		Hardware interface		Y	
		Groups		Y	
		Port counters		Y	
4.4	Logical Ports	NA	The OpenFlow logical ports are switch defined ports that don't correspond directly to a hardware interface of the switch.		
		Map to various physical ports		Y	
		LAG		Y	
		Tunnel (GRE)		Y	
		Lookback interface		Y	
		Ingress		Y	
		Output		Y	
		Groups		Y	
4.5	Reserved Ports	NA			
		All	Represents all ports the switch can use for forwarding a specific packet. Can be used only as an output port.	Y	
		Controller	Represents the control channel with the OpenFlow controller. Can be used as an ingress port or as an output port.	Y	
		Table	Represents the start of the OpenFlow pipeline.	Y	

	In_port	Represents the packet ingress port. Can be used only as an output port, send the packet out through its ingress port.		Y	Matching must specify the ingress port.
	Any	Special value used in some OpenFlow commands when no port is specified. Can neither be used as an ingress port nor as an output port.		N	
	Local	Represents the switch's local networking stack and its management stack.	O	Y	
	Normal	Represents the traditional non-OpenFlow pipeline of the switch. Can be used only as an output port and processes the packet using the normal pipeline.	O	Y	
	Flood	Represents flooding using the normal pipeline of the switch. Can be used only as an output port.	O	Y	
5	OpenFlow Tables	NA			
5.1	Pipeline Processing				
	Openflow-only	All packets are processed by the OpenFlow pipeline.		Y	
	Openflow-hybrid	OpenFlow operation and normal Ethernet switching operation.		N	
		L2 Ethernet switching, L3 routing (IPv4 routing, IPv6 routing...), ACL and QoS processing.		N	
		VLAN isolation		N	
		A classification mechanism outside of OpenFlow that routes traffic to either the OpenFlow pipeline or the normal pipeline.		N	
		VLAN tag or input port whether to process the packet using which pipeline.		N	
		Normal and flood		N	
	Multiple flow tables, each flow table containing multiple flow entries			Y	
	Sequentially numbered, start at 0			Y	
	Goto instruction			Y	
	Go forward and not backward			Y	
				N	

		Last table can not include goto instruction			
		Table miss		Y	
5.2	Flow Table	NA			
		Match fields	To match against packets..	Y	
		Priority	Matching precedence of the flow entry.	Y	
		Counters	Updated when packets are matched.	Y	Only count by Byte
		Instructions	To modify the action set or pipeline processing.	Y	
		Timeouts	Maximum amount of time or idle time before flow is expired by the switch.	Y	
		Cookie	Opaque data value chosen by the controller.	Y	
		Wildcards all fields and priority equal 0 is table-miss		Y	
5.3	Matching	NA			
		Ingress port		Y	
		Metadata fields		N	
		Apply-actions		Y	
		Any		N	
		Highest priority matches packets be selected		Y	
		Counters update and instructions applied		Y	
		OFPFF_CHECK_OVERLAP		Y	
		OFCF_FRAG_REASM		N	
		Behavior when a switch receives a corrupted packet		N	
5.4	Table-miss	NA			
		Every flow table support table-miss		Y	
		Send packets to controller		Y	
		Drop packets		Y	
		Direct packets to a subsequent table		Y	

		Wildcard all match fields		N	MAC/IP/Port can support wildcard.
		Priority= 0		Y	
		Not exist by default		Y	
		Add or remove by controller at any time		Y	
		May expire		N	
		Match packets unmatched by others		Y	
		Instructions applied		Y	
		Packet-in reason is table-miss		Y	
		Packets unmatched are dropped is not exist table-miss		Y	
5.5	Flow Removal	NA			
			Is run by the switch independently of the controller and is based on the state and configuration of flow entries.	Y	
			A non-zero hard_timeout field causes the flow entry to be removed after the given number of seconds, regardless of how many packets it has matched.	Y	
			A non-zero idle_timeout field causes the flow entry to be removed when it has matched no packets in the given number of seconds.	Y	
			The switch must implement flow expiry and remove flow entries from the flow table when one of their timeouts is exceeded.	Y	
		OFPFF_SEND_FLOW_Rem flag	When a flow entry is removed, the switch must check the flow entry's OFPFF_SEND_FLOW_Rem flag. If this flag is set, the switch must send a flow removed message to the controller.	Y	
			Each flow removed message contains a complete description of the flow entry, the reason for removal (expiry or delete), the flow entry duration at the time of removal, and the flow statistics at time of removal.	Y	
		Eviction	Flow entries may be evicted from flow tables when the switch needs to reclaim resources.	Y	
5.6	Group Table	NA			

		Group identifier		Y	
		Group type		Y	
		Counters		Y	
		Action buckets		Y	
5.6.1	Group Types	All	Used for multicast or broadcast.	Y	
			Effectively cloned for each bucket..	Y	
			Process for each bucket.	Y	Some limitation, please see configuration guide.
			Direct out the ingress, packet is dropped.	Y	
			Output action to OFPP_IN_PORT	Y	Need to specify the in_port in matching.
		Select	Processed by a single bucket.	Y	
			Switch-computed selection algorithm.	Y	
			Bucket weight	N	
			Forward to live ports	Y	
		Indirect	Supports only a single bucket.	Y	
			Multiple flow entries or groups to point to a common group identifier.	Y	
			Supporting faster, more efficient convergence.	N	
			Effectively identical to an all group with one bucket.	Y	
		Fast failover	Execute the first live bucket.	Y	
			Associated with a port and/or group.	Y	
			Change forwarding without request to controller.	Y	
			No bucket live, packet dropped.	Y	
5.7	Meter Table				
		Meter entries		Y	
		Per-flow meters		Y	
		Rate-limit		Y	

		Combine with per-port queue		Y	
		Measure and control packet rate		Y	
		Attached to flow entries		Y	
		In flow instruction set		Y	
		Multiple meters in the same table		Y	
		Multiple meters on the same set of packets		Y	
		Meter identifier		Y	
5.7.1	Meter Bands				
	One band			Y	
	More meter bands			N	
	The rate band applies and the way packets be process			Y	
	Processed by a single meter band based on the current measured meter rate			Y	
	Configure rate lower than current rate			Y	
	No meter band applied if current rate lower than specified rate			Y	
	Band type			Y	
	Rate			Y	
	Counters			Y	
5.8	Counters	NA			
	Per Table Counters	Reference count (active entries)	32 bits	Y	
		Packet Lookups	64 bits	Y	
		Packet Matches	64 bits	Y	
	Per Flow Counters	Received Packets	64 bits	Y	
		Received Bytes	64 bits	Y	
		Duration (seconds)	32 bits	Y	
		Duration (nanoseconds)	32 bits		

	Per Port Counters	Received Packets	64 bits		Y	
		Transmitted Packets	64 bits		Y	
		Received Bytes	64 bits		Y	
		Transmitted Bytes	64 bits		Y	
		Receive Drops	64 bits		Y	
		Transmit Drops	64 bits		Y	
		Receive Errors	64 bits		Y	
		Transmit Errors	64 bits		Y	
		Receive Frame Alignment Errors	64 bits		Y	
		Receive Overrun Errors	64 bits		Y	
		Receive CRC Errors	64 bits		Y	
		Collisions	64 bits		Y	
	Per Queue Counters	Transmit Packets	64 bits		N	
		Transmit Bytes	64 bits		N	
		Transmit Overrun Errors	64 bits		N	
	Per Group Counters	Reference Count (flow entries)	32 bits		Y	
		Packet Count	64 bits		Y	
		Byte Count	64 bits		N	
	Per Bucket Counters	Packet Count	64 bits		N	
		Byte Count	64 bits		N	
5.9	Instructions					
		The controller can query the switch about which of the "Optional Instructions" it supports.			Y	
		Apply-Actions action(s)			Y	
		Clear-Actions			Y	
		Write-Actions action(s)			Y	
		Write-Metadata metadata/mask			Y	

		Goto-Table next-table-id		Y	
		Clear-Actions instruction is executed before the Write-Actions instruction.		Y	
		Goto-Table is executed last.		Y	
		Reject a flow entry if it is unable to execute the instructions & return an unsupported flow error.		N	
5.10	Action Set				
		Action set is associated with each packet.		Y	
		This set is empty by default.		Y	
		Action set is carried between flow tables.		Y	
		When the instruction set of a flow entry does not contain a Goto-Table instruction, pipeline processing stops and the actions in the action set of the packet are executed.		Y	
		Action set contains a maximum of one action of each type.		Y	
		The actions in an action set are applied in the order specified below.		Y	
		1. copy TTL inwards		N	
		2. pop		Y	
		3. push		Y	
		4. copy TTL outwards		N	
		5. decrement TTL		N	
		6. set		Y	
		7. qos		Y	
		8. group	If a group action is specified, apply the actions of the relevant group bucket(s) in the order specified by this list.	Y	
		9. output	If no group action is specified, forward the packet on the port specified by the output action. The output action in the action set is executed last.	N	

		If both an output action and a group action are specified in an action set, the output action is ignored and the group action takes precedence.		N	
		If no output action and no group action were specified in an action set, the packet is dropped.		N	
		The execution of groups is recursive if the switch supports it; a group bucket may specify another group, in which case the execution of actions traverses all the groups specified by the group configuration.		N	
5.11	Action list				
		Apply-Actions instruction and the Packet-out message include an action list	The actions of an action list are executed in the order specified by the list, and are applied immediately to the packet.	Y	
			The effect of those actions is cumulative.	Y	
			If the action list contains an output action, a copy of the packet is forwarded in its current state to the desired port.	Y	
			If the list contains a group actions, a copy of the packet in its current state is processed by the relevant group buckets.	Y	
5.12	Actions				
		Output	Support forwarding to physical ports, switch-defined logical ports and the required reserved ports.	Y	
		Set-Queue	The set-queue action sets the queue id for a packet and is used to provide basic Quality-of-Service (QoS) support.	Y	
		Drop		Y	
		Group		Y	
		Push-Tag/Pop-Tag	Order of header fields - Ethernet, VLAN, MPLS, ARP/IP, TCP/UDP/SCTP (IP-only).	Y	
		Push VLAN header	Push a new VLAN header onto the packet. The Ethertype is used as the Ethertype for the tag. Only Ethertype 0x8100 and 0x88a8 should be used.	Y	
		Pop VLAN header	Pop the outer-most VLAN header from the packet.	Y	
		Push MPLS header	Push a new MPLS shim header onto the packet. Only Ethertype 0x8847 and 0x8848 should be used.	Y	

	Pop MPLS header	Pop the outer-most MPLS tag or shim header from the packet.	Y	
	Push PBB header		Y	
	Pop PBB header		Y	
	Set-Field		Y	
	Set VLAN ID		Y	
	Strip VLAN ID		Y	
	Change-TTL	Modify the values of the IPv4 TTL, IPv6 Hop Limit or MPLS TTL in the packet.	N	
		If it is supported, applied to the outermost-possible header.	N	
	Set MPLS TTL	8 bits: New MPLS TTL, Replace the existing MPLS TTL. Only applies to packets with an existing MPLS shim header.	Y	
	Decrement MPLS TTL	Decrement the MPLS TTL. Only applies to packets with an existing MPLS shim header.	N	
	Set IP TTL	Replace the existing IPv4 TTL or IPv6 Hop Limit and update the IP checksum. Only applies to IPv4 and IPv6 packets.	N	
	Decrement IP TTL	Decrement the IPv4 TTL or IPv6 Hop Limit field and update the IP checksum. Only applies to IPv4 and IPv6 packets	N	
	Copy TTL outwards	Copy the TTL from next-to-outermost to outermost header with TTL. Copy can be IP-to-IP, MPLS-to-MPLS, or IP-to-MPLS.	N	
	Copy TTL inwards	Copy the TTL from outermost to next-to-outermost header with TTL. Copy can be IP-to-IP, MPLS-to-MPLS, or MPLS-to-IP.	N	
5.12.1	Default values for field on push			
	Field values for all fields specified in Table 6 should be copied from existing outer headers to new outer headers.	VLAN ID VLAN ID	Y	
	New fields listed in Table 6 without corresponding existing fields should be set to zero.	VLAN priority VLAN priority	Y	
		MPLS label MPLS label	Y	

		Fields in new headers may be overridden by specifying a "set" action for the appropriate field(s) after the push operation.				
		Fields in new headers may be overridden by specifying a "set" action for the appropriate field(s) after the push operation.	PBB label	PBB label	Y	
6	OpenFlow Channel					
6.1	OpenFlow Protocol Overview	The OpenFlow protocol supports three message types, controller-to-switch, asynchronous, and sym-metric, each with multiple sub-types.				
6.1.1		Controller-to-Switch	Controller/switch messages are initiated by the controller and may or may not require a response from the switch.		Y	
6.1.2		Asynchronous	Asynchronous messages are sent without a controller soliciting them from a switch.		Y	
			Switches send asynchronous messages to controllers to denote a packet arrival or switch state change.		Y	
6.1.3		Symmetric	Symmetric messages are sent without solicitation, in either direction., including Hello, Echo, Error, Experimenter message.		Y	
6.2	Message Handling	The OpenFlow protocol provides reliable message delivery and processing, but does not automatically provide acknowledgements or ensure ordered message processing.			Y	
6.3	OpenFlow Channel Connections	The OpenFlow channel is used to exchange OpenFlow message between an OpenFlow switch and an OpenFlow controller.				
6.3.1		Connection Setup	The switch must be able to establish communication with a controller at a user-configurable IP address, using either a user-specified transport port or the default transport port.		Y	
6.3.2		Connection Interruption	In the case that a switch loses contact with all controllers the switch must immediately enter either \fail secure mode" or \fail standalone mode", depending upon the switch implementation and configuration.		Y	
6.3.3		Encryption			Y	

			The switch and controller may communicate through a TLS connection.		
6.3.4		Multiple Controllers	The switch may establish communication with a single controller, or may establish communication with multiple controllers.	Y	
6.3.5		Auxiliary Connections	The OpenFlow channel may also be composed of a main connection and multiple auxiliary connections.	Y	
6.4	Flow Table Modification Messages	Flow table modification messages are used to add, modify, delete flow.		Y	
6.5	Flow Table Synchronization	A flow table may be synchronized with another flow table. with Flow Table Synchronization		N	
6.6	Group Table Modification Messages	Action of group (including add, modify, delete) can be done by Group table modification messages.		Y	
6.7	Meter Modification Messages	Action of meter (including add, modify, delete) can be done by Meter modification messages.		Y	
6.8	Bundle Messages				
6.8.1		Bundle overview	A bundle is a sequence of OpenFlow modification requests from the controller that is applied as a single OpenFlow operation.	Y	
6.8.2		Bundle example usage		Y	
6.8.3		Bundle error processing	The OpenFlow messages part of a bundle must be pre-validated before they are stored in the bundle.	Y	
6.8.4		Bundle atomic modifications	Committing the bundle must be controller atomic.	N	
6.8.5		Bundle parallelism	The switch must support exchanging echo request and echo reply messages during the creation and population of the bundle, the switch must reply to an echo request without waiting for the end of the bundle.	Y	
7	The OpenFlow Protocol				
7.1	OpenFlow Header	Each Openflow message begins with the OpenFlow header		Y	
7.1.1		Padding	Most OpenFlow messages contain padding fields.	Y	

7.2	Common Structures				
7.2.1		Port Structures	The switch may define physical and logical ports.	Y	
7.2.1.1		Port Description Structures, the physical ports, switch-defined logical ports, and the OFPP_LOCAL reserved port	Ports includes: OFPP_IN_PORT,OFPP_TABLE, OFPP_NORMAL, OFPP_FLOOD, OFPP_ALL, OFPP_CONTROLLER, OFPP_LOCAL,OFPP_ANY	Y	
7.2.1.2		Port Description Properties. A property definition contains the property type, length, and any associated data.	Associated date includes curr, advertised, supported, peer, each one consists of speed, duplexity.	Y	
7.2.2		Flow Match Structures	An OpenFlow match is composed of a flow match header and a sequence of zero or more flow match fields.	Y	
7.2.2.1		Flow Match Header, Fields to match against flows	The flow match header is described by the ofp_match structure, The type field is set to OFPMT_OXM and length field is set to the actual length of ofp_match structure including all match fields. The payload of the OpenFlow match is a set of OXM Flow match fields.	Y	
7.2.2.2		Flow Match Field Structures	The flow match fields are described using the OpenFlow Extensible Match (OXM) format, which is a compact type-length-value (TLV) format.	Y	
7.2.2.3		OXM classes	The match types are structured using OXM match classes, The OpenFlow specification distinguishes two types of OXM match classes, ONF member classes and ONF reserved classes, differentiated by their high order bit.	Y	
7.2.2.4		Flow Matching	A zero-length OpenFlow match (one with no OXM TLVs) matches every packet. Match fields that should be wildcarded are omitted from the OpenFlow match.	Y	
7.2.2.5		Flow Match Field Masking	The masks are defined such that a 0 in a given bit position indicates a 'don't care' match for the same bit in the corresponding field, whereas a 1 means match the bit exactly.	Y	Some limitation, please see the configuration guide.
7.2.2.6		Flow Match Field Prerequisite	In general, matching header fields of a protocol can only be done if the OpenFlow match explicitly matches the corresponding protocol.	Y	
7.2.2.7		Flow Match Fields	Match fields contains: OFPXMT_OFB_IN_PORT, OFPXMT_OFB_IN_PHY_PORT,	Y	

	OXM_OF_IN_PORT	/* Switch input port. */		Y	
	OXM_OF_IN_PHY_PORT	/* Switch physical input port. */	O		
	OXM_OF_METADATA	/* Metadata passed between tables. */	O		
	OXM_OF_ETH_DST	/* Ethernet destination address. */		Y	
	OXM_OF_ETH_SRC	/* Ethernet source address. */		Y	
	OXM_OF_ETH_TYPE	/* Ethernet frame type. */		Y	
	OXM_OF_VLAN_VID	/* VLAN id. */	O	Y	
	OXM_OF_VLAN_PCP	/* VLAN priority. */	O	Y	
	OXM_OF_IP_DSCP	/* IP DSCP (6 bits in ToS field). */	O	Y	
	OXM_OF_IP_ECN	/* IP ECN (2 bits in ToS field). */	O	N	
	OXM_OF_IP_PROTO	/* IP protocol. */		Y	
	OXM_OF_IPV4_SRC	/* IPv4 source address. */		Y	
	OXM_OF_IPV4_DST	/* IPv4 destination address. */		Y	
	OXM_OF_TCP_SRC	/* TCP source port. */		Y	
	OXM_OF_TCP_DST	/* TCP destination port. */		Y	
	OXM_OF_UDP_SRC	/* UDP source port. */		Y	
	OXM_OF_UDP_DST	/* UDP destination port. */		Y	
	OXM_OF_SCTP_SRC	/* SCTP source port. */	O	N	
	OXM_OF_SCTP_DST	/* SCTP destination port. */	O	N	
	OXM_OF_ICMPV4_TYPE	/* ICMP type. */	O	Y	
	OXM_OF_ICMPV4_CODE	/* ICMP code. */	O	Y	
	OXM_OF_ARP_OP	/* ARP opcode. */	O	Y	
	OXM_OF_ARP_SPA	/* ARP source IPv4 address. */	O	Y	
	OXM_OF_ARP_TPA	/* ARP target IPv4 address. */	O	Y	
	OXM_OF_ARP_SHA	/* ARP source hardware address. */	O	N	
	OXM_OF_ARP_THA	/* ARP target hardware address. */	O	N	
	OXM_OF_IPV6_SRC	/* IPv6 source address. */		Y	
	OXM_OF_IPV6_DST	/* IPv6 destination address. */		Y	
	OXM_OF_IPV6_FLABEL	/* IPv6 Flow Label */	O	N	
	OXM_OF_ICMPV6_TYPE	/* ICMPv6 type. */	O	N	

		OXM_OF_ICMPV6_CODE /* ICMPv6 code. */	O	N	
		OXM_OF_IPV6_ND_TARGET /* Target address for ND. */	O	N	
		OXM_OF_IPV6_ND_SLL /* Source link-layer for ND. */	O	N	
		OXM_OF_IPV6_ND_TLL /* Target link-layer for ND. */	O	N	
		OXM_OF_MPLS_LABEL /* MPLS label. */	O	N	
		OXM_OF_MPLS_TC /* MPLS TC. */	O	N	
7.2.2.8	Experimenter Flow Match Fields	Experimenter-specific flow match fields, may be defined using the oxm_class=OFPXMC_EXPERIMENTER	O	N	
7.2.3	Flow Instruction Structures	Flow instructions associated with a flow table entry are executed when a flow matches the entry.			
7.2.4	Action Structures				
	OFPAT_OUTPUT = 0, /* Output to switch port. */		Y		
	OFPAT_COPY_TTL_OUT = 11, /* Copy TTL "outwards" -- from next-to-outermost to outermost */		Y		
	OFPAT_COPY_TTL_IN = 12, /* Copy TTL "inwards" -- from outermost to next-to-outermost */		Y		
	OFPAT_SET_MPLS_TTL = 15, /* MPLS TTL */		Y		
	OFPAT_DEC_MPLS_TTL = 16, /* Decrement MPLS TTL */		N		
	OFPAT_PUSH_VLAN = 17, /* Push a new VLAN tag */		Y		
	OFPAT_POP_VLAN = 18, /* Pop the outer VLAN tag */		Y		
	OFPAT_PUSH_MPLS = 19, /* Push a new MPLS tag */		Y		
	OFPAT_POP_MPLS = 20, /* Pop the outer MPLS tag */		Y		
	OFPAT_SET_QUEUE = 21, /* Set queue id when outputting to a port */		Y		
	OFPAT_GROUP = 22, /* Apply group. */		Y		
	OFPAT_SET_NW_TTL = 23, /* IP TTL. */		N		
	OFPAT_DEC_NW_TTL = 24, /* Decrement IP TTL. */		N		
	OFPAT_SET_FIELD = 25, /* Set a header field using OXM TLV format. */		Y		
	OFPAT_PUSH_PBB = 26 /* Push a new PBB service tag (I-TAG) */		Y		
	OFPAT_POP_PBB = 27 /* Pop the outer PBB service tag (I-TAG) */		Y		
	OFPAT_EXPERIMENTER = 0xffff			N	

7.2.5		Experimenter Structure	Experimenter extensions provide a standard way for OpenFlow switches to offer additional functionality within the OpenFlow message type space.		N	
7.3	Controller-to-Switch Messages					
7.3.1		Handshake	The OFPT_FEATURES_REQUEST message is used by the controller to identify the switch and read its basic capabilities.		Y	
7.3.2		Switch Configuration	The controller is able to set and query configuration parameters in the switch with the OFPT_SET_CONFIG and OFPT_GET_CONFIG_REQUEST messages, respectively.		Y	
7.3.3		Flow Table Configuration	Flow entries are modified in the flow table using the OFP_FLOW_MOD request.		Y	
7.3.4		Modify State Messages			Y	
7.3.4.1		Modify Flow Table Message	The controller can configure the dynamic state in a flow table with the OFP_TABLE_MOD request.		Y	
7.3.4.2		Modify Flow Entry Message	Modifications to a flow table from the controller are done with the OFPT_FLOW_MOD message.		Y	
7.3.4.3		Modify Group Entry Message	Modifications to the group table from the controller are done with the OFPT_GROUP_MOD message.		Y	
7.3.4.4		Port Modification Message	The controller uses the OFPT_PORT_MOD message to modify the behavior of the port.		Y	
7.3.4.5		Meter Modification Messages	Modifications to a meter from the controller are done with the OFPT_METER_MOD message.		Y	
7.3.5		Multipart Messages	Multipart messages are used to encode requests or replies that potentially carry a large amount of data and would not always fit in a single OpenFlow message, which is limited to 64KB.			
7.3.5.1		Description	Information about the switch manufacturer, hardware revision, software revision, serial number, and a description field is available from the OFPMP_DESC multipart request type.		Y	
7.3.5.2		Individual Flow Statistics	Information about individual flow entries is requested with the OFPMP_FLOW multipart request type.		Y	
7.3.5.3		Aggregate Flow Statistics			Y	

			Aggregate information about multiple flow entries is requested with the OFPMP_AGGREGATE multipart request type.			
7.3.5.4		able Statistics	Information about tables is requested with the OFPMP_TABLE multipart request type.		Y	
7.3.5.5		Table Description	The OFPMP_TABLE_DESC multipart request message provides a way to list the current configuration of the tables on a switch, which is set using the OFPT_TABLE_MOD message.		Y	
7.3.5.6		Table Features	The OFPMP_TABLE_FEATURES multipart type allows a controller to both query for the capabilities of existing tables, and to optionally ask the switch to reconfigure its tables to match a supplied configuration.		N	
		Table Features request and reply	If the OFPMP_TABLE_FEATURES request body is empty the switch will return an array of struct ofp_table_features containing the capabilities of the currently configured flow tables.		N	
		Table Features properties	A property definition contains the property type, length, and any associated data.		N	
7.3.5.7		Port Statistics	Information about ports statistics is requested with the OFPMP_PORT_STATS multipart request type.		Y	
7.3.5.8		Port Description	The port description request OFPMP_PORT_DESCRIPTION enables the controller to get a description of all the ports in the system that support OpenFlow.		Y	
7.3.5.9		Queue Statistics	The OFPMP_QUEUE_STATS multipart request message provides queue statistics for one or more ports and one or more queues.		N	
7.3.5.10		Queue Descriptions	The controller can query the switch for configured queues on a port using OFPMP_QUEUE_DESC multipart request.		Y	
7.3.5.11		Group Statistics	The OFPMP_GROUP multipart request message provides statistics for one or more groups.		Y	
7.3.5.12		Group Description	The OFPMP_GROUP_DESC multipart request message provides a way to list the set of groups on a switch along with their corresponding bucket actions.		Y	
7.3.5.13		Group Features	The OFPMP_GROUP_FEATURES multipart request message provides a way to list the capabilities of groups on a switch.		Y	
7.3.5.14		Meter Statistics			Y	

			The OFPMT_METER stats request message provides statistics for one or more meters.		
7.3.5.15		Meter Configuration Statistics	The OFPMT_METER_CONFIG stats request message provides configuration for one or more meter.	Y	
7.3.5.16		Meter Features Statistics	The OFPMT_METER_FEATURES stats request message provides the set of features of the metering subsystem.	Y	
7.3.5.17		Flow monitoring	The OFPMP_FLOW_MONITOR multipart type allows a controller to manage flow monitors, that keep track of changes to the flow tables.	N	
		Flow monitoring request	Flow monitor configuration is done with a OFPMP_FLOW_MONITOR multipart request	N	
		Flow monitoring reply	When the switch received a OFPMP_FLOW_MONITOR multipart request, it replies to it using a OFPMP_FLOW_MONITOR multipart reply, the transaction id (xid) of this reply must be the same as the request.	N	
		Flow monitoring pause	OpenFlow messages for flow monitor notifications can overflow the buffer space available to the switch either temporarily or more permanently.	N	
7.3.5.18		Experimenter Multipart	Experimenter-specific multipart messages are requested with the OFPMP_EXPERIMENTER multipart type.	N	
7.3.6		Packet-Out Message	When the controller wishes to send a packet out through the datapath, it uses the OFPT_PACKET_OUT message.	Y	
7.3.7		Barrier Message	When the controller wants to ensure message dependencies have been met or wants to receive notifications for completed operations, it may use an OFPT_BARRIER_REQUEST message.	Y	
7.3.8		Role Request Message	When the controller wants to change its role, it uses the OFPT_ROLE_REQUEST message.	Y	
7.3.9		Bundle messages			
7.3.9.1		Bundle control messages	The controller can create, destroy and commit bundles with the OFPT_BUNDLE_CONTROL request.	Y	
7.3.9.2		Bundle Add message	The controller can add requests to a bundle using the OFPT_BUNDLE_ADD_MESSAGE message.	Y	

7.3.9.3		Bundle flags	Bundle flags enable to modify the behavior of a bundle.		Y	
7.3.9.4		Bundle properties	A property definition contains the property type, length, and any associated data.		Y	
7.3.9.5		Creating and opening a bundle	To create a bundle, the controller sends a OFPT_BUNDLE_CONTROL message with type OFPBCT_OPEN_REQUEST.		Y	
7.3.9.6		Adding messages to a bundle	The switch adds message to a bundle using the OFPT_BUNDLE_ADD_MESSAGE.		Y	
7.3.9.7		Closing a bundle	To finish recording a bundle, the controller may sends a OFPT_BUNDLE_CONTROL message with type OFPBCT_CLOSE_REQUEST.		Y	
7.3.9.8		Committing Bundles	To finish and apply the bundle, the controller sends a OFPT_BUNDLE_CONTROL message with type OFPBCT_COMMIT_REQUEST.		Y	
7.3.9.9		Discarding Bundles	To finish and discard the bundle, the controller sends a OFPT_BUNDLE_CONTROL message with type OFPBCT_DISCARD_REQUEST.		Y	
7.3.9.10		Other bundle error conditions	If a OFPT_BUNDLE_CONTROL message contains an invalid type, the switch must reject the request and send an ofp_error_msg with OFPET_BUNDLE_FAILED type and OFPBFC_BAD_TYPE code.		N	
7.3.10		Set Asynchronous Configuration Message	The switch manages a per-controller asynchronous configuration, which defines the asynchronous messages that it wants to receive (other than error messages) on a given OpenFlow channel.		Y	
7.4	Asynchronous Messages					
7.4.1		Packet-In Message	When packets are received by the datapath and sent to the controller, they use the OFPT_PACKET_IN.		Y	
7.4.2		Flow Removed Message	If the controller has requested to be notified when flow entries time out or are deleted from table, the data path does this with the OFPT_FLOW_Removed message.		Y	
7.4.3		Port Status Message	As ports are added, modified, and removed from the datapath, the controller needs to be informed with the OFPT_PORT_STATUS message		Y	
7.4.4		Controller Role Status Message			Y	

			When a controller has its role changed by the switch, and not directly changed by that controller using a OFPT_ROLE_REQUEST message, the corresponding controller must be informed with a OFPT_ROLE_STATUS message.			
7.4.5		Table Status Message	When the table state changes, the controller needs to be informed with the OFPT_TABLE_STATUS message.	Y		
7.4.6		Request Forward Message	When a controller modifies the state a groups and meters, the request that successfully modifies this state may be forwarded to other controller.	Y		
7.5	Symmetric Messages					
7.5.1		Hello	The OFPT_HELLO message consists of an OpenFlow header plus a set of variable size hello elements.	Y		
7.5.2		Echo Request	An Echo Request message consists of an OpenFlow header plus an arbitrary-length data field.	Y		
7.5.3		Echo Reply	An Echo Reply message consists of an OpenFlow header plus the unmodified data field of an echo request message.	Y		
7.5.4		Error Message	Error messages are used by the switch or the controller to notify the other side of the connection of problems.	Y		
7.5.5		Experimenter Message		N		
A	Header file openflow.h					
B	Release Notes					
B.14	OpenFlow version 1.4.0					
B.14.1		More extensible wire protocol		N		
B.14.2		More descriptive reasons for packet-in		Y		
B.14.3		Optical port properties		N		
B.14.4		Flow-removed reason for meter delete.		Y		
B.14.5		Flow monitoring		N		
B.14.6		Role status events		Y		
B.14.7		Eviction		Y		

B.14.8	Vacancy events		Y	
B.14.9	Bundles		Y	
B.14.10	Synchronized tables		N	
B.14.11	Group and Meter change notifications		Y	
B.14.12	Error code for bad priority		N	
B.14.13	Error code for Set-async-config		Y	
B.14.14	PBB UCA header field		N	
B.14.15	Error code for duplicate instruction		Y	
B.14.16	Error code for multipart timeout		Y	
B.14.17	Change default TCP port to 6653		N	

PicOS Support for OpenFlow 1.4.0

This document contains OpenFlow 1.4.0 features supported by the Pica8 PicOS software. For clarity, the feature names in this table are identical to the feature names found in OpenFlow Switch Specification, Version 1.4.0.

OpenFlow Messages

Each OpenFlow message begins with the OpenFlow header. The OpenFlow header has several fields, including the **type** field. The **type** field identifies the type of OpenFlow message.

The OpenFlow protocol has three message types: *symmetric*, *controller-to-switch*, and *asynchronous*. Each message type has multiple sub-types.

Symmetric Messages

Symmetric messages are unsolicited messages that may be initiated by either the switch or the controller. Symmetric messages are sent without explicit solicitation in either direction. The following table describes PicOS support for symmetric messages:

Table 1 PicOS Support for OpenFlow Symmetric Messages

Message	Support	Comments
OFPT_HELLO	Supported	
OFPT_ERROR	Supported	
OFPT_ECHO_REQUEST	Supported	
OFPT_ECHO_REPLY	Supported	

OFPT_EXPERIMENTER	Not Supported	

Controller-to-Switch Messages

As the title suggests, Controller-to-switch messages are sent from the controller to the switch. These messages are used to directly manage the state of a switch. Controller-to-switch messages may or may not require a response from the switch.

The following table describes PicOS support for controller-to-switch messages:

Table 2 PicOS Support for OpenFlow Controller-to-Switch Messages

Message	Support	Comments
OFPT_FEATURES_REQUEST	Supported	
OFPT_FEATURES_REPLY		
See Capabilities Supported by Datapath		
OFPT_GET_CONFIG_REQUEST	Supported	
OFPT_GET_CONFIG_REPLY		
OFPT_SET_CONFIG	Supported	
OFPT_PACKET_OUT	Supported	
OFPT_FLOW_MOD	Supported	
OFPT_GROUP_MOD	Supported	
OFPT_PORT_MOD	Supported	
OFPT_TABLE_MOD	Supported	
OFPT_MULTIPART_REQUEST	Supported	
See Multipart Messages		
OFPT_MULTIPART_REPLY	Supported	
See Multipart Messages		
OFPT_BARRIER_REQUEST	Supported	
OFPT_BARRIER_REPLY		
OFPT_ROLE_REQUEST	Supported	
OFPT_ROLE_REPLY		
OFPT_GET_ASYNC_REQUEST	Supported	
OFPT_GET_ASYNC_REPLY	Supported	
OFPT_SET_ASYNC	Supported	

OFPT_METER_MOD	Supported
----------------	-----------

Asynchronous Messages

Asynchronous messages are sent from the switch to the controller. These messages are used to communicate network events and switch state changes to the controller. Asynchronous messages are sent without a controller explicitly requesting them from a switch.

The following table describes PicOS support for asynchronous messages:

Table 3 PicOS Support for OpenFlow Asynchronous Messages

Message	Support	Comments
OFPT_PACKET_IN	Supported	
OFPT_FLOW_REMOVED	Supported	
OFPT_PORT_STATUS	Supported	
OFPT_ROLE_STATUS	Supported	
OFPT_TABLE_STATUS	Supported	
OFPT_REQUESTFORWARD		

Bundle Operations

The following table describes PicOS support for bundle operations:

Table 4 PicOS Support for OpenFlow Bundle Operations

Message	Support	Comments
OFPT_BUNDLE_CONTROL	Supported	
OFPT_BUNDLE_ADD_MESSAGE	Supported	

Multipart Messages

The following table describes PicOS support for multipart messages: a single OpenFlow message cannot be larger than 64 kilobytes. Multipart messages are used to encode requests or replies that would carry a large amount of data, and would not always fit in a single OpenFlow message. The sender encodes the request or reply as a sequence of multipart messages with a specific multipart type. The receiver re-assembles the request or reply.

Table 5 PicOS Support for OpenFlow Multipart Messages

Message	Support	Comments
OFPMP_DESC	Supported	

OFPMP_FLOW	Supported	
OFPMP_TABLE	Supported	
OFPMP_TABLE_DESC	Supported	
OFPMP_TABLE_FEATURES	Supported	
OFPMP_PORT_STATS	Supported	
OFPMP_PORT_DESCRIPTION	Supported	
OFPMP_QUEUE_STATS	Not Supported	
OFPMP_QUEUE_DESC	Supported	
OFPMP_GROUP	Supported	
OFPMP_GROUP_DESC	Supported	
OFPMP_GROUP_FEATURES	Supported	
OFPMT_METER	Supported	
OFPMT_METER_CONFIG	Supported	
OFPMT_METER_FEATURES	Supported	
OFPMP_FLOW_MONITOR	Supported	
OFPMP_EXPERIMENTER	Not Supported	

Capabilities Supported by Datapath

The controller sends an **OFPT_FEATURES_REQUEST** message to the switch once a session is established. The switch responds to the **OFPT_FEATURES_REQUEST** message with an **OFPT_FEATURES_REPLY** message. The **OFPT_FEATURES_REPLY** message has several fields, including the **capabilities** field. The **capabilities** field identifies the capabilities supported by the switch datapath.

The **capabilities** field is a combination of the following bits/flags:

Table 6 PicOS Support for OpenFlow Capabilities Supported by Datapath

Specification	Support	Comments
OFPC_FLOW_STATS	Supported	
OFPC_TABLE_STATS	Supported	
OFPC_PORT_STATS	Supported	
OFPC_GROUP_STATS	Supported	

Specification	Support	Comments
OFPC_IP_REASM		
OFPC_QUEUE_STATS		
OFPC_PORT_BLOCKED		

OpenFlow Ports

OpenFlow ports are abstract network interfaces used for passing traffic between the OpenFlow switch and the rest of the network. An OpenFlow switch makes a number of OpenFlow ports available. The set of OpenFlow ports does not have to be identical to the set of physical network interfaces on the switch hardware.

Port Descriptions

The OpenFlow switch receives and sends packets on OpenFlow ports. The switch may define physical and logical ports. OpenFlow specification also defines some reserved ports.

Table 7 PicOS Support for OpenFlow Ports

Specification	Support	Comments
OFPP_MAX		
OFPP_IN_PORT	Supported	
OFPP_TABLE	Supported	
OFPP_NORMAL	Supported	
OFPP_FLOOD	Supported	
OFPP_ALL	Supported	
OFPP_CONTROLLER	Supported	
OFPP_LOCAL	Supported	
OFPP_ANY	Supported	

Port Administrative Settings

The following table describes PicOS support for administrative settings of OpenFlow ports:

Table 8 PicOS Support for OpenFlow Port Administrative Settings

Specification	Support	Comments
OFPPC_PORT_DOWN	Supported	
OFPPC_NO_RECV		

Specification	Support	Comments
OFPPC_NO_FWD		
OFPPC_NO_PACKET_IN		

Port States

The following table describes PicOS support for OpenFlow port states:

Table 9 PicOS Support for OpenFlow Port States

Specification	Support	Comments
OFPPS_LINK_DOWN	Supported	
OFPPS_BLOCKED		
OFPPS_LIVE		

Port Features

The following table describes PicOS support for OpenFlow port features available in the datapath:

Table 10 PicOS Support for OpenFlow Port Features

Specification	Support	Comments
OFPPF_10MB_HD		
OFPPF_10MB_FD	Supported	
OFPPF_100MB_HD		
OFPPF_100MB_FD	Supported	
OFPPF_1GB_HD		
OFPPF_1GB_FD	Supported	
OFPPF_10GB_FD	Supported	
OFPPF_40GB_FD	Supported	
OFPPF_100GB_FD	Supported	
OFPPF_1TB_FD		
OFPPF_OTHER		
OFPPF_COPPER	Supported	
OFPPF_FIBER	Supported	
OFPPF_AUTONEG	Supported	

Specification	Support	Comments
OFPPF_PAUSE	Supported	
OFPPF_PAUSE_ASYM		

OpenFlow Instructions

Each flow entry contains a set of instructions that are executed to match a packet to an entry. The following table details OpenFlow instructions supported by PicOS:

Table 11 PicOS Support for OpenFlow Instructions

Specification	Support	Comments
OFPIT_GOTO_TABLE	Supported	only next-table-id; is executed last.
OFPIT_WRITE_METADATA	Supported	
OFPIT_WRITE_ACTIONS	Supported	
OFPIT_APPLY_ACTIONS	Supported	
OFPIT_CLEAR_ACTIONS	Supported	
OFPIT_METER	Supported	
OFPIT_EXPERIMENTER		

OpenFlow Action Types

The following table details OpenFlow action types supported by PicOS:

Table 12 PicOS Support for OpenFlow Action Types

Specification	Support	Comments
OFPAT_OUTPUT	Supported	
OFPAT_COPY_TTL_OUT	Supported	
OFPAT_COPY_TTL_IN	Supported	
OFPAT_SET_MPLS_TTL	Supported	
OFPAT_DEC_MPLS_TTL	Not Supported	
OFPAT_PUSH_VLAN	Supported	
OFPAT_POP_VLAN	Supported	
OFPAT_PUSH_MPLS	Supported	
OFPAT_POP_MPLS	Supported	

Specification	Support	Comments
OFPAT_SET_QUEUE	Supported	
OFPAT_GROUP	Supported	
OFPAT_SET_NW_TTL	Not Supported	
OFPAT_DEC_NW_TTL	Not Supported	
OFPAT_SET_FIELD	Supported	
OFPAT_PUSH_PBB	Supported	
OFPAT_POP_PBB	Supported	
OFPAT_EXPERIMENTER	Not Supported	

OpenFlow Match Fields

A match field may include the packet header, the ingress port, and the metadata value. A match field may use wildcards to match any value and, in some cases, bitmasks. A packet is matched against a match field.

The following table details the OXM flow match field types supported by PicOS:

Table 13 PicOS Support for OpenFlow Flow Match Fields

Specification	Support	Comments
OFPXMT_OFB_IN_PORT	Supported	
OFPXMT_OFB_IN_PHY_PORT	Supported	
OFPXMT_OFB_METADATA		
OFPXMT_OFB_ETH_DST	Supported	
OFPXMT_OFB_ETH_SRC	Supported	
OFPXMT_OFB_ETH_TYPE	Supported	
OFPXMT_OFB_VLAN_VID	Supported	
OFPXMT_OFB_VLAN_PCP	Supported	
OFPXMT_OFB_IP_DSCP	Supported	
OFPXMT_OFB_IP_ECN		
OFPXMT_OFB_IP_PROTO	Supported	
OFPXMT_OFB_IPV4_SRC	Supported	
OFPXMT_OFB_IPV4_DST	Supported	
OFPXMT_OFB_TCP_SRC	Supported	

Specification	Support	Comments
OFPXMT_OFB_TCP_DST	Supported	
OFPXMT_OFB_UDP_SRC	Supported	
OFPXMT_OFB_UDP_DST	Supported	
OFPXMT_OFB_SCTP_SRC		
OFPXMT_OFB_SCTP_DST		
OFPXMT_OFB_ICMPV4_TYPE	Supported	
OFPXMT_OFB_ICMPV4_CODE	Supported	
OFPXMT_OFB_ARP_OP	Supported	
OFPXMT_OFB_ARP_SPA	Supported	
OFPXMT_OFB_ARP_TPA	Supported	
OFPXMT_OFB_ARP_SHA		
OFPXMT_OFB_ARP_THA		
OFPXMT_OFB_IPV6_SRC	Supported	
OFPXMT_OFB_IPV6_DST	Supported	
OFPXMT_OFB_IPV6_FLABEL	Supported	
OFPXMT_OFB_ICMPV6_TYPE		
OFPXMT_OFB_ICMPV6_CODE		
OFPXMT_OFB_IPV6_ND_TARGET		
OFPXMT_OFB_IPV6_ND_SLL		
OFPXMT_OFB_IPV6_ND_TLL		
OFPXMT_OFB_MPLS_LABEL		
OFPXMT_OFB_MPLS_TC		
OFPXMT_OFP_MPLS_BOS		
OFPXMT_OFB_PBB_ISID	Supported	
OFPXMT_OFB_TUNNEL_ID	Supported	
OFPXMT_OFB_IPV6_EXTHDR		
OFPXMT_OFB_PBB_UCA		

OpenFlow Group Types

The following table describes PicOS support for OpenFlow group types:

Table 14 PicOS Support for OpenFlow Group Types

Specification	Support	Comments
OFPGT_ALL	Supported	
OFPGT_SELECT	Supported	
OFPGT_INDIRECT	Supported	
OFPGT_FF	Supported	

Introduction to Open vSwitch

Open vSwitch, sometimes abbreviated as OVS, is an open-source multilayer virtual switch. Open vSwitch can operate both as a soft switch running within the hypervisor and as the control stack for switching silicon. Learn more about Open vSwitch [here](#).

Open vSwitch is already included in the Pica8 PicOS software and runs as a process in PicOS.

The Open vSwitch version available on a PicOS switch can be determined using the **ovs-appctl version** command from the Linux shell.

```
admin@Switch$ ovs-appctl version
ovs-vswitchd (Open vSwitch) 2.3.0
Compiled May 24 2015 21:37:14
```

Open vSwitch Components

The main components of Open vSwitch are:

1. Kernel Module
2. Open vSwitch Database
3. Open vSwitch Daemon

Kernel Module

Kernel modules are pieces of code that can be loaded and unloaded into the kernel on demand. They extend the functionality of the kernel without having to reboot the system. Without modules, new functionality would have to be added directly into the kernel image, resulting in monolithic, larger kernels.

The kernel module handles switching and tunneling. It is designed to be fast and simple. When a packet is received and a match is found, associated actions are executed and counters are updated. Otherwise, packets are sent to userspace.

To manage the kernel module, use the **ovs-dpctl** command-line tool.

Open vSwitch Database

The Open vSwitch Database (OVSDB) holds switch configuration including bridge, interface, and tunnel definitions. The OVSDB and OpenFlow controller addresses are also held by the OVSDB. The OVSDB, and hence switch configuration, is stored on persistent storage and survives a reboot.

The **ovsdb-server** communicates with **ovs-vswitchd** and the controller using the OVSDB management protocol, defined in RFC 7047.

To manage **ovsdb-server**, use the **ovsdb-tool** and **ovs-vsctl** command-line tools.

Open vSwitch Daemon

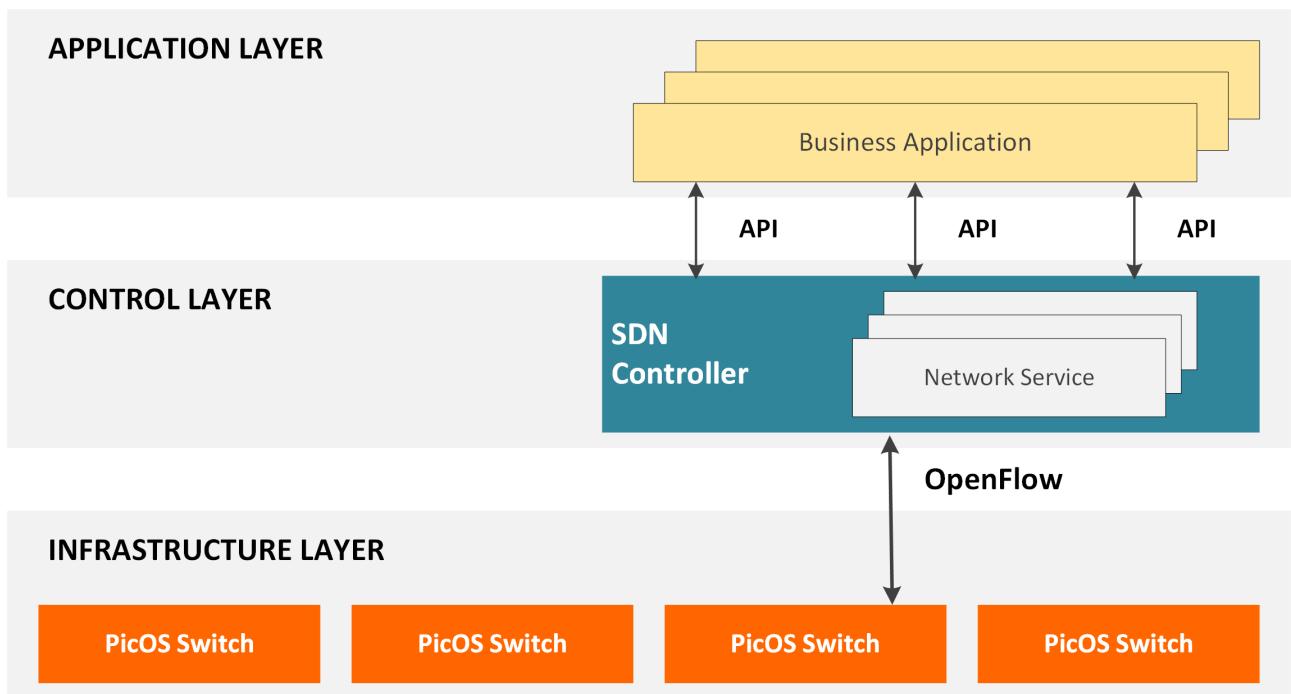
The Open vSwitch daemon (**ovs-vswitchd**) is the core component of an Open vSwitch incarnation. It communicates with the controller using OpenFlow. It communicates with the Open vSwitch Database Server (**ovsdb-server**) using the Open vSwitch Database (OVSDB) management protocol, defined in RFC 7047. The **ovs-vswitchd** communicates with the kernel module over netlink - a Linux kernel interface used for communication between userspace processes and the kernel.

The **ovs-vswitchd** supports multiple independent data paths, known as bridges.

To manage **ovs-vswitchd**, use the **ovs-ofctl** and **ovs-appctl** command-line tools.

Introduction to OpenFlow

The separation of control and data planes is one of the fundamental principles of SDN (Software-Defined Networking). OpenFlow is the first standard interface for communication between the control and data planes of an SDN architecture. The following graphic presents the role OpenFlow plays in the SDN framework:



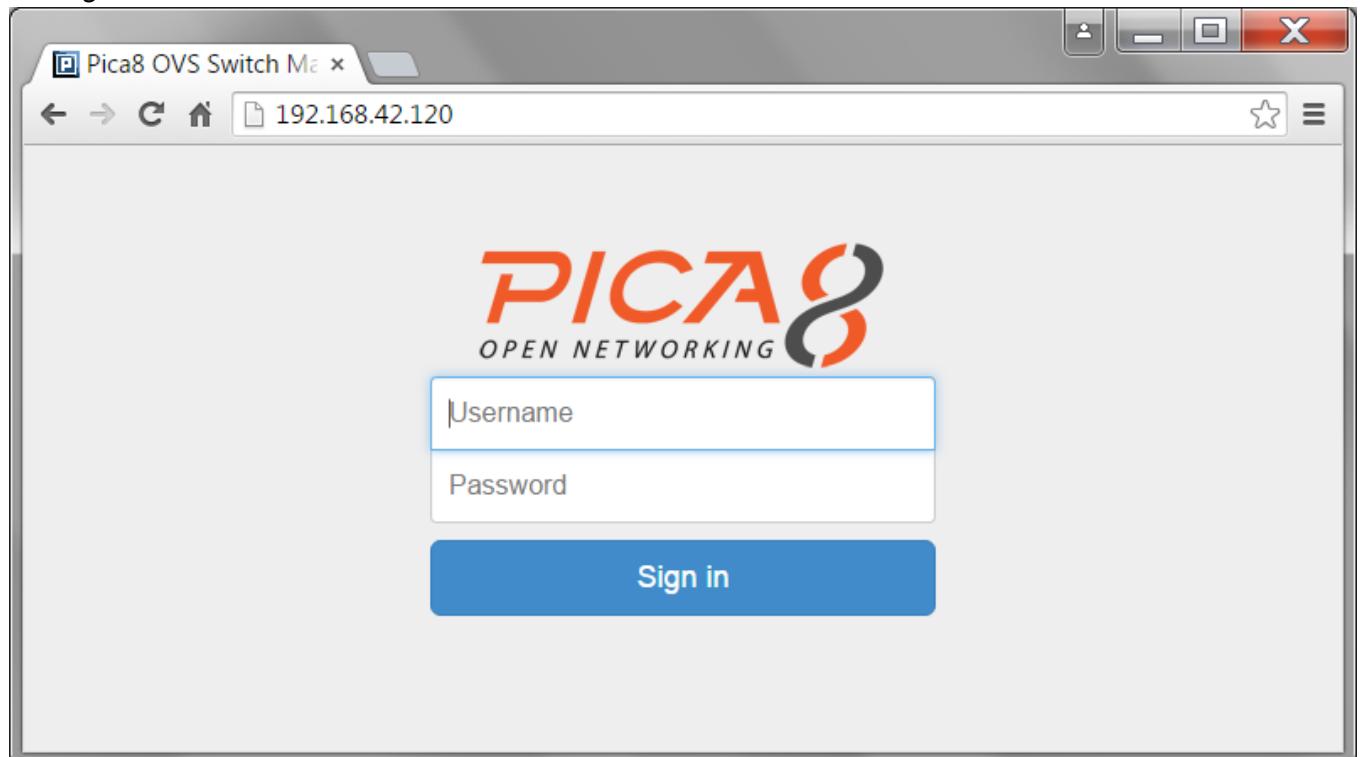
Learn more about OpenFlow [here](#).

OVS Web User Interface

- Login Interface
- Monitoring the Switch
- Adding a Bridge
- Add a Port
- Add GRE Port
- Add Group Table
- Add or Edit a Controller
- Edit Flow Tables
- Edit Lag Interface

Login Interface

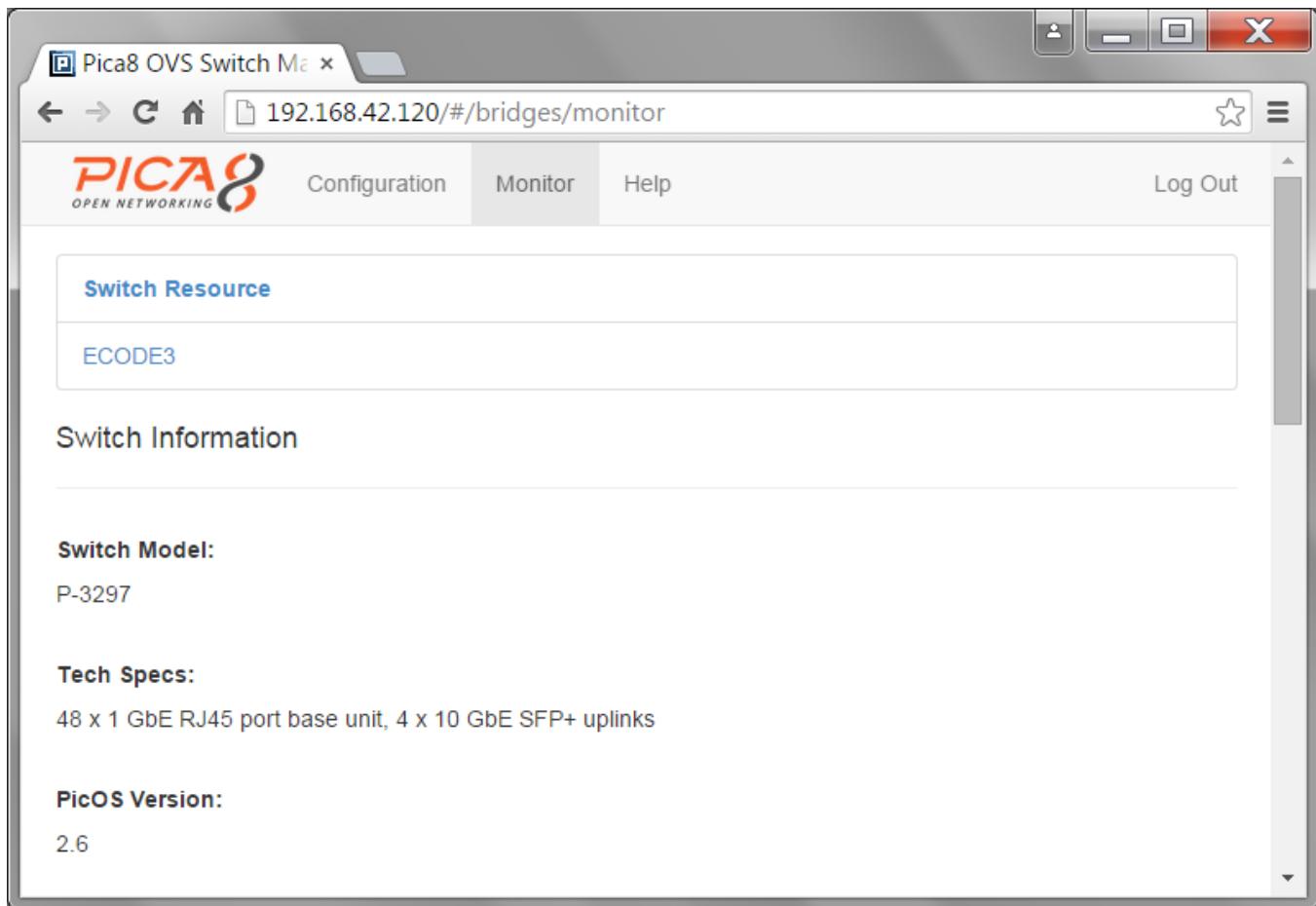
User can use the management IP address of the PicOS switch to launch the *Pica8 OVS Switch Management Panel*.



Monitoring the Switch

The **Monitor** tab provides basic switch information and allows users to monitor the switch operation.

The **Monitor** tab may have several sections including the default *Switch Resource*, as shown below:



The default *Switch Resource* section, provides the following pieces of information:

- Switch Model
- Tech Specs
- PicOS Version
- Software Revision
- Web Interface Version
- Open vSwitch Version
- Maximum Flow Numbers
- Storage Capacity
- MAC Address
- IP Address
- Gateway
- OVSDB Config File Location
- CPU Load

The **Monitor** tab also allows users to monitor any bridges created on the switch. The following screenshot displays information about a bridge that has been named *ECODE3*.

The screenshot shows the Pica8 OVS Switch Management interface. The title bar reads "Pica8 OVS Switch M" and the address bar shows "192.168.42.120/#/bridges/monitor". The top navigation menu includes "Configuration", "Monitor", "Help", and "Log Out". On the left, there's a sidebar with "Switch Resource" and "ECODE3". The main content area has tabs: "Basic" (selected), "Controller", "Port", "Tunnel", "LAG", "Flow Table", "Group Table", "Meter Table", and "Visibility". Below the tabs is a table with one row for the bridge "ECODE3". The table columns are "Name", "Datapath ID", "Fail Mode", and "OpenFlow Version". The values are "ECODE3", "4c3e486e730203da", "", and "OpenFlow13" respectively. At the bottom, there's a checkbox for "Auto refresh every" with a dropdown set to "5" seconds, and a "Manually Refresh" button. A copyright notice at the bottom reads "Copyright © 2008-2015 Pica8 All Rights Reserved."

Name	Datapath ID	Fail Mode	OpenFlow Version
ECODE3	4c3e486e730203da		OpenFlow13

User can refresh bridge information manually or set it to auto-refresh at the specified interval.

Adding a Bridge

Once user has successfully launched the user interface, the ***Configuration*** tab reveals the switch resource section that provides basic switch information. To create a bridge, click on the ***Create a bridge*** icon.

Switch Model: P-3290

Tech Specs: 48 x 1 GbE RJ45 port base unit, 4 x 10 GbE SFP+ uplinks

PicOS Version: 2.2

Software Revision: 13602

Web Interface Version: 1.0

Open vSwitch Version: 2.0.90

Maximum Flow Numbers: 2048

Storage Capacity: 2.4G / 3.3G (Avail / Total)

MAC Address: 08:9e:01:a8:00:4a

IP Address: 10.10.50.154

Gateway: 10.10.50.1

OVSDB config file location: /ovsdb/ovs-vswitchd.conf.db

CPU Load: 12.66%

Multiple logic bridges are supported on the switch. Choose a logic bridge listed on the left pane to configure. Or create a logic bridge to proceed and add ports to the bridge after creation.

Create a bridge

Once user has created a new bridge (br0 in example below), user can edit properties or delete it. The menu on the left (in the graphic below) allows users to view, edit, and change any of the modules listed in the menu.

Switch Resource

- br0
- Basic Info
- Controllers
- Ports
- Tunnels
- Link Aggregation
- Group Table
- Meter Table
- Flow Table
- Mirrors
- NetFlow
- sFlow

The very basic information of this logic bridge.

Datapath ID: 5e3e8e0f05d081fe

Fail Mode: standalone

OpenFlow Version: OpenFlow10,OpenFlow12,OpenFlow13

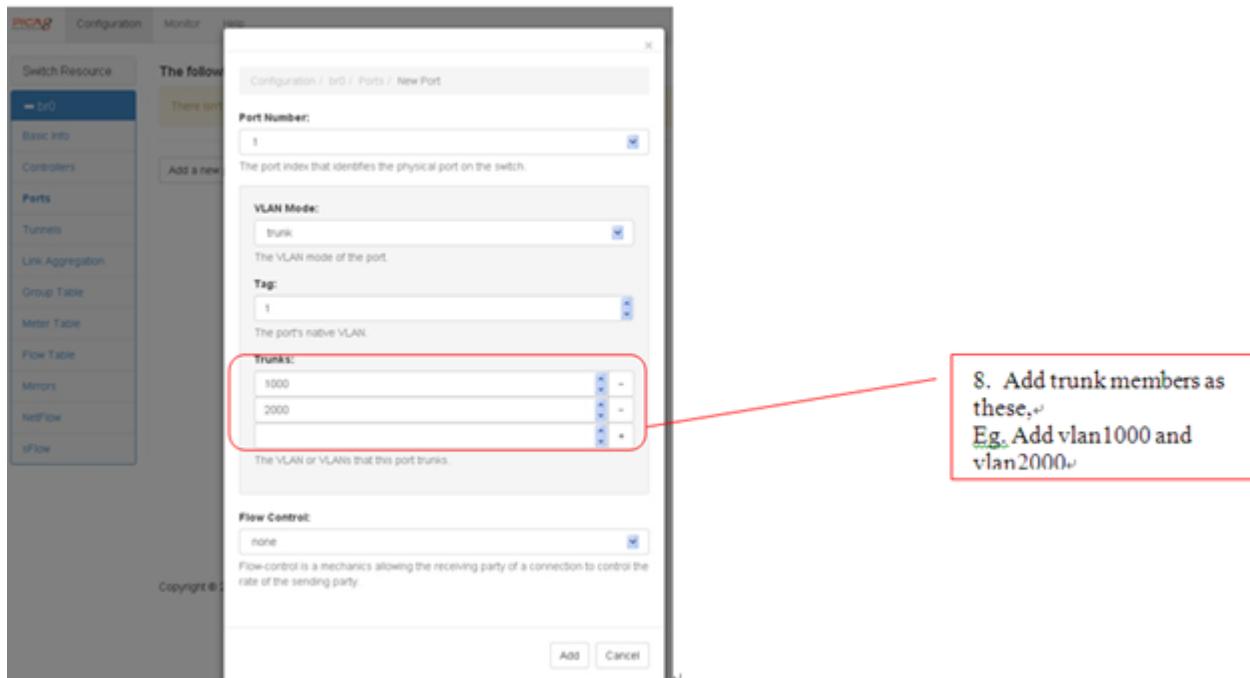
Edit **Delete**

5. To edit ↴

4. Choose and enter difference modules to check and edit info. ↴

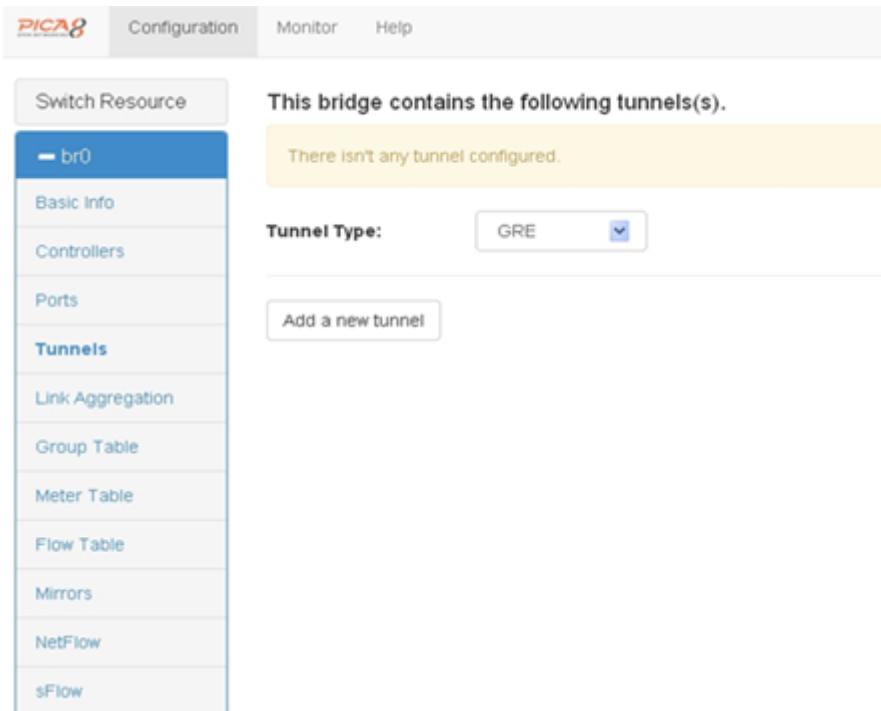
Add a Port

To add a new port, click on **Port** from the menu on the left of the screen. Complete the port number, VLAN mode, tag, and trunks. Then, click **Add** to save it.



Add GRE Port

Select **Tunnels** from the menu to view the bridge's tunnel type or to add or edit a tunnel.



Add Group Table

The screenshot shows the PicOS configuration interface with the 'Groups' dialog open. The 'Actions' section of the dialog is highlighted with a red box. A callout box to the right contains the following text:

10. Add group actions,
Eg. bucket1:actions=output:3 ;
Bucket2:actions=output:4;

Add or Edit a Controller

The following controllers are configured for this bridge.				
Method	Connection Mode	IP Address	Port Number	Operations
tcp	out-of-band	10.10.50.47	6633	[Edit] [Delete]

6. Add a new controller

7. Edit a existing controller

Edit Flow Tables

View the flow table attached to the bridge. Then, select required tabs to delete, edit, download, or add to the flow table.

The screenshot shows the PicOS Open vSwitch configuration interface. On the left, a sidebar lists resources: Basic Info, Controllers, Ports, Tunnels, Link Aggregation, Group Table, Meter Table, Flow Table (selected), Mirrors, NetFlow, and sFlow. The main area displays flow tables attached to the bridge br0. A table titled 'Table: 0' shows one entry: Priority 0, Cookie 0x0, Match Fields (empty), Actions NORMAL, and Operation [Edit] (circled in red). Below the table are buttons for Refresh, Download Flows, New Flow Table Entry (circled in red), and Add new flows (also circled in red). A red box labeled '13. Edit an existed flow' points to the [Edit] button. Another red box labeled '11. Add a new flow' points to the 'New Flow Table Entry' button. A third red box labeled '12. Add new flows' points to the 'Add new flows' button.

Edit Lag Interface

The screenshot shows the PicOS Open vSwitch configuration interface. On the left, a sidebar lists resources: Basic Info, Controllers, Ports, Tunnels, Link Aggregation (selected), Group Table, Meter Table, Flow Table, Mirrors, NetFlow, and sFlow. The main area shows a dialog for 'Configuration / br0 / Link Aggregation / New Bond'. It includes fields for LAG Number (set to 1), Type (static), Members (a list box containing 1, 2, 3, 11, circled in red), VLAN Mode (access), Tag (1), and Trunks (an empty list box). A red box labeled '9. Add lag members' points to the 'Members' field.

Configuring Open vSwitch

Port Ranges

The port ranges in PicOS are as follows:

Port Type	Port Number
Physical	1-1023
LAG	1025-2047
Bond	2049-3071
GRE	3073-4095
VXLAN	4097-5119
L2GRE	5121-6143

Get switch feature

Due to ASIC limit, the different platforms supports is different. User can use command **ovs-vsctl list switch_feature** to get the feature messages.

```
admin@PicOS-OVS$ ovs-vsctl list switch_feature
_uuid : 41daac2b-1764-4ae8-b7da-8c8252ecec6b
hardware_model : "HP5712"
l2l3_buffer_stats : {host-route=12000, "l2-size"=32768, route=12000}
support_egress_meter: true
support_l2gre : true
support_l2l3_buffer : true
support_multiple_groups: true
support_policy_route: true
support_port_modes : [flexible, max, normal]
support_vxlan : true
xovs_running : false
admin@PicOS-OVS$ 
admin@PicOS-OVS$ version
Copyright (C) 2009-2016 Pica8, Inc.
=====
Hardware Model : HP5712
Linux System Version/Revision : 2.0.6master/ffb6936
Linux System Released Date : 11/23/2016
L2/L3 Version/Revision : 2.0.6master/ffb6936
L2/L3 Released Date : 11/23/2016
OVS/OF Version/Revision : 2.0.6master/ffb6936
OVS/OF Released Date : 11/23/2016
```

Configuring ovsdb Locally

User can check the status of the **ovsdb-server** process using the **ps** command from the Linux shell.

```
admin@PicOS-OVS$ps aux|grep ovs
root      3422  0.0  0.1    7020  2788 ?          S     10:42   0:00 ovsdb-server
/ovs/ovs-vswitchd.conf.db /tmp/inventory.conf.db /ovs/function.conf.db --pidfile
--remote=ptcp:6640:127.0.0.1 --remote=unix:/ovs/var/run/openvswitch/db.sock
--remote=db:Open_vSwitch,Manager,target
root      3435 15.7  1.5  122860  32320 ?          Sl     10:42   2:12 ovs-vswitchd
--pidfile=ovs-vswitchd.pid --overwrite-pidfile
root      3494  0.0  0.2   12960  5752 ?          Ss     10:43   0:00 /usr/bin/python
/ovs/share/openvswitch/scripts/ovs-invd --pidfile=ovs-invd.pid --overwrite-pidfile --detach
--root-prefix=/ovs/
admin     3738  0.0  0.0    2200   684  ttys0     S+    10:56   0:00 grep --color=auto ovs
admin@PicOS-OVS$
```

The string '--remote=ptcp:6640:127.0.0.1' in the output above shows that the **ovsdb-server** is listening the local IP 127.0.0.1 and port 6640.

User can configure **ovs-vswitchd** locally as shown below:

```
root@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
root@PicOS-OVS$ovs-vsctl add-port br0 qe-1/1/1 vlan_mode=trunk tag=1 -- set interface
qe-1/1/1 type=pica8
root@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac=C8:0A:A9:04:49:1A
options:dst_mac=C8:0A:A9:9E:14:A5 options:egress_port=qe-1/1/1
root@PicOS-OVS$ovs-vsctl set-controller br0 tcp:10.10.51.51:6633
```

Configuring ovsdb Remotely

Check the state of **ovsdb-server** on the switch.

```
admin@PicOS-OVS$ps aux|grep ovs
root      3422  0.0  0.1    7020  2788 ?          S     10:42   0:00 ovsdb-server
/ovs/ovs-vswitchd.conf.db /tmp/inventory.conf.db /ovs/function.conf.db --pidfile
--remote=ptcp:6640:127.0.0.1 --remote=unix:/ovs/var/run/openvswitch/db.sock
--remote=db:Open_vSwitch,Manager,target
root      3435 14.9  1.5  122860  32388 ?          Sl     10:42   3:16 ovs-vswitchd
--pidfile=ovs-vswitchd.pid --overwrite-pidfile
root      3494  0.0  0.2   12960  5752 ?          Ss     10:43   0:00 /usr/bin/python
/ovs/share/openvswitch/scripts/ovs-invd --pidfile=ovs-invd.pid --overwrite-pidfile --detach
--root-prefix=/ovs/
admin     3835  0.0  0.0    2200   680  ttys0     S+    11:04   0:00 grep --color=auto ovs
```

The string '--remote=db:Open_vSwitch,Manager,target' in the output above shows that the **ovsdb-server** is listening remote Manager and target default. User can configure switch ovsdb at remote side.

User can configure **ovs-vswitchd** remotely as shown below, switch's management ip is 10.10.51.138.

```
root@dev-42:~# ovs-vsctl --db=tcp:10.10.51.138:6640 add-br br0 -- set bridge br0
datapath_type=pica8
root@dev-42:~# ovs-vsctl --db=tcp:10.10.51.138:6640 add-port br0 qe-1/1/1 vlan_mode=trunk
tag=1 -- set interface qe-1/1/1 type=pica8
root@dev-42:~# ovs-vsctl --db=tcp:10.10.51.138:6640 add-port br0 vxlan1 -- set interface
vxlan1 type=pica8_vxlan options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1
options:vlan=1 options:vnid=1122867 options:udp_dst_port=4789
options:src_mac=C8:0A:A9:04:49:1A options:dst_mac=C8:0A:A9:9E:14:A5
options:egress_port=qe-1/1/1
root@dev-42:~# ovs-vsctl --db=tcp:10.10.51.138:6640 set-controller br0 tcp:10.10.51.51:6633
```

Add flow locally

User can add flow entry in switch cli.

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=31,actions=output:32
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  flow_id=29, cookie=0x0, duration=3.313s, table=0, n_packets=n/a, n_bytes=3481169152,
  in_port=31 actions=output:32
admin@PicOS-OVS$
```

Add flow remotely

User can add flow entry at remote side, and must configure controller before, switch's management ip is 10.10.51.138.

```
Switch,
admin@PicOS-OVS$ovs-vsctl set-controller br0 ptcp:6633:10.10.51.138

Remote server,
root@dev-42:~# ovs-ofctl add-flow tcp:10.10.51.138:6633 in_port=32,actions=output:31
root@dev-42:~#
```

- Basic Configuration in OVS Mode
- Creating Bridge and add ports
- Connecting to a Controller
- 40G Changes to 4*10G in OVS
 - 40G Changes to 4*10G in OVS mode on P-5101
 - 40G Changes to 4*10G in OVS Mode on P-5401
 - 40G Changes to 4*10G in OVS mode on accton_as6701_32x
 - 40G Changes to 4*10G in OVS mode on P-3922
 - 40G Changes to 4*10G in OVS mode on P-3920
 - 40G Changes to 4*10G in OVS mode on accton_as5712_54x/HP5712
 - 40G Changes to 4*10G in OVS mode on P-3930
 - 40G Changes to 4*10G in OVS mode on Niagara2632XL
 - 40G Changes to 4*10G in OVS Mode on as6712_32x
 - 40G Changes to 4*10G in OVS Mode on dcs7032q28
 - Port SFP

- 40G Changes to 4*10G in OVS Mode on as7712_32x
- 40G Changes to 4*10G in OVS mode on as5812_54t
- 40G Changes to 4*10G in OVS mode on arctica4806xp
- Configuring sFlow v5
- Configuring NetFlow
- Configuring Port Mirroring
- Configuring IPv4 OpenFlow
- Configure GRE Tunneling
- Configuring MPLS
- Configuring LAG and LACP
- Creating a Group Table
- Priority of Arp Group
- Configuring ECMP
- Class of Service Mapping for QoS
- Configuring QoS Queue
- Configuring OpenFlow Meter
- Configuring QinQ
- Configuring OpenFlow Provider Backbone Bridge
- Configuring Loopback
- Optimizing TCAM Usage
- Configuring Layer 2 over GRE on Trident-2 based switches
- Configuring VXLAN
- Configuring Multi-Table
 - Multitable Resources
- Configuring Network Address Translation
- ASIC Limitation
- Configuring CFM
- Open vSwitch Configuration File
- OVS LLDP
- OVS Udf
- Combinated Mode
- Enabling Radius in PicOS OVS Mode
- Creating SSL Connection to a Controller
- Configuring L2MPLS

- Inventory Database
- Configuring option-match-vlan-type
- Connecting to Manager
- OVS LFS
- Configure TPID in port
- Broadcom Chip Limitation in OVS
- Goto_table
- Clear counter
- Ovs CLI Enhancements
- Table Type Pattern
 - TTP Multicast
 - TTP Unicast
- Flow Handling Mode
- Drop counter
- ecmp-select and lag-select group
- ingress-mirror-group and egress-mirror-group
- Configuring Meter
- Configuration saving

Basic Configuration in OVS Mode

This section describes the basic configuration of PicOS switches in OVS (Open vSwitch) mode.

Once users have access to the switch and have PicOS running in OVS mode , users need to configure the IP address and default gateway.

To configure the management IP address, users should use the **picos_boot** configuration script as described in PicOS Mode Selection. An alternative to using the configuration script is to manually edit the PicOS configuration file.

For accessing the switch through a user interface, instead of the management interface, some OpenFlow flows need to be configured to redirect management traffic to the control plane. This is needed only if the switch cannot be managed through the management interface.

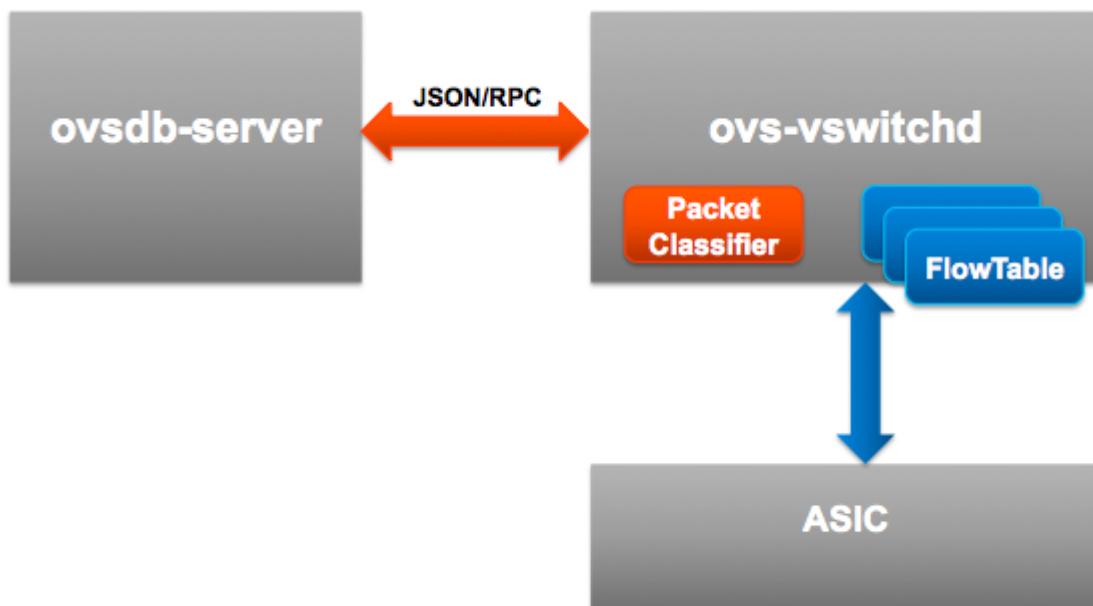
For example, users need to add the following flows to access the switch inband. The bridge **br0** has the MAC address *c8:0a:a9:04:49:19*.

```
root@PicOS-OVS#ovs-ofctl add-flow br0
priority=65300,in_port=local,dl_src=c8:0a:a9:04:49:19,actions=all
root@PicOS-OVS#ovs-ofctl add-flow br0 priority=65300,dl_dst=c8:0a:a9:04:49:19,actions=local
root@PicOS-OVS#ovs-ofctl add-flow br0
priority=65300,dl_dst=FF:FF:FF:FF:FF:FF,actions=all,local
```

Understanding OVS Components

The OVS has several components:

- **ovsdb-server:** A lightweight database server that provides switch level information (for example, information about switch ports).
- **ovs-vswitchd:** It is the core system component, which stores OpenFlow rules and performs flow based switching along with a companion Linux kernel module.
- **openvswitch_mod.ko:** It is a Linux kernel module doing most of the flow-based switching in PicOS OVS mode. This module is not loaded in the PicOS accelerated OVS, and is replaced by the ASIC (application-specific integrated circuit) for packet forwarding.
- **CLI:** The CLI (command-line interface) is used to control and manipulate other OVS components.



Understanding OVS CLI

The following three commands are used to control and monitor OVS:

- **ovs-vsctl Commands:** ovs-vsctl commands are used to control the **ovsdb-server** to create bridges, add interfaces, and configure interfaces.
- **ovs-appctl Commands:** ovs-appctl commands are used to control the **ovs-vswitchd**.
- **ovs-ofctl Commands:** ovs-ofctl commands are used to send OpenFlow queries. This can be used to manipulate the flows in **ovs-vswitchd**.

Each of those commands have a main page that can be viewed from the Linux shell on users' PicOS switch.

```

admin@Leaf1$man ovs-ofctl
ovs-ofctl(8)                         Open vSwitch Manual          ovs-ofctl(8)
NAME
    ovs-ofctl - administer OpenFlow switches
SYNOPSIS
    ovs-ofctl [options] command [switch] [args...]
DESCRIPTION

```

The ovs-ofctl program is a command line tool for monitoring and administering OpenFlow switches. It can also show the current state of an OpenFlow switch, including features, configuration, and table entries. It should work with any OpenFlow switch, not just Open vSwitch.

OpenFlow Switch Management Commands

These commands allow ovs-ofctl to monitor and administer an OpenFlow switch. It is able to show the current state of a switch, including features, configuration, and table entries.

Most of these commands take an argument that specifies the method for connecting to an OpenFlow switch. The following connection methods are supported:

<Some output omitted>

Creating Bridge and add ports

Users can create one or more bridges on a PICA8 switch. Note that each physical port can only be added to one bridge.

User can use command 'ovs-appctl pica/show' to show valid ports.

```
admin@PicOS-OVS$ovs-appctl pica/show
Max Hardware Flow Entry Limitation:
  TCAM Table      : 2048
  Egress Table    : 256
  VFilter Table   : 1024
  L2 Table        : 32768
  L3 Table        : 24000
Valid Interfaces On Switch P3290:
  Physical interfaces:
    ge-1/1/1(1)      ge-1/1/2(2)      ge-1/1/3(3)      ge-1/1/4(4)
    ge-1/1/5(5)      ge-1/1/6(6)      ge-1/1/7(7)      ge-1/1/8(8)
    ge-1/1/9(9)      ge-1/1/10(10)    ge-1/1/11(11)    ge-1/1/12(12)
    ge-1/1/13(13)    ge-1/1/14(14)    ge-1/1/15(15)    ge-1/1/16(16)
    ge-1/1/17(17)    ge-1/1/18(18)    ge-1/1/19(19)    ge-1/1/20(20)
    ge-1/1/21(21)    ge-1/1/22(22)    ge-1/1/23(23)    ge-1/1/24(24)
    ge-1/1/25(25)    ge-1/1/26(26)    ge-1/1/27(27)    ge-1/1/28(28)
    ge-1/1/29(29)    ge-1/1/30(30)    ge-1/1/31(31)    ge-1/1/32(32)
    ge-1/1/33(33)    ge-1/1/34(34)    ge-1/1/35(35)    ge-1/1/36(36)
    ge-1/1/37(37)    ge-1/1/38(38)    ge-1/1/39(39)    ge-1/1/40(40)
    ge-1/1/41(41)    ge-1/1/42(42)    ge-1/1/43(43)    ge-1/1/44(44)
    ge-1/1/45(45)    ge-1/1/46(46)    ge-1/1/47(47)    ge-1/1/48(48)
    te-1/1/49(49)    te-1/1/50(50)    te-1/1/51(51)    te-1/1/52(52)
  LAG interfaces:    ae1(1025) - ae1023(2047)
  Bond interfaces:   bond1(2049) - bond1023(3071)
  GRE interfaces:   gre1(3073) - gre1023(4095)
  VXLAN interfaces: vxlan1(4097) - vxlan1023(5119)
  L2GRE interfaces: l2gre1(5121) - l2gre1023(6143)
```

Create a Bridge

User can command " **ovs-vsctl add-br <bridge> [-- set bridge br0 datapath_type=pica8]**".

From PicOS2.6.5, due to the default datapath_type is pica8, so "**-- set bridge br0 datapath_type=pica8**" is optional.

```
root@PicOS-OV$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
root@PicOS-OVS$ovs-vsctl add-br br1
```

Adding Ports to a Bridge

In the example below, the user creates a bridge named **br0**, using the **set bridge** command. With the **add-port** command, add access ports, **ge-1/1/1** and **ge-1/1/2**, to **br0**. The default VLAN-ID for both ports is 1.

```
root@PicOS-OVS$ ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=access tag=1 -- set Interface
ge-1/1/1 type=pica8
root@PicOS-OVS$ ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=access tag=1 -- set Interface
ge-1/1/2 type=pica8
root@PicOS-OVS$
```

If the user wants to allow use of a DAC line, user should enable DAC.

```
root@PicOS-OVS$ ovs-vsctl add-port br0 te-1/1/49 vlan_mode=trunk tag=1 -- set Interface
te-1/1/49 type=pica8 options:is_dac=true
```

Adding the Default VLAN-ID

In the example below, the user adds the trunk port **ge-1/1/3** to bridge **br0** with the default VLAN-ID as 1000.

```
root@PicOS-OVS$ ovs-vsctl add-port br0 ge-1/1/3 vlan_mode=trunk tag=1000 trunks=1000 -- set
Interface ge-1/1/3 type=pica8
root@PicOS-OVS$
```

Configuring a Trunk Port

PicOS supports 802.1Q trunk ports (since PicOS 2.1). Each port has a default VLAN-ID; and by default, the VLAN-ID is 1. If user wants a port to belong to more than one VLAN, use the **vlan mode=trunk** command. When user specifies one port to a trunk port (tag=1), this port is the trunk port, the PVID is the tag number, and this port belongs to all the other VLANs (2-4094).

In the example below, the user specifies the VLAN mode to equal *trunk*, and then specifies the VLANs in the trunks:

```
root@PicOS-OVS$ ovs-vsctl add-port br0 ge-1/1/4 vlan_mode=trunk tag=1 -- set Interface
ge-1/1/4 type=pica8
root@PicOS-OVS$
```

Setting the Port Link Speed

The trunk port can carry all VLANs if the user does not specify the *trunks* field, as shown below.

```
root@PicOS-OVS$ ovs-vsctl add-port br0 te-1/1/49 vlan_mode=access tag=1 -- set Interface
te-1/1/49 type=pica8 options:link_speed=1G
root@PicOS-OVS$
```

Add Bond Ports

PicOS supports bond multiple ports as in_port or output, and this virtual port does not influence each physical port's forwarding packets. That mean the physical port which adds in bond port also can be configured as lag/gre/l2gre/vxlan and others.

Bonding port ge-1/1/1 ~ ge-1/1/4 as one port.

```
// add port
ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set Interface ge-1/1/1 type=pica8
ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1 -- set Interface ge-1/1/2 type=pica8
ovs-vsctl add-port br0 ge-1/1/3 vlan_mode=trunk tag=1 -- set Interface ge-1/1/3 type=pica8
ovs-vsctl add-port br0 ge-1/1/4 vlan_mode=trunk tag=1 -- set Interface ge-1/1/4 type=pica8

// add bond ports
ovs-vsctl add-port br0 bond1 -- set Interface bond1 type=pica8_bond
ovs-vsctl set Interface bond1 options:members=ge-1/1/2,ge-1/1/3,ge-1/1/4

// add flows
ovs-ofctl add-flow br0 in_port=2049,actions=1
ovs-ofctl add-flow br0 in_port=1,actions=2049
```

Viewing the Bridge Settings

Use the **show <bridge_name>** command to view the bridge details.

```
root@PicOS-OVS$ ovs-ofctl show br0
OFPT_FEATURES_REPLY (xid=0x1): ver:0x1, dpid:0000e89a8f503d30
n_tables:1, n_buffers:256
features: capabilities:0x87, actions:0x3f
1(ge-1/1/1): addr:e8:9a:8f:50:3d:30
config: 0
state: LINK_DOWN
current: 10MB-FD COPPER AUTO_NEG AUTO_PAUSE AUTO_PAUSE_ASYM
advertised: 10MB-FD AUTO_PAUSE
supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD AUTO_NEG AUTO_PAUSE AUTO_PAUSE_ASYM
peer: 10MB-FD AUTO_PAUSE
2(ge-1/1/2): addr:e8:9a:8f:50:3d:30
config: 0
state: LINK_DOWN
current: 10MB-FD COPPER AUTO_NEG AUTO_PAUSE AUTO_PAUSE_ASYM
advertised: 10MB-FD AUTO_PAUSE
supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD AUTO_NEG AUTO_PAUSE AUTO_PAUSE_ASYM
peer: 10MB-FD AUTO_PAUSE
3(ge-1/1/3): addr:e8:9a:8f:50:3d:30
config: 0
state: LINK_DOWN
current: 10MB-FD COPPER AUTO_NEG AUTO_PAUSE AUTO_PAUSE_ASYM
advertised: 10MB-FD AUTO_PAUSE
supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD AUTO_NEG AUTO_PAUSE AUTO_PAUSE_ASYM
peer: 10MB-FD AUTO_PAUSE
LOCAL(br0): addr:e8:9a:8f:50:3d:30
config: PORT_DOWN
state: LINK_DOWN
current: 10MB-FD COPPER
OFPT_GET_CONFIG_REPLY (xid=0x3): frags=normal miss_send_len=0
root@PicOS-OVS$ 
root@PicOS-OVS$ 
root@PicOS-OVS$ ovs-vsctl list-ports br0
ge-1/1/1
ge-1/1/2
```

```
ge-1/1/3
root@PicOS-OVS$ 
root@PicOS-OVS$ 
root@PicOS-OVS$ ovs-vsctl list-ifaces br0
ge-1/1/1
ge-1/1/2
ge-1/1/3
root@PicOS-OVS$ 
root@PicOS-OVS$
```

Deleting the Bridge

To delete the bridge and its ports, use the **del-port** command, then the **del-br <bridge_name>** command.

```
root@PicOS-OVS$ ovs-vsctl del-port br0 ge-1/1/1
root@PicOS-OVS$ ovs-vsctl del-port br0 ge-1/1/2
root@PicOS-OVS$ ovs-vsctl del-port br0 ge-1/1/3
root@PicOS-OVS$ ovs-vsctl del-br br0
```

View software table flows

Normally, we view software table flows using below command without options.

```
admin@PicOS-OVS$ ovs-ofctl dump-flows br0
```

And picos add a cli view to group flows into certain application categories (LLDP, VRRP, IP, Controller, Miscellaneous) or a user-configurable value from Picos 2.8.0. User can filter flows using the flow group name.

`ovs-ofctl dump-flows br0 --filter=<VRRP | LLDP | IPV4 | IPV6 | CONTROLLER | MISC>`

You can use one filter or more.

For example:

```
admin@PicOS-OVS$ ovs-ofctl dump-flows br0 --filter=VRRP,LLDP,IPV4,IPV6,CONTROLLER,MISC

VRRP flows (count=1):
 flow_id=9, cookie=0x0, duration=64.268s, table=0, n_packets=n/a, n_bytes=0,
 priority=22016, ip, in_port=1, dl_dst=01:00:00:00:00:00/01:00:00:00:00:00, nw_proto=112
 actions=output:2

LLDP flows (count=1):
 flow_id=6, cookie=0x0, duration=92.408s, table=0, n_packets=n/a, n_bytes=0,
 in_port=3, dl_dst=01:80:c2:00:00:0e, dl_type=0x88cc actions=output:4

IPV4 flows (count=2):
 flow_id=9, cookie=0x0, duration=64.268s, table=0, n_packets=n/a, n_bytes=0,
 priority=22016, ip, in_port=1, dl_dst=01:00:00:00:00:00/01:00:00:00:00:00, nw_proto=112
 actions=output:2
 flow_id=8, cookie=0x0, duration=72.908s, table=0, n_packets=n/a, n_bytes=0,
 ip, in_port=2, nw_dst=192.168.100.200 actions=CONTROLLER:65535

IPV6 flows (count=1):
 flow_id=10, cookie=0x0, duration=57.592s, table=0, n_packets=n/a, n_bytes=0,
 tcp6, in_port=3 actions=CONTROLLER:65535
```

```

Controller flows (count=2):
 flow_id=8, cookie=0x0, duration=72.908s, table=0, n_packets=n/a, n_bytes=0,
 ip,in_port=2,nw_dst=192.168.100.200 actions=CONTROLLER:65535
 flow_id=10, cookie=0x0, duration=57.592s, table=0, n_packets=n/a, n_bytes=0,
 tcp6,in_port=3 actions=CONTROLLER:65535

MISC flows (count=1):
 flow_id=7, cookie=0x0, duration=80.295s, table=0, n_packets=n/a, n_bytes=0,
 priority=40000,in_port=3,dl_dst=01:80:c2:00:00:02,dl_type=0x8809
 actions=set_field:00:00:00:00:22:33->eth_src,output:4
admin@PicOS-OVS$
```

Enable STP in bridge.

Defaultly, STP is disabled in ovs mode. User can use following command to enable/disable STP in bridge.

ovs-vsctl set bridge <bridge> stp_enable=<true/false>

Example:

```

admin@PicOS-OVS$ovs-vsctl set bridge br-12 stp_enable=true

admin@PicOS-OVS$ovs-ofctl show br-12
OFPT_FEATURES_REPLY (OF1.4) (xid=0x2):
dpid:0xaa5e60eb69d29cd7(12276356198467542231)
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS
OFPST_PORT_DESC reply (OF1.4) (xid=0x3):
 51(te-1/1/51): addr:60:eb:69:d2:9c:d7
    config: 0
    state: BLOCKED
    current: 10GB-FD FIBER
    advertised: 1GB-FD 10GB-FD FIBER
    supported: 1GB-FD 10GB-FD FIBER AUTO_NEG
    speed: 10000 Mbps now, 10000 Mbps max
 52(te-1/1/52): addr:60:eb:69:d2:9c:d7
    config: 0
    state: LINK_UP
    current: 10GB-FD FIBER
    advertised: 1GB-FD 10GB-FD FIBER
    supported: 1GB-FD 10GB-FD FIBER AUTO_NEG
    speed: 10000 Mbps now, 10000 Mbps max
 LOCAL(br-12): addr:60:eb:69:d2:9c:d7
    config: 0
    state: LINK_UP
    current: 10MB-FD COPPER
    supported: 10MB-FD COPPER
    speed: 10 Mbps now, 10 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.4) (xid=0x5): frags=normal miss_send_len=0
```

Connecting to a Controller

Use the OVSDB protocol to connect to a controller. The **ovs-vsctl** command requires an IP address and a port number on the OVS database server. In the example below, the switch connects to an OF controller with an IP address of 10.10.53.50, and port number of 6633.

```
root@PicOS-OVS# ovs-vsctl set-controller br0 tcp:10.10.53.50:6633
root@PicOS-OVS#
```

Verify connectivity with the controller:

```
admin@PicOS-OVS$ovs-vsctl show

101c4a95-2973-4aeb-9c0a-a04380950b4d
    Bridge "br0"
        Controller "tcp:10.10.53.50:6633"
            is_connected: true
        Port "te-1/1/50"
            tag: 1
            Interface "te-1/1/50"
                type: "pica8"
        Port "ge-1/1/48"
            tag: 1
            Interface "ge-1/1/48"
                type: "pica8"
        Port "br0"
            Interface "br0"
                type: internal
```

Check detailed controller status. A properly working controller configuration would appear as:

```
admin@PicOS-OVS$ovs-vsctl list controller

_uuid          : f35735f3-1d62-45ba-967e-59e324d1e150
auxiliary      : {}
connection_mode: []
controller_burst_limit: []
controller_rate_limit: []
enable_async_messages: []
external_ids    : {}
inactivity_probe: []
is_connected    : true
local_gateway   : []
local_ip        : []
local_netmask   : []
max_backoff     : []
other_config    : {}
role            : other
status          : {current_version="OpenFlow13", sec_since_connect="20", state=ACTIVE}
target          : "tcp:10.10.53.50:6633"
```

In the event of an error in the bridge configuration, such as a mismatch in the open flow version, the output would appear as:

```
admin@PicOS-OVS$ovs-vsctl list controller

_uuid          : f35735f3-1d62-45ba-967e-59e324d1e150
auxiliary      : {}
connection_mode: []
controller_burst_limit: []
controller_rate_limit: []
enable_async_messages: []
external_ids    : {}
inactivity_probe: []
is_connected    : true
local_gateway   : []
local_ip        : []
local_netmask   : []
```

```

max_backoff      : []
other_config     : {}
role             : other
status           : {last_error="Connection refused",
sec_since_connect="7713",sec_since_disconnect="7721", state=ACTIVE}
target           : "tcp:10.10.53.50:6633"

```

Modify Openflow Protocol

PicOS supports openflow1.0, openflow1.2, openflow1.3, openflow1.4 default, if user wants special configuration openflow protocol number, the command is below.

ovs-vsctl set bridge br0 protocol=<OpenFlow protocol number>

```

admin@PicOS-OVS$ovs-vsctl set bridge br0 protocol=OpenFlow13
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-vsctl set bridge br0 protocol=OpenFlow10,OpenFlow13

```

Connection Mode between Bridge and Controller

From PicOS 2.4, the default connection between one bridge and all the controllers is in-band mode. User can configure to disable it, using the command as shown below.

ovs-vsctl set bridge br0 other_config=disable-in-band=<true|false>

Disable in-band mode. After disable in-band is done, all the controllers must use out-of-band mode connection to bridge, which means only the management port can be used to connect.

```
ovs-vsctl set bridge br0 other_config=disable-in-band=true
```

Enable in-band mode, all the controllers can use in-band connect to bridge.

```
ovs-vsctl set bridge br0 other_config=disable-in-band=false
```

If enabling in-band in bridge, user also can configure in-band or out-of-band in one single controller. That mean that this controller only uses out-of-band, Others still use in-band connect to bridge.

ovs-vsctl set controller [_uuid] connection_mode=out-of-band

```

admin@PicOS-OVS$ovs-vsctl list controller
_uuid          : 7f651b3b-4b0d-4d9b-b6e5-fe67499be1c4
auxiliary      : {}
connection_mode : in-band
controller_burst_limit: []
controller_rate_limit: []
enable_async_messages: []
external_ids    : {}
inactivity_probe: []
is_connected    : false
local_gateway   : []
local_ip        : []
local_netmask   : []
max_backoff    : []
other_config    : {}
role            : other

```

```

status          : {last_error="Network is unreachable", state=BACKOFF}
target         : "tcp:10.10.50.42:6653"
admin@PicOS-OVS$ ovs-vsctl set controller 7f651b3b-4b0d-4d9b-b6e5-fe67499be1c4
connection_mode=out-of-band
admin@PicOS-OVS$ ovs-vsctl list controller
_uuid          : 7f651b3b-4b0d-4d9b-b6e5-fe67499be1c4
auxiliary      : {}
connection_mode : out-of-band
controller_burst_limit: []
controller_rate_limit: []
enable_async_messages: []
external_ids    : {}
inactivity_probe: []
is_connected    : false
local_gateway   : []
local_ip        : []
local_netmask   : []
max_backoff     : []
other_config    : {}
role            : other
status          : {last_error="Network is unreachable", state=BACKOFF}
target         : "tcp:10.10.50.42:6653"

```

Auxiliary Connections

PicOS OVS supports Auxiliary connections to the controller. Auxiliary connections configuration is based on the OVS controller. When the user wants to use it, they must first configure auxiliary in the controller list.

```
ovs-vsctl set controller {uuid} auxiliary:{id}={udp | tcp}
```

Auxiliary configuration has two parameters:

- The first is Auxiliary ID, an integer that identifies auxiliary connections.

And the range is [1, 255].

- The second is a string that specifies the transport of auxiliary connection.

And now we only support “udp” and “tcp”.

Example:

Step 1: Assume there is a controller “tcp:10.10.51.16:6633” on Bridge br0 using this:

Firstly, get uuid of the controller using this:

```
ovs-vsctl set-controller br0 tcp:10.10.50.50:6633
```

Step 2: Get controller uuid:

```

ovs-vsctl list controller
admin@PicOS-OVS$ ovs-vsctl list controller
_uuid          : 6eb5d036-87af-44ca-aa58-2b916ae126f4
auxiliary      : {}
connection_mode : []
controller_burst_limit: []
controller_rate_limit: []
enable_async_messages: []
external_ids    : {}

```

```

inactivity_probe      : []
is_connected          : false
local_gateway         : []
local_ip              : []
local_netmask         : []
max_backoff           : []
other_config          : {}
role                  : other
status                : {last_error="Connection refused", sec_since_disconnect="1",
state=BACKOFF}
target                : "tcp:10.10.51.50:6633"

```

Step 3:

```
admin@PicOS-OVS$ovs-vsctl set controller 6eb5d036-87af-44ca-aa58-2b916ae126f4
auxiliary:1=udp
```

Configure Flow Table Flush once set or delete controller

ovs-vsctl set bridge br0 other_config:enable-flush=<true|false>

In PicOS OVS switch, the flow table will be cleared when user uses the **set-controller** command for the bridge. Starting with PicOS 2.6, user can define if the flow table is flushed or not by the **set-controller** and **del-controller** commands.

The flow table is flushed when **enable-flush=true**. The flow table is not flushed when **enable-flush=false**. The default value is **enable-flush=true**.

```
ovs-vsctl set bridge br0 other_config:enable-flush=true
ovs-vsctl set bridge br0 other_config:enable-flush=false
```

Display the configuration.

```
ovs-vsctl list Bridge
ovs-vsctl list bridge br0
```

40G Changes to 4*10G in OVS

In this document, flexible port speed configuration in OVS mode for various switches is described. There are four port speed modes, Normal, Half, Max, and Flexible. The details of these port modes and its effect on port configuration is different for different platforms and explained below. Its important to note that not all of the switches support all the four modes as some switches do not support the half mode. For a quick reference of supported modes, please refer to Table 1.

Table 1: Port mode support for various platforms

Platform ➡	P-5401	P-5101	AS6701_32X	P-3922	P-3920	AS5712_54X	Niagara 2632XL	AS7712_32X	DCS7032Q28
Feature ➡									
Normal	✓	✓	✓	✓	✓	✓	✓	✓	✓

Platform →	P-5401	P-5101	AS6701_32X	P-3922	P-3920	AS5712_54X	Niagara 2632XL	AS7712_32X	DCS7032Q28
Feature ↓									
Half	✓	✗	✗	✗	✗	✗	✓	✗	✗
Max	✓	✓	✓	✓	✓	✓	✓	✓	✓
Flexible	✓	✓	✓	✓	✓	✓	✓	✓	✓

In OVS mode the P-5401 switch ports can be configured to one of the following settings. By default, it is in normal mode.

1. **Normal:** All 32 ports operate in 40G QSFP mode.
2. **Half:** Ports from 1-8 and 17-24 operate in 4*10G, ports 9-16 and 25-32 operate in 40G.
3. **Max :** Ports from 1-12 and 17-28 operate in 4*10G, ports from 13-16 and 29-32 operate in 40G.
4. **Flexible:** Ports from 1-12 and 17-28 support 4*10G configuration, user can use the command "**ovs-vsctl set-port-breakout 1 true**" to configure a breakout port. User must restart the switch for this configuration to take effect. Ports other than the mentioned ports can not be breakout and must operate in 40G. After applying the breakout command and restarting the switch, the names of all the ports that are set as flexible ports, changes to "**xe-1/1/...**". For example ge-1/1/1 will become xe-1/1/1 indicating this is a breakout port.

In OVS mode the P-5101 switch ports can be configured to one of the following settings. By default, it is in normal mode.

1. **Normal :** Ports from 1-40 operate in 10G, ports 41-48 operate in 40G.
2. **Max :** Ports from 1-40 operate in 10G, ports 41-48 operate in 4*10G.
3. **Flexible:** Ports from 41-48 support 4*10G, user can use the command "**ovs-vsctl set-port-breakout 1 true**" to configure a breakout port. User must restart the switch for this configuration to take effect. Ports other than the mentioned ports can not be breakout and must operate in 40G. After applying the breakout command and restarting the switch, the names of all the ports that are set as flexible ports, changes to "**xe-1/1/...**". For example ge-1/1/1 will become xe-1/1/1 indicating this is a breakout port.

In OVS mode the AS6701_32X switch ports can be configured to one of the following settings. By default, it is in normal mode.

1. **Normal:** All 32 ports operate in 40G.
2. **Max:** Ports from 5-16 and 21-32 operate in 4*10G, ports 1-4 and 17-20 operate in 40G.

3. **Flexible:** Ports from 5-16 and 21-32 supports 4*10G, user can use the command "**ovs-vsctl set-port-breakout 1 true**" to configure a breakout port. User must restart the switch for this configuration to take effect. Ports other than the mentioned ports can not be breakout and must operate in 40G. After applying the breakout command and restarting the switch, the names of all the ports that are set as flexible ports, changes to "**xe-1/1/...**". For example ge-1/1/1 will become xe-1/1/1 indicating this is a breakout port.

In OVS mode the P-3922 switch ports can be configured to one of the following settings. By default, it is in normal mode.

1. **Normal:** Ports from 1-48 operate in 10G, ports 48-52 operate in 40G.
2. **Max:** Ports from 1-48 operate in 10G, ports 48-52 operate in 4*10G.
3. **Flexible:** Ports from 48-52 support 4*10G, user can use the command "**ovs-vsctl set-port-breakout 1 true**" to configure a breakout port. User must restart the switch for this configuration to take effect. Ports other than the mentioned ports can not be breakout and must operate in 40G. After applying the breakout command and restarting the switch, the names of all the ports that are set as flexible ports, changes to "**xe-1/1/...**". For example ge-1/1/1 will become xe-1/1/1 indicating this is a breakout port.

In OVS mode, P-3920 ports can be configured to one of the following settings. By default, it is in normal mode.

1. **Normal:** Ports from 1-48 operate in 10G, ports 48-52 operate in 40G.
2. **Max:** Ports from 1-48 operate in 10G, ports 48-52 operate in 4*10G.
3. **Flexible:** Ports from 48-52 support 4*10G, user can use the command "**ovs-vsctl set-port-breakout 1 true**" to configure a breakout port. User must restart the switch for this configuration to take effect. Ports other than the mentioned ports can not be breakout and must operate in 40G. After applying the breakout command and restarting the switch, the names of all the ports that are set as flexible ports, changes to "**xe-1/1/...**". For example ge-1/1/1 will become xe-1/1/1 indicating this is a breakout port.

In OVS mode the AS5712_54X switch ports can be configured to one of the following settings. By default, it is in normal mode.

1. **Normal:** Ports from 1-48 operate in 10G, ports 48-54 operate in 40G.
2. **Max:** Ports from 1-48 operate in 10G, ports 48-54 operate in 4*10G.
3. **Flexible:** Ports from 48-54 support 4*10G, user can use the command "**ovs-vsctl set-port-breakout 1 true**" to configure a breakout port. User must restart the switch for this configuration to take effect. Ports other than the mentioned ports can not be breakout and must operate in 40G. After applying the breakout command and restarting the switch, the names of all the ports that are set as flexible ports, changes to "**xe-1/1/...**". For example ge-1/1/1 will become xe-1/1/1 indicating this is a breakout port.

In OVS mode the Niagara 2632XL switch ports can be configured to one of the following settings. By default, it is in normal mode.

1. **Normal:** All 32 ports operate in 40G QSFP mode.
2. **Half:** Ports from 1-8,17-24 operate in 4*10G, ports 9-16 and 25-32 operate in 40G.
3. **Max:** Ports from 1-12 and 17-28 operate in 4*10G, ports 13-16 and 29-32 operate in 40G.
4. **Flexible:** Ports from 1-12 and 17-28 support 4*10G, user can use the command "**ovs-vsctl set-port-breakout 1 true**" to configure a breakout port. User must restart the switch for this configuration to take effect. Ports other than the mentioned ports can not be breakout and must operate in 40G. After applying the breakout command and restarting the switch, the names of all the ports that are set as flexible ports, changes to "**xe-1/1/...**". For example ge-1/1/1 will become xe-1/1/1 indicating this is a breakout port.

In OVS mode the AS7712_32X and DCS7032Q28 switch ports can be configured to one of the following settings. By default, it is in normal mode.

1. **Normal:** All 32 ports operate in 100G QSFP mode.
2. **Max:** All 32 ports operate in 4*10G or 4*25G. When user plugs in a 40G transceiver module into the switch, the 100G port is reduced to 40G and can then be split into 4*10G ports. If user plugs in a 100G module into the switch, the total port speed remains 100G and can be split it into 4*25G. User can also configure a 100G port to 40G and then split it as 4*10G ports. This 100G to 40G downgrade is sometimes necessary if user wants connectivity between another device that does not support 100G port speed.
3. **Flexible:** All ports support 100G, 40G, 4*10G and 4*25G configurations. The default port speed is 100G, but if user plugs in a 40G transceiver module or manually set the link speed to 40G, the port will then operate in 40G mode. In 40G, user can split the port in 4*10G ports. In case of 100G, the port can be split into 4*25G ports. User can use the command "**ovs-vsctl set-port-breakout 1 true**" to configure a breakout port. User must restart the switch for this configuration to take effect. Ports other than the mentioned ports can not be breakout and must operate in 40G. After applying the breakout command and restarting the switch, the names of all the ports that are set as flexible ports, changes to "**xe-1/1/...**". For example ge-1/1/1 will become xe-1/1/1 indicating this is a breakout port.

i Note:

P-5101, AS6701_32X, P-3922, P-3920, AS7712_32X and DCS7032Q28 do not support the half mode.

PORt SFP

Since PicOS version 2.7.2, the 40G and 100G ports support flexible mode. This gives the user flexibility in choosing which ports to breakout compared to Half or Max modes where once the Half or Max mode is selected, the respective ports are put into breakout mode. User can use the command "**ovs-vsctl set-qe-port-mode flexible**" if user wants to set some ports in flexible mode. After the restarting the switch, not all QSFP ports are put into flexible mode. User can check all the valid ports by "**ovs-vsctl**

show-valid-port all" or "**ovs-vsctl show-valid-port 1**". If user does not want to use the default port name, user can rename the port by "**ovs-vsctl set-port-name 1 1=fn1.1**" "**ovs-vsctl set-port-name 1 2=fn1.2**". User can restore the default port name by "**ovs-vsctl set-port-name 1 default**" or "**ovs-vsctl set-port-name all default**". Users can set breakout true or false for each QSFP port by a new command "**ovs-vsctl set-port-breakout 49 true**". Don't forget to restart the switch after setting all the configurations.

Commands

Set port mode to flexible:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode flexible
Please reboot for the change to take effect!
```

Port breakout command:

```
#ovs-vsctl set-port-breakout 49 true
#ovs-vsctl set-port-breakout 49 false
```

If the command is successful user will see the following message:

Please reboot for the change to take effect!

If failed, then:

The device does not support this command!

i Note: Parameter "49" is the physical port number.

Set port name command:

```
ovs-vsctl set-port-name 49 1=fn49.1
ovs-vsctl set-port-name 49 2=fn49.2
```

i Note

Parameter "1" or "49" is the physical port number.

Parameter "1=xxx"/"2=xxx"/"3=xxx"/"4=xxx" is the sub port numbers.

If successful:

Please reboot for the change to take effect!

If failed:

The device does not support this command!

Show valid port command:

```
#ovs-vsctl show-valid-port all
#ovs-vsctl show-valid-port 49
```

note

Parameter "all" will display all valid ports, including ifindex, fname, breakout
 Parameter "49" is the physical port umber.

Restore default configuration command:

```
Ovs-vsctl set-port-name all default
Ovs-vsctl set-port-name 49 default
ovs-vsctl set-port-name 50 1=default
```

If success:

Please reboot for the change to take effect!

- 40G Changes to 4*10G in OVS mode on P-5101
- 40G Changes to 4*10G in OVS Mode on P-5401
- 40G Changes to 4*10G in OVS mode on accton_as6701_32x
- 40G Changes to 4*10G in OVS mode on P-3922
- 40G Changes to 4*10G in OVS mode on P-3920
- 40G Changes to 4*10G in OVS mode on accton_as5712_54x/HP5712
- 40G Changes to 4*10G in OVS mode on P-3930
- 40G Changes to 4*10G in OVS mode on Niagara2632XL
- 40G Changes to 4*10G in OVS Mode on as6712_32x
- 40G Changes to 4*10G in OVS Mode on dcs7032q28
- Port SFP
- 40G Changes to 4*10G in OVS Mode on as7712_32x
- 40G Changes to 4*10G in OVS mode on as5812_54t
- 40G Changes to 4*10G in OVS mode on arctica4806xp

OVS Mode Configuration

Users can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | max]
```

After setting ports to a different mode, it is mandatory to restart the OVS service in order to make the new state to take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

Users can take a look of the current QSFP port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

Normal (8 x 40G+40*10G)

When ports are in normal mode, the mapping between physical port number, Interface index, interface name and interface support speed are in the following table.

Physical Port number	Interface index	Interface name	Interface support speed
1	1	te-1/1/1	10Gb/s and 1Gb/s
2	2	te-1/1/2	10Gb/s and 1Gb/s
3	3	te-1/1/3	10Gb/s and 1Gb/s
4	4	te-1/1/4	10Gb/s and 1Gb/s
5	5	te-1/1/5	10Gb/s and 1Gb/s
6	6	te-1/1/6	10Gb/s and 1Gb/s
7	7	te-1/1/7	10Gb/s and 1Gb/s
8	8	te-1/1/8	10Gb/s and 1Gb/s
9	9	te-1/1/9	10Gb/s and 1Gb/s
10	10	te-1/1/10	10Gb/s and 1Gb/s
11	11	te-1/1/11	10Gb/s and 1Gb/s
12	12	te-1/1/12	10Gb/s and 1Gb/s
13	13	te-1/1/13	10Gb/s and 1Gb/s
14	14	te-1/1/14	10Gb/s and 1Gb/s
15	15	te-1/1/15	10Gb/s and 1Gb/s
16	16	te-1/1/16	10Gb/s and 1Gb/s
17	17	te-1/1/17	10Gb/s and 1Gb/s
18	18	te-1/1/18	10Gb/s and 1Gb/s
19	19	te-1/1/19	10Gb/s and 1Gb/s
20	20	te-1/1/20	10Gb/s and 1Gb/s
21	21	te-1/1/21	10Gb/s and 1Gb/s
22	22	te-1/1/22	10Gb/s and 1Gb/s
23	23	te-1/1/23	10Gb/s and 1Gb/s
24	24	te-1/1/24	10Gb/s and 1Gb/s
25	25	te-1/1/25	10Gb/s and 1Gb/s

26	26	te-1/1/26	10Gb/s and 1Gb/s
27	27	te-1/1/27	10Gb/s and 1Gb/s
28	28	te-1/1/28	10Gb/s and 1Gb/s
29	29	te-1/1/29	10Gb/s and 1Gb/s
30	30	te-1/1/30	10Gb/s and 1Gb/s
31	31	te-1/1/31	10Gb/s and 1Gb/s
32	32	te-1/1/32	10Gb/s and 1Gb/s
33	33	te-1/1/33	10Gb/s and 1Gb/s
34	34	te-1/1/34	10Gb/s and 1Gb/s
35	35	te-1/1/35	10Gb/s and 1Gb/s
36	36	te-1/1/36	10Gb/s and 1Gb/s
37	37	te-1/1/37	10Gb/s and 1Gb/s
38	38	te-1/1/38	10Gb/s and 1Gb/s
39	39	te-1/1/39	10Gb/s and 1Gb/s
40	40	te-1/1/40	10Gb/s and 1Gb/s
41	73	qe -1/1/41	40Gb/s
42	74	qe -1/1/42	40Gb/s
43	75	qe -1/1/43	40Gb/s
44	76	qe -1/1/44	40Gb/s
45	77	qe -1/1/45	40Gb/s
46	78	qe -1/1/46	40Gb/s
47	79	qe -1/1/47	40Gb/s
48	80	qe -1/1/48	40Gb/s

Max (72x 10G)

When ports are in max mode, the mapping between physical port number, Interface index, interface name and interface support speed are in the following table.

Physical Port number	Interface index	Interface name	Interface support speed
1	1	te-1/1/1	10Gb/s and 1Gb/s
2	2	te-1/1/2	10Gb/s and 1Gb/s

3	3	te-1/1/3	10Gb/s and 1Gb/s
4	4	te-1/1/4	10Gb/s and 1Gb/s
5	5	te-1/1/5	10Gb/s and 1Gb/s
6	6	te-1/1/6	10Gb/s and 1Gb/s
7	7	te-1/1/7	10Gb/s and 1Gb/s
8	8	te-1/1/8	10Gb/s and 1Gb/s
9	9	te-1/1/9	10Gb/s and 1Gb/s
10	10	te-1/1/10	10Gb/s and 1Gb/s
11	11	te-1/1/11	10Gb/s and 1Gb/s
12	12	te-1/1/12	10Gb/s and 1Gb/s
13	13	te-1/1/13	10Gb/s and 1Gb/s
14	14	te-1/1/14	10Gb/s and 1Gb/s
15	15	te-1/1/15	10Gb/s and 1Gb/s
16	16	te-1/1/16	10Gb/s and 1Gb/s
17	17	te-1/1/17	10Gb/s and 1Gb/s
18	18	te-1/1/18	10Gb/s and 1Gb/s
19	19	te-1/1/19	10Gb/s and 1Gb/s
20	20	te-1/1/20	10Gb/s and 1Gb/s
21	21	te-1/1/21	10Gb/s and 1Gb/s
22	22	te-1/1/22	10Gb/s and 1Gb/s
23	23	te-1/1/23	10Gb/s and 1Gb/s
24	24	te-1/1/24	10Gb/s and 1Gb/s
25	25	te-1/1/25	10Gb/s and 1Gb/s
26	26	te-1/1/26	10Gb/s and 1Gb/s
27	27	te-1/1/27	10Gb/s and 1Gb/s
28	28	te-1/1/28	10Gb/s and 1Gb/s
29	29	te-1/1/29	10Gb/s and 1Gb/s
30	30	te-1/1/30	10Gb/s and 1Gb/s
31	31	te-1/1/31	10Gb/s and 1Gb/s
32	32	te-1/1/32	10Gb/s and 1Gb/s

33	33	te-1/1/33	10Gb/s and 1Gb/s
34	34	te-1/1/34	10Gb/s and 1Gb/s
35	35	te-1/1/35	10Gb/s and 1Gb/s
36	36	te-1/1/36	10Gb/s and 1Gb/s
37	37	te-1/1/37	10Gb/s and 1Gb/s
38	38	te-1/1/38	10Gb/s and 1Gb/s
39	39	te-1/1/39	10Gb/s and 1Gb/s
40	40	te-1/1/40	10Gb/s and 1Gb/s
41		4 x 10G	
	41	te -1/1/41	10Gb/s
	42	te -1/1/42	10Gb/s
	43	te -1/1/43	10Gb/s
	44	te -1/1/44	10Gb/s
42		4 x 10G	
	45	te -1/1/45	10Gb/s
	46	te -1/1/46	10Gb/s
	47	te -1/1/47	10Gb/s
	48	te -1/1/48	10Gb/s
43		4 x 10G	
	49	te -1/1/49	10Gb/s
	50	te -1/1/50	10Gb/s
	51	te -1/1/51	10Gb/s
	52	te -1/1/52	10Gb/s
44		4 x 10G	
	53	te -1/1/53	10Gb/s
	54	te -1/1/54	10Gb/s
	55	te -1/1/55	10Gb/s
	56	te -1/1/56	10Gb/s
45		4 x 10G	
	57	te -1/1/57	10Gb/s

	58	te -1/1/58	10Gb/s
	59	te -1/1/59	10Gb/s
	60	te -1/1/60	10Gb/s
46		4 x 10G	
	61	te -1/1/61	10Gb/s
	62	te -1/1/62	10Gb/s
	63	te -1/1/63	10Gb/s
	64	te -1/1/64	10Gb/s
47		4 x 10G	
	65	te -1/1/65	10Gb/s
	66	te -1/1/66	10Gb/s
	67	te -1/1/67	10Gb/s
	68	te -1/1/68	10Gb/s
48		4 x 10G	
	69	te -1/1/69	10Gb/s
	70	te -1/1/70	10Gb/s
	71	te -1/1/71	10Gb/s
	72	te -1/1/72	10Gb/s

i Note

P-5101 does not support half mode.

40G Changes to 4*10G in OVS Mode on P-5401

OVS Mode Configuration

Users can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | half | max]
```

After setting ports to a different mode, it is mandatory to restart the OVS service in order to make the new state take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

Users can take a look at the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

Normal (32 x 40G)

When ports are in normal mode, the mapping between physical port number, Interface index, interface name and interface support speed are in the following table.

Physical Port Number	Interface Index	Interface Name	Interface Support Speed
1	97	qe-1/1/1	40Gb/s
2	98	qe-1/1/2	40Gb/s
3	99	qe-1/1/3	40Gb/s
4	100	qe-1/1/4	40Gb/s
5	101	qe-1/1/5	40Gb/s
6	102	qe-1/1/6	40Gb/s
7	103	qe-1/1/7	40Gb/s
8	104	qe-1/1/8	40Gb/s
9	105	qe-1/1/9	40Gb/s
10	106	qe-1/1/10	40Gb/s
11	107	qe-1/1/11	40Gb/s
12	108	qe-1/1/12	40Gb/s
13	109	qe-1/1/13	40Gb/s
14	110	qe-1/1/14	40Gb/s
15	111	qe-1/1/15	40Gb/s
16	112	qe-1/1/16	40Gb/s
17	113	qe-1/1/17	40Gb/s
18	114	qe-1/1/18	40Gb/s
19	115	qe-1/1/19	40Gb/s
20	116	qe-1/1/20	40Gb/s
21	117	qe-1/1/21	40Gb/s
22	118	qe-1/1/22	40Gb/s
23	119	qe-1/1/23	40Gb/s

24	120	qe-1/1/24	40Gb/s
25	121	qe-1/1/25	40Gb/s
26	122	qe-1/1/26	40Gb/s
27	123	qe-1/1/27	40Gb/s
28	124	qe-1/1/28	40Gb/s
29	125	qe-1/1/29	40Gb/s
30	126	qe-1/1/30	40Gb/s
31	127	qe-1/1/31	40Gb/s
32	128	qe-1/1/32	40Gb/s

Half (16 x 40G + 64 x 10G)

When ports are in half mode, the mapping between physical port number, Interface index, interface name and interface support speed are in the following table.

Physical Port Number	Interface Index	Interface Name	Interface Support Speed
1	4 x 10G		
	1	te-1/1/1	10Gb/s, 1Gb/s
	2	te-1/1/2	10Gb/s, 1Gb/s
	3	te-1/1/3	10Gb/s, 1Gb/s
	4	te-1/1/4	10Gb/s, 1Gb/s
2	4 x 10G		
	5	te-1/1/5	10Gb/s, 1Gb/s
	6	te-1/1/6	10Gb/s, 1Gb/s
	7	te-1/1/7	10Gb/s, 1Gb/s
	8	te-1/1/8	10Gb/s, 1Gb/s
3	4 x 10G		
	9	te-1/1/9	10Gb/s, 1Gb/s
	10	te-1/1/10	10Gb/s, 1Gb/s
	11	te-1/1/11	10Gb/s, 1Gb/s
	12	te-1/1/12	10Gb/s, 1Gb/s
4	4 x 10G		

	13	te-1/1/13	10Gb/s, 1Gb/s	
	14	te-1/1/14	10Gb/s, 1Gb/s	
	15	te-1/1/15	10Gb/s, 1Gb/s	
	16	te-1/1/16	10Gb/s, 1Gb/s	
5		4 x 10G		
	17	te-1/1/17	10Gb/s, 1Gb/s	
	18	te-1/1/18	10Gb/s, 1Gb/s	
	19	te-1/1/19	10Gb/s, 1Gb/s	
	20	te-1/1/20	10Gb/s, 1Gb/s	
6		4 x 10G		
	21	te-1/1/21	10Gb/s, 1Gb/s	
	22	te-1/1/22	10Gb/s, 1Gb/s	
	23	te-1/1/23	10Gb/s, 1Gb/s	
	24	te-1/1/24	10Gb/s, 1Gb/s	
7		4 x 10G		
	25	te-1/1/25	10Gb/s, 1Gb/s	
	26	te-1/1/26	10Gb/s, 1Gb/s	
	27	te-1/1/27	10Gb/s, 1Gb/s	
	28	te-1/1/28	10Gb/s, 1Gb/s	
8		4 x 10G		
	29	te-1/1/29	10Gb/s, 1Gb/s	
	30	te-1/1/30	10Gb/s, 1Gb/s	
	31	te-1/1/31	10Gb/s, 1Gb/s	
	32	te-1/1/32	10Gb/s, 1Gb/s	
9	105	qe-1/1/9	40Gb/s	
10	106	qe-1/1/10	40Gb/s	
11	107	qe-1/1/11	40Gb/s	
12	108	qe-1/1/12	40Gb/s	
13	109	qe-1/1/13	40Gb/s	
14	110	qe-1/1/14	40Gb/s	

15	111	qe-1/1/15	40Gb/s
16	112	qe-1/1/16	40Gb/s
17		4 x 10G	
	33	te-1/1/33	10Gb/s, 1Gb/s
	34	te-1/1/34	10Gb/s, 1Gb/s
	35	te-1/1/35	10Gb/s, 1Gb/s
	36	te-1/1/36	10Gb/s, 1Gb/s
18		4 x 10G	
	37	te-1/1/37	10Gb/s, 1Gb/s
	38	te-1/1/38	10Gb/s, 1Gb/s
	39	te-1/1/39	10Gb/s, 1Gb/s
	40	te-1/1/40	10Gb/s, 1Gb/s
19		4 x 10G	
	41	te-1/1/41	10Gb/s, 1Gb/s
	42	te-1/1/42	10Gb/s, 1Gb/s
	43	te-1/1/43	10Gb/s, 1Gb/s
	44	te-1/1/44	10Gb/s, 1Gb/s
20		4 x 10G	
	45	te-1/1/45	10Gb/s, 1Gb/s
	46	te-1/1/46	10Gb/s, 1Gb/s
	47	te-1/1/47	10Gb/s, 1Gb/s
	48	te-1/1/48	10Gb/s, 1Gb/s
21		4 x 10G	
	49	te-1/1/49	10Gb/s, 1Gb/s
	50	te-1/1/50	10Gb/s, 1Gb/s
	51	te-1/1/51	10Gb/s, 1Gb/s
	52	te-1/1/52	10Gb/s, 1Gb/s
22		4 x 10G	
	53	te-1/1/53	10Gb/s, 1Gb/s
	54	te-1/1/54	10Gb/s, 1Gb/s

	55	te-1/1/55	10Gb/s, 1Gb/s	
	56	te-1/1/56	10Gb/s, 1Gb/s	
23		4 x 10G		
	57	te-1/1/57	10Gb/s, 1Gb/s	
	58	te-1/1/58	10Gb/s, 1Gb/s	
	59	te-1/1/59	10Gb/s, 1Gb/s	
	60	te-1/1/60	10Gb/s, 1Gb/s	
24		4 x 10G		
	61	te-1/1/61	10Gb/s, 1Gb/s	
	62	te-1/1/62	10Gb/s, 1Gb/s	
	63	te-1/1/63	10Gb/s, 1Gb/s	
	64	te-1/1/64	10Gb/s, 1Gb/s	
25	121	qe-1/1/25	40Gb/s	
26	122	qe-1/1/26	40Gb/s	
27	123	qe-1/1/27	40Gb/s	
28	124	qe-1/1/28	40Gb/s	
29	125	qe-1/1/29	40Gb/s	
30	126	qe-1/1/30	40Gb/s	
31	127	qe-1/1/31	40Gb/s	
32	128	qe-1/1/32	40Gb/s	

Max (8 x 40G + 96 x 10G)

When ports are in max mode, the mapping between physical port number, interface index, interface name and interface support speed are in the following table.

Physical Port Number	Interface Index	Interface Name	Interface Support Speed	
1		4 x 10G		
	1	te-1/1/1	10Gb/s, 1Gb/s	
	2	te-1/1/2	10Gb/s, 1Gb/s	
	3	te-1/1/3	10Gb/s, 1Gb/s	
	4	te-1/1/4	10Gb/s, 1Gb/s	

2		4 x 10G	
	5	te-1/1/5	10Gb/s, 1Gb/s
	6	te-1/1/6	10Gb/s, 1Gb/s
	7	te-1/1/7	10Gb/s, 1Gb/s
	8	te-1/1/8	10Gb/s, 1Gb/s
3		4 x 10G	
	9	te-1/1/9	10Gb/s, 1Gb/s
	10	te-1/1/10	10Gb/s, 1Gb/s
	11	te-1/1/11	10Gb/s, 1Gb/s
	12	te-1/1/12	10Gb/s, 1Gb/s
4		4 x 10G	
	13	te-1/1/13	10Gb/s, 1Gb/s
	14	te-1/1/14	10Gb/s, 1Gb/s
	15	te-1/1/15	10Gb/s, 1Gb/s
	16	te-1/1/16	10Gb/s, 1Gb/s
5		4 x 10G	
	17	te-1/1/17	10Gb/s, 1Gb/s
	18	te-1/1/18	10Gb/s, 1Gb/s
	19	te-1/1/19	10Gb/s, 1Gb/s
	20	te-1/1/20	10Gb/s, 1Gb/s
6		4 x 10G	
	21	te-1/1/21	10Gb/s, 1Gb/s
	22	te-1/1/22	10Gb/s, 1Gb/s
	23	te-1/1/23	10Gb/s, 1Gb/s
	24	te-1/1/24	10Gb/s, 1Gb/s
7		4 x 10G	
	25	te-1/1/25	10Gb/s, 1Gb/s
	26	te-1/1/26	10Gb/s, 1Gb/s
	27	te-1/1/27	10Gb/s, 1Gb/s
	28	te-1/1/28	10Gb/s, 1Gb/s

8		4 x 10G	
	29	te-1/1/29	10Gb/s, 1Gb/s
	30	te-1/1/30	10Gb/s, 1Gb/s
	31	te-1/1/31	10Gb/s, 1Gb/s
	32	te-1/1/32	10Gb/s, 1Gb/s
9		4 x 10G	
	33	te-1/1/33	10Gb/s, 1Gb/s
	34	te-1/1/34	10Gb/s, 1Gb/s
	35	te-1/1/35	10Gb/s, 1Gb/s
	36	te-1/1/36	10Gb/s, 1Gb/s
10		4 x 10G	
	37	te-1/1/37	10Gb/s, 1Gb/s
	38	te-1/1/38	10Gb/s, 1Gb/s
	39	te-1/1/39	10Gb/s, 1Gb/s
	40	te-1/1/40	10Gb/s, 1Gb/s
11		4 x 10G	
	41	te-1/1/41	10Gb/s, 1Gb/s
	42	te-1/1/42	10Gb/s, 1Gb/s
	43	te-1/1/43	10Gb/s, 1Gb/s
	44	te-1/1/44	10Gb/s, 1Gb/s
12		4 x 10G	
	45	te-1/1/45	10Gb/s, 1Gb/s
	46	te-1/1/46	10Gb/s, 1Gb/s
	47	te-1/1/47	10Gb/s, 1Gb/s
	48	te-1/1/48	10Gb/s, 1Gb/s
13	109	qe-1/1/13	40Gb/s
14	110	qe-1/1/14	40Gb/s
15	111	qe-1/1/15	40Gb/s
16	112	qe-1/1/16	40Gb/s
17		4 x 10G	

	49	te-1/1/49	10Gb/s, 1Gb/s
	50	te-1/1/50	10Gb/s, 1Gb/s
	51	te-1/1/51	10Gb/s, 1Gb/s
	52	te-1/1/52	10Gb/s, 1Gb/s
18		4 x 10G	
	53	te-1/1/53	10Gb/s, 1Gb/s
	54	te-1/1/54	10Gb/s, 1Gb/s
	55	te-1/1/55	10Gb/s, 1Gb/s
	56	te-1/1/56	10Gb/s, 1Gb/s
19		4 x 10G	
	57	te-1/1/57	10Gb/s, 1Gb/s
	58	te-1/1/58	10Gb/s, 1Gb/s
	59	te-1/1/59	10Gb/s, 1Gb/s
	60	te-1/1/60	10Gb/s, 1Gb/s
20		4 x 10G	
	61	te-1/1/61	10Gb/s, 1Gb/s
	62	te-1/1/62	10Gb/s, 1Gb/s
	63	te-1/1/63	10Gb/s, 1Gb/s
	64	te-1/1/64	10Gb/s, 1Gb/s
21		4 x 10G	
	65	te-1/1/65	10Gb/s, 1Gb/s
	66	te-1/1/66	10Gb/s, 1Gb/s
	67	te-1/1/67	10Gb/s, 1Gb/s
	68	te-1/1/68	10Gb/s, 1Gb/s
22		4 x 10G	
	69	te-1/1/69	10Gb/s, 1Gb/s
	70	te-1/1/70	10Gb/s, 1Gb/s
	71	te-1/1/71	10Gb/s, 1Gb/s
	72	te-1/1/72	10Gb/s, 1Gb/s
23		4 x 10G	

	73	te-1/1/73	10Gb/s, 1Gb/s
	74	te-1/1/74	10Gb/s, 1Gb/s
	75	te-1/1/75	10Gb/s, 1Gb/s
	76	te-1/1/76	10Gb/s, 1Gb/s
24		4 x 10G	
	77	te-1/1/77	10Gb/s, 1Gb/s
	78	te-1/1/78	10Gb/s, 1Gb/s
	79	te-1/1/79	10Gb/s, 1Gb/s
	80	te-1/1/80	10Gb/s, 1Gb/s
25		4 x 10G	
	81	te-1/1/81	10Gb/s, 1Gb/s
	82	te-1/1/82	10Gb/s, 1Gb/s
	83	te-1/1/83	10Gb/s, 1Gb/s
	84	te-1/1/84	10Gb/s, 1Gb/s
26		4 x 10G	
	85	te-1/1/85	10Gb/s, 1Gb/s
	86	te-1/1/86	10Gb/s, 1Gb/s
	87	te-1/1/87	10Gb/s, 1Gb/s
	88	te-1/1/88	10Gb/s, 1Gb/s
27		4 x 10G	
	89	te-1/1/89	10Gb/s, 1Gb/s
	90	te-1/1/90	10Gb/s, 1Gb/s
	91	te-1/1/91	10Gb/s, 1Gb/s
	92	te-1/1/92	10Gb/s, 1Gb/s
28		4 x 10G	
	93	te-1/1/93	10Gb/s, 1Gb/s
	94	te-1/1/94	10Gb/s, 1Gb/s
	95	te-1/1/95	10Gb/s, 1Gb/s
	96	te-1/1/96	10Gb/s, 1Gb/s
29	125	qe-1/1/29	40Gb/s

30	126	qe-1/1/30	40Gb/s
31	127	qe-1/1/31	40Gb/s
32	128	qe-1/1/32	40Gb/s

OVS Mode Configuration

Users can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | max]
```

After setting ports to a different mode, it is mandatory to restart the OVS service in order to make the new state to take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

Users can take a look of the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

Normal (32 x 40G)

When ports are in normal mode, the mapping between physical ports number, Interface index, interface name and interface support speed are in the following table.

Physical Port number	Interface index	Interface name	Interface support speed
1	97	qe-1/1/1	40Gb/s
2	98	qe-1/1/2	40Gb/s
3	99	qe-1/1/3	40Gb/s
4	100	qe-1/1/4	40Gb/s
5	101	qe-1/1/5	40Gb/s
6	102	qe-1/1/6	40Gb/s
7	103	qe-1/1/7	40Gb/s
8	104	qe-1/1/8	40Gb/s
9	105	qe-1/1/9	40Gb/s
10	106	qe-1/1/10	40Gb/s
11	107	qe-1/1/11	40Gb/s
12	108	qe-1/1/12	40Gb/s

13	109	qe-1/1/13	40Gb/s
14	110	qe-1/1/14	40Gb/s
15	111	qe-1/1/15	40Gb/s
16	112	qe-1/1/16	40Gb/s
17	113	qe-1/1/17	40Gb/s
18	114	qe-1/1/18	40Gb/s
19	115	qe-1/1/19	40Gb/s
20	116	qe-1/1/20	40Gb/s
21	117	qe-1/1/21	40Gb/s
22	118	qe-1/1/22	40Gb/s
23	119	qe-1/1/23	40Gb/s
24	120	qe-1/1/24	40Gb/s
25	121	qe-1/1/25	40Gb/s
26	122	qe-1/1/26	40Gb/s
27	123	qe-1/1/27	40Gb/s
28	124	qe-1/1/28	40Gb/s
29	125	qe-1/1/29	40Gb/s
30	126	qe-1/1/30	40Gb/s
31	127	qe-1/1/31	40Gb/s
32	128	qe-1/1/32	40Gb/s

Max (8 x 40G + 96 x 10G)

When ports are in max mode, the mapping between physical ports number, Interface index, the logical ports/interfaces name and interface support speed are in the following table.

Physical Port number	Interface index	OVS port/interface name	interface support speed
1	97	qe-1/1/1	40Gb/s
2	98	qe-1/1/2	40Gb/s
3	99	qe-1/1/3	40Gb/s
4	100	qe-1/1/4	40Gb/s
5		4 x 10G	

	1	te-1/1/1	10Gb/s	
	2	te-1/1/2	10Gb/s	
	3	te-1/1/3	10Gb/s	
	4	te-1/1/4	10Gb/s	
6		4 x 10G		
	5	te-1/1/5	10Gb/s	
	6	te-1/1/6	10Gb/s	
	7	te-1/1/7	10Gb/s	
	8	te-1/1/8	10Gb/s	
7		4 x 10G		
	9	te-1/1/9	10Gb/s	
	10	te-1/1/10	10Gb/s	
	11	te-1/1/11	10Gb/s	
	12	te-1/1/12	10Gb/s	
8		4 x 10G		
	13	te-1/1/13	10Gb/s	
	14	te-1/1/14	10Gb/s	
	15	te-1/1/15	10Gb/s	
	16	te-1/1/16	10Gb/s	
9		4 x 10G		
	17	te-1/1/17	10Gb/s	
	18	te-1/1/18	10Gb/s	
	19	te-1/1/19	10Gb/s	
	20	te-1/1/20	10Gb/s	
10		4 x 10G		
	21	te-1/1/21	10Gb/s	
	22	te-1/1/22	10Gb/s	
	23	te-1/1/23	10Gb/s	
	24	te-1/1/24	10Gb/s	
11		4 x 10G		

	25	te-1/1/25	10Gb/s	
	26	te-1/1/26	10Gb/s	
	27	te-1/1/27	10Gb/s	
	28	te-1/1/28	10Gb/s	
12		4 x 10G		
	29	te-1/1/29	10Gb/s	
	30	te-1/1/30	10Gb/s	
	31	te-1/1/31	10Gb/s	
	32	te-1/1/32	10Gb/s	
13		4 x 10G		
	33	te-1/1/33	10Gb/s	
	34	te-1/1/34	10Gb/s	
	35	te-1/1/35	10Gb/s	
	36	te-1/1/36	10Gb/s	
14		4 x 10G		
	37	te-1/1/37	10Gb/s	
	38	te-1/1/38	10Gb/s	
	39	te-1/1/39	10Gb/s	
	40	te-1/1/40	10Gb/s	
15		4 x 10G		
	41	te-1/1/41	10Gb/s	
	42	te-1/1/42	10Gb/s	
	43	te-1/1/43	10Gb/s	
	44	te-1/1/44	10Gb/s	
16		4 x 10G		
	45	te-1/1/45	10Gb/s	
	46	te-1/1/46	10Gb/s	
	47	te-1/1/47	10Gb/s	
	48	te-1/1/48	10Gb/s	
17	113	qe-1/1/17	40Gb/s	

18	114	qe-1/1/18	40Gb/s	
19	115	qe-1/1/19	40Gb/s	
20	116	qe-1/1/20	40Gb/s	
21		4 x 10G		
	49	te-1/1/49	10Gb/s	
	50	te-1/1/50	10Gb/s	
	51	te-1/1/51	10Gb/s	
	52	te-1/1/52	10Gb/s	
22		4 x 10G		
	53	te-1/1/53	10Gb/s	
	54	te-1/1/54	10Gb/s	
	55	te-1/1/55	10Gb/s	
	56	te-1/1/56	10Gb/s	
23		4 x 10G		
	57	te-1/1/57	10Gb/s	
	58	te-1/1/58	10Gb/s	
	59	te-1/1/59	10Gb/s	
	60	te-1/1/60	10Gb/s	
24		4 x 10G		
	61	te-1/1/61	10Gb/s	
	62	te-1/1/62	10Gb/s	
	63	te-1/1/63	10Gb/s	
	64	te-1/1/64	10Gb/s	
25		4 x 10G		
	65	te-1/1/65	10Gb/s	
	66	te-1/1/66	10Gb/s	
	67	te-1/1/67	10Gb/s	
	68	te-1/1/68	10Gb/s	
26		4 x 10G		
	69	te-1/1/69	10Gb/s	

	70	te-1/1/70	10Gb/s
	71	te-1/1/71	10Gb/s
	72	te-1/1/72	10Gb/s
27		4 x 10G	
	73	te-1/1/73	10Gb/s
	74	te-1/1/74	10Gb/s
	75	te-1/1/75	10Gb/s
	76	te-1/1/76	10Gb/s
28		4 x 10G	
	77	te-1/1/77	10Gb/s
	78	te-1/1/78	10Gb/s
	79	te-1/1/79	10Gb/s
	80	te-1/1/80	10Gb/s
29		4 x 10G	
	81	te-1/1/81	10Gb/s
	82	te-1/1/82	10Gb/s
	83	te-1/1/83	10Gb/s
	84	te-1/1/84	10Gb/s
30		4 x 10G	
	85	te-1/1/85	10Gb/s
	86	te-1/1/86	10Gb/s
	87	te-1/1/87	10Gb/s
	88	te-1/1/88	10Gb/s
31		4 x 10G	
	89	te-1/1/89	10Gb/s
	90	te-1/1/90	10Gb/s
	91	te-1/1/91	10Gb/s
	92	te-1/1/92	10Gb/s
32		4 x 10G	
	93	te-1/1/93	10Gb/s

	94	te-1/1/94	10Gb/s
	95	te-1/1/95	10Gb/s
	96	te-1/1/96	10Gb/s

OVS mode Configuration

User can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | max]
```

After setting ports to a different mode, it is mandatory to restart the OVS service in order to make the new state to take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

User can take a look of the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

Normal(48*10G+4*40G)

When ports are in normal mode, the mapping between physical port number, interface index, interface names and interface support speed is in the following table.

Physical Port number	Interface index	Interface name	interface support speed
1	1	te-1/1/1	10Gb/s and 1Gb/s
2	2	te-1/1/2	10Gb/s and 1Gb/s
3	3	te-1/1/3	10Gb/s and 1Gb/s
4	4	te-1/1/4	10Gb/s and 1Gb/s
5	5	te-1/1/5	10Gb/s and 1Gb/s
6	6	te-1/1/6	10Gb/s and 1Gb/s
7	7	te-1/1/7	10Gb/s and 1Gb/s
8	8	te-1/1/8	10Gb/s and 1Gb/s
9	9	te-1/1/9	10Gb/s and 1Gb/s
10	10	te-1/1/10	10Gb/s and 1Gb/s
11	11	te-1/1/11	10Gb/s and 1Gb/s

12	12	te-1/1/12	10Gb/s and 1Gb/s
13	13	te-1/1/13	10Gb/s and 1Gb/s
14	14	te-1/1/14	10Gb/s and 1Gb/s
15	15	te-1/1/15	10Gb/s and 1Gb/s
16	16	te-1/1/16	10Gb/s and 1Gb/s
17	17	te-1/1/17	10Gb/s and 1Gb/s
18	18	te-1/1/18	10Gb/s and 1Gb/s
19	19	te-1/1/19	10Gb/s and 1Gb/s
20	20	te-1/1/20	10Gb/s and 1Gb/s
21	21	te-1/1/21	10Gb/s and 1Gb/s
22	22	te-1/1/22	10Gb/s and 1Gb/s
23	23	te-1/1/23	10Gb/s and 1Gb/s
24	24	te-1/1/24	10Gb/s and 1Gb/s
25	25	te-1/1/25	10Gb/s and 1Gb/s
26	26	te-1/1/26	10Gb/s and 1Gb/s
27	27	te-1/1/27	10Gb/s and 1Gb/s
28	28	te-1/1/28	10Gb/s and 1Gb/s
29	29	te-1/1/29	10Gb/s and 1Gb/s
30	30	te-1/1/30	10Gb/s and 1Gb/s
31	31	te-1/1/31	10Gb/s and 1Gb/s
32	32	te-1/1/32	10Gb/s and 1Gb/s
33	33	te-1/1/33	10Gb/s and 1Gb/s
34	34	te-1/1/34	10Gb/s and 1Gb/s
35	35	te-1/1/35	10Gb/s and 1Gb/s
36	36	te-1/1/36	10Gb/s and 1Gb/s
37	37	te-1/1/37	10Gb/s and 1Gb/s
38	38	te-1/1/38	10Gb/s and 1Gb/s
39	39	te-1/1/39	10Gb/s and 1Gb/s
40	40	te-1/1/40	10Gb/s and 1Gb/s
41	41	te-1/1/41	10Gb/s and 1Gb/s

42	42	te-1/1/42	10Gb/s and 1Gb/s
43	43	te-1/1/43	10Gb/s and 1Gb/s
44	44	te-1/1/44	10Gb/s and 1Gb/s
45	45	te-1/1/45	10Gb/s and 1Gb/s
46	46	te-1/1/46	10Gb/s and 1Gb/s
47	47	te-1/1/47	10Gb/s and 1Gb/s
48	48	te-1/1/48	10Gb/s and 1Gb/s
49	65	qe-1/1/49	40Gb/s
50	66	qe-1/1/50	40Gb/s
51	67	qe-1/1/51	40Gb/s
52	68	qe-1/1/52	40Gb/s

Max(64*10G)

When ports are in max mode, the mapping between physical port number, interface index, interface names and interface support speed is in the following table.

Physical Port number	Interface index	interface name	interface support speed
1	1	te-1/1/1	10Gb/s and 1Gb/s
2	2	te-1/1/2	10Gb/s and 1Gb/s
3	3	te-1/1/3	10Gb/s and 1Gb/s
4	4	te-1/1/4	10Gb/s and 1Gb/s
5	5	te-1/1/5	10Gb/s and 1Gb/s
6	6	te-1/1/6	10Gb/s and 1Gb/s
7	7	te-1/1/7	10Gb/s and 1Gb/s
8	8	te-1/1/8	10Gb/s and 1Gb/s
9	9	te-1/1/9	10Gb/s and 1Gb/s
10	10	te-1/1/10	10Gb/s and 1Gb/s
11	11	te-1/1/11	10Gb/s and 1Gb/s
12	12	te-1/1/12	10Gb/s and 1Gb/s
13	13	te-1/1/13	10Gb/s and 1Gb/s
14	14	te-1/1/14	10Gb/s and 1Gb/s

15	15	te-1/1/15	10Gb/s and 1Gb/s
16	16	te-1/1/16	10Gb/s and 1Gb/s
17	17	te-1/1/17	10Gb/s and 1Gb/s
18	18	te-1/1/18	10Gb/s and 1Gb/s
19	19	te-1/1/19	10Gb/s and 1Gb/s
20	20	te-1/1/20	10Gb/s and 1Gb/s
21	21	te-1/1/21	10Gb/s and 1Gb/s
22	22	te-1/1/22	10Gb/s and 1Gb/s
23	23	te-1/1/23	10Gb/s and 1Gb/s
24	24	te-1/1/24	10Gb/s and 1Gb/s
25	25	te-1/1/25	10Gb/s and 1Gb/s
26	26	te-1/1/26	10Gb/s and 1Gb/s
27	27	te-1/1/27	10Gb/s and 1Gb/s
28	28	te-1/1/28	10Gb/s and 1Gb/s
29	29	te-1/1/29	10Gb/s and 1Gb/s
30	30	te-1/1/30	10Gb/s and 1Gb/s
31	31	te-1/1/31	10Gb/s and 1Gb/s
32	32	te-1/1/32	10Gb/s and 1Gb/s
33	33	te-1/1/33	10Gb/s and 1Gb/s
34	34	te-1/1/34	10Gb/s and 1Gb/s
35	35	te-1/1/35	10Gb/s and 1Gb/s
36	36	te-1/1/36	10Gb/s and 1Gb/s
37	37	te-1/1/37	10Gb/s and 1Gb/s
38	38	te-1/1/38	10Gb/s and 1Gb/s
39	39	te-1/1/39	10Gb/s and 1Gb/s
40	40	te-1/1/40	10Gb/s and 1Gb/s
41	41	te-1/1/41	10Gb/s and 1Gb/s
42	42	te-1/1/42	10Gb/s and 1Gb/s
43	43	te-1/1/43	10Gb/s and 1Gb/s
44	44	te-1/1/44	10Gb/s and 1Gb/s

45	45	te-1/1/45	10Gb/s and 1Gb/s
46	46	te-1/1/46	10Gb/s and 1Gb/s
47	47	te-1/1/47	10Gb/s and 1Gb/s
48	48	te-1/1/48	10Gb/s and 1Gb/s
49		4 x 10G	
	49	te-1/1/49	10Gb/s
	50	te-1/1/50	10Gb/s
	51	te-1/1/51	10Gb/s
	52	te-1/1/52	10Gb/s
50		4 x 10G	
	53	te-1/1/53	10Gb/s
	54	te-1/1/54	10Gb/s
	55	te-1/1/55	10Gb/s
	56	te-1/1/56	10Gb/s
51		4 x 10G	
	57	te-1/1/57	10Gb/s
	58	te-1/1/58	10Gb/s
	59	te-1/1/59	10Gb/s
	60	te-1/1/60	10Gb/s
52		4 x 10G	
	61	te-1/1/61	10Gb/s
	62	te-1/1/62	10Gb/s
	63	te-1/1/63	10Gb/s
	64	te-1/1/64	10Gb/s

OVS mode Configuration

User can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | max]
```

After setting ports to a different mode, it is mandatory to restart the OVS service in order to make the new state to take effect.

```
admin@PicOS-OVS$ sudo service picos restart
```

User can take a look of the current port mode by issuing:

```
admin@PicOS-OVS$ ovs-vsctl show-qe-port-mode
```

Normal(48*10G+4*40G)

When ports are in normal mode, the mapping between physical port, physical port index, interface names and interfaces support speed is in the following table.

Physical Port number	physical port index	interface name	interface support speed
1	1	te-1/1/1	10Gb/s and 1Gb/s
2	2	te-1/1/2	10Gb/s and 1Gb/s
3	3	te-1/1/3	10Gb/s and 1Gb/s
4	4	te-1/1/4	10Gb/s and 1Gb/s
5	5	te-1/1/5	10Gb/s and 1Gb/s
6	6	te-1/1/6	10Gb/s and 1Gb/s
7	7	te-1/1/7	10Gb/s and 1Gb/s
8	8	te-1/1/8	10Gb/s and 1Gb/s
9	9	te-1/1/9	10Gb/s and 1Gb/s
10	10	te-1/1/10	10Gb/s and 1Gb/s
11	11	te-1/1/11	10Gb/s and 1Gb/s
12	12	te-1/1/12	10Gb/s and 1Gb/s
13	13	te-1/1/13	10Gb/s and 1Gb/s
14	14	te-1/1/14	10Gb/s and 1Gb/s
15	15	te-1/1/15	10Gb/s and 1Gb/s
16	16	te-1/1/16	10Gb/s and 1Gb/s
17	17	te-1/1/17	10Gb/s and 1Gb/s
18	18	te-1/1/18	10Gb/s and 1Gb/s
19	19	te-1/1/19	10Gb/s and 1Gb/s
20	20	te-1/1/20	10Gb/s and 1Gb/s
21	21	te-1/1/21	10Gb/s and 1Gb/s

22	22	te-1/1/22	10Gb/s and 1Gb/s
23	23	te-1/1/23	10Gb/s and 1Gb/s
24	24	te-1/1/24	10Gb/s and 1Gb/s
25	25	te-1/1/25	10Gb/s and 1Gb/s
26	26	te-1/1/26	10Gb/s and 1Gb/s
27	27	te-1/1/27	10Gb/s and 1Gb/s
28	28	te-1/1/28	10Gb/s and 1Gb/s
29	29	te-1/1/29	10Gb/s and 1Gb/s
30	30	te-1/1/30	10Gb/s and 1Gb/s
31	31	te-1/1/31	10Gb/s and 1Gb/s
32	32	te-1/1/32	10Gb/s and 1Gb/s
33	33	te-1/1/33	10Gb/s and 1Gb/s
34	34	te-1/1/34	10Gb/s and 1Gb/s
35	35	te-1/1/35	10Gb/s and 1Gb/s
36	36	te-1/1/36	10Gb/s and 1Gb/s
37	37	te-1/1/37	10Gb/s and 1Gb/s
38	38	te-1/1/38	10Gb/s and 1Gb/s
39	39	te-1/1/39	10Gb/s and 1Gb/s
40	40	te-1/1/40	10Gb/s and 1Gb/s
41	41	te-1/1/41	10Gb/s and 1Gb/s
42	42	te-1/1/42	10Gb/s and 1Gb/s
43	43	te-1/1/43	10Gb/s and 1Gb/s
44	44	te-1/1/44	10Gb/s and 1Gb/s
45	45	te-1/1/45	10Gb/s and 1Gb/s
46	46	te-1/1/46	10Gb/s and 1Gb/s
47	47	te-1/1/47	10Gb/s and 1Gb/s
48	48	te-1/1/48	10Gb/s and 1Gb/s
49	65	qe-1/1/49	40Gb/s
50	66	qe-1/1/50	40Gb/s
51	67	qe-1/1/51	40Gb/s

52	68	qe-1/1/52	40Gb/s
----	----	-----------	--------

Max(64*10G)

When ports are in max mode, the mapping between physical port, logical port index, the logical ports/interface names and interfaces support speed is in the following table.

Physical Port number	logical port index	interface name	interface support speed
1	1	te-1/1/1	10Gb/s and 1Gb/s
2	2	te-1/1/2	10Gb/s and 1Gb/s
3	3	te-1/1/3	10Gb/s and 1Gb/s
4	4	te-1/1/4	10Gb/s and 1Gb/s
5	5	te-1/1/5	10Gb/s and 1Gb/s
6	6	te-1/1/6	10Gb/s and 1Gb/s
7	7	te-1/1/7	10Gb/s and 1Gb/s
8	8	te-1/1/8	10Gb/s and 1Gb/s
9	9	te-1/1/9	10Gb/s and 1Gb/s
10	10	te-1/1/10	10Gb/s and 1Gb/s
11	11	te-1/1/11	10Gb/s and 1Gb/s
12	12	te-1/1/12	10Gb/s and 1Gb/s
13	13	te-1/1/13	10Gb/s and 1Gb/s
14	14	te-1/1/14	10Gb/s and 1Gb/s
15	15	te-1/1/15	10Gb/s and 1Gb/s
16	16	te-1/1/16	10Gb/s and 1Gb/s
17	17	te-1/1/17	10Gb/s and 1Gb/s
18	18	te-1/1/18	10Gb/s and 1Gb/s
19	19	te-1/1/19	10Gb/s and 1Gb/s
20	20	te-1/1/20	10Gb/s and 1Gb/s
21	21	te-1/1/21	10Gb/s and 1Gb/s
22	22	te-1/1/22	10Gb/s and 1Gb/s
23	23	te-1/1/23	10Gb/s and 1Gb/s
24	24	te-1/1/24	10Gb/s and 1Gb/s

25	25	te-1/1/25	10Gb/s and 1Gb/s
26	26	te-1/1/26	10Gb/s and 1Gb/s
27	27	te-1/1/27	10Gb/s and 1Gb/s
28	28	te-1/1/28	10Gb/s and 1Gb/s
29	29	te-1/1/29	10Gb/s and 1Gb/s
30	30	te-1/1/30	10Gb/s and 1Gb/s
31	31	te-1/1/31	10Gb/s and 1Gb/s
32	32	te-1/1/32	10Gb/s and 1Gb/s
33	33	te-1/1/33	10Gb/s and 1Gb/s
34	34	te-1/1/34	10Gb/s and 1Gb/s
35	35	te-1/1/35	10Gb/s and 1Gb/s
36	36	te-1/1/36	10Gb/s and 1Gb/s
37	37	te-1/1/37	10Gb/s and 1Gb/s
38	38	te-1/1/38	10Gb/s and 1Gb/s
39	39	te-1/1/39	10Gb/s and 1Gb/s
40	40	te-1/1/40	10Gb/s and 1Gb/s
41	41	te-1/1/41	10Gb/s and 1Gb/s
42	42	te-1/1/42	10Gb/s and 1Gb/s
43	43	te-1/1/43	10Gb/s and 1Gb/s
44	44	te-1/1/44	10Gb/s and 1Gb/s
45	45	te-1/1/45	10Gb/s and 1Gb/s
46	46	te-1/1/46	10Gb/s and 1Gb/s
47	47	te-1/1/47	10Gb/s and 1Gb/s
48	48	te-1/1/48	10Gb/s and 1Gb/s
49		4 x 10G	
	49	te-1/1/49	10Gb/s
	50	te-1/1/50	10Gb/s
	51	te-1/1/51	10Gb/s
	32	te-1/1/52	10Gb/s
50		4 x 10G	

	53	te-1/1/53	10Gb/s
	54	te-1/1/54	10Gb/s
	55	te-1/1/55	10Gb/s
	56	te-1/1/56	10Gb/s
51		4 x 10G	
	57	te-1/1/57	10Gb/s
	58	te-1/1/58	10Gb/s
	59	te-1/1/59	10Gb/s
	60	te-1/1/60	10Gb/s
52		4 x 10G	
	61	te-1/1/61	10Gb/s
	62	te-1/1/62	10Gb/s
	63	te-1/1/63	10Gb/s
	64	te-1/1/64	10Gb/s

OVS mode Configuration

User can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | max]
```

After setting ports to a different mode, it is mandatory to restart the OVS service in order to make the new state to take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

User can take a look of the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

Normal (48*10G+6*40G)

When ports are in normal mode, the mapping between physical port, interface index, interface names and interfaces support speed is in the following table.

Physical Port number	Interface index	Interface name	Interface support speed
1	1	te-1/1/1	10Gb/s and 1Gb/s

2	2	te-1/1/2	10Gb/s and 1Gb/s
3	3	te-1/1/3	10Gb/s and 1Gb/s
4	4	te-1/1/4	10Gb/s and 1Gb/s
5	5	te-1/1/5	10Gb/s and 1Gb/s
6	6	te-1/1/6	10Gb/s and 1Gb/s
7	7	te-1/1/7	10Gb/s and 1Gb/s
8	8	te-1/1/8	10Gb/s and 1Gb/s
9	9	te-1/1/9	10Gb/s and 1Gb/s
10	10	te-1/1/10	10Gb/s and 1Gb/s
11	11	te-1/1/11	10Gb/s and 1Gb/s
12	12	te-1/1/12	10Gb/s and 1Gb/s
13	13	te-1/1/13	10Gb/s and 1Gb/s
14	14	te-1/1/14	10Gb/s and 1Gb/s
15	15	te-1/1/15	10Gb/s and 1Gb/s
16	16	te-1/1/16	10Gb/s and 1Gb/s
17	17	te-1/1/17	10Gb/s and 1Gb/s
18	18	te-1/1/18	10Gb/s and 1Gb/s
19	19	te-1/1/19	10Gb/s and 1Gb/s
20	20	te-1/1/20	10Gb/s and 1Gb/s
21	21	te-1/1/21	10Gb/s and 1Gb/s
22	22	te-1/1/22	10Gb/s and 1Gb/s
23	23	te-1/1/23	10Gb/s and 1Gb/s
24	24	te-1/1/24	10Gb/s and 1Gb/s
25	25	te-1/1/25	10Gb/s and 1Gb/s
26	26	te-1/1/26	10Gb/s and 1Gb/s
27	27	te-1/1/27	10Gb/s and 1Gb/s
28	28	te-1/1/28	10Gb/s and 1Gb/s
29	29	te-1/1/29	10Gb/s and 1Gb/s
30	30	te-1/1/30	10Gb/s and 1Gb/s
31	31	te-1/1/31	10Gb/s and 1Gb/s

32	32	te-1/1/32	10Gb/s and 1Gb/s
33	33	te-1/1/33	10Gb/s and 1Gb/s
34	34	te-1/1/34	10Gb/s and 1Gb/s
35	35	te-1/1/35	10Gb/s and 1Gb/s
36	36	te-1/1/36	10Gb/s and 1Gb/s
37	37	te-1/1/37	10Gb/s and 1Gb/s
38	38	te-1/1/38	10Gb/s and 1Gb/s
39	39	te-1/1/39	10Gb/s and 1Gb/s
40	40	te-1/1/40	10Gb/s and 1Gb/s
41	41	te-1/1/41	10Gb/s and 1Gb/s
42	42	te-1/1/42	10Gb/s and 1Gb/s
43	43	te-1/1/43	10Gb/s and 1Gb/s
44	44	te-1/1/44	10Gb/s and 1Gb/s
45	45	te-1/1/45	10Gb/s and 1Gb/s
46	46	te-1/1/46	10Gb/s and 1Gb/s
47	47	te-1/1/47	10Gb/s and 1Gb/s
48	48	te-1/1/48	10Gb/s and 1Gb/s
49	73	qe-1/1/49	40Gb/s
50	74	qe-1/1/50	40Gb/s
51	75	qe-1/1/51	40Gb/s
52	76	qe-1/1/52	40Gb/s
53	77	qe-1/1/53	40Gb/s
54	78	qe-1/1/54	40Gb/s

Max (72*10G)

When ports are in max mode, the mapping between physical port, interface index, interface names and interfaces support speed is in the following table.

Physical Port number	interface index	interface name	interface support speed
1	1	te-1/1/1	10Gb/s and 1Gb/s
2	2	te-1/1/2	10Gb/s and 1Gb/s

3	3	te-1/1/3	10Gb/s and 1Gb/s
4	4	te-1/1/4	10Gb/s and 1Gb/s
5	5	te-1/1/5	10Gb/s and 1Gb/s
6	6	te-1/1/6	10Gb/s and 1Gb/s
7	7	te-1/1/7	10Gb/s and 1Gb/s
8	8	te-1/1/8	10Gb/s and 1Gb/s
9	9	te-1/1/9	10Gb/s and 1Gb/s
10	10	te-1/1/10	10Gb/s and 1Gb/s
11	11	te-1/1/11	10Gb/s and 1Gb/s
12	12	te-1/1/12	10Gb/s and 1Gb/s
13	13	te-1/1/13	10Gb/s and 1Gb/s
14	14	te-1/1/14	10Gb/s and 1Gb/s
15	15	te-1/1/15	10Gb/s and 1Gb/s
16	16	te-1/1/16	10Gb/s and 1Gb/s
17	17	te-1/1/17	10Gb/s and 1Gb/s
18	18	te-1/1/18	10Gb/s and 1Gb/s
19	19	te-1/1/19	10Gb/s and 1Gb/s
20	20	te-1/1/20	10Gb/s and 1Gb/s
21	21	te-1/1/21	10Gb/s and 1Gb/s
22	22	te-1/1/22	10Gb/s and 1Gb/s
23	23	te-1/1/23	10Gb/s and 1Gb/s
24	24	te-1/1/24	10Gb/s and 1Gb/s
25	25	te-1/1/25	10Gb/s and 1Gb/s
26	26	te-1/1/26	10Gb/s and 1Gb/s
27	27	te-1/1/27	10Gb/s and 1Gb/s
28	28	te-1/1/28	10Gb/s and 1Gb/s
29	29	te-1/1/29	10Gb/s and 1Gb/s
30	30	te-1/1/30	10Gb/s and 1Gb/s
31	31	te-1/1/31	10Gb/s and 1Gb/s
32	32	te-1/1/32	10Gb/s and 1Gb/s

33	33	te-1/1/33	10Gb/s and 1Gb/s
34	34	te-1/1/34	10Gb/s and 1Gb/s
35	35	te-1/1/35	10Gb/s and 1Gb/s
36	36	te-1/1/36	10Gb/s and 1Gb/s
37	37	te-1/1/37	10Gb/s and 1Gb/s
38	38	te-1/1/38	10Gb/s and 1Gb/s
39	39	te-1/1/39	10Gb/s and 1Gb/s
40	40	te-1/1/40	10Gb/s and 1Gb/s
41	41	te-1/1/41	10Gb/s and 1Gb/s
42	42	te-1/1/42	10Gb/s and 1Gb/s
43	43	te-1/1/43	10Gb/s and 1Gb/s
44	44	te-1/1/44	10Gb/s and 1Gb/s
45	45	te-1/1/45	10Gb/s and 1Gb/s
46	46	te-1/1/46	10Gb/s and 1Gb/s
47	47	te-1/1/47	10Gb/s and 1Gb/s
48	48	te-1/1/48	10Gb/s and 1Gb/s
49		4 x 10G	
	49	te-1/1/49	10Gb/s
	50	te-1/1/50	10Gb/s
	51	te-1/1/51	10Gb/s
	52	te-1/1/52	10Gb/s
50		4 x 10G	
	53	te-1/1/53	10Gb/s
	54	te-1/1/54	10Gb/s
	55	te-1/1/55	10Gb/s
	56	te-1/1/56	10Gb/s
51		4 x 10G	
	57	te-1/1/57	10Gb/s
	58	te-1/1/58	10Gb/s
	59	te-1/1/59	10Gb/s

	60	te-1/1/60	10Gb/s
52		4 x 10G	
	61	te-1/1/61	10Gb/s
	62	te-1/1/62	10Gb/s
	63	te-1/1/63	10Gb/s
	64	te-1/1/64	10Gb/s
53		4 x 10G	
	65	te-1/1/65	10Gb/s
	66	te-1/1/66	10Gb/s
	67	te-1/1/67	10Gb/s
	68	te-1/1/68	10Gb/s
54		4 x 10G	
	69	te-1/1/69	10Gb/s
	70	te-1/1/70	10Gb/s
	71	te-1/1/71	10Gb/s
	72	te-1/1/72	10Gb/s

OVS mode Configuration

User can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | half | max]
```

After setting ports to a different mode, it is mandatory to restart the OVS service in order to make the new state to take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

User can take a look of the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

Normal(48*10G+4*40G)

When ports are in normal mode, the mapping between physical port number, interface index, interface names and interface support speed is in the following table.

Physical Port number	Interface index	Interface name	interface support speed
----------------------	-----------------	----------------	-------------------------

1	1	te-1/1/1	10Gb/s , 1Gb/s and 100Mb/s
2	2	te-1/1/2	10Gb/s , 1Gb/s and 100Mb/s
3	3	te-1/1/3	10Gb/s , 1Gb/s and 100Mb/s
4	4	te-1/1/4	10Gb/s , 1Gb/s and 100Mb/s
5	5	te-1/1/5	10Gb/s , 1Gb/s and 100Mb/s
6	6	te-1/1/6	10Gb/s , 1Gb/s and 100Mb/s
7	7	te-1/1/7	10Gb/s , 1Gb/s and 100Mb/s
8	8	te-1/1/8	10Gb/s , 1Gb/s and 100Mb/s
9	9	te-1/1/9	10Gb/s , 1Gb/s and 100Mb/s
10	10	te-1/1/10	10Gb/s , 1Gb/s and 100Mb/s
11	11	te-1/1/11	10Gb/s , 1Gb/s and 100Mb/s
12	12	te-1/1/12	10Gb/s , 1Gb/s and 100Mb/s
13	13	te-1/1/13	10Gb/s , 1Gb/s and 100Mb/s
14	14	te-1/1/14	10Gb/s , 1Gb/s and 100Mb/s
15	15	te-1/1/15	10Gb/s , 1Gb/s and 100Mb/s
16	16	te-1/1/16	10Gb/s , 1Gb/s and 100Mb/s
17	17	te-1/1/17	10Gb/s , 1Gb/s and 100Mb/s
18	18	te-1/1/18	10Gb/s , 1Gb/s and 100Mb/s
19	19	te-1/1/19	10Gb/s , 1Gb/s and 100Mb/s
20	20	te-1/1/20	10Gb/s , 1Gb/s and 100Mb/s
21	21	te-1/1/21	10Gb/s , 1Gb/s and 100Mb/s
22	22	te-1/1/22	10Gb/s , 1Gb/s and 100Mb/s
23	23	te-1/1/23	10Gb/s , 1Gb/s and 100Mb/s
24	24	te-1/1/24	10Gb/s , 1Gb/s and 100Mb/s
25	25	te-1/1/25	10Gb/s , 1Gb/s and 100Mb/s
26	26	te-1/1/26	10Gb/s , 1Gb/s and 100Mb/s
27	27	te-1/1/27	10Gb/s , 1Gb/s and 100Mb/s
28	28	te-1/1/28	10Gb/s , 1Gb/s and 100Mb/s
29	29	te-1/1/29	10Gb/s , 1Gb/s and 100Mb/s
30	30	te-1/1/30	10Gb/s , 1Gb/s and 100Mb/s

31	31	te-1/1/31	10Gb/s , 1Gb/s and 100Mb/s
32	32	te-1/1/32	10Gb/s , 1Gb/s and 100Mb/s
33	33	te-1/1/33	10Gb/s , 1Gb/s and 100Mb/s
34	34	te-1/1/34	10Gb/s , 1Gb/s and 100Mb/s
35	35	te-1/1/35	10Gb/s , 1Gb/s and 100Mb/s
36	36	te-1/1/36	10Gb/s , 1Gb/s and 100Mb/s
37	37	te-1/1/37	10Gb/s , 1Gb/s and 100Mb/s
38	38	te-1/1/38	10Gb/s , 1Gb/s and 100Mb/s
39	39	te-1/1/39	10Gb/s , 1Gb/s and 100Mb/s
40	40	te-1/1/40	10Gb/s , 1Gb/s and 100Mb/s
41	41	te-1/1/41	10Gb/s , 1Gb/s and 100Mb/s
42	42	te-1/1/42	10Gb/s , 1Gb/s and 100Mb/s
43	43	te-1/1/43	10Gb/s , 1Gb/s and 100Mb/s
44	44	te-1/1/44	10Gb/s , 1Gb/s and 100Mb/s
45	45	te-1/1/45	10Gb/s , 1Gb/s and 100Mb/s
46	46	te-1/1/46	10Gb/s , 1Gb/s and 100Mb/s
47	47	te-1/1/47	10Gb/s , 1Gb/s and 100Mb/s
48	48	te-1/1/48	10Gb/s , 1Gb/s and 100Mb/s
49	65	qe-1/1/49	40Gb/s
50	66	qe-1/1/50	40Gb/s
51	67	qe-1/1/51	40Gb/s
52	68	qe-1/1/52	40Gb/s

Max(64*10G)

When ports are in max mode, the mapping between physical port number, interface index, interface names and interfaces support speed is in the following table.

Physical Port number	Interface index	Interface name	Interface support speed
1	1	te-1/1/1	10Gb/s , 1Gb/s and 100Mb/s
2	2	te-1/1/2	10Gb/s , 1Gb/s and 100Mb/s

3	3	te-1/1/3	10Gb/s , 1Gb/s and 100Mb/s
4	4	te-1/1/4	10Gb/s , 1Gb/s and 100Mb/s
5	5	te-1/1/5	10Gb/s , 1Gb/s and 100Mb/s
6	6	te-1/1/6	10Gb/s , 1Gb/s and 100Mb/s
7	7	te-1/1/7	10Gb/s , 1Gb/s and 100Mb/s
8	8	te-1/1/8	10Gb/s , 1Gb/s and 100Mb/s
9	9	te-1/1/9	10Gb/s , 1Gb/s and 100Mb/s
10	10	te-1/1/10	10Gb/s , 1Gb/s and 100Mb/s
11	11	te-1/1/11	10Gb/s , 1Gb/s and 100Mb/s
12	12	te-1/1/12	10Gb/s , 1Gb/s and 100Mb/s
13	13	te-1/1/13	10Gb/s , 1Gb/s and 100Mb/s
14	14	te-1/1/14	10Gb/s , 1Gb/s and 100Mb/s
15	15	te-1/1/15	10Gb/s , 1Gb/s and 100Mb/s
16	16	te-1/1/16	10Gb/s , 1Gb/s and 100Mb/s
17	17	te-1/1/17	10Gb/s , 1Gb/s and 100Mb/s
18	18	te-1/1/18	10Gb/s , 1Gb/s and 100Mb/s
19	19	te-1/1/19	10Gb/s , 1Gb/s and 100Mb/s
20	20	te-1/1/20	10Gb/s , 1Gb/s and 100Mb/s
21	21	te-1/1/21	10Gb/s , 1Gb/s and 100Mb/s
22	22	te-1/1/22	10Gb/s , 1Gb/s and 100Mb/s
23	23	te-1/1/23	10Gb/s , 1Gb/s and 100Mb/s
24	24	te-1/1/24	10Gb/s , 1Gb/s and 100Mb/s
25	25	te-1/1/25	10Gb/s , 1Gb/s and 100Mb/s
26	26	te-1/1/26	10Gb/s , 1Gb/s and 100Mb/s
27	27	te-1/1/27	10Gb/s , 1Gb/s and 100Mb/s
28	28	te-1/1/28	10Gb/s , 1Gb/s and 100Mb/s
29	29	te-1/1/29	10Gb/s , 1Gb/s and 100Mb/s
30	30	te-1/1/30	10Gb/s , 1Gb/s and 100Mb/s
31	31	te-1/1/31	10Gb/s , 1Gb/s and 100Mb/s
32	32	te-1/1/32	10Gb/s , 1Gb/s and 100Mb/s

33	33	te-1/1/33	10Gb/s , 1Gb/s and 100Mb/s
34	34	te-1/1/34	10Gb/s , 1Gb/s and 100Mb/s
35	35	te-1/1/35	10Gb/s , 1Gb/s and 100Mb/s
36	36	te-1/1/36	10Gb/s , 1Gb/s and 100Mb/s
37	37	te-1/1/37	10Gb/s , 1Gb/s and 100Mb/s
38	38	te-1/1/38	10Gb/s , 1Gb/s and 100Mb/s
39	39	te-1/1/39	10Gb/s , 1Gb/s and 100Mb/s
40	40	te-1/1/40	10Gb/s , 1Gb/s and 100Mb/s
41	41	te-1/1/41	10Gb/s , 1Gb/s and 100Mb/s
42	42	te-1/1/42	10Gb/s , 1Gb/s and 100Mb/s
43	43	te-1/1/43	10Gb/s , 1Gb/s and 100Mb/s
44	44	te-1/1/44	10Gb/s , 1Gb/s and 100Mb/s
45	45	te-1/1/45	10Gb/s , 1Gb/s and 100Mb/s
46	46	te-1/1/46	10Gb/s , 1Gb/s and 100Mb/s
47	47	te-1/1/47	10Gb/s , 1Gb/s and 100Mb/s
48	48	te-1/1/48	10Gb/s , 1Gb/s and 100Mb/s
49		4 x 10G	
	49	te-1/1/49	10Gb/s
	50	te-1/1/50	10Gb/s
	51	te-1/1/51	10Gb/s
	52	te-1/1/52	10Gb/s
50		4 x 10G	
	53	te-1/1/53	10Gb/s
	54	te-1/1/54	10Gb/s
	55	te-1/1/55	10Gb/s
	56	te-1/1/56	10Gb/s
51		4 x 10G	
	57	te-1/1/57	10Gb/s
	58	te-1/1/58	10Gb/s
	59	te-1/1/59	10Gb/s

	60	te-1/1/60	10Gb/s
52		4 x 10G	
	61	te-1/1/61	10Gb/s
	62	te-1/1/62	10Gb/s
	63	te-1/1/63	10Gb/s
	64	te-1/1/64	10Gb/s

OVS Mode Configuration

User can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | half | max]
```

After setting ports to a different mode, it is mandatory to restart the OVS service in order to make the new state to take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

User can take a look of the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

Normal (32 x 40G)

When ports are in normal mode, the mapping between physical port number, Interface index, interface name and interface support speed are in the following table.

Physical Port number	Interface index	Interface name	Interface support speed
1	97	qe-1/1/1	40Gb/s
2	98	qe-1/1/2	40Gb/s
3	99	qe-1/1/3	40Gb/s
4	100	qe-1/1/4	40Gb/s
5	101	qe-1/1/5	40Gb/s
6	102	qe-1/1/6	40Gb/s
7	103	qe-1/1/7	40Gb/s
8	104	qe-1/1/8	40Gb/s
9	105	qe-1/1/9	40Gb/s

10	106	qe-1/1/10	40Gb/s
11	107	qe-1/1/11	40Gb/s
12	108	qe-1/1/12	40Gb/s
13	109	qe-1/1/13	40Gb/s
14	110	qe-1/1/14	40Gb/s
15	111	qe-1/1/15	40Gb/s
16	112	qe-1/1/16	40Gb/s
17	113	qe-1/1/17	40Gb/s
18	114	qe-1/1/18	40Gb/s
19	115	qe-1/1/19	40Gb/s
20	116	qe-1/1/20	40Gb/s
21	117	qe-1/1/21	40Gb/s
22	118	qe-1/1/22	40Gb/s
23	119	qe-1/1/23	40Gb/s
24	120	qe-1/1/24	40Gb/s
25	121	qe-1/1/25	40Gb/s
26	122	qe-1/1/26	40Gb/s
27	123	qe-1/1/27	40Gb/s
28	124	qe-1/1/28	40Gb/s
29	125	qe-1/1/29	40Gb/s
30	126	qe-1/1/30	40Gb/s
31	127	qe-1/1/31	40Gb/s
32	128	qe-1/1/32	40Gb/s

Half (16 x 40G + 64 x 10G)

When ports are in half mode, the mapping between physical port number, Interface index, interface name and interface support speed are in the following table.

Physical Port number	Interface index	Interface name	Interface support speed
1		4 x 10G	
	1	te-1/1/1	10Gb/s

	2	te-1/1/2	10Gb/s
	3	te-1/1/3	10Gb/s
	4	te-1/1/4	10Gb/s
2		4 x 10G	
	5	te-1/1/5	10Gb/s
	6	te-1/1/6	10Gb/s
	7	te-1/1/7	10Gb/s
	8	te-1/1/8	10Gb/s
3		4 x 10G	
	9	te-1/1/9	10Gb/s
	10	te-1/1/10	10Gb/s
	11	te-1/1/11	10Gb/s
	12	te-1/1/12	10Gb/s
4		4 x 10G	
	13	te-1/1/13	10Gb/s
	14	te-1/1/14	10Gb/s
	15	te-1/1/15	10Gb/s
	16	te-1/1/16	10Gb/s
5		4 x 10G	
	17	te-1/1/17	10Gb/s
	18	te-1/1/18	10Gb/s
	19	te-1/1/19	10Gb/s
	20	te-1/1/20	10Gb/s
6		4 x 10G	
	21	te-1/1/21	10Gb/s
	22	te-1/1/22	10Gb/s
	23	te-1/1/23	10Gb/s
	24	te-1/1/24	10Gb/s
7		4 x 10G	
	25	te-1/1/25	10Gb/s

	26	te-1/1/26	10Gb/s
	27	te-1/1/27	10Gb/s
	28	te-1/1/28	10Gb/s
8		4 x 10G	
	29	te-1/1/29	10Gb/s
	30	te-1/1/30	10Gb/s
	31	te-1/1/31	10Gb/s
	32	te-1/1/32	10Gb/s
9	105	qe-1/1/9	40Gb/s
10	106	qe-1/1/10	40Gb/s
11	107	qe-1/1/11	40Gb/s
12	108	qe-1/1/12	40Gb/s
13	109	qe-1/1/13	40Gb/s
14	110	qe-1/1/14	40Gb/s
15	111	qe-1/1/15	40Gb/s
16	112	qe-1/1/16	40Gb/s
17		4 x 10G	
	33	te-1/1/33	10Gb/s
	34	te-1/1/34	10Gb/s
	35	te-1/1/35	10Gb/s
	36	te-1/1/36	10Gb/s
18		4 x 10G	
	37	te-1/1/37	10Gb/s
	38	te-1/1/38	10Gb/s
	39	te-1/1/39	10Gb/s
	40	te-1/1/40	10Gb/s
19		4 x 10G	
	41	te-1/1/41	10Gb/s
	42	te-1/1/42	10Gb/s
	43	te-1/1/43	10Gb/s

	44	te-1/1/44	10Gb/s
20		4 x 10G	
	45	te-1/1/45	10Gb/s
	46	te-1/1/46	10Gb/s
	47	te-1/1/47	10Gb/s
	48	te-1/1/48	10Gb/s
21		4 x 10G	
	49	te-1/1/49	10Gb/s
	50	te-1/1/50	10Gb/s
	51	te-1/1/51	10Gb/s
	52	te-1/1/52	10Gb/s
22		4 x 10G	
	53	te-1/1/53	10Gb/s
	54	te-1/1/54	10Gb/s
	55	te-1/1/55	10Gb/s
	56	te-1/1/56	10Gb/s
23		4 x 10G	
	57	te-1/1/57	10Gb/s
	58	te-1/1/58	10Gb/s
	59	te-1/1/59	10Gb/s
	60	te-1/1/60	10Gb/s
24		4 x 10G	
	61	te-1/1/61	10Gb/s
	62	te-1/1/62	10Gb/s
	63	te-1/1/63	10Gb/s
	64	te-1/1/64	10Gb/s
25	121	qe-1/1/25	40Gb/s
26	122	qe-1/1/26	40Gb/s
27	123	qe-1/1/27	40Gb/s
28	124	qe-1/1/28	40Gb/s

29	125	qe-1/1/29	40Gb/s
30	126	qe-1/1/30	40Gb/s
31	127	qe-1/1/31	40Gb/s
32	128	qe-1/1/32	40Gb/s

Max (8 x 40G + 96 x 10G)

When ports are in max mode, the mapping between physical port number, Interface index, interface name and interface support speed are in the following table.

Physical Port number	Interface index	Interface name	Interface support speed
1	4 x 10G		
	1	te-1/1/1	10Gb/s
	2	te-1/1/2	10Gb/s
	3	te-1/1/3	10Gb/s
	4	te-1/1/4	10Gb/s
2	4 x 10G		
	5	te-1/1/5	10Gb/s
	6	te-1/1/6	10Gb/s
	7	te-1/1/7	10Gb/s
	8	te-1/1/8	10Gb/s
3	4 x 10G		
	9	te-1/1/9	10Gb/s
	10	te-1/1/10	10Gb/s
	11	te-1/1/11	10Gb/s
	12	te-1/1/12	10Gb/s
4	4 x 10G		
	13	te-1/1/13	10Gb/s
	14	te-1/1/14	10Gb/s
	15	te-1/1/15	10Gb/s
	16	te-1/1/16	10Gb/s
5	4 x 10G		

	17	te-1/1/17	10Gb/s
	18	te-1/1/18	10Gb/s
	19	te-1/1/19	10Gb/s
	20	te-1/1/20	10Gb/s
6		4 x 10G	
	21	te-1/1/21	10Gb/s
	22	te-1/1/22	10Gb/s
	23	te-1/1/23	10Gb/s
	24	te-1/1/24	10Gb/s
7		4 x 10G	
	25	te-1/1/25	10Gb/s
	26	te-1/1/26	10Gb/s
	27	te-1/1/27	10Gb/s
	28	te-1/1/28	10Gb/s
8		4 x 10G	
	29	te-1/1/29	10Gb/s
	30	te-1/1/30	10Gb/s
	31	te-1/1/31	10Gb/s
	32	te-1/1/32	10Gb/s
9		4 x 10G	
	33	te-1/1/33	10Gb/s
	34	te-1/1/34	10Gb/s
	35	te-1/1/35	10Gb/s
	36	te-1/1/36	10Gb/s
10		4 x 10G	
	37	te-1/1/37	10Gb/s
	38	te-1/1/38	10Gb/s
	39	te-1/1/39	10Gb/s
	40	te-1/1/40	10Gb/s
11		4 x 10G	

	41	te-1/1/41	10Gb/s
	42	te-1/1/42	10Gb/s
	43	te-1/1/43	10Gb/s
	44	te-1/1/44	10Gb/s
12		4 x 10G	
	45	te-1/1/45	10Gb/s
	46	te-1/1/46	10Gb/s
	47	te-1/1/47	10Gb/s
	48	te-1/1/48	10Gb/s
13	109	qe-1/1/13	40Gb/s
14	110	qe-1/1/14	40Gb/s
15	111	qe-1/1/15	40Gb/s
16	112	qe-1/1/16	40Gb/s
17		4 x 10G	
	49	te-1/1/49	10Gb/s
	50	te-1/1/50	10Gb/s
	51	te-1/1/51	10Gb/s
	52	te-1/1/52	10Gb/s
18		4 x 10G	
	53	te-1/1/53	10Gb/s
	54	te-1/1/54	10Gb/s
	55	te-1/1/55	10Gb/s
	56	te-1/1/56	10Gb/s
19		4 x 10G	
	57	te-1/1/57	10Gb/s
	58	te-1/1/58	10Gb/s
	59	te-1/1/59	10Gb/s
	60	te-1/1/60	10Gb/s
20		4 x 10G	
	61	te-1/1/61	10Gb/s

	62	te-1/1/62	10Gb/s
	63	te-1/1/63	10Gb/s
	64	te-1/1/64	10Gb/s
21		4 x 10G	
	65	te-1/1/65	10Gb/s
	66	te-1/1/66	10Gb/s
	67	te-1/1/67	10Gb/s
	68	te-1/1/68	10Gb/s
22		4 x 10G	
	69	te-1/1/69	10Gb/s
	70	te-1/1/70	10Gb/s
	71	te-1/1/71	10Gb/s
	72	te-1/1/72	10Gb/s
23		4 x 10G	
	73	te-1/1/73	10Gb/s
	74	te-1/1/74	10Gb/s
	75	te-1/1/75	10Gb/s
	76	te-1/1/76	10Gb/s
24		4 x 10G	
	77	te-1/1/77	10Gb/s
	78	te-1/1/78	10Gb/s
	79	te-1/1/79	10Gb/s
	80	te-1/1/80	10Gb/s
25		4 x 10G	
	81	te-1/1/81	10Gb/s
	82	te-1/1/82	10Gb/s
	83	te-1/1/83	10Gb/s
	84	te-1/1/84	10Gb/s
26		4 x 10G	
	85	te-1/1/85	10Gb/s

	86	te-1/1/86	10Gb/s
	87	te-1/1/87	10Gb/s
	88	te-1/1/88	10Gb/s
27		4 x 10G	
	89	te-1/1/89	10Gb/s
	90	te-1/1/90	10Gb/s
	91	te-1/1/91	10Gb/s
	92	te-1/1/92	10Gb/s
28		4 x 10G	
	93	te-1/1/93	10Gb/s
	94	te-1/1/94	10Gb/s
	95	te-1/1/95	10Gb/s
	96	te-1/1/96	10Gb/s
29	125	qe-1/1/29	40Gb/s
30	126	qe-1/1/30	40Gb/s
31	127	qe-1/1/31	40Gb/s
32	128	qe-1/1/32	40Gb/s

OVS mode Configuration

User can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | half | max]
```

After setting ports to a different mode, it is mandatory to restart the OVS service in order to make the new state take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

User can take a look at the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

Normal (32 x 40G)

When ports are in normal mode, the mapping between physical port number, interface index, interface name and interface support speed are in the following table.

Physical Port Number	Interface Index	Interface Name	Interface Support Speed
1	97	qe-1/1/1	40Gb/s
2	98	qe-1/1/2	40Gb/s
3	99	qe-1/1/3	40Gb/s
4	100	qe-1/1/4	40Gb/s
5	101	qe-1/1/5	40Gb/s
6	102	qe-1/1/6	40Gb/s
7	103	qe-1/1/7	40Gb/s
8	104	qe-1/1/8	40Gb/s
9	105	qe-1/1/9	40Gb/s
10	106	qe-1/1/10	40Gb/s
11	107	qe-1/1/11	40Gb/s
12	108	qe-1/1/12	40Gb/s
13	109	qe-1/1/13	40Gb/s
14	110	qe-1/1/14	40Gb/s
15	111	qe-1/1/15	40Gb/s
16	112	qe-1/1/16	40Gb/s
17	113	qe-1/1/17	40Gb/s
18	114	qe-1/1/18	40Gb/s
19	115	qe-1/1/19	40Gb/s
20	116	qe-1/1/20	40Gb/s
21	117	qe-1/1/21	40Gb/s
22	118	qe-1/1/22	40Gb/s
23	119	qe-1/1/23	40Gb/s
24	120	qe-1/1/24	40Gb/s
25	121	qe-1/1/25	40Gb/s

26	122	qe-1/1/26	40Gb/s
27	123	qe-1/1/27	40Gb/s
28	124	qe-1/1/28	40Gb/s
29	125	qe-1/1/29	40Gb/s
30	126	qe-1/1/30	40Gb/s
31	127	qe-1/1/31	40Gb/s
32	128	qe-1/1/32	40Gb/s

Half (16 x 40G + 64 x 10G)

When ports are in half mode, the mapping between physical port number, Interface index, interface name and interface support speed are in the following table.

Physical Port Number	Interface Index	Interface Name	Interface Support Speed
1		4 x 10G	
	1	te-1/1/1	10Gb/s, 1Gb/s
	2	te-1/1/2	10Gb/s, 1Gb/s
	3	te-1/1/3	10Gb/s, 1Gb/s
	4	te-1/1/4	10Gb/s, 1Gb/s
2		4 x 10G	
	5	te-1/1/5	10Gb/s, 1Gb/s
	6	te-1/1/6	10Gb/s, 1Gb/s
	7	te-1/1/7	10Gb/s, 1Gb/s
	8	te-1/1/8	10Gb/s, 1Gb/s
3		4 x 10G	
	9	te-1/1/9	10Gb/s, 1Gb/s
	10	te-1/1/10	10Gb/s, 1Gb/s
	11	te-1/1/11	10Gb/s, 1Gb/s
	12	te-1/1/12	10Gb/s, 1Gb/s
4		4 x 10G	
	13	te-1/1/13	10Gb/s, 1Gb/s
	14	te-1/1/14	10Gb/s, 1Gb/s

	15	te-1/1/15	10Gb/s, 1Gb/s
	16	te-1/1/16	10Gb/s, 1Gb/s
5		4 x 10G	
	17	te-1/1/17	10Gb/s, 1Gb/s
	18	te-1/1/18	10Gb/s, 1Gb/s
	19	te-1/1/19	10Gb/s, 1Gb/s
	20	te-1/1/20	10Gb/s, 1Gb/s
6		4 x 10G	
	21	te-1/1/21	10Gb/s, 1Gb/s
	22	te-1/1/22	10Gb/s, 1Gb/s
	23	te-1/1/23	10Gb/s, 1Gb/s
	24	te-1/1/24	10Gb/s, 1Gb/s
7		4 x 10G	
	25	te-1/1/25	10Gb/s, 1Gb/s
	26	te-1/1/26	10Gb/s, 1Gb/s
	27	te-1/1/27	10Gb/s, 1Gb/s
	28	te-1/1/28	10Gb/s, 1Gb/s
8		4 x 10G	
	29	te-1/1/29	10Gb/s, 1Gb/s
	30	te-1/1/30	10Gb/s, 1Gb/s
	31	te-1/1/31	10Gb/s, 1Gb/s
	32	te-1/1/32	10Gb/s, 1Gb/s
9	105	qe-1/1/9	40Gb/s
10	106	qe-1/1/10	40Gb/s
11	107	qe-1/1/11	40Gb/s
12	108	qe-1/1/12	40Gb/s
13	109	qe-1/1/13	40Gb/s
14	110	qe-1/1/14	40Gb/s
15	111	qe-1/1/15	40Gb/s
16	112	qe-1/1/16	40Gb/s

17		4 x 10G	
	33	te-1/1/33	10Gb/s, 1Gb/s
	34	te-1/1/34	10Gb/s, 1Gb/s
	35	te-1/1/35	10Gb/s, 1Gb/s
	36	te-1/1/36	10Gb/s, 1Gb/s
18		4 x 10G	
	37	te-1/1/37	10Gb/s, 1Gb/s
	38	te-1/1/38	10Gb/s, 1Gb/s
	39	te-1/1/39	10Gb/s, 1Gb/s
	40	te-1/1/40	10Gb/s, 1Gb/s
19		4 x 10G	
	41	te-1/1/41	10Gb/s, 1Gb/s
	42	te-1/1/42	10Gb/s, 1Gb/s
	43	te-1/1/43	10Gb/s, 1Gb/s
	44	te-1/1/44	10Gb/s, 1Gb/s
20		4 x 10G	
	45	te-1/1/45	10Gb/s, 1Gb/s
	46	te-1/1/46	10Gb/s, 1Gb/s
	47	te-1/1/47	10Gb/s, 1Gb/s
	48	te-1/1/48	10Gb/s, 1Gb/s
21		4 x 10G	
	49	te-1/1/49	10Gb/s, 1Gb/s
	50	te-1/1/50	10Gb/s, 1Gb/s
	51	te-1/1/51	10Gb/s, 1Gb/s
	52	te-1/1/52	10Gb/s, 1Gb/s
22		4 x 10G	
	53	te-1/1/53	10Gb/s, 1Gb/s
	54	te-1/1/54	10Gb/s, 1Gb/s
	55	te-1/1/55	10Gb/s, 1Gb/s
	56	te-1/1/56	10Gb/s, 1Gb/s

23		4 x 10G	
	57	te-1/1/57	10Gb/s, 1Gb/s
	58	te-1/1/58	10Gb/s, 1Gb/s
	59	te-1/1/59	10Gb/s, 1Gb/s
	60	te-1/1/60	10Gb/s, 1Gb/s
24		4 x 10G	
	61	te-1/1/61	10Gb/s, 1Gb/s
	62	te-1/1/62	10Gb/s, 1Gb/s
	63	te-1/1/63	10Gb/s, 1Gb/s
	64	te-1/1/64	10Gb/s, 1Gb/s
25	121	qe-1/1/25	40Gb/s
26	122	qe-1/1/26	40Gb/s
27	123	qe-1/1/27	40Gb/s
28	124	qe-1/1/28	40Gb/s
29	125	qe-1/1/29	40Gb/s
30	126	qe-1/1/30	40Gb/s
31	127	qe-1/1/31	40Gb/s
32	128	qe-1/1/32	40Gb/s

Max (8 x 40G + 96 x 10G)

When ports are in max mode, the mapping between physical port number, Interface index, interface name and interface support speed are in the following table.

Physical Port number	Interface Index	Interface Name	Interface Support Speed
1		4 x 10G	
	1	te-1/1/1	10Gb/s, 1Gb/s
	2	te-1/1/2	10Gb/s, 1Gb/s
	3	te-1/1/3	10Gb/s, 1Gb/s
	4	te-1/1/4	10Gb/s, 1Gb/s
2		4 x 10G	
	5	te-1/1/5	10Gb/s, 1Gb/s

	6	te-1/1/6	10Gb/s, 1Gb/s
	7	te-1/1/7	10Gb/s, 1Gb/s
	8	te-1/1/8	10Gb/s, 1Gb/s
3		4 x 10G	
	9	te-1/1/9	10Gb/s, 1Gb/s
	10	te-1/1/10	10Gb/s, 1Gb/s
	11	te-1/1/11	10Gb/s, 1Gb/s
	12	te-1/1/12	10Gb/s, 1Gb/s
4		4 x 10G	
	13	te-1/1/13	10Gb/s, 1Gb/s
	14	te-1/1/14	10Gb/s, 1Gb/s
	15	te-1/1/15	10Gb/s, 1Gb/s
	16	te-1/1/16	10Gb/s, 1Gb/s
5		4 x 10G	
	17	te-1/1/17	10Gb/s, 1Gb/s
	18	te-1/1/18	10Gb/s, 1Gb/s
	19	te-1/1/19	10Gb/s, 1Gb/s
	20	te-1/1/20	10Gb/s, 1Gb/s
6		4 x 10G	
	21	te-1/1/21	10Gb/s, 1Gb/s
	22	te-1/1/22	10Gb/s, 1Gb/s
	23	te-1/1/23	10Gb/s, 1Gb/s
	24	te-1/1/24	10Gb/s, 1Gb/s
7		4 x 10G	
	25	te-1/1/25	10Gb/s, 1Gb/s
	26	te-1/1/26	10Gb/s, 1Gb/s
	27	te-1/1/27	10Gb/s, 1Gb/s
	28	te-1/1/28	10Gb/s, 1Gb/s
8		4 x 10G	
	29	te-1/1/29	10Gb/s, 1Gb/s

	30	te-1/1/30	10Gb/s, 1Gb/s
	31	te-1/1/31	10Gb/s, 1Gb/s
	32	te-1/1/32	10Gb/s, 1Gb/s
9		4 x 10G	
	33	te-1/1/33	10Gb/s, 1Gb/s
	34	te-1/1/34	10Gb/s, 1Gb/s
	35	te-1/1/35	10Gb/s, 1Gb/s
	36	te-1/1/36	10Gb/s, 1Gb/s
10		4 x 10G	
	37	te-1/1/37	10Gb/s, 1Gb/s
	38	te-1/1/38	10Gb/s, 1Gb/s
	39	te-1/1/39	10Gb/s, 1Gb/s
	40	te-1/1/40	10Gb/s, 1Gb/s
11		4 x 10G	
	41	te-1/1/41	10Gb/s, 1Gb/s
	42	te-1/1/42	10Gb/s, 1Gb/s
	43	te-1/1/43	10Gb/s, 1Gb/s
	44	te-1/1/44	10Gb/s, 1Gb/s
12		4 x 10G	
	45	te-1/1/45	10Gb/s, 1Gb/s
	46	te-1/1/46	10Gb/s, 1Gb/s
	47	te-1/1/47	10Gb/s, 1Gb/s
	48	te-1/1/48	10Gb/s, 1Gb/s
13	109	qe-1/1/13	40Gb/s
14	110	qe-1/1/14	40Gb/s
15	111	qe-1/1/15	40Gb/s
16	112	qe-1/1/16	40Gb/s
17		4 x 10G	
	49	te-1/1/49	10Gb/s, 1Gb/s
	50	te-1/1/50	10Gb/s, 1Gb/s

	51	te-1/1/51	10Gb/s, 1Gb/s
	52	te-1/1/52	10Gb/s, 1Gb/s
18		4 x 10G	
	53	te-1/1/53	10Gb/s, 1Gb/s
	54	te-1/1/54	10Gb/s, 1Gb/s
	55	te-1/1/55	10Gb/s, 1Gb/s
	56	te-1/1/56	10Gb/s, 1Gb/s
19		4 x 10G	
	57	te-1/1/57	10Gb/s, 1Gb/s
	58	te-1/1/58	10Gb/s, 1Gb/s
	59	te-1/1/59	10Gb/s, 1Gb/s
	60	te-1/1/60	10Gb/s, 1Gb/s
20		4 x 10G	
	61	te-1/1/61	10Gb/s, 1Gb/s
	62	te-1/1/62	10Gb/s, 1Gb/s
	63	te-1/1/63	10Gb/s, 1Gb/s
	64	te-1/1/64	10Gb/s, 1Gb/s
21		4 x 10G	
	65	te-1/1/65	10Gb/s, 1Gb/s
	66	te-1/1/66	10Gb/s, 1Gb/s
	67	te-1/1/67	10Gb/s, 1Gb/s
	68	te-1/1/68	10Gb/s, 1Gb/s
22		4 x 10G	
	69	te-1/1/69	10Gb/s, 1Gb/s
	70	te-1/1/70	10Gb/s, 1Gb/s
	71	te-1/1/71	10Gb/s, 1Gb/s
	72	te-1/1/72	10Gb/s, 1Gb/s
23		4 x 10G	
	73	te-1/1/73	10Gb/s, 1Gb/s
	74	te-1/1/74	10Gb/s, 1Gb/s

	75	te-1/1/75	10Gb/s, 1Gb/s
	76	te-1/1/76	10Gb/s, 1Gb/s
24		4 x 10G	
	77	te-1/1/77	10Gb/s, 1Gb/s
	78	te-1/1/78	10Gb/s, 1Gb/s
	79	te-1/1/79	10Gb/s, 1Gb/s
	80	te-1/1/80	10Gb/s, 1Gb/s
25		4 x 10G	
	81	te-1/1/81	10Gb/s, 1Gb/s
	82	te-1/1/82	10Gb/s, 1Gb/s
	83	te-1/1/83	10Gb/s, 1Gb/s
	84	te-1/1/84	10Gb/s, 1Gb/s
26		4 x 10G	
	85	te-1/1/85	10Gb/s, 1Gb/s
	86	te-1/1/86	10Gb/s, 1Gb/s
	87	te-1/1/87	10Gb/s, 1Gb/s
	88	te-1/1/88	10Gb/s, 1Gb/s
27		4 x 10G	
	89	te-1/1/89	10Gb/s, 1Gb/s
	90	te-1/1/90	10Gb/s, 1Gb/s
	91	te-1/1/91	10Gb/s, 1Gb/s
	92	te-1/1/92	10Gb/s, 1Gb/s
28		4 x 10G	
	93	te-1/1/93	10Gb/s, 1Gb/s
	94	te-1/1/94	10Gb/s, 1Gb/s
	95	te-1/1/95	10Gb/s, 1Gb/s
	96	te-1/1/96	10Gb/s, 1Gb/s
29	125	qe-1/1/29	40Gb/s
30	126	qe-1/1/30	40Gb/s
31	127	qe-1/1/31	40Gb/s

32	128	qe-1/1/32	40Gb/s
----	-----	-----------	--------

OVS Mode Configuration

User can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | max]
```

After setting ports to a different mode, it is mandatory to restart the OVS service in order to make the new state take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

User can take a look of the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

Normal: 32 x 100G (or 40G)

When ports are in normal mode, the mapping between physical port number, interface index, interface name and interface support speed are in the following table.

Physical Port Number	Interface Index	Interface Name	Interface Support Speed
1	129	he-1/1/1	100Gb/s, 40Gb/s
2	130	he-1/1/2	100Gb/s, 40Gb/s
3	131	he-1/1/3	100Gb/s, 40Gb/s
4	132	he-1/1/4	100Gb/s, 40Gb/s
5	133	he-1/1/5	100Gb/s, 40Gb/s
6	134	he-1/1/6	100Gb/s, 40Gb/s
7	135	he-1/1/7	100Gb/s, 40Gb/s
8	136	he-1/1/8	100Gb/s, 40Gb/s
9	137	he-1/1/9	100Gb/s, 40Gb/s
10	138	he-1/1/10	100Gb/s, 40Gb/s
11	139	he-1/1/11	100Gb/s, 40Gb/s
12	140	he-1/1/12	100Gb/s, 40Gb/s
13	141	he-1/1/13	100Gb/s, 40Gb/s
14	142	he-1/1/14	100Gb/s, 40Gb/s

15	143	he-1/1/15	100Gb/s, 40Gb/s
16	144	he-1/1/16	100Gb/s, 40Gb/s
17	145	he-1/1/17	100Gb/s, 40Gb/s
18	146	he-1/1/18	100Gb/s, 40Gb/s
19	147	he-1/1/19	100Gb/s, 40Gb/s
20	148	he-1/1/20	100Gb/s, 40Gb/s
21	149	he-1/1/21	100Gb/s, 40Gb/s
22	150	he-1/1/22	100Gb/s, 40Gb/s
23	151	he-1/1/23	100Gb/s, 40Gb/s
24	152	he-1/1/24	100Gb/s, 40Gb/s
25	153	he-1/1/25	100Gb/s, 40Gb/s
26	154	he-1/1/26	100Gb/s, 40Gb/s
27	155	he-1/1/27	100Gb/s, 40Gb/s
28	156	he-1/1/28	100Gb/s, 40Gb/s
29	157	he-1/1/29	100Gb/s, 40Gb/s
30	158	he-1/1/30	100Gb/s, 40Gb/s
31	159	he-1/1/31	100Gb/s, 40Gb/s
32	160	he-1/1/32	100Gb/s, 40Gb/s

SFP/max: 128 x 25G (or 10G)

When ports are in SFP/max mode, the mapping between physical port number, interface name and interface support speed are in the following table.

Physical Port Number	Interface Index	Interface Name	Interface Support Speed
1		4 x 25G, 4 x 10G	
	1	te-1/1/1	25Gb/s,10Gb/s
	2	te-1/1/2	25Gb/s,10Gb/s
	3	te-1/1/3	25Gb/s,10Gb/s
	4	te-1/1/4	25Gb/s,10Gb/s
2		4 x 25G, 4 x 10G	
	5	te-1/1/5	25Gb/s,10Gb/s

	6	te-1/1/6	25Gb/s,10Gb/s
	7	te-1/1/7	25Gb/s,10Gb/s
	8	te-1/1/8	25Gb/s,10Gb/s
3	4 x 25G, 4 x 10G		
	9	te-1/1/9	25Gb/s,10Gb/s
	10	te-1/1/10	25Gb/s,10Gb/s
	11	te-1/1/11	25Gb/s,10Gb/s
	12	te-1/1/12	25Gb/s,10Gb/s
4	4 x 25G, 4 x 10G		
	13	te-1/1/13	25Gb/s,10Gb/s
	14	te-1/1/14	25Gb/s,10Gb/s
	15	te-1/1/15	25Gb/s,10Gb/s
	16	te-1/1/16	25Gb/s,10Gb/s
5	4 x 25G, 4 x 10G		
	17	te-1/1/17	25Gb/s,10Gb/s
	18	te-1/1/18	25Gb/s,10Gb/s
	19	te-1/1/19	25Gb/s,10Gb/s
	20	te-1/1/20	25Gb/s,10Gb/s
6	4 x 25G, 4 x 10G		
	21	te-1/1/21	25Gb/s,10Gb/s
	22	te-1/1/22	25Gb/s,10Gb/s
	23	te-1/1/23	25Gb/s,10Gb/s
	24	te-1/1/24	25Gb/s,10Gb/s
7	4 x 25G, 4 x 10G		
	25	te-1/1/25	25Gb/s,10Gb/s
	26	te-1/1/26	25Gb/s,10Gb/s
	27	te-1/1/27	25Gb/s,10Gb/s
	28	te-1/1/28	25Gb/s,10Gb/s
8	4 x 25G, 4 x 10G		
	29	te-1/1/29	25Gb/s,10Gb/s

	30	te-1/1/30	25Gb/s,10Gb/s
	31	te-1/1/31	25Gb/s,10Gb/s
	32	te-1/1/32	25Gb/s,10Gb/s
9	4 x 25G, 4 x 10G		
	33	te-1/1/33	25Gb/s,10Gb/s
	34	te-1/1/34	25Gb/s,10Gb/s
	35	te-1/1/35	25Gb/s,10Gb/s
	36	te-1/1/36	25Gb/s,10Gb/s
10	4 x 25G, 4 x 10G		
	37	te-1/1/37	25Gb/s,10Gb/s
	38	te-1/1/38	25Gb/s,10Gb/s
	39	te-1/1/39	25Gb/s,10Gb/s
	40	te-1/1/40	25Gb/s,10Gb/s
11	4 x 25G, 4 x 10G		
	41	te-1/1/41	25Gb/s,10Gb/s
	42	te-1/1/42	25Gb/s,10Gb/s
	43	te-1/1/43	25Gb/s,10Gb/s
	44	te-1/1/44	25Gb/s,10Gb/s
12	4 x 25G, 4 x 10G		
	45	te-1/1/45	25Gb/s,10Gb/s
	46	te-1/1/46	25Gb/s,10Gb/s
	47	te-1/1/47	25Gb/s,10Gb/s
	48	te-1/1/48	25Gb/s,10Gb/s
13	4 x 25G, 4 x 10G		
	49	te-1/1/49	25Gb/s,10Gb/s
	50	te-1/1/50	25Gb/s,10Gb/s
	51	te-1/1/51	25Gb/s,10Gb/s
	52	te-1/1/52	25Gb/s,10Gb/s
14	4 x 25G, 4 x 10G		
	53	te-1/1/53	25Gb/s,10Gb/s

	54	te-1/1/54	25Gb/s,10Gb/s
	55	te-1/1/55	25Gb/s,10Gb/s
	56	te-1/1/56	25Gb/s,10Gb/s
15	4 x 25G, 4 x 10G		
	57	te-1/1/57	25Gb/s,10Gb/s
	58	te-1/1/58	25Gb/s,10Gb/s
	59	te-1/1/59	25Gb/s,10Gb/s
	60	te-1/1/60	25Gb/s,10Gb/s
16	4 x 25G, 4 x 10G		
	61	te-1/1/61	25Gb/s,10Gb/s
	62	te-1/1/62	25Gb/s,10Gb/s
	63	te-1/1/63	25Gb/s,10Gb/s
	64	te-1/1/64	25Gb/s,10Gb/s
17	4 x 25G, 4 x 10G		
	65	te-1/1/65	25Gb/s,10Gb/s
	66	te-1/1/66	25Gb/s,10Gb/s
	67	te-1/1/67	25Gb/s,10Gb/s
	68	te-1/1/68	25Gb/s,10Gb/s
18	4 x 25G, 4 x 10G		
	69	te-1/1/69	25Gb/s,10Gb/s
	70	te-1/1/70	25Gb/s,10Gb/s
	71	te-1/1/71	25Gb/s,10Gb/s
	72	te-1/1/72	25Gb/s,10Gb/s
19	4 x 25G, 4 x 10G		
	73	te-1/1/73	25Gb/s,10Gb/s
	74	te-1/1/74	25Gb/s,10Gb/s
	75	te-1/1/75	25Gb/s,10Gb/s
	76	te-1/1/76	25Gb/s,10Gb/s
20	4 x 25G, 4 x 10G		
	77	te-1/1/77	25Gb/s,10Gb/s

	78	te-1/1/78	25Gb/s,10Gb/s
	79	te-1/1/79	25Gb/s,10Gb/s
	80	te-1/1/80	25Gb/s,10Gb/s
21	4 x 25G, 4 x 10G		
	81	te-1/1/81	25Gb/s,10Gb/s
	82	te-1/1/82	25Gb/s,10Gb/s
	83	te-1/1/83	25Gb/s,10Gb/s
	84	te-1/1/84	25Gb/s,10Gb/s
22	4 x 25G, 4 x 10G		
	85	te-1/1/85	25Gb/s,10Gb/s
	86	te-1/1/86	25Gb/s,10Gb/s
	87	te-1/1/87	25Gb/s,10Gb/s
	88	te-1/1/88	25Gb/s,10Gb/s
23	4 x 25G, 4 x 10G		
	89	te-1/1/89	25Gb/s,10Gb/s
	90	te-1/1/90	25Gb/s,10Gb/s
	91	te-1/1/91	25Gb/s,10Gb/s
	92	te-1/1/92	25Gb/s,10Gb/s
24	4 x 25G, 4 x 10G		
	93	te-1/1/93	25Gb/s,10Gb/s
	94	te-1/1/94	25Gb/s,10Gb/s
	95	te-1/1/95	25Gb/s,10Gb/s
	96	te-1/1/96	25Gb/s,10Gb/s
25	4 x 25G, 4 x 10G		
	97	te-1/1/97	25Gb/s,10Gb/s
	98	te-1/1/98	25Gb/s,10Gb/s
	99	te-1/1/99	25Gb/s,10Gb/s
	100	te-1/1/100	25Gb/s,10Gb/s
26	4 x 25G, 4 x 10G		
	101	te-1/1/101	25Gb/s,10Gb/s

	102	te-1/1/102	25Gb/s,10Gb/s
	103	te-1/1/103	25Gb/s,10Gb/s
	104	te-1/1/104	25Gb/s,10Gb/s
27	4 x 25G, 4 x 10G		
	105	te-1/1/105	25Gb/s,10Gb/s
	106	te-1/1/106	25Gb/s,10Gb/s
	107	te-1/1/107	25Gb/s,10Gb/s
	108	te-1/1/108	25Gb/s,10Gb/s
28	4 x 25G, 4 x 10G		
	109	te-1/1/109	25Gb/s,10Gb/s
	110	te-1/1/110	25Gb/s,10Gb/s
	111	te-1/1/111	25Gb/s,10Gb/s
	112	te-1/1/112	25Gb/s,10Gb/s
29	4 x 25G, 4 x 10G		
	113	te-1/1/113	25Gb/s,10Gb/s
	114	te-1/1/114	25Gb/s,10Gb/s
	115	te-1/1/115	25Gb/s,10Gb/s
	116	te-1/1/116	25Gb/s,10Gb/s
30	4 x 25G, 4 x 10G		
	117	te-1/1/117	25Gb/s,10Gb/s
	118	te-1/1/118	25Gb/s,10Gb/s
	119	te-1/1/119	25Gb/s,10Gb/s
	120	te-1/1/120	25Gb/s,10Gb/s
31	4 x 25G, 4 x 10G		
	121	te-1/1/121	25Gb/s,10Gb/s
	122	te-1/1/122	25Gb/s,10Gb/s
	123	te-1/1/123	25Gb/s,10Gb/s
	124	te-1/1/124	25Gb/s,10Gb/s
32	4 x 25G, 4 x 10G		
	125	te-1/1/125	25Gb/s,10Gb/s

	126	te-1/1/126	25Gb/s,10Gb/s
	127	te-1/1/127	25Gb/s,10Gb/s
	128	te-1/1/128	25Gb/s,10Gb/s

Examples

100G port

Step 1: set port flexible

```
ovs-vsctl set-qe-port-mode flexible
sudo /etc/init.d/picos restart
```

Step 2: add 100G port

```
ovs-vsctl add-port br0 xe-1/1/1.1 vlan_mode=trunk tag=1 - set interface xe-1/1/1.1
type=pica8
ovs-vsctl add-port br0 xe-1/1/3.1 vlan_mode=trunk tag=1 - set interface xe-1/1/3.1
type=pica8
```

Step 3: check the port light color and port speed

```
ovs-ofctl show br0
check ".*xe-1/1/1.1.*speed.*100000Mbps"
check ".*xe-1/1/3.1.*speed.*100000Mbps"
check the light color----xe-1/1/1.1 and xe-1/1/3.1 blue
```

Step 4: set port breakout true on 100G' port that can be split

```
ovs-vsctl set-port-breakout 1 true
ovs-vsctl set-port-breakout 3 true
sudo /etc/init.d/picos restart
```

Step 5: add bridge and port

```
ovs-vsctl add-br br0 - set bridge br0 datapath_type=pica8
ovs-vsctl add-port br0 xe-1/1/1.1 vlan_mode=trunk tag=1 - set interface xe-1/1/1.1
type=pica8
ovs-vsctl add-port br0 xe-1/1/1.2 vlan_mode=trunk tag=1 - set interface xe-1/1/1.2
type=pica8
ovs-vsctl add-port br0 xe-1/1/1.3 vlan_mode=trunk tag=1 - set interface xe-1/1/1.3
type=pica8
ovs-vsctl add-port br0 xe-1/1/1.4 vlan_mode=trunk tag=1 - set interface xe-1/1/1.4
type=pica8
ovs-vsctl add-port br0 xe-1/1/3.1 vlan_mode=trunk tag=1 - set interface xe-1/1/3.1
type=pica8
ovs-vsctl add-port br0 xe-1/1/3.2 vlan_mode=trunk tag=1 - set interface xe-1/1/3.2
type=pica8
ovs-vsctl add-port br0 xe-1/1/3.3 vlan_mode=trunk tag=1 - set interface xe-1/1/3.3
```

```
type=pica8
ovs-vsctl add-port br0 xe-1/1/3.4 vlan_mode=trunk tag=1 - set interface xe-1/1/3.4
type=pica8
```

Step 6: check the port speed

```
ovs-ofctl show br0
```

```
check “.*xe-1/1/1.1.*speed.*25000Mbps”
check “.*xe-1/1/1.2.*speed.*25000Mbps”
check “.*xe-1/1/1.3.*speed.*25000Mbps”
check “.*xe-1/1/1.4.*speed.*25000Mbps”
check “.*xe-1/1/3.1.*speed.*25000Mbps”
check “.*xe-1/1/3.2.*speed.*25000Mbps”
check “.*xe-1/1/3.3.*speed.*25000Mbps”
check “.*xe-1/1/3.4.*speed.*25000Mbps”
```

Step 7: check the light color

```
xe-1/1/1.1-----white
xe-1/1/1.2-----white
xe-1/1/1.3-----white
xe-1/1/1.4-----white
```

Step 8: set the sub port to 10G

```
ovs-vsctl set interface xe-1/1/1.1 options:link_speed=10G
ovs-vsctl set interface xe-1/1/1.2 options:link_speed=10G
ovs-vsctl set interface xe-1/1/1.3 options:link_speed=10G
ovs-vsctl set interface xe-1/1/1.4 options:link_speed=10G
ovs-vsctl set interface xe-1/1/3.1 options:link_speed=10G
ovs-vsctl set interface xe-1/1/3.2 options:link_speed=10G
ovs-vsctl set interface xe-1/1/3.3 options:link_speed=10G
ovs-vsctl set interface xe-1/1/3.4 options:link_speed=10G
```

Step 9: check the port speed

```
ovs-ofctl show br0
```

```
check “.*xe-1/1/1.1.*speed.*10000Mbps”
check “.*xe-1/1/1.2.*speed.*10000Mbps”
check “.*xe-1/1/1.3.*speed.*10000Mbps”
check “.*xe-1/1/1.4.*speed.*10000Mbps”
check “.*xe-1/1/3.1.*speed.* 10000Mbps”
```

```
check “.*xe-1/1/3.2.*speed.* 10000Mbps”
check “.*xe-1/1/3.3.*speed.* 10000Mbps”
check “.*xe-1/1/3.4.*speed.* 10000Mbps”
```

Step 10: check the light color

```
xe-1/1/1.1-----green
xe-1/1/1.2----- green
xe-1/1/1.3----- green
xe-1/1/1.4----- green
```

Step 11: clear the configuration

```
ovs-vsctl set-port-breakout all false
sudo /etc/init.d/picos restart
```

Step 12: set the 100G port to 40G,check the speed and port light color

```
ovs-vsctl set interface xe-1/1/1.1 options:link_speed=40G
ovs-vsctl set interface xe-1/1/3.1 options:link_speed=40G
ovs-ofctl show br0
```

```
check “.*xe-1/1/1.1.*speed.*40000Mbps”
check “.*xe-1/1/3.1.*speed.*40000Mbps”
xe-1/1/1.1-----orange
xe-1/1/3.1-----orange
```

40G Changes to 4*10G in OVS Mode on as7712_32x

OVS Mode Configuration

You can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | max]
```

After setting port to different mode, it is mandatory to restart the OVS service in order to make the new state take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

You can take a look of the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

Normal: 32 x 100G (or 40G)

When ports are in normal mode, the mapping between physical port number, interface index, interface name and interface support speed are in the following table.

Physical Port Number	Interface Index	Interface Name	Interface Support Speed
1	129	he-1/1/1	100Gb/s, 40Gb/s
2	130	he-1/1/2	100Gb/s, 40Gb/s
3	131	he-1/1/3	100Gb/s, 40Gb/s
4	132	he-1/1/4	100Gb/s, 40Gb/s
5	133	he-1/1/5	100Gb/s, 40Gb/s
6	134	he-1/1/6	100Gb/s, 40Gb/s
7	135	he-1/1/7	100Gb/s, 40Gb/s
8	136	he-1/1/8	100Gb/s, 40Gb/s
9	137	he-1/1/9	100Gb/s, 40Gb/s
10	138	he-1/1/10	100Gb/s, 40Gb/s
11	139	he-1/1/11	100Gb/s, 40Gb/s
12	140	he-1/1/12	100Gb/s, 40Gb/s
13	141	he-1/1/13	100Gb/s, 40Gb/s
14	142	he-1/1/14	100Gb/s, 40Gb/s
15	143	he-1/1/15	100Gb/s, 40Gb/s
16	144	he-1/1/16	100Gb/s, 40Gb/s
17	145	he-1/1/17	100Gb/s, 40Gb/s
18	146	he-1/1/18	100Gb/s, 40Gb/s
19	147	he-1/1/19	100Gb/s, 40Gb/s
20	148	he-1/1/20	100Gb/s, 40Gb/s
21	149	he-1/1/21	100Gb/s, 40Gb/s
22	150	he-1/1/22	100Gb/s, 40Gb/s
23	151	he-1/1/23	100Gb/s, 40Gb/s
24	152	he-1/1/24	100Gb/s, 40Gb/s
25	153	he-1/1/25	100Gb/s, 40Gb/s

26	154	he-1/1/26	100Gb/s, 40Gb/s
27	155	he-1/1/27	100Gb/s, 40Gb/s
28	156	he-1/1/28	100Gb/s, 40Gb/s
29	157	he-1/1/29	100Gb/s, 40Gb/s
30	158	he-1/1/30	100Gb/s, 40Gb/s
31	159	he-1/1/31	100Gb/s, 40Gb/s
32	160	he-1/1/32	100Gb/s, 40Gb/s

SFP/max: 128 x 25G (or 10G)

When ports are in SFP/max mode, the mapping between physical port number, interface name and interface support speed are in the following table.

Physical Port Number	Interface Index	Interface Name	Interface Support Speed
1	4 x 25G, 4 x 10G		
	1	te-1/1/1	25Gb/s,10Gb/s
	2	te-1/1/2	25Gb/s,10Gb/s
	3	te-1/1/3	25Gb/s,10Gb/s
	4	te-1/1/4	25Gb/s,10Gb/s
2	4 x 25G, 4 x 10G		
	5	te-1/1/5	25Gb/s,10Gb/s
	6	te-1/1/6	25Gb/s,10Gb/s
	7	te-1/1/7	25Gb/s,10Gb/s
	8	te-1/1/8	25Gb/s,10Gb/s
3	4 x 25G, 4 x 10G		
	9	te-1/1/9	25Gb/s,10Gb/s
	10	te-1/1/10	25Gb/s,10Gb/s
	11	te-1/1/11	25Gb/s,10Gb/s
	12	te-1/1/12	25Gb/s,10Gb/s
4	4 x 25G, 4 x 10G		
	13	te-1/1/13	25Gb/s,10Gb/s
	14	te-1/1/14	25Gb/s,10Gb/s

	15	te-1/1/15	25Gb/s,10Gb/s
	16	te-1/1/16	25Gb/s,10Gb/s
5	4 x 25G, 4 x 10G		
	17	te-1/1/17	25Gb/s,10Gb/s
	18	te-1/1/18	25Gb/s,10Gb/s
	19	te-1/1/19	25Gb/s,10Gb/s
	20	te-1/1/20	25Gb/s,10Gb/s
6	4 x 25G, 4 x 10G		
	21	te-1/1/21	25Gb/s,10Gb/s
	22	te-1/1/22	25Gb/s,10Gb/s
	23	te-1/1/23	25Gb/s,10Gb/s
	24	te-1/1/24	25Gb/s,10Gb/s
7	4 x 25G, 4 x 10G		
	25	te-1/1/25	25Gb/s,10Gb/s
	26	te-1/1/26	25Gb/s,10Gb/s
	27	te-1/1/27	25Gb/s,10Gb/s
	28	te-1/1/28	25Gb/s,10Gb/s
8	4 x 25G, 4 x 10G		
	29	te-1/1/29	25Gb/s,10Gb/s
	30	te-1/1/30	25Gb/s,10Gb/s
	31	te-1/1/31	25Gb/s,10Gb/s
	32	te-1/1/32	25Gb/s,10Gb/s
9	4 x 25G, 4 x 10G		
	33	te-1/1/33	25Gb/s,10Gb/s
	34	te-1/1/34	25Gb/s,10Gb/s
	35	te-1/1/35	25Gb/s,10Gb/s
	36	te-1/1/36	25Gb/s,10Gb/s
10	4 x 25G, 4 x 10G		
	37	te-1/1/37	25Gb/s,10Gb/s
	38	te-1/1/38	25Gb/s,10Gb/s

	39	te-1/1/39	25Gb/s,10Gb/s
	40	te-1/1/40	25Gb/s,10Gb/s
11	4 x 25G, 4 x 10G		
	41	te-1/1/41	25Gb/s,10Gb/s
	42	te-1/1/42	25Gb/s,10Gb/s
	43	te-1/1/43	25Gb/s,10Gb/s
	44	te-1/1/44	25Gb/s,10Gb/s
12	4 x 25G, 4 x 10G		
	45	te-1/1/45	25Gb/s,10Gb/s
	46	te-1/1/46	25Gb/s,10Gb/s
	47	te-1/1/47	25Gb/s,10Gb/s
	48	te-1/1/48	25Gb/s,10Gb/s
13	4 x 25G, 4 x 10G		
	49	te-1/1/49	25Gb/s,10Gb/s
	50	te-1/1/50	25Gb/s,10Gb/s
	51	te-1/1/51	25Gb/s,10Gb/s
	52	te-1/1/52	25Gb/s,10Gb/s
14	4 x 25G, 4 x 10G		
	53	te-1/1/53	25Gb/s,10Gb/s
	54	te-1/1/54	25Gb/s,10Gb/s
	55	te-1/1/55	25Gb/s,10Gb/s
	56	te-1/1/56	25Gb/s,10Gb/s
15	4 x 25G, 4 x 10G		
	57	te-1/1/57	25Gb/s,10Gb/s
	58	te-1/1/58	25Gb/s,10Gb/s
	59	te-1/1/59	25Gb/s,10Gb/s
	60	te-1/1/60	25Gb/s,10Gb/s
16	4 x 25G, 4 x 10G		
	61	te-1/1/61	25Gb/s,10Gb/s
	62	te-1/1/62	25Gb/s,10Gb/s

	63	te-1/1/63	25Gb/s,10Gb/s
	64	te-1/1/64	25Gb/s,10Gb/s
17		4 x 25G, 4 x 10G	
	65	te-1/1/65	25Gb/s,10Gb/s
	66	te-1/1/66	25Gb/s,10Gb/s
	67	te-1/1/67	25Gb/s,10Gb/s
	68	te-1/1/68	25Gb/s,10Gb/s
18		4 x 25G, 4 x 10G	
	69	te-1/1/69	25Gb/s,10Gb/s
	70	te-1/1/70	25Gb/s,10Gb/s
	71	te-1/1/71	25Gb/s,10Gb/s
	72	te-1/1/72	25Gb/s,10Gb/s
19		4 x 25G, 4 x 10G	
	73	te-1/1/73	25Gb/s,10Gb/s
	74	te-1/1/74	25Gb/s,10Gb/s
	75	te-1/1/75	25Gb/s,10Gb/s
	76	te-1/1/76	25Gb/s,10Gb/s
20		4 x 25G, 4 x 10G	
	77	te-1/1/77	25Gb/s,10Gb/s
	78	te-1/1/78	25Gb/s,10Gb/s
	79	te-1/1/79	25Gb/s,10Gb/s
	80	te-1/1/80	25Gb/s,10Gb/s
21		4 x 25G, 4 x 10G	
	81	te-1/1/81	25Gb/s,10Gb/s
	82	te-1/1/82	25Gb/s,10Gb/s
	83	te-1/1/83	25Gb/s,10Gb/s
	84	te-1/1/84	25Gb/s,10Gb/s
22		4 x 25G, 4 x 10G	
	85	te-1/1/85	25Gb/s,10Gb/s
	86	te-1/1/86	25Gb/s,10Gb/s

	87	te-1/1/87	25Gb/s,10Gb/s
	88	te-1/1/88	25Gb/s,10Gb/s
23	4 x 25G, 4 x 10G		
	89	te-1/1/89	25Gb/s,10Gb/s
	90	te-1/1/90	25Gb/s,10Gb/s
	91	te-1/1/91	25Gb/s,10Gb/s
	92	te-1/1/92	25Gb/s,10Gb/s
24	4 x 25G, 4 x 10G		
	93	te-1/1/93	25Gb/s,10Gb/s
	94	te-1/1/94	25Gb/s,10Gb/s
	95	te-1/1/95	25Gb/s,10Gb/s
	96	te-1/1/96	25Gb/s,10Gb/s
25	4 x 25G, 4 x 10G		
	97	te-1/1/97	25Gb/s,10Gb/s
	98	te-1/1/98	25Gb/s,10Gb/s
	99	te-1/1/99	25Gb/s,10Gb/s
	100	te-1/1/100	25Gb/s,10Gb/s
26	4 x 25G, 4 x 10G		
	101	te-1/1/101	25Gb/s,10Gb/s
	102	te-1/1/102	25Gb/s,10Gb/s
	103	te-1/1/103	25Gb/s,10Gb/s
	104	te-1/1/104	25Gb/s,10Gb/s
27	4 x 25G, 4 x 10G		
	105	te-1/1/105	25Gb/s,10Gb/s
	106	te-1/1/106	25Gb/s,10Gb/s
	107	te-1/1/107	25Gb/s,10Gb/s
	108	te-1/1/108	25Gb/s,10Gb/s
28	4 x 25G, 4 x 10G		
	109	te-1/1/109	25Gb/s,10Gb/s
	110	te-1/1/110	25Gb/s,10Gb/s

	111	te-1/1/111	25Gb/s,10Gb/s
	112	te-1/1/112	25Gb/s,10Gb/s
29	4 x 25G, 4 x 10G		
	113	te-1/1/113	25Gb/s,10Gb/s
	114	te-1/1/114	25Gb/s,10Gb/s
	115	te-1/1/115	25Gb/s,10Gb/s
	116	te-1/1/116	25Gb/s,10Gb/s
30	4 x 25G, 4 x 10G		
	117	te-1/1/117	25Gb/s,10Gb/s
	118	te-1/1/118	25Gb/s,10Gb/s
	119	te-1/1/119	25Gb/s,10Gb/s
	120	te-1/1/120	25Gb/s,10Gb/s
31	4 x 25G, 4 x 10G		
	121	te-1/1/121	25Gb/s,10Gb/s
	122	te-1/1/122	25Gb/s,10Gb/s
	123	te-1/1/123	25Gb/s,10Gb/s
	124	te-1/1/124	25Gb/s,10Gb/s
32	4 x 25G, 4 x 10G		
	125	te-1/1/125	25Gb/s,10Gb/s
	126	te-1/1/126	25Gb/s,10Gb/s
	127	te-1/1/127	25Gb/s,10Gb/s
	128	te-1/1/128	25Gb/s,10Gb/s

In OVS mode Configuration

You can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | max]
```

After setting port to different mode, it is mandatory to restart the OVS service in order to make the new state take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

You can take a look of the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

normal(48*10G+6*40G)

When ports are in normal mode, the mapping between physical port number, interface index, interface names and interface support speed is in the following table.

Physical Port number	Interface index	Interface name	interface support speed
1	1	te-1/1/1	10Gb/s , 1Gb/s and 100Mb/s
2	2	te-1/1/2	10Gb/s , 1Gb/s and 100Mb/s
3	3	te-1/1/3	10Gb/s , 1Gb/s and 100Mb/s
4	4	te-1/1/4	10Gb/s , 1Gb/s and 100Mb/s
5	5	te-1/1/5	10Gb/s , 1Gb/s and 100Mb/s
6	6	te-1/1/6	10Gb/s , 1Gb/s and 100Mb/s
7	7	te-1/1/7	10Gb/s , 1Gb/s and 100Mb/s
8	8	te-1/1/8	10Gb/s , 1Gb/s and 100Mb/s
9	9	te-1/1/9	10Gb/s , 1Gb/s and 100Mb/s
10	10	te-1/1/10	10Gb/s , 1Gb/s and 100Mb/s
11	11	te-1/1/11	10Gb/s , 1Gb/s and 100Mb/s
12	12	te-1/1/12	10Gb/s , 1Gb/s and 100Mb/s
13	13	te-1/1/13	10Gb/s , 1Gb/s and 100Mb/s
14	14	te-1/1/14	10Gb/s , 1Gb/s and 100Mb/s
15	15	te-1/1/15	10Gb/s , 1Gb/s and 100Mb/s
16	16	te-1/1/16	10Gb/s , 1Gb/s and 100Mb/s
17	17	te-1/1/17	10Gb/s , 1Gb/s and 100Mb/s
18	18	te-1/1/18	10Gb/s , 1Gb/s and 100Mb/s
19	19	te-1/1/19	10Gb/s , 1Gb/s and 100Mb/s
20	20	te-1/1/20	10Gb/s , 1Gb/s and 100Mb/s
21	21	te-1/1/21	10Gb/s , 1Gb/s and 100Mb/s
22	22	te-1/1/22	10Gb/s , 1Gb/s and 100Mb/s
23	23	te-1/1/23	10Gb/s , 1Gb/s and 100Mb/s
24	24	te-1/1/24	10Gb/s , 1Gb/s and 100Mb/s

25	25	te-1/1/25	10Gb/s , 1Gb/s and 100Mb/s
26	26	te-1/1/26	10Gb/s , 1Gb/s and 100Mb/s
27	27	te-1/1/27	10Gb/s , 1Gb/s and 100Mb/s
28	28	te-1/1/28	10Gb/s , 1Gb/s and 100Mb/s
29	29	te-1/1/29	10Gb/s , 1Gb/s and 100Mb/s
30	30	te-1/1/30	10Gb/s , 1Gb/s and 100Mb/s
31	31	te-1/1/31	10Gb/s , 1Gb/s and 100Mb/s
32	32	te-1/1/32	10Gb/s , 1Gb/s and 100Mb/s
33	33	te-1/1/33	10Gb/s , 1Gb/s and 100Mb/s
34	34	te-1/1/34	10Gb/s , 1Gb/s and 100Mb/s
35	35	te-1/1/35	10Gb/s , 1Gb/s and 100Mb/s
36	36	te-1/1/36	10Gb/s , 1Gb/s and 100Mb/s
37	37	te-1/1/37	10Gb/s , 1Gb/s and 100Mb/s
38	38	te-1/1/38	10Gb/s , 1Gb/s and 100Mb/s
39	39	te-1/1/39	10Gb/s , 1Gb/s and 100Mb/s
40	40	te-1/1/40	10Gb/s , 1Gb/s and 100Mb/s
41	41	te-1/1/41	10Gb/s , 1Gb/s and 100Mb/s
42	42	te-1/1/42	10Gb/s , 1Gb/s and 100Mb/s
43	43	te-1/1/43	10Gb/s , 1Gb/s and 100Mb/s
44	44	te-1/1/44	10Gb/s , 1Gb/s and 100Mb/s
45	45	te-1/1/45	10Gb/s , 1Gb/s and 100Mb/s
46	46	te-1/1/46	10Gb/s , 1Gb/s and 100Mb/s
47	47	te-1/1/47	10Gb/s , 1Gb/s and 100Mb/s
48	48	te-1/1/48	10Gb/s , 1Gb/s and 100Mb/s
49	65	qe-1/1/49	40Gb/s
50	66	qe-1/1/50	40Gb/s
51	67	qe-1/1/51	40Gb/s
52	68	qe-1/1/52	40Gb/s
53	69	qe-1/1/53	40Gb/s
54	70	qe-1/1/54	40Gb/s

max(72*10G)

When ports are in max mode, the mapping between physical port number, interface index, interface names and interfaces support speed is in the following table.

Physical Port number	Interface index	Interface name	Interface support speed
1	1	te-1/1/1	10Gb/s , 1Gb/s and 100Mb/s
2	2	te-1/1/2	10Gb/s , 1Gb/s and 100Mb/s
3	3	te-1/1/3	10Gb/s , 1Gb/s and 100Mb/s
4	4	te-1/1/4	10Gb/s , 1Gb/s and 100Mb/s
5	5	te-1/1/5	10Gb/s , 1Gb/s and 100Mb/s
6	6	te-1/1/6	10Gb/s , 1Gb/s and 100Mb/s
7	7	te-1/1/7	10Gb/s , 1Gb/s and 100Mb/s
8	8	te-1/1/8	10Gb/s , 1Gb/s and 100Mb/s
9	9	te-1/1/9	10Gb/s , 1Gb/s and 100Mb/s
10	10	te-1/1/10	10Gb/s , 1Gb/s and 100Mb/s
11	11	te-1/1/11	10Gb/s , 1Gb/s and 100Mb/s
12	12	te-1/1/12	10Gb/s , 1Gb/s and 100Mb/s
13	13	te-1/1/13	10Gb/s , 1Gb/s and 100Mb/s
14	14	te-1/1/14	10Gb/s , 1Gb/s and 100Mb/s
15	15	te-1/1/15	10Gb/s , 1Gb/s and 100Mb/s
16	16	te-1/1/16	10Gb/s , 1Gb/s and 100Mb/s
17	17	te-1/1/17	10Gb/s , 1Gb/s and 100Mb/s
18	18	te-1/1/18	10Gb/s , 1Gb/s and 100Mb/s
19	19	te-1/1/19	10Gb/s , 1Gb/s and 100Mb/s
20	20	te-1/1/20	10Gb/s , 1Gb/s and 100Mb/s
21	21	te-1/1/21	10Gb/s , 1Gb/s and 100Mb/s
22	22	te-1/1/22	10Gb/s , 1Gb/s and 100Mb/s
23	23	te-1/1/23	10Gb/s , 1Gb/s and 100Mb/s
24	24	te-1/1/24	10Gb/s , 1Gb/s and 100Mb/s
25	25	te-1/1/25	10Gb/s , 1Gb/s and 100Mb/s
26	26	te-1/1/26	10Gb/s , 1Gb/s and 100Mb/s

27	27	te-1/1/27	10Gb/s , 1Gb/s and 100Mb/s
28	28	te-1/1/28	10Gb/s , 1Gb/s and 100Mb/s
29	29	te-1/1/29	10Gb/s , 1Gb/s and 100Mb/s
30	30	te-1/1/30	10Gb/s , 1Gb/s and 100Mb/s
31	31	te-1/1/31	10Gb/s , 1Gb/s and 100Mb/s
32	32	te-1/1/32	10Gb/s , 1Gb/s and 100Mb/s
33	33	te-1/1/33	10Gb/s , 1Gb/s and 100Mb/s
34	34	te-1/1/34	10Gb/s , 1Gb/s and 100Mb/s
35	35	te-1/1/35	10Gb/s , 1Gb/s and 100Mb/s
36	36	te-1/1/36	10Gb/s , 1Gb/s and 100Mb/s
37	37	te-1/1/37	10Gb/s , 1Gb/s and 100Mb/s
38	38	te-1/1/38	10Gb/s , 1Gb/s and 100Mb/s
39	39	te-1/1/39	10Gb/s , 1Gb/s and 100Mb/s
40	40	te-1/1/40	10Gb/s , 1Gb/s and 100Mb/s
41	41	te-1/1/41	10Gb/s , 1Gb/s and 100Mb/s
42	42	te-1/1/42	10Gb/s , 1Gb/s and 100Mb/s
43	43	te-1/1/43	10Gb/s , 1Gb/s and 100Mb/s
44	44	te-1/1/44	10Gb/s , 1Gb/s and 100Mb/s
45	45	te-1/1/45	10Gb/s , 1Gb/s and 100Mb/s
46	46	te-1/1/46	10Gb/s , 1Gb/s and 100Mb/s
47	47	te-1/1/47	10Gb/s , 1Gb/s and 100Mb/s
48	48	te-1/1/48	10Gb/s , 1Gb/s and 100Mb/s
49		4 x 10G	
	49	te-1/1/49	10Gb/s
	50	te-1/1/50	10Gb/s
	51	te-1/1/51	10Gb/s
	52	te-1/1/52	10Gb/s
50		4 x 10G	
	53	te-1/1/53	10Gb/s
	54	te-1/1/54	10Gb/s

	55	te-1/1/55	10Gb/s
	56	te-1/1/56	10Gb/s
51	4 x 10G		
	57	te-1/1/57	10Gb/s
	58	te-1/1/58	10Gb/s
	59	te-1/1/59	10Gb/s
	60	te-1/1/60	10Gb/s
52	4 x 10G		
	61	te-1/1/61	10Gb/s
	62	te-1/1/62	10Gb/s
	63	te-1/1/63	10Gb/s
	64	te-1/1/64	10Gb/s
53	4 x 10G		
	65	te-1/1/65	10Gb/s
	66	te-1/1/66	10Gb/s
	67	te-1/1/67	10Gb/s
	68	te-1/1/68	10Gb/s
54	4 x 10G		
	69	te-1/1/69	10Gb/s
	70	te-1/1/70	10Gb/s
	71	te-1/1/71	10Gb/s
	72	te-1/1/72	10Gb/s

In OVS mode Configuration

You can set the port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl set-qe-port-mode [normal | max]
```

After setting port to different mode, it is mandatory to restart the OVS service in order to make the new state to take effect.

```
admin@PicOS-OVS$sudo service picos restart
```

You can take a look of the current port mode by issuing:

```
admin@PicOS-OVS$ovs-vsctl show-qe-port-mode
```

normal (48*10G+6*40G)

When ports are in normal mode, the mapping between physical port, interface index, interface names and interfaces support speed is in the following table.

Physical Port number	Interface index	Interface name	Interface support speed
1	1	te-1/1/1	10Gb/s and 1Gb/s
2	2	te-1/1/2	10Gb/s and 1Gb/s
3	3	te-1/1/3	10Gb/s and 1Gb/s
4	4	te-1/1/4	10Gb/s and 1Gb/s
5	5	te-1/1/5	10Gb/s and 1Gb/s
6	6	te-1/1/6	10Gb/s and 1Gb/s
7	7	te-1/1/7	10Gb/s and 1Gb/s
8	8	te-1/1/8	10Gb/s and 1Gb/s
9	9	te-1/1/9	10Gb/s and 1Gb/s
10	10	te-1/1/10	10Gb/s and 1Gb/s
11	11	te-1/1/11	10Gb/s and 1Gb/s
12	12	te-1/1/12	10Gb/s and 1Gb/s
13	13	te-1/1/13	10Gb/s and 1Gb/s
14	14	te-1/1/14	10Gb/s and 1Gb/s
15	15	te-1/1/15	10Gb/s and 1Gb/s
16	16	te-1/1/16	10Gb/s and 1Gb/s
17	17	te-1/1/17	10Gb/s and 1Gb/s
18	18	te-1/1/18	10Gb/s and 1Gb/s
19	19	te-1/1/19	10Gb/s and 1Gb/s
20	20	te-1/1/20	10Gb/s and 1Gb/s
21	21	te-1/1/21	10Gb/s and 1Gb/s
22	22	te-1/1/22	10Gb/s and 1Gb/s
23	23	te-1/1/23	10Gb/s and 1Gb/s

24	24	te-1/1/24	10Gb/s and 1Gb/s
25	25	te-1/1/25	10Gb/s and 1Gb/s
26	26	te-1/1/26	10Gb/s and 1Gb/s
27	27	te-1/1/27	10Gb/s and 1Gb/s
28	28	te-1/1/28	10Gb/s and 1Gb/s
29	29	te-1/1/29	10Gb/s and 1Gb/s
30	30	te-1/1/30	10Gb/s and 1Gb/s
31	31	te-1/1/31	10Gb/s and 1Gb/s
32	32	te-1/1/32	10Gb/s and 1Gb/s
33	33	te-1/1/33	10Gb/s and 1Gb/s
34	34	te-1/1/34	10Gb/s and 1Gb/s
35	35	te-1/1/35	10Gb/s and 1Gb/s
36	36	te-1/1/36	10Gb/s and 1Gb/s
37	37	te-1/1/37	10Gb/s and 1Gb/s
38	38	te-1/1/38	10Gb/s and 1Gb/s
39	39	te-1/1/39	10Gb/s and 1Gb/s
40	40	te-1/1/40	10Gb/s and 1Gb/s
41	41	te-1/1/41	10Gb/s and 1Gb/s
42	42	te-1/1/42	10Gb/s and 1Gb/s
43	43	te-1/1/43	10Gb/s and 1Gb/s
44	44	te-1/1/44	10Gb/s and 1Gb/s
45	45	te-1/1/45	10Gb/s and 1Gb/s
46	46	te-1/1/46	10Gb/s and 1Gb/s
47	47	te-1/1/47	10Gb/s and 1Gb/s
48	48	te-1/1/48	10Gb/s and 1Gb/s
49	73	qe-1/1/49	40Gb/s
50	74	qe-1/1/50	40Gb/s
51	75	qe-1/1/51	40Gb/s
52	76	qe-1/1/52	40Gb/s
53	77	qe-1/1/53	40Gb/s

54	78	qe-1/1/54	40Gb/s
----	----	-----------	--------

max (72*10G)

When ports are in max mode, the mapping between physical port, interface index, interface names and interfaces support speed is in the following table.

Physical Port number	interface index	interface name	interface support speed
1	1	te-1/1/1	10Gb/s and 1Gb/s
2	2	te-1/1/2	10Gb/s and 1Gb/s
3	3	te-1/1/3	10Gb/s and 1Gb/s
4	4	te-1/1/4	10Gb/s and 1Gb/s
5	5	te-1/1/5	10Gb/s and 1Gb/s
6	6	te-1/1/6	10Gb/s and 1Gb/s
7	7	te-1/1/7	10Gb/s and 1Gb/s
8	8	te-1/1/8	10Gb/s and 1Gb/s
9	9	te-1/1/9	10Gb/s and 1Gb/s
10	10	te-1/1/10	10Gb/s and 1Gb/s
11	11	te-1/1/11	10Gb/s and 1Gb/s
12	12	te-1/1/12	10Gb/s and 1Gb/s
13	13	te-1/1/13	10Gb/s and 1Gb/s
14	14	te-1/1/14	10Gb/s and 1Gb/s
15	15	te-1/1/15	10Gb/s and 1Gb/s
16	16	te-1/1/16	10Gb/s and 1Gb/s
17	17	te-1/1/17	10Gb/s and 1Gb/s
18	18	te-1/1/18	10Gb/s and 1Gb/s
19	19	te-1/1/19	10Gb/s and 1Gb/s
20	20	te-1/1/20	10Gb/s and 1Gb/s
21	21	te-1/1/21	10Gb/s and 1Gb/s
22	22	te-1/1/22	10Gb/s and 1Gb/s
23	23	te-1/1/23	10Gb/s and 1Gb/s
24	24	te-1/1/24	10Gb/s and 1Gb/s

25	25	te-1/1/25	10Gb/s and 1Gb/s
26	26	te-1/1/26	10Gb/s and 1Gb/s
27	27	te-1/1/27	10Gb/s and 1Gb/s
28	28	te-1/1/28	10Gb/s and 1Gb/s
29	29	te-1/1/29	10Gb/s and 1Gb/s
30	30	te-1/1/30	10Gb/s and 1Gb/s
31	31	te-1/1/31	10Gb/s and 1Gb/s
32	32	te-1/1/32	10Gb/s and 1Gb/s
33	33	te-1/1/33	10Gb/s and 1Gb/s
34	34	te-1/1/34	10Gb/s and 1Gb/s
35	35	te-1/1/35	10Gb/s and 1Gb/s
36	36	te-1/1/36	10Gb/s and 1Gb/s
37	37	te-1/1/37	10Gb/s and 1Gb/s
38	38	te-1/1/38	10Gb/s and 1Gb/s
39	39	te-1/1/39	10Gb/s and 1Gb/s
40	40	te-1/1/40	10Gb/s and 1Gb/s
41	41	te-1/1/41	10Gb/s and 1Gb/s
42	42	te-1/1/42	10Gb/s and 1Gb/s
43	43	te-1/1/43	10Gb/s and 1Gb/s
44	44	te-1/1/44	10Gb/s and 1Gb/s
45	45	te-1/1/45	10Gb/s and 1Gb/s
46	46	te-1/1/46	10Gb/s and 1Gb/s
47	47	te-1/1/47	10Gb/s and 1Gb/s
48	48	te-1/1/48	10Gb/s and 1Gb/s
49		4 x 10G	
	49	te-1/1/49	10Gb/s
	50	te-1/1/50	10Gb/s
	51	te-1/1/51	10Gb/s
	52	te-1/1/52	10Gb/s
50		4 x 10G	

	53	te-1/1/53	10Gb/s
	54	te-1/1/54	10Gb/s
	55	te-1/1/55	10Gb/s
	56	te-1/1/56	10Gb/s
51		4 x 10G	
	57	te-1/1/57	10Gb/s
	58	te-1/1/58	10Gb/s
	59	te-1/1/59	10Gb/s
	60	te-1/1/60	10Gb/s
52		4 x 10G	
	61	te-1/1/61	10Gb/s
	62	te-1/1/62	10Gb/s
	63	te-1/1/63	10Gb/s
	64	te-1/1/64	10Gb/s
53		4 x 10G	
	65	te-1/1/65	10Gb/s
	66	te-1/1/66	10Gb/s
	67	te-1/1/67	10Gb/s
	68	te-1/1/68	10Gb/s
54		4 x 10G	
	69	te-1/1/69	10Gb/s
	70	te-1/1/70	10Gb/s
	71	te-1/1/71	10Gb/s
	72	te-1/1/72	10Gb/s

Configuring sFlow v5

PicOS OVS supports sFlow v5. User can configure the sFlow as follows:

```
root@PicOS-OVS$ ovs-vsctl --id=@s create sFlow agent=eth0 target=\"10.10.50.207:9901\" header=128 sampling=5000 polling=30 -- set Bridge br0 sflow=@s
root@PicOS-OVS$
```

In the above example, the parameters are as follows:

```
COLLECTOR_IP=10.10.50.207
COLLECTOR_PORT=9901
AGENT_IP=eth0
HEADER_BYTES=128
SAMPLING_N=5000
POLLING_SECS=30
```

List the configuration of sflow:

```
root@PicOS-OVS$ ovs-vsctl list sflow
_uuid          : 88d94294-4bb3-44f3-8a12-6055bf458de6
agent          : "eth0"
external_ids   : {}
header         : 128
polling        : 30
sampling       : 5000
targets        : ["10.10.50.207:9901"]
root@PicOS-OVS$
```

To delete an sFlow, use the clear command, as shown in the following example.:

```
root@PicOS-OVS$ ovs-vsctl -- clear Bridge br0 sflow
root@PicOS-OVS$
```

Configuring NetFlow

PicOS OVS supports NetFlow. User can configure the NetFlow as follows.

```
root@PicOS-OVS# ovs-vsctl -- set Bridge br0 netflow=@nf -- --id=@nf create NetFlow
targets=\"10.10.50.207:5566\" active-timeout=30
root@PicOS-OVS#
```

In the command above, the parameters are as follows.

```
COLLECTOR_IP=10.10.50.207
COLLECTOR_PORT=5566
ACTIVE_TIMEOUT=30
```

To delete NetFlow, use the clear command as shown in the following example.

```
root@PicOS-OVS# ovs-vsctl -- clear Bridge br0 netflow
```

Configuration Examples

The following examples show how to configure the port mirroring feature.

```
admin@PicOS-OVS$ ovs-vsctl -- set bridge br0 mirrors=@m -- --id=@te-1/1/1 get Port te-1/1/1
-- --id=@te-1/1/2 get Port te-1/1/2 -- --id=@te-1/1/3 get Port te-1/1/3 -- --id=@m create
```

```
Mirror name=mymirror select-dst-port=@te-1/1/1,@te-1/1/2
select-src-port=@te-1/1/1,@te-1/1/2 output-port=@te-1/1/3
admin@PicOS-OVS$
```

The configuration above includes ports te-1/1/1, te-1/1/2 and te-1/1/3. The source ports are te-1/1/1 and te-1/1/2 (including ingress and egress), and the output port (monitor port) is te-1/1/3.

The "select-dst-port" means some packets (in switch chip) will go-out from the specified port (egress).

The "select-src-port" means some packets will enter the specified port (ingress).



PicOS can support multiple mirrors, but some limitations exist.

1. Only has one output port, at this time the specified port can be configured multiple ways..
2. Has different output ports, and only monitors ingress or egress. If monitoring both ingress and egress the number of ports which can be monitored at most is (that is the select-dst-port and select-src-port).

The command set 2 mirrors, m1 and m2:

```
admin@PicOS-OVS$ovs-vsctl -- set bridge br0 mirrors=@m1,@m2 -- --id=@te-1/1/11 get Port
te-1/1/11 -- --id=@te-1/1/13 get Port te-1/1/13 -- --id=@te-1/1/14 get Port te-1/1/14 --
--id=@m1 create Mirror name=mymirror1 select-src-port=@te-1/1/11 output-port=@te-1/1/13 --
--id=@m2 create Mirror name=mymirror2 select-src-port=@te-1/1/11 output-port=@te-1/1/14
```

The command add 2 new mirrors, m3 and m4:

```
admin@PicOS-OVS$ovs-vsctl add bridge br0 mirrors @m3,@m4 -- --id=@te-1/1/11 get Port
te-1/1/11 -- --id=@te-1/1/12 get Port te-1/1/12 -- --id=@te-1/1/16 get Port te-1/1/16 --
--id=@m3 create Mirror name=mymirror3 select-dst-port=@te-1/1/11 output-port=@te-1/1/12 --
--id=@m4 create Mirror name=mymirror4 select-dst-port=@te-1/1/11 output-port=@te-1/1/16
```

Deleting the Mirroring

```
root@PicOS-OVS# ovs-vsctl clear Bridge br0 mirrors
```

Configuring IPv4 OpenFlow

PicOS OVS supports IPv4 flow in OpenFlow.

Creating an IPv4 flow

```
root@PicOS-OVS# ovs-ofctl add-flow br0
dl_src=22:11:11:11:11:11,dl_dst=22:00:00:00:00:00,in_port=1,dl_type=0x0800,nw_src=128.1.1.1
,nw_dst=128.1.1.2,nw_proto=6,actions=output:2,3,4
root@PicOS-OVS#
root@PicOS-OVS# ovs-ofctl dump-flows br0
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=12.758s, table=0, n_packets=0, n_bytes=0,
tcp,in_port=1,dl_src=22:11:11:11:11:11,dl_dst=22:00:00:00:00:00,nw_src=128.1.1.1,nw_dst=128
.1.1.2 actions=output:2,output:3,output:4
cookie=0x0, duration=2180.111s, table=0, n_packets=0, n_bytes=0, priority=0 actions=NORMAL
root@PicOS-OVS#
```

Deleting an IPv4 flow

```
root@PicOS-OVS# ovs-ofctl del-flows br0
dl_src=22:11:11:11:11:11,dl_dst=22:00:00:00:00:00,in_port=1,dl_type=0x0800,nw_src=128.1.1.1
,nw_dst=128.1.1.2,nw_proto=6
root@PicOS-OVS#
```

Removing all flows

```
root@PicOS-OVS# ovs-ofctl del-flows br0
root@XorPlus
```

Configure GRE Tunneling

PicOS OVS supports IP GRE tunneling.

GRE ON Physical PORT

Creating a GRE tunnel

```
root@PicOS-OVS# ovs-vsctl add-port br0 gre1 -- set Interface gre1 type=pica8_gre
options:remote_ip=10.10.60.10 options:local_ip=10.10.61.10 options:vlan=1
options:src_mac=00:11:11:11:11:11 options:dst_mac=00:22:22:22:22:22
options:egress_port=ge-1/1/5
```

If the user wants to create a GRE tunnel, the user will need to configure a GRE tunnel along with two flows, which are used for sending traffic to the GRE and sending output from the GRE respectively.

```
root@PicOS-OVS# ovs-ofctl add-flow br0 in_port=1,actions=output:3073
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=3073,actions=mod_dl_src:00:11:11:11:11:11,mod_dl_dst:00:33:33:33:33:33,output:1
```

The GRE port number starts from 3073, which is the port number of GRE1. The first flow in the example above, is configured so that all traffic from port ge-1/1/1 will be sent to the GRE tunnel, whose port number is 3073. The second flow is configured so that all of the traffic coming out from the GRE tunnel will be forwarded to port ge-1/1/1 and the source MAC address modified to the switch's MAC address and the destination MAC address to the MAC address of the internal target. (If user doesn't specify the dl_src in the action of second flow, the src mac will be the switch's mac, but the dst mac must be specified.)

Description

1. br0: bridge name
2. remote_ip=10.10.60.10: the IP address of the peer GRE tunnel interface; this IP address will be the destination IP of the encapsulated GRE packets

3. local_ip=10.10.61.10: the IP address of this GRE tunnel interface; this IP address will be the source IP of the encapsulated GRE packets
4. src_mac=00:11:11:11:11:11: the logical MAC address of the GRE tunnel interface; this MAC address will be the source MAC of the encapsulated GRE packets to next-hop
5. dst_mac=00:22:22:22:22:22: the next-hop MAC address; this MAC address will be the destination MAC the encapsulated GRE packets to next-hop
6. egress_port=ge-1/1/5: the output port of the encapsulated GRE packets

GRE ON LAG/LACP PORT

Static Lag and GRE Tunnel

Command:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 ael vlan_mode=trunk tag=1 -- set interface ael type=pica8_lag options:members=te-1/1/78,te-1/1/80
admin@PicOS-OVS$ovs-vsctl -- set interface ael options:lag_type=static
admin@PicOS-OVS$ovs-vsctl add-port br0 gre1 -- set Interface gre1 type=pica8_gre
options:local_ip=10.10.60.10 options:remote_ip=10.10.61.10 options:vlan=1012
options:dst_mac=C8:0A:A9:9E:14:A5 options:src_mac=C8:0A:A9:04:49:1A options:egress_port=ael
```

Flows:

```
root@PicOS-OVS# ovs-ofctl add-flow br0 in_port=1,actions=output:3073
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=3073,actions=mod_dl_src:00:11:11:11:11:11,mod_dl_dst:00:33:33:33:33:33,output:1
```

LACP and GRE Tunnel

Command:

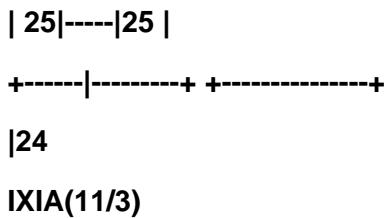
```
admin@PicOS-OVS$ovs-vsctl add-port br0 ael vlan_mode=trunk tag=1 -- set interface ael type=pica8_lag options:members=te-1/1/78,te-1/1/80
admin@PicOS-OVS$ovs-vsctl -- set interface ael options:lag_type=static
admin@PicOS-OVS$ovs-vsctl add-port br0 gre1 -- set Interface gre1 type=pica8_gre
options:local_ip=10.10.60.10 options:remote_ip=10.10.61.10 options:vlan=1012
options:dst_mac=C8:0A:A9:9E:14:A5 options:src_mac=C8:0A:A9:04:49:1A options:egress_port=ael
```

Flows:

```
root@PicOS-OVS# ovs-ofctl add-flow br0 in_port=1,actions=output:3073
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=3073,actions=mod_dl_src:00:11:11:11:11:11,mod_dl_dst:00:33:33:33:33:33,output:1
```

Examples:

ixia(11/1) — |22 sw1 23|----|23 sw2 |22-----ixia(11/2)



SW1 CONFIG:

Step1: Create a bridge and add two ports (PX1,PX2,PX3)

```

ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
ovs-vsctl add-port br0 ge-1/1/22 vlan_mode=trunk tag=1 -- set interface ge-1/1/22 type=pica8
ovs-vsctl add-port br0 ge-1/1/23 vlan_mode=trunk tag=1 -- set interface ge-1/1/23 type=pica8
ovs-vsctl add-port br0 ge-1/1/25 vlan_mode=trunk tag=1 -- set interface ge-1/1/25 type=pica8
  
```

Step 2: Add static lag port and gre port

```

ovs-vsctl add-port br0 ae1 vlan_mode=trunk tag=1 -- set interface ae1 type=pica8_lag
options:members=ge-1/1/23,ge-1/1/25
ovs-vsctl -- set interface ae1 options:lag_type=static
ovs-vsctl add-port br0 gre1 -- set Interface gre1 type=pica8_gre options:local_ip=10.10.60.10
options:remote_ip=10.10.61.10 options:vlan=1012 options:dst_mac=C8:0A:A9:9E:14:A5
options:src_mac=C8:0A:A9:04:49:1A options:egress_port=ae1
  
```

Step 3: Add flow

```

ovs-ofctl add-flow br0 in_port=22,actions=output:3073
ovs-ofctl add-flow br0
in_port=3073,actions=set_field:66:66:66:44:44:44->dl_dst,set_field:66:66:66:33:33:33->dl_src,out
  
```

SW2 CONFIG:

Step1: Create a bridge and add two ports (PX1,PX2,PX3)

```

ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
ovs-vsctl add-port br0 ge-1/1/22 vlan_mode=trunk tag=1 -- set interface ge-1/1/22 type=pica8
ovs-vsctl add-port br0 ge-1/1/23 vlan_mode=trunk tag=1 -- set interface ge-1/1/23 type=pica8
ovs-vsctl add-port br0 ge-1/1/25 vlan_mode=trunk tag=1 -- set interface ge-1/1/25 type=pica8
  
```

Step 2: Add flow

```

ovs-ofctl add-flow br0 in_port=23,actions=output:22
ovs-ofctl add-flow br0 in_port=25,actions=output:22
  
```

SEND Packets

Step 4: Send ipv4 packets,check the result(limitation:the src mac and vlan of the packets must be the same with gre tunnel)

(1) send no changing packets to PX1,

Result:

then PX2 or PX3 will transmit all the encapsulated packets

```
check nw_dst=10.10.61.10,nw_src= 10.10.60.10,vlan=1012,src_mac=C8:0A:A9:04:49:1A
,dst_mac=C8:0A:A9:9E:14:A5
```

Check:

dst mac: C8 0A A9 9E 14 A5

src mac: C8 0A A9 04 49 1A

offset:12

vlan: 81 00 03F4

dst ip: 0A 0A 3D 0A

src ip: 0A 0A 3C 0A

(2) sending increasing src ip to PX1,(first enable hash mapping mode:ovs-vsctl -- set Interface ae1 options:hash-mapping=advance ovs-vsctl set-lag-advance-hash-mapping-fields nw_src)

Result:

then both PX2 and PX3 will transmit the encapsulated packets

```
check nw_dst=10.10.61.10,nw_src= 10.10.60.10,vlan=1012,src_mac=C8:0A:A9:04:49:1A
,dst_mac=C8:0A:A9:9E:14:A5
```

Then reconfigure sw2

Step1: Add static lag port and gre port

```
ovs-vsctl add-port br0 ae1 vlan_mode=trunk tag=1 -- set interface ae1 type=pica8_lag
options:members=ge-1/1/23,ge-1/1/25
```

```
ovs-vsctl -- set interface ae1 options:lag_type=static
```

```
ovs-vsctl add-port br0 gre1 -- set Interface gre1 type=pica8_gre options:remote_ip=10.10.60.10
options:local_ip=10.10.61.10 options:vlan=1012 options:src_mac=C8:0A:A9:9E:14:A5
options:dst_mac=C8:0A:A9:04:49:1A options:egress_port=ae1
```

Step 2: Add flow

```
ovs-ofctl add-flow br0 in_port=22,actions=output:3073
```

```
ovs-ofctl add-flow br0
```

```
in_port=3073,actions=set_field:66:66:66:44:44:44->dl_dst,set_field:66:66:66:33:33:33->dl_src,out
```

SEND Packets

(1) sending no changing packets to ge-1/1/22(SW1),

Result:

ge-1/1/22(SW2) will transmit the decapsulated packets.

check `dl_src=66:66:66:33:33:33,dl_dst=66:66:66:44:44:44,vlan 1012`

Configuring MPLS

The basic MPLS actions are Push, Swap and Pop. Beginning with PicOS 2.4, user does not need to configure the `dl_src`. The packet `src_mac` pushes the MPLS to the MAC address of this switch.

User can add flows to modify and copy the MPLS TTL and IP TTL.

User can push/pop 2 MPLS labels per flow.



Every un-tagged packet is tagged with the default VLAN-ID before Push, Pop and Swap.

Hardware or Software based Forwarding

The flow is pre-installed in hardware if there is enough information on the Flow to be processed by the ASIC.

Here is the minimal set of information required to process the packet on hardware only,**before version 2.8.1:**

If the Flow action is a POP : `dl_dst, vlan_id, mpls_lse`

If the Flow action is a PUSH : `dl_dst, vlan_id, dl_type,mpls_lse` (only being needed when `dl_type` is `0x8847` or `0x8848`)

If the Flow action is a SWAP :`dl_dst, vlan_id, mpls_lse`

Since version 2.8.1, configuration is changed, the minimal set on hardware only as shown below:

If the Flow action is a PUSH:

`push mpls for ip packet: in_port,dl_vlan,dl_dst,dl_type`

`push mpls for mpls packet: in_port,dl_vlan,dl_dst,dl_type(mpls),mpls_label`

If the Flow action is a SWAP: `in_port,dl_vlan,dl_dst,dl_type(mpls),mpls_label`

If the Flow action is a POP: `in_port,dl_vlan,dl_dst,dl_type(mpls),mpls_label`



1. There is one exception. If the action is "pop_mpls:0x8847" and matches enough fields (`dl_dst,vlan_id,mpls_label`), the flow becomes a direct flow (hardware only). If the action is "pop_mpls:0x0800", the flow is packet-driven flow.
2. If the flow actions a push, flow match must include `dl_type=0x0800/dl_type=0x86dd/dl_type=0x8847/dl_type=0x8848`. If the flow match not include one of them, the flow entry cannot add.

If there is some information missing to process the packet, the flow becomes a "packet-driven" flow. This means that the first MPLS packet is sent to the CPU, which analyzes it and downloads a new flow on the hardware with the missing information to handle the packet in hardware. Following packets for this specific flow will then be handled by the hardware (ASIC) without reaching the Switch CPU.

PUSH MPLS

Pushing an MPLS Label

In the following configuration, user specifies a flow, which should match:

```
{ in_port=1,dl_type=0x0800, dl_dst=22:00:00:00:00:00,dl_vlan=1}
```

The action is to push an MPLS label (i.e. 10) and forward to port te-1/1/2

Note that MPLS TTL will copy from the IP header and decrease

Pushing two MPLS Labels

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,dl_dst=22:00:00:00:00:00,dl_vlan=1,actions=push_mpls:0x8847,set_field:10->mpls_label,output:2
```

In the following configuration, specify a flow, which should match {

in_port=1,dl_type=0x0800,dl_dst=22:00:00:00:00:00,dl_vlan=1}, the action is to push two labels (i.e. 10 and 20) and forward to port te-1/1/2

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,dl_dst=22:00:00:00:00:00,dl_vlan=1,actions=push_mpls:0x8847,set_field:10->mpls_label,push_mpls:0x8847,set_field:20->mpls_label,output:2
root@PicOS-OVS#
```

SWAP MPLS Label

Swapping MPLS Labels

In following configuration, user specifies a flow, which should match {

in_port=1,dl_type=0x8847,dl_dst=22:00:00:00:00:00,dl_vlan=1,mpls_label=10}, the action is to swap label 10 with 20 and forward to port te-1/1/2

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=1,dl_type=0x8847,dl_dst=22:00:00:00:00:00,dl_vlan=1,mpls_label=10,actions=set_field:20->mpls_label,output:2
root@PicOS-OVS#
```

POP MPLS Label

Popping an MPLS Label of the flow

In following configuration, specify a flow, which should match {
 in_port=1,dl_type=0x8847,dl_dst=22:00:00:00:00:00,dl_vlan=1,mpls_label=10}, the action is to pop
 MPLS label and forward to port te-1/1/2.
 Note that MPLS TTL will be copied to IP header TTL and decremented by 1.

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=1,dl_type=0x8847,dl_dst=22:00:00:00:00:00,dl_vlan=1,mpls_label=10,actions=pop_mpls:
0x8847,output:2
```

Popping one MPLS Label for flows with Two MPLS Labels

In the following configuration, specify a flow that has two MPLS labels (i.e. 10 and 20). The pop action is always popping the outer MPLS header. Note that the two label flow pops only one label, the output packet is also a MPLS packet. Thus, the "pop_mpls:0x8847" must be configured.

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=1,dl_type=0x8847,dl_dst=22:00:00:00:00:00,dl_vlan=1,mpls_label=10,actions=pop_mpls:
0x8847,output:2
```

Popping two MPLS Labels for flows with two MPLS Labels

In following configuration, specify a flow which has two labels to pop. The output flow is IP packet. Configure two pop entries to pop the flow.

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=1,dl_type=0x8847,dl_dst=22:00:00:00:00:00,dl_vlan=1,actions=pop_mpls:0x0800,output:
2
```

PUSH MPLS Label and VLAN

Pushing one MPLS Label and one VLAN

In following configuration, specify flows to push one MPLS Label and one VLAN

```
ovs-ofctl add-flow br0
in_port=2,dl_type=0x0800,dl_vlan=2999,dl_dst=22:22:22:22:22:22,actions=push_mpls:0x8847,set_
_field:333->mpls_label,push_vlan:0x8100,set_field:1999->vlan_vid,output:4
ovs-ofctl add-flow br0
in_port=2,dl_type=0x8847,dl_vlan=2999,dl_dst=22:22:22:22:22:22,mpls_label=66,actions=push_m
pls:0x8847,set_field:333->mpls_label,push_vlan:0x8100,set_field:1999->vlan_vid,output:4
```

Pushing two MPLS Labels and one VLAN

In following configuration, specify flows to push two mpls labels and one VLAN

```
ovs-ofctl add-flow br0
in_port=2,dl_type=0x0800,dl_vlan=2999,actions=push_mpls:0x8847,set_field:333->mpls_label,p
ush_mpls:0x8847,set_field:66->mpls_label,push_vlan:0x8100,set_field:1999->vlan_vid,output
:4
```

POP One or Two MPLS Labels

In following configuration, specify flows which should match dl_type,dl_vlan and the MPLS label. The action is to pop one MPLS label and set a new src mac address. The first flow will pop one MPLS label and the second flow will pop two MPLS labels.

```
ovs-ofctl add-flow br0
in_port=2,dl_type=0x8847,dl_vlan=2999,mpls_label=333,dl_dst=22:22:22:22:22:22,actions=pop_mpls:0x8847,mac_src:00:0c:29:99:99:99,mac_dst:00:0c:29:99:99:99,actions=set_field:22:22:22:44:44:44->eth_src,output:4
ovs-ofctl add-flow br0
in_port=2,dl_type=0x8847,dl_vlan=2999,actions=pop_mpls:0x0800,output:4
```

POP One or Two MPLS Labels and PUSH VLAN

The following flow should match dl_type and dl_vlan, action is to pop one MPLS label and push one VLAN.

```
ovs-ofctl add-flow br0
in_port=2,dl_type=0x8847,dl_vlan=2999,actions=pop_mpls:0x8847,push_vlan:0x8100,set_field:1999->vlan_vid,output:4
```

The following flow should match dl_type and dl_vlan, with the action of popping two MPLS labels and pushing one VLAN.

```
ovs-ofctl add-flow br0
in_port=2,dl_type=0x8847,dl_vlan=2999,actions=pop_mpls:0x8847,pop_mpls:0x8847,push_vlan:0x8100,set_field:1999->vlan_vid,output:4
```

The following flow should match dl_type and dl_vlan and mpls label, and the action is to pop two MPLS labels and push one VLAN.

```
ovs-ofctl add-flow br0
in_port=2,dl_type=0x8847,dl_vlan=2999,mpls_label=333,actions=pop_mpls:0x0800,push_vlan:0x8100,set_field:1999->vlan_vid,output:4
```

The following flow should match dl_type, the action is to pop two MPLS labels and push one VLAN.

```
ovs-ofctl add-flow br0
in_port=2,dl_type=0x8847,actions=pop_mpls:0x8847,push_vlan:0x8100,set_field:1999->vlan_vid,output:4
```

PUSH MPLS and POP VLAN

In the following configuration, push MPLS label and pop VLAN have been supported. Following flow, the action is to push one MPLS label and pop one VLAN.

```
ovs-ofctl add-flow br0
in_port=2,dl_type=0x8847,dl_vlan=2999,actions=push_mpls:0x8847,set_field:333->mpls_label,p
op_vlan,output:4
```

PUSH Two MPLS Labels and POP VLAN

Following flow, action is to push two MPLS labels and pop VLAN.

```
ovs-ofctl add-flow br0
in_port=2,dl_type=0x8847,dl_vlan=2999,actions=push_mpls:0x8847,set_field:333->mpls_label,p
ush_mpls:0x8847,set_field:222->mpls_label,pop_vlan,output:4
```

PUSH Two MPLS Labels and POP Two VLANs

Following flow, the action is to push two MPLS labels and pop two VLANs.

```
ovs-ofctl add-flow br0
in_port=1,dl_type=0x8847,dl_vlan=1666,actions=push_mpls:0x8847,set_field:333->mpls_label,p
ush_mpls:0x8847,set_field:222->mpls_label,pop_vlan,pop_vlan,output:2
```

NOTICE:

1. If flows match dl_vlan and the actions have push_vlan, then receives packets only carry the pushed vlan.
2. Hardware can't support pop_mpls and pop_vlan at the same time, and the packets can't forward with line-speed.
3. When actions don't appoint to modify dl_src, the src mac of received packets should be modified to bridge mac whatever for direct flows or packet-driven flow.
4. Push two mpls labels is supported, but push two vlans at the same time is not supported.
5. Hardware can't support push two vlan_header.

Configuring LAG and LACP

LAG/LACP

Link aggregation (LAG) refers to *aggregating* multiple physical network links in parallel in order to increase throughput beyond what a single link could sustain, and provide redundancy in case any of the links should fail.

Link Aggregation Control Protocol (LACP) refers to bundling of several physical ports together to form a single logical channel. Network devices use LACP to negotiate automatic bundling of links by sending LACP packets to its directly connected peer(s)

LACP features and practical examples:

1. Maximum number of bundled ports allowed in the port channel: valid values are from 1 to 8.
2. LACP packets are sent with a well known multicast group MAC address 01:80:c2:00:00:02
3. LACP detection period:
 - LACP packets are transmitted every second
 - Keep alive mechanism for link members: (default: slow = 30s, fast=1s)
4. LACP can have the port-channel load-balance mode:
 - load-balancing can be configured using a combination of in_port, L2, L3 and L4 headers.
5. LACP mode:
 - active: Enables LACP unconditionally.
 - passive: Enables LACP only when an LACP device is detected. (This is the default state)



PicOS OVS supports LAG and LACP.

PicOS can support 48 LAG or LACP at most. Each LAG has a maximum of 8 member ports.

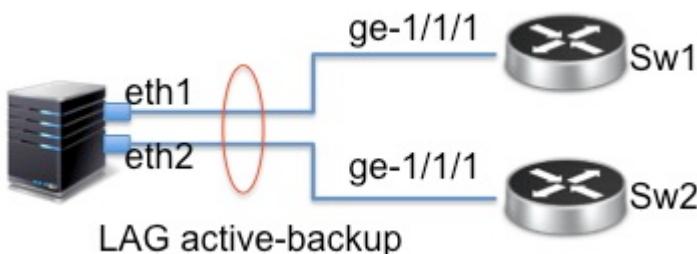
Create a Static LAG

In following configuration, user can create LAG ae1, and add port 2 and port 3 into this LAG

```
root@PicOS-OVS# ovs-vsctl add-port br0 ae1 vlan_mode=trunk tag=1 -- set Interface ae1 type=pica8 Lag
root@PicOS-OVS# ovs-vsctl -- set Interface ae1 options:lag_type=static
root@PicOS-OVS# ovs-vsctl -- set Interface ae1 options:members=ge-1/1/2,ge-1/1/3
```

Dual Homing a Host

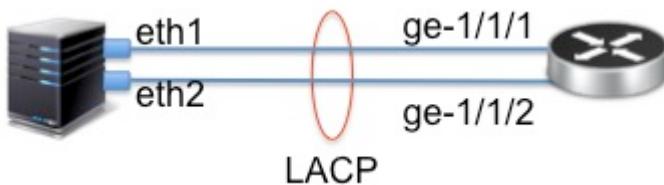
While dual-homing a host does not require a LAG configuration on the switch, this configuration illustrates steps required to setup LAG on the host connected to two different switches.



Host LAG Configuration	Switch Port Configuration
<pre>pica8@Ubuntu\$ more /etc/modprobe.d/bonding.conf bonding mode=1 miimon=100 pica8@Ubuntu\$ more /etc/network/interfaces # NIC bonding config for LAG active-backup START auto eth1 iface eth1 inet manual bond-master bond0 bond-primary eth1 auto eth2 iface eth2 inet manual bond-master bond0 auto bond0 iface bond0 inet static address 10.0.0.20 netmask 255.255.255.0 bond-mode active-backup bond-miimon 100 bond-slaves none # NIC bonding config for LAG active-backup END pica8@Ubuntu\$ cat /proc/net/bonding/bond0 Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011) Bonding Mode: fault-tolerance (active-backup) Primary Slave: eth1 (primary_reselect always) Currently Active Slave: eth1 MII Status: up MII Polling Interval (ms): 100 Up Delay (ms): 0 Down Delay (ms): 0 Slave Interface: eth2 MII Status: up Speed: 1000 Mbps Duplex: full Link Failure Count: 0 Permanent HW addr: 00:0c:29:70:2e:c2 Slave queue ID: 0 Slave Interface: eth1 MII Status: up Speed: 1000 Mbps</pre>	<pre>OVS-Sw1\$ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set interface ge-1/1/1 type=pica8 OVS-Sw2\$ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set interface ge-1/1/1 type=pica8</pre>

Host LAG Configuration	Switch Port Configuration
Duplex: full	
Link Failure Count: 0	
Permanent HW addr: 00:0c:29:70:2e:b8	
Slave queue ID: 0	

Create a LACP Port



Following is the configuration on the host (running Ubuntu):

```
pica8@Ubuntu:~$ more /etc/modprobe.d/bonding.conf
bonding mode=4 miimon=100 lACP_rate=1

pica8@Ubuntu:~$ more /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet dhcp

# NIC bonding config with LACP START
auto eth1
iface eth1 inet manual
    bond-master bond0
auto eth2
iface eth2 inet manual
    bond-master bond0
auto bond0
iface bond0 inet static
    address 10.0.0.10
    netmask 255.255.255.0
bond-mode 802.3ad
bond-miimon 100
bond-lACP-rate 1
bond-slaves none
# NIC bonding config with LACP END

pica8@Ubuntu:~$ /etc/init.d/networking restart

pica8@Ubuntu:~$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)
Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2 (0)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
```

```

Down Delay (ms): 0

802.3ad info
LACP rate: fast
Min links: 0
Aggregator selection policy (ad_select): stable
Active Aggregator Info:
Aggregator ID: 3
Number of ports: 2
Actor Key: 17
Partner Key: 1
Partner Mac Address: 00:11:11:11:11:11

Slave Interface: eth1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:0c:29:fd:75:ae
Aggregator ID: 3
Slave queue ID: 0

Slave Interface: eth2
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:0c:29:fd:75:b8
Aggregator ID: 3
Slave queue ID: 0

```

Following is the LACP configuration on the switch.

The default (slow) LACP timer is 30s whereas fast timer is 1s. LACP mode determines which node initiates the negotiation: active sets the switch to initiate the LACP handshake, while in passive mode the switch accepts all incoming LACP requests.

```

admin@PicOS-OVS$ ovs-vsctl add-port br0 ae1 vlan_mode=trunk tag=1 -- set Interface ae1
type=pica8_lag
admin@PicOS-OVS$ ovs-vsctl -- set Interface ae1 options:lag_type=lACP
admin@PicOS-OVS$ ovs-vsctl -- set Interface ae1 options:members=ge-1/1/1,ge-1/1/2
admin@PicOS-OVS$ ovs-vsctl -- set Interface ae1 options:lACP-system-id=00:11:11:11:11:11
admin@PicOS-OVS$ ovs-vsctl -- set Interface ae1 options:lACP-time=fast | slow

Optional Settings:
admin@PicOS-OVS$ ovs-vsctl -- set Interface ae1 options:lACP-system-priority=32768
admin@PicOS-OVS$ ovs-vsctl -- set Interface ae1 options:lACP-mode=active | passive
admin@PicOS-OVS$ ovs-vsctl -- set Interface ge-1/1/2 options:lACP-port-id=2
admin@PicOS-OVS$ ovs-vsctl -- set Interface ge-1/1/2 options:lACP-port-priority=32768
admin@PicOS-OVS$ ovs-vsctl -- set Interface ge-1/1/2 options:lACP-aggregation-key=0

```

From PicOS version 2.7.1, support is added to configure minimum number port in a LACP or LAG. The command is as follows:

```
ovs-vsctl set Interface <lag port> options:lag_selected_min=<value>
```

```
admin@PicOS-OVS$ ovs-vsctl set Interface ae1 options:lag_selected_min=1
```

Create Static Flow for LAG or LACP

In following configuration, user can create static flow whose output port is a LAG or LACP.

```
root@PicOS-OVS# ovs-ofctl add-flow br0 in_port=1025,actions=output:1
root@PicOS-OVS# ovs-ofctl add-flow br0 in_port=1,actions=output:1025
```

LAG number index is shown as following:

LAG Name	LAG Number
ae1	1025
ae2	1026
..	..
ae1023	2047

Display the LACP Information

User can display the LACP information with the following CLI commands.

```
admin@PicOS-OVS$ovs-appctl lacp/show

---- ae1 ---
status: active negotiated
sys_id:00:11:11:11:11:11
sys_priority: 32768
aggregation key: 1
lacp_time: fast
slave: ge-1/1/1: current attached
port_id: 1
port_priority: 32768
may_enable: true
actor sys_id:00:11:11:11:11:11
actor sys_priority:32768
actor port_id: 1
actor port_priority:32768
actor key: 1
actor state: activity timeout synchronized collecting distributing
partner sys_id:00:0c:29:fd:75:b8
partner sys_priority:65535
partner port_id: 2
partner port_priority:255
partner key: 17
partner state:activity timeout aggregation synchronized collecting distributing

slave: ge-1/1/2: current attached
port_id: 1
port_priority: 32768
may_enable: true
actor sys_id:00:11:11:11:11:11
actor sys_priority:32768
actor port_id: 1
actor port_priority:32768
actor key: 1
actor state: activity timeout synchronized collecting distributing
partner sys_id:00:0c:fe:ae:69:ac
```

```

partner sys_priority:65535
partner port_id: 2
partner port_priority:255
partner key: 17
partner state:activity timeout aggregation synchronized collecting distributing

```

Lag Hash Configuration

Lag hash commands are as follows.

```

# Config command
ovs-vsctl -- set Interface ae1 options:hash-mapping=dl_dst
ovs-vsctl -- set Interface ae1 options:hash-mapping=dl_src_dst
ovs-vsctl -- set Interface ae1 options:hash-mapping=dl_src
ovs-vsctl -- set Interface ae1 options:hash-mapping=nw_dst
ovs-vsctl -- set Interface ae1 options:hash-mapping=nw_src_dst
ovs-vsctl -- set Interface ae1 options:hash-mapping=nw_src
ovs-vsctl -- set Interface ae1 options:hash-mapping=resilient
ovs-vsctl -- set Interface ae1 options:hash-mapping=advance
ovs-vsctl -- set-lag-advance-hash-mapping-fields dl_dst dl_src ether_type in_port nw_dst
nw_proto nw_src port_dst port_src
vlan
# Show command
ovs-vsctl show-lag-advance-hash-mapping-fields

```

If not special the hash-mapping in one lag interface, dl_src_dst will be set by default; And if not special the hash-mapping in set-lag-advance-hash-mapping-fields, all field will be set.

LAG speed

From PicOS version 2.7.2, support for LAG speed has been added. User can check the LAG speed configuration with the command "**ovs-ofctl show br0**".

Examples

Topology

```

ge-1/1/1-----connect ixia1
ge-1/1/2-----connect ixia2
ge-1/1/3-----connect ixia3
ge-1/1/4-----connect ixia4

```

Step 1: Create a bridge and add four ports (PX1,PX2,PX3)

```

ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set interface ge-1/1/1 type=pica8
ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1 -- set interface ge-1/1/2 type=pica8
ovs-vsctl add-port br0 ge-1/1/3 vlan_mode=trunk tag=1 -- set interface ge-1/1/3 type=pica8
ovs-vsctl add-port br0 ge-1/1/4 vlan_mode=trunk tag=1 -- set interface ge-1/1/4 type=pica8

```

Step 2: Add a lag port

```
ovs-vsctl add-port br0 ae1 vlan_mode=trunk tag=1 -- set interface ae1 type=pica8_lag
options:members=ge-1/1/2,ge-1/1/3,ge-1/1/4
ovs-vsctl -- set interface ae1 options:lag_type=static
```

Check the lag port speed

```
ovs-ofctl show br0
check "1025(ae1).*state.* LINK_UP"
check "speed.*300 Mbps"
```

Step 3: Add flow

```
ovs-ofctl add-flow br0 in_port=1,actions=output:1025
check the flows:
ovs-ofctl dump-flows br0
check "in_port=1"
ovs-appctl pica/dump-flows
check "in_port=1"
```

Step 4: Send ipv4 packets and check the result

(1) Send no changing packets to

PX1,(dl_src=22:11:11:11:11:11,dl_dst=22:22:22:22:22:22,dl_vlan=10,ip,nw_src=192.168.1.10,nw_dst=192.168.1.11)

Result:

```
then PX2,PX3 or PX4 will transmit all the packets.
Check:
dl_src: 22 11 11 11 11 11
dl_dst:22 22 22 22 22 22
offset:30
ip:C0 A8 01 0A C0 A8 02 0A
```

(2)sending increasing src mac to PX1,

(dl_src=22:11:11:11:11:11(increase),dl_dst=22:22:22:22:22:22,dl_vlan=10,ip,nw_src=192.168.1.10,nw_dst=192.168.1.11)

Result:

```
then both PX2, PX3 and PX4 will transmit the packets.
Check:
dl_dst:22 22 22 22 22 22
offset:30
ip:C0 A8 01 0A C0 A8 02 0A
```

Step 5: Down one or more port of lag and check the lag speed

(1)ovs-ofctl mod-port br0 ge-1/1/2 down

```
ovs-ofctl show br0
check "1025(ae1).*state.* LINK_UP"
check "speed.*200Mbps"
```

(2)ovs-ofctl mod-port br0 ge-1/1/3 down

```
ovs-ofctl show br0
check "1025(ae1).*state.* LINK_UP"
check "speed.*100Mbps"
```

(3)ovs-ofctl mod-port br0 ge-1/1/4 down

```
ovs-ofctl show br0
check "1025(ae1).*state.* LINK_DOWN"
check "speed.*0Mbps"
```

Step 6: Up all the ports of lag and check the lag speed

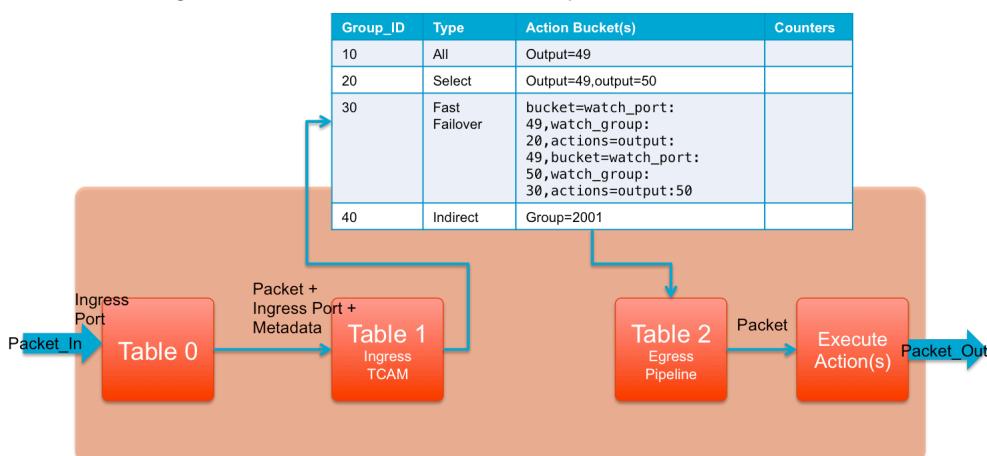
```
ovs-ofctl mod-port br0 ge-1/1/2 up
ovs-ofctl mod-port br0 ge-1/1/3 up
ovs-ofctl mod-port br0 ge-1/1/4 up
ovs-ofctl show br0
check "1025(ae1).*state.* LINK_UP"
check "speed.*300Mbps"
```

Step 7: Test end and clear the configure

```
ovs-vsctl del-br br0
```

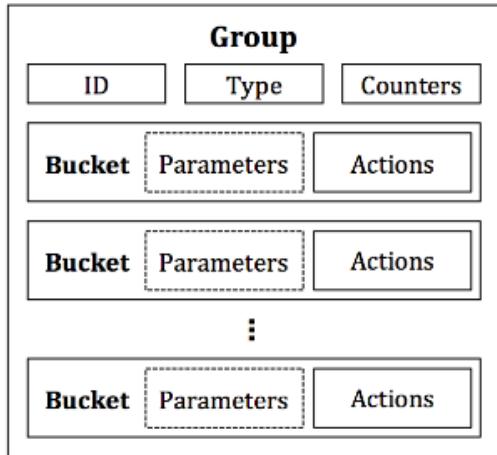
Group Tables

Group tables enable Openflow to process forwarding decisions on multiple links. Examples include: load-balancing, multicast, and active/standby.



The above figure illustrates how group-tables can simplify the configuration required to consolidate forwarding decisions for flows in an Openflow pipeline.

An **OpenFlow group** is an abstraction that facilitates more complex and specialized packet operations that cannot easily be performed through a flow table entry. Each group receives packets as input, and performs any OpenFlow actions on these packets. A group is not capable of performing any OpenFlow instructions, so it cannot send packets to other flow tables or meters. Furthermore, it is expected that packets have been matched appropriately prior to entry to a group, as groups do not support matching on packets. Groups are merely mechanisms to perform advanced actions, or sets of actions.



As shown in the above figure, the power of a group is that it contains separate lists of actions, and each individual action list is referred to as an **OpenFlow bucket**. Thus, it is said that a group contains a bucket list (or a list of lists of actions). Each bucket or list of buckets can be applied to entering packets; the exact behavior depends on the group type. There are certain types of groups that make use of additional parameters within a bucket. The details of these parameters will be discussed with each group type, where applicable.

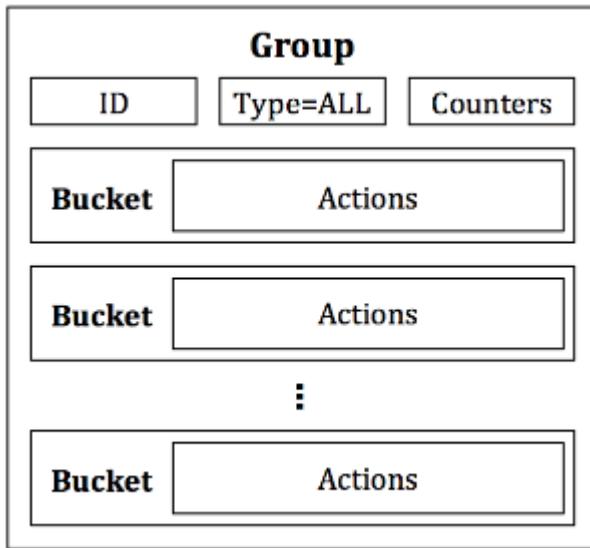
There are four types of groups: **ALL**, **SELECT**, **INDIRECT**, and **FAST-FAILOVER**.



1. PicOS OVS supports group tables in Openflow 1.2 Openflow 1.3 and Openflow 1.4
2. The number of buckets supported is dependent on the TCAM size in the ASIC. So there is a possibility that all defined group tables will not be installed in hardware.

The ALL Group

Starting with one of the simplest, the **ALL group**, illustrated in the following figure, will take any packet received as input, and duplicate it to be operated on independently by each bucket in the bucket list. In this way, an ALL group can be used to replicate and then operate on separate copies of the packet defined by the actions in each bucket. Different and distinct actions can be in each bucket, which allows different operations to be performed on different copies of the packet.



Limitation in PicOS switch:

Due to limitation in the ASIC, PicOS OVS switches do not support replicating the packet and then operating by the actions in each bucket. So if user wants to configure type=all group, the actions of different buckets must be same. In other word, all the buckets need be uniform.

Example:

```

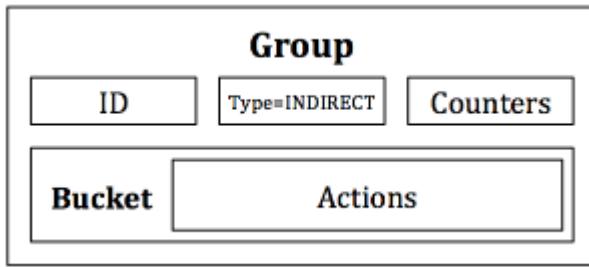
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=20,type=all,bucket=mod_dl_src=00:00:00:11:11:11,mod_dl_dst=22:22:22:00:00:00,output:13,bucket=mod_dl_src=00:00:00:11:11:11,mod_dl_dst=22:22:22:00:00:00,output:14
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=11,dl_src=22:11:11:11:11:11,dl_dst=22:00:00:00:00:00,dl_type=0x0800,nw_proto=6,nw_src=1.1.2.100,nw_dst=2.2.2.100,actions=group:20

admin@PicOS-OVS$ovs-ofctl dump-groups br0 20
OFPST_GROUP_DESC reply (OF1.4) (xid=0x2):
group_id=20,type=all,bucket=actions=set_field:00:00:00:11:11:11->eth_src,set_field:22:22:22:00:00->eth_dst,output:13,bucket=actions=set_field:00:00:00:11:11:11->eth_src,set_field:22:22:22:00:00->eth_dst,output:14

admin@PicOS-OVS$ovs-appctl pica/dump-flows
#134 normal permanent
recirc_id=0,tcp,in_port=11,dl_src=22:11:11:11:11:11,dl_dst=22:00:00:00:00:00,nw_src=1.1.2.100,nw_dst=2.2.2.100,actions:set(eth(src=00:00:00:11:11:11,dst=22:22:22:00:00:00)),13,14
  
```

The INDIRECT Group

The **INDIRECT group** illustrated in the figure below, can be difficult to comprehend as a “group,” since it contains only a single bucket, where all packets received by the group are sent to this lone bucket. In other words, the INDIRECT group does not contain a list of buckets but a single bucket (or single list of actions) instead. The purpose of the INDIRECT group is to encapsulate a common set of actions used by many flows. For example, if flows A, B, and C match on different packet headers but have a common set or subset of actions, these flows can send packets to the single INDIRECT group as opposed to having to duplicate the list of common actions for each flow. The INDIRECT group is used to simplify an OpenFlow deployment and reduce the memory footprint of a set of similar flows.



```

admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=30,type=indirect,bucket=mod_dl_src=00:00:00:11:11:11,mod_dl_dst=22:22:22:00:00:00,
output:14
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=11,dl_src=22:11:11:11:11:11,dl_dst=22:00:00:00:00:00,dl_type=0x0800,nw_proto=6,nw_s
rc=1.1.2.100,nw_dst=2.2.2.100,actions=group:30
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=11,dl_src=22:11:11:11:22,dl_dst=22:00:00:00:00:00,dl_type=0x0800,nw_proto=6,nw_s
rc=1.1.2.200,nw_dst=2.2.2.100,actions=group:30

admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  cookie=0x0, duration=7.342s, table=0, n_packets=n/a, n_bytes=0,
  tcp,in_port=11,dl_src=22:11:11:11:22,dl_dst=22:00:00:00:00:00,nw_src=1.1.2.200,nw_dst=2.
  2.2.100 actions=group:30
  cookie=0x0, duration=31.446s, table=0, n_packets=n/a, n_bytes=0,
  tcp,in_port=11,dl_src=22:11:11:11:11,dl_dst=22:00:00:00:00:00,nw_src=1.1.2.100,nw_dst=2.
  2.2.100 actions=group:30

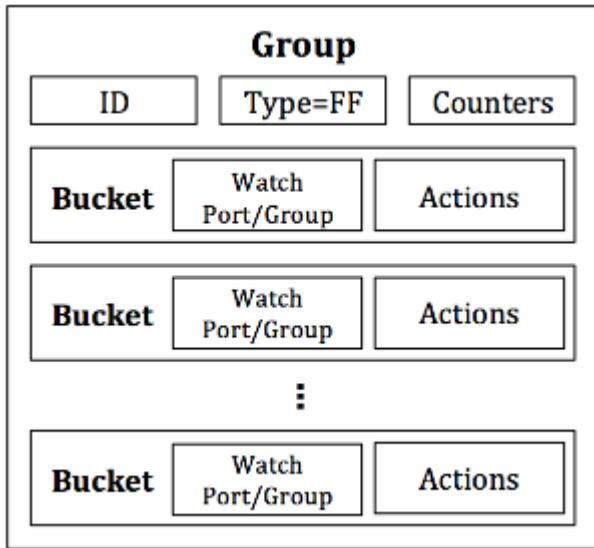
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#136 normal permanent
recirc_id=0,tcp,in_port=11,dl_src=22:11:11:11:11:22,dl_dst=22:00:00:00:00:00,nw_src=1.1.2.2
00,nw_dst=2.2.2.100,actions:set(eth(src=00:00:00:11:11:11,dst=22:22:22:00:00:00)),14
#135 normal permanent
recirc_id=0,tcp,in_port=11,dl_src=22:11:11:11:11:11,dl_dst=22:00:00:00:00:00,nw_src=1.1.2.1
00,nw_dst=2.2.2.100,actions:set(eth(src=00:00:00:11:11:11,dst=22:22:22:00:00:00)),14
  
```

The FAST-FAILOVER Group

Lastly, the **FAST-FAILOVER group** is the topic of conversation for this tutorial and is designed specifically to detect and overcome port failures. Like the SELECT and ALL groups, the FAST-FAILOVER group, as indicated in **Figure 5**, has a list of buckets. In addition to this list of actions, each bucket has a **watch port** and/or **watch group** as a special parameter. The watch port/group will monitor the “liveness” or up/down status of the indicated port/group. If the status is deemed to be down, then the bucket will not be used. If it is determined to be up, then the bucket can be used. Only one bucket can be used at a time, and the bucket in use will not be changed unless the status of the watch port/group transitions from up to down. When such an event occurs, the FAST-FAILOVER group will quickly select the next bucket in the bucket list with a watch port/group that is up.

Actually, watch group is not supported in PicOS.

There is no guarantee on the transition time to select a new bucket when a failure occurs. The transition time is dependent on search time to find a watch port/group that is up and on the switch implementation. However, the motivation behind using a FAST-FAILOVER group is that it is almost guaranteed to be quicker than consulting the control plane to handle the port down event, and inserting a new flow or set of flows. With FAST-FAILOVER groups, link failure detection and recovery takes place entirely in the data plane.



```

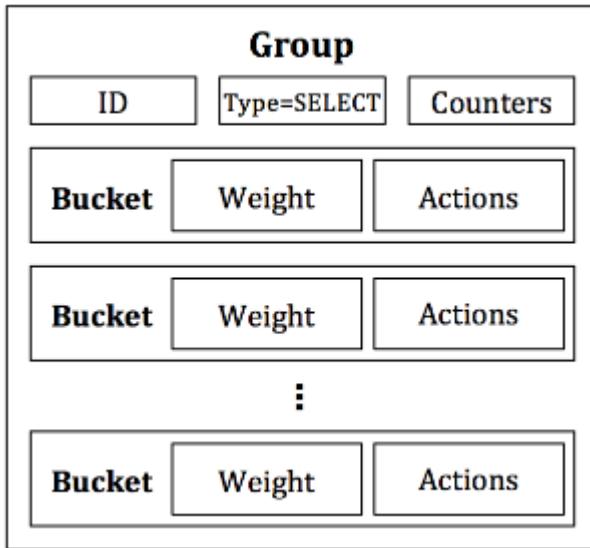
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=fast_failover,bucket=watch_port:1,output:1

admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=2,type=fast_failover,bucket=watch_port:2,output:2
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=3,type=fast_failover,bucket=watch_port:3,watch_group:1,output:5,bucket=watch_port:
4,watch_group:2,output:6

admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=11,dl_src=22:11:11:11:11,dl_dst=22:00:00:00:00:00,dl_type=0x0800,nw_proto=6,nw_s
rc=1.1.2.100,nw_dst=2.2.2.100,actions=group:3
  
```

The SELECT Group

Next, there is the **SELECT group**, which is primarily designed for load balancing. As indicated in the following figure, each bucket in a SELECT group has an assigned weight, and each packet that enters the group is sent to a single bucket. The bucket selection algorithm is undefined and is dependent on the switch's implementation; however, weighted round robin is perhaps the most obvious and simplest choice of packet distribution to buckets. The weight of a bucket is provided as a special parameter to each bucket. Each bucket in a SELECT group is still a list of actions, so any actions supported by OpenFlow can be used in each bucket, and the buckets need not be uniform.



⚠ PicOS does **not** support weight. Because OVS forward is based on priority of entries in TCAM, the traditional ECMP in routing table cannot be used. For a typical IP flow, PicOS implements a "dummy ECMP" by splitting the matching fields of a flow. For a group-table with type=select, PicOS support set-select-group-hash-fields and the packets will hash according to the value of set-select-group-hash-fields.

The command added for setting select-group hash fields is:

```
ovs-vsctl set-select-group-hash-fields [FIELDS]
FIELDS: default or some of [nw_src, nw_dst, dl_src, dl_dst] spliced by ",".
```

In the default mode the order is "nw_src, nw_dst, dl_src, dl_dst", so will check the first filed "nw_src" and find all mask bits of nw_src in flow entry are "1", in order will check the second filed "nw_dst" find the some mask bits are not "1", then the hardware flow entry will hash by "nw_dst".

Description: This command takes at most 1 argument that specifies match fields to do hash for select-group, these match fields should be spliced by “,” with descending priority and must use the constrained fields list above. If there is 0 argument token or 1 argument with string “default”, the default mode will take effect.

The priority for match fields means that when match fields are set, they will be checked in sequence to select one can do hash. If select fails, the first field in the configured match fields will be picked on and only one bucket can be used.

In default mode, the check order is “nw_src, nw_dst, dl_src, dl_dst” with non-zero field mask. If select failed, check the order “nw_src, nw_dst, dl_src, dl_dst” with field mask equals zero again.

If the selected field is an exact value, means all mask bits are “1”, then only one bucket can be used.



*When user want the traffic match nw_src or nw_dst to hash, the flow entry match field must include dl_type=0x0800. If not include dl_type=0x0800, the flow entry only hash by dl_src and dl_dst.
ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,actions=output:2*

Example

```

1. If the match field does not include dl_type=0x0800.

set-select-group-hash-fields: dl_src,dl_dst
admin@PicOS-OVS$ovs-vsctl set-select-group-hash-fields dl_src,dl_dst
admin@PicOS-OVS$ovs-vsctl show-select-group-hash-fields
select_group_hash_fields: dl_src,dl_dst

1> There is no dl_src or dl_dst in flow match field, use dl_src for hashing,
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:14,bucket=output:25,bucket=output:38
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=group:1
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#4 normal permanent recirc_id=0,in_port=1,dl_src=00:00:00:00:01/00:00:00:00:00:03,
actions:25
#5 normal permanent recirc_id=0,in_port=1,dl_src=00:00:00:00:00:00/00:00:00:00:00:03,
actions:14
#2 normal permanent recirc_id=0,in_port=1,dl_src=00:00:00:00:00:03/00:00:00:00:00:03,
actions:14
#3 normal permanent recirc_id=0,in_port=1,dl_src=00:00:00:00:02/00:00:00:00:00:03,
actions:38

2> dl_src has hashable mask in flow match field, use dl_src for hashing,
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:14,bucket=output:25,bucket=output:38
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_src=22:11:11:11:11:ff:ff:ff:00:00:00,actions=group:1
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#9 normal permanent recirc_id=0,in_port=1,dl_src=22:11:11:00:00:00/ff:ff:ff:00:00:03,
actions:14
#8 normal permanent recirc_id=0,in_port=1,dl_src=22:11:11:00:00:01/ff:ff:ff:00:00:03,
actions:25
#7 normal permanent recirc_id=0,in_port=1,dl_src=22:11:11:00:00:02/ff:ff:ff:00:00:03,
actions:38
#6 normal permanent recirc_id=0,in_port=1,dl_src=22:11:11:00:00:03/ff:ff:ff:00:00:03,
actions:14

3> dl_src has no hashable mask and there is no dl_dst in flow match field, use dl_dst for
hashing,
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:14,bucket=output:25,bucket=output:38
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,dl_src=22:11:11:11:11,actions=group:1
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#13 normal permanent
recirc_id=0,in_port=1,dl_src=22:11:11:11:11,dl_dst=00:00:00:00:00:00/00:00:00:00:00:03,
actions:14
#12 normal permanent
recirc_id=0,in_port=1,dl_src=22:11:11:11:11,dl_dst=00:00:00:00:00:01/00:00:00:00:00:03,
actions:25
#11 normal permanent
recirc_id=0,in_port=1,dl_src=22:11:11:11:11,dl_dst=00:00:00:00:02/00:00:00:00:00:03,
actions:38
#10 normal permanent
recirc_id=0,in_port=1,dl_src=22:11:11:11:11,dl_dst=00:00:00:00:03/00:00:00:00:00:03,
actions:14

4> There is no dl_src, and dl_dst has hashable mask in flow match field, use dl_src for
hashing,
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:14,bucket=output:25,bucket=output:38
admin@PicOS-OVS$ovs-ofctl add-flow br0

```

```

in_port=1,dl_dst=22:22:22:22:22:22/ff:ff:ff:00:00:00,actions=group:1
admin@PicOS-OVS$ 
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#83 normal permanent
recirc_id=0,in_port=1,dl_src=00:00:00:00:00:01/00:00:00:00:00:03,dl_dst=22:22:22:00:00:00/f
f:ff:ff:00:00:00, actions:25
#81 normal permanent
recirc_id=0,in_port=1,dl_src=00:00:00:00:00:03/00:00:00:00:00:03,dl_dst=22:22:22:00:00:f
f:ff:ff:00:00:00, actions:14
#84 normal permanent
recirc_id=0,in_port=1,dl_src=00:00:00:00:00:00/00:00:00:00:00:03,dl_dst=22:22:22:00:00:f
f:ff:ff:00:00:00, actions:14
#82 normal permanent
recirc_id=0,in_port=1,dl_src=00:00:00:00:00:02/00:00:00:00:00:03,dl_dst=22:22:22:00:00:f
f:ff:ff:00:00:00, actions:38

5> dl_src and dl_dst have no hashable mask in flow match field, won't hash,
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:14,bucket=output:25,bucket=output:38
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_src=22:11:11:11:11:11,dl_dst=22:22:22:22:22,actions=group:1
admin@PicOS-OVS$ 
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#18 normal permanent
recirc_id=0,in_port=1,dl_src=22:11:11:11:11:11,dl_dst=22:22:22:22:22:22, actions:14

```

2. If the match field include dl_type=0x0800.

```

set-select-group-hash-fields: nw_dst,nw_src,
admin@PicOS-OVS$ovs-vsctl set-select-group-hash-fields nw_dst,nw_src
admin@PicOS-OVS$ovs-vsctl show-select-group-hash-fields
select_group_hash_fields: nw_dst,nw_src

1> There is no nw_dst or nw_src in flow match field, use nw_dst for hashing,
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:14,bucket=output:25,bucket=output:38
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,actions=group:1
admin@PicOS-OVS$ 
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#28 normal permanent recirc_id=0,ip,in_port=1,nw_dst=0.0.0.2/0.0.0.3, actions:38
#30 normal permanent recirc_id=0,ip,in_port=1,nw_dst=0.0.0.0/0.0.0.3, actions:14
#29 normal permanent recirc_id=0,ip,in_port=1,nw_dst=0.0.0.1/0.0.0.3, actions:25
#27 normal permanent recirc_id=0,ip,in_port=1,nw_dst=0.0.0.3/0.0.0.3, actions:14
2> nw_dst has hashable mask, use nw_dst for hashing,
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:14,bucket=output:25,bucket=output:38
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,nw_dst=192.168.2.0/24,actions=group:1
admin@PicOS-OVS$ 
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#35 normal permanent recirc_id=0,ip,in_port=1,nw_dst=192.168.2.3/255.255.255.3, actions:14
#37 normal permanent recirc_id=0,ip,in_port=1,nw_dst=192.168.2.1/255.255.255.3, actions:25
#38 normal permanent recirc_id=0,ip,in_port=1,nw_dst=192.168.2.0/255.255.255.3, actions:14
#36 normal permanent recirc_id=0,ip,in_port=1,nw_dst=192.168.2.2/255.255.255.3, actions:38

3> nw_dst has no hashable mask and there is no nw_src in match field, use nw_src for
hashing,
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:14,bucket=output:25,bucket=output:38
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,nw_dst=192.168.2.100,actions=group:1
admin@PicOS-OVS$ 
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#40 normal permanent recirc_id=0,ip,in_port=1,nw_src=0.0.0.2/0.0.0.3,nw_dst=192.168.2.100,
actions:38
#42 normal permanent recirc_id=0,ip,in_port=1,nw_src=0.0.0.0/0.0.0.3,nw_dst=192.168.2.100,
actions:14
#41 normal permanent recirc_id=0,ip,in_port=1,nw_src=0.0.0.1/0.0.0.3,nw_dst=192.168.2.100,
actions:25

```

```
#39 normal permanent recirc_id=0,ip,in_port=1,nw_src=0.0.0.3/0.0.0.3,nw_dst=192.168.2.100,
actions:14

4> There is no nw_dst, and nw_src has hashable mask, use nw_dst for hashing,
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:14,bucket=output:25,bucket=output:38
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,nw_src=192.168.1.0/24,actions=group:1
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#46 normal permanent recirc_id=0,ip,in_port=1,nw_src=192.168.1.0/24,nw_dst=0.0.0.1/0.0.0.3,
actions:25
#44 normal permanent recirc_id=0,ip,in_port=1,nw_src=192.168.1.0/24,nw_dst=0.0.0.3/0.0.0.3,
actions:14
#47 normal permanent recirc_id=0,ip,in_port=1,nw_src=192.168.1.0/24,nw_dst=0.0.0.0/0.0.0.3,
actions:14
#45 normal permanent recirc_id=0,ip,in_port=1,nw_src=192.168.1.0/24,nw_dst=0.0.0.2/0.0.0.3,
actions:38

5> nw_src and nw_dst have no hashable mask in match field, won't hash,
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:14,bucket=output:25,bucket=output:38
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,nw_src=192.168.1.100,nw_dst=192.168.2.100,actions=group:1
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#48 normal permanent recirc_id=0,ip,in_port=1,nw_src=192.168.1.100,nw_dst=192.168.2.100,
actions:14
```

3. set-select-group-hash-fields: nw_src,dl_src

```
admin@PicOS-OVS$ovs-vsctl set-select-group-hash-fields nw_src,dl_src
admin@PicOS-OVS$ovs-vsctl show-select-group-hash-fields
select_group_hash_fields: nw_src,dl_src

admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:12,bucket=output:13,bucket=output:14
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=11,ip,nw_src=10.10.10.10,nw_dst=10.10.20.0/24,dl_src=00:00:00:11:11:11/ff:ff:ff:ff:ff:ff
ff:00,dl_dst=00:00:00:22:22:22,actions=group:1
```

The order is "nw_src, dl_src", so will check the first filed "nw_src" and find all mask bits of nw_src in flow entry are "1", in order will check the second filed "dl_src" find the some mask bits are not "1", then the hardware flow entry will hash by "dl_src".

```
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#29 normal permanent
recirc_id=0,ip,in_port=11,dl_src=00:00:00:11:11:00/ff:ff:ff:ff:ff:03,dl_dst=00:00:00:22:22:
22,nw_src=10.10.10.10,nw_dst=10.10.20.0/24, actions:12
#27 normal permanent
recirc_id=0,ip,in_port=11,dl_src=00:00:00:11:11:02/ff:ff:ff:ff:ff:03,dl_dst=00:00:00:22:22:
22,nw_src=10.10.10.10,nw_dst=10.10.20.0/24, actions:14
#28 normal permanent
recirc_id=0,ip,in_port=11,dl_src=00:00:00:11:11:01/ff:ff:ff:ff:ff:03,dl_dst=00:00:00:22:22:
22,nw_src=10.10.10.10,nw_dst=10.10.20.0/24, actions:13
#26 normal permanent
recirc_id=0,ip,in_port=11,dl_src=00:00:00:11:11:03/ff:ff:ff:ff:ff:03,dl_dst=00:00:00:22:22:
22,nw_src=10.10.10.10,nw_dst=10.10.20.0/24, actions:12
```

4. set-select-group-hash-fields: nw_src,dl_dst

```
admin@PicOS-OVS$ovs-vsctl set-select-group-hash-fields nw_src,dl_dst
admin@PicOS-OVS$ovs-vsctl show-select-group-hash-fields
select_group_hash_fields: nw_src,dl_dst

admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:12,bucket=output:13,bucket=output:14
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=11,ip,nw_src=10.10.10.10,nw_dst=10.10.20.0/24,dl_src=00:00:00:11:11:11/ff:ff:ff:ff:
```

```
ff:00,dl_dst=00:00:00:22:22:22,actions=group:1

The order is "nw_src, dl_dst", so will check the first filed "nw_src" and find all mask bits of nw_src in flow entry are "1", in order will check the second filed "dl_dst" find all the mask bits are "1", selected field in this time then only one bucket can be used.

admin@PicOS-OVS$ovs-appctl pica/dump-flows
#30 normal permanent
recirc_id=0,ip,in_port=11,dl_src=00:00:00:11:11:00/ff:ff:ff:ff:ff:00,dl_dst=00:00:00:22:22:22,nw_src=10.10.10.10,nw_dst=10.10.20.0/24, actions:14
```

Modify Bucket in a Group Table

The following configuration shows modification of buckets in a group table.

```
admin@PicOS-OVS# ovs-ofctl add-group br0 group_id=2238,type=all,bucket=output:3
admin@PicOS-OVS# ovs-ofctl mod-group br0
group_id=2238,type=all,bucket=output:2,bucket=output:3
admin@PicOS-OVS# ovs-ofctl mod-group br0
group_id=2238,type=all,bucket=mod_dl_src:22:11:11:22:22:22,mod_dl_dst:22:00:00:11:11:11,output:2,bucket=mod_dl_src:22:11:11:22:22:22,mod_dl_dst:22:00:00:11:11:11,output:3
```

Delete Group Table

In following configuration, users can delete the group table with following CLI.

```
admin@PicOS-OVS$ovs-ofctl del-groups br0 group_id=2238
admin@PicOS-OVS$ovs-ofctl del-groups br0
```

Display the Information of Group Table

Users can display the information of all group tables.

```
admin@PicOS-OVS$ovs-ofctl dump-groups br0
OFPST_GROUP_DESC reply (OF1.4) (xid=0x2):
  group_id=20,type=all,bucket=actions=output:49
  group_id=30,type=all,bucket=actions=output:50
  group_id=10,type=ff,bucket=watch_port:49,watch_group:20,actions=output:49,bucket=watch_port:50,watch_group:30,actions=output:50

admin@PicOS-OVS$ovs-ofctl dump-group-stats br0
OFPST_GROUP reply (OF1.4) (xid=0x2):
  group_id=20,duration=163370.539s,ref_count=0,packet_count=n/a,byte_count=0,bucket0:packet_count=n/a,byte_count=0
  group_id=30,duration=163365.129s,ref_count=0,packet_count=n/a,byte_count=0,bucket0:packet_count=n/a,byte_count=0
  group_id=10,duration=163375.513s,ref_count=0,packet_count=n/a,byte_count=13984159200,bucket0:packet_count=n/a,byte_count=8247745704,bucket1:packet_count=n/a,byte_count=5736413496

admin@PicOS-OVS$ovs-ofctl dump-group-features br0
OFPST_GROUP_FEATURES reply (OF1.4) (xid=0x2):
  Group table:
    Types: 0xf
```

```

Capabilities: 0xe
all group:
  max_groups=0x64
  actions: output group set_field strip_vlan push_vlan mod_nw_ttl dec_ttl set_mpls_ttl
dec_mpls_ttl dec_mpls_ttl dec_mpls_ttl dec_mpls_ttl
  select group:
    max_groups=0x64
    actions: output group set_field strip_vlan push_vlan mod_nw_ttl dec_ttl set_mpls_ttl
dec_mpls_ttl dec_mpls_ttl dec_mpls_ttl dec_mpls_ttl
  indirect group:
    max_groups=0x64
    actions: output group set_field strip_vlan push_vlan mod_nw_ttl dec_ttl set_mpls_ttl
dec_mpls_ttl dec_mpls_ttl dec_mpls_ttl dec_mpls_ttl
  fast failover group:
    max_groups=0x64
    actions: output group set_field strip_vlan push_vlan mod_nw_ttl dec_ttl set_mpls_ttl
dec_mpls_ttl dec_mpls_ttl dec_mpls_ttl dec_mpls_ttl

```

Priority of Arp Group

Generally, the priority of an arp group is higher than a mac group. If one flow matches arp group and mac group at the same time, even if the priority of the arp is lower than the mac, packets will be forwarded according to arp group.

Example 1

Basic configuration:

```

admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1-- set interface
ge-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/4 vlan_mode=trunk tag=1-- set interface
ge-1/1/4 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/5 vlan_mode=trunk tag=1-- set interface
ge-1/1/5 type=pica8

```

Add flow:

```

admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,dl_dst=ff:ff:ff:ff:ff:ff,actions=5
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,dl_type=0x0806,arp_op=1,actions=4

```

Check table:

```

admin@PicOS-OVS$ovs-appctl pica/dump-flows
#3720 normal permanent recirc_id=0,arp,in_port=1,arp_op=1, actions:4
#3719 normal permanent recirc_id=0,in_port=1,dl_dst=ff:ff:ff:ff:ff:ff, actions:5
#3718 normal permanent priority=0,recirc_id=0, actions:drop
Total 3 flows in HW.
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
cookie=0x0, duration=28.919s, table=0, n_packets=n/a, n_bytes=0,
in_port=1,dl_dst=ff:ff:ff:ff:ff:ff actions=output:5
cookie=0x0, duration=15.884s, table=0, n_packets=n/a, n_bytes=0, arp,in_port=1,arp_op=1
actions=output:4
admin@PicOS-OVS$

```

Send arp request packets and then check table:

```
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
cookie=0x0, duration=62.999s, table=0, n_packets=n/a, n_bytes=0,
in_port=1,dl_dst=ff:ff:ff:ff:ff:ff actions=output:5
cookie=0x0, duration=49.964s, table=0, n_packets=n/a, n_bytes=2560000,
arp,in_port=1,arp_op=1 actions=output:4
```

From example above, if packets match files of above flows, packets will be forwarded according to arp.

Example 2

Add flow:

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,priority=333,dl_dst=ff:ff:ff:ff:ff:ff,actions=5
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,priority=222,dl_type=0x0806,arp_op=1,actions=4
```

Send arp request packets and check table:

```
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
cookie=0x0, duration=19.544s, table=0, n_packets=n/a, n_bytes=0,
priority=333,in_port=1,dl_dst=ff:ff:ff:ff:ff:ff actions=output:5
cookie=0x0, duration=8.422s, table=0, n_packets=n/a, n_bytes=19846144,
priority=222,arp,in_port=1,arp_op=1 actions=output:4
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
cookie=0x0, duration=20.931s, table=0, n_packets=n/a, n_bytes=0,
priority=333,in_port=1,dl_dst=ff:ff:ff:ff:ff:ff actions=output:5
cookie=0x0, duration=9.809s, table=0, n_packets=n/a, n_bytes=26813440,
priority=222,arp,in_port=1,arp_op=1 actions=output:4
admin@PicOS-OVS$
```

Even if arp group priority is lower than mac group, packets still are forwarded according to arp group.

Configuring ECMP

Command

ovs-vsctl set-max-ecmp-ports *[numbers]*

ovs-vsctl show-max-ecmp-ports

Parameters

Numbers:[2-32],max is 32 ,and this must be 2^n(n=1,2,3,4,5). Default number is 4.

Example

PicOS OVS supports ecmp (nw_src, nw_dst), the default ecmp ports is 4.

IP packets (nw_src=192.168.1.0/255.255.255.1) will forward to port 2.

IP packets (nw_src=192.168.1.1/255.255.255.1) will forward to port 3.

```
root@PicOS-OVS#ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:2,bucket=output:3
root@PicOS-OVS#ovs-ofctl add-flow br0 dl_type=0x0800,nw_src=192.168.1.0/24,actions=group:1
```

If port 2 is down, all packets will forward to port 3.

Class of Service Mapping for QoS

In PicOS-2.1, if user enables the Class of Service (CoS) mapping, the packet will be mapped to a physical queue (0-7). With DSCP (0-7), it maps to queue-0 and with DSCP (8-16), it maps to queue-1 and so on. Queue-7 has the highest priority. Enable the CoS Mapping as follows:

```
admin@PicOS-OVS$ovs-vsctl set-cos-map true
admin@PicOS-OVS$ovs-vsctl show-cos-map
cos mapping: enabled
```

Display the configuration by entering the following:

```
admin@PicOS-OVS$ovs-vsctl show-cos-map ge-1/1/1
cos mapping: enabled
{
    dscp    queue
    -----
    0 - 7:   q0
    8 - 15:  q1
    16 - 23: q2
    24 - 31: q3
    32 - 39: q4
    40 - 47: q5
    48 - 55: q6
    56 - 63: q7
}
```

To configure a flow, use the following command:

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_src=22:11:11:11:11:11,actions=set_queue:7,output=3
```

The action of "**set-queue:7**" will take the place of the default CoS mapping.

Modify the dscp value of port:

```
admin@XorPlus$ovs-vsctl set-cos-map true
admin@XorPlus$ovs-vsctl set interface ge-1/1/1 dscp_map=0=q1,1=q1,2=q2
admin@XorPlus$ovs-vsctl show-cos-map ge-1/1/1
```

```

cos mapping: enabled
{
dscp queue
-----
0: q1
1: q1
2: q2
others: q0
}
admin@XorPlus$
```

The following command is also permitted:

```

admin@XorPlus$ovs-vsctl set interface ge-1/1/2 dscp_map:0=q1 dscp_map:1=q1 dscp_map:2=q2
admin@XorPlus$
admin@XorPlus$ovs-vsctl show-cos-map ge-1/1/2
cos mapping: enabled
{
dscp queue
-----
0: q1
1: q1
2: q2
others: q0
}
admin@XorPlus$
```

Configuring QoS Queue

PicOS OVS Supports qos/queue

Flow (dl_src is 22:11:11:11:11:11) will be forwarded to queue 0 of port 3

Flow (dl_src is 22:11:11:11:11:12) will be forwarded to queue 7 of port 3.

Min and max rate of queue 0 and queue 7 is set as 10M

```

root@PicOS-OVS# ovs-ofctl del-flows br0
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=1,dl_src=22:11:11:11:11,actions=set_queue:0,output=3
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=2,dl_src=22:11:11:11:12,actions=set_queue:7,output=3
root@PicOS-OVS# ovs-vsctl -- set port ge-1/1/3 qos=@newqos -- --id=@newqos create qos
type=PRONTO_STRICT queues:0=@newqueue queues:7=@newqueue1 -- --id=@newqueue create queue
other-config:min-rate=10000000 other-config:max-rate=10000000 -- --id=@newqueue1 create
queue other-config:min-rate=10000000 other-config:max-rate=10000000
```

WRR

WRR(short for Weighted Round Robin)is a scheduler mode of Qos. It uses a round robin scheduling algorithm between the queues and can avoid the lowest priority queues not being serviced for long time when traffic congestion happens. When you use WRR scheduling mode, you can define your own weighted value for each queue to distribute different service time for queue.Generaly,greater the weight, longer the serviced time.

1.1 Create a qos and add two queues (0 and 3)to qos for port te-1/1/3

```
root@PicOS-OVS$ovs-vsctl -- set port te-1/1/3 qos=@newqos -- --id=@newqos create qos
type=PRONTO_WEIGHTED_ROUND_ROBIN queues:0=@newqueue queues:3=@newqueue1 -- --id=@newqueue1 create queue
create queue other-config:min-rate=200000000,weight=3 -- --id=@newqueue1 create queue
other-config:min-rate=200000000,weight=3
b41dbaabb-dc60-40ff-8a11-7a241bfd5a5c
f7c5e8ff-f81f-4e69-9096-9b1251f29618
5d57cffb-b9b7-4324-9402-034ab9fac050
root@PicOS-OVS$
```

1.2 Flow config

```
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_src=22:11:11:11:11:11,actions=set_queue:0,output=3
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=2,dl_src=22:11:11:11:11:12,actions=set_queue:3,output=3
root@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  flow_id=18, cookie=0x0, duration=422.520s, table=0, n_packets=n/a, n_bytes=0,
  in_port=2,dl_src=22:11:11:11:11:12 actions=set_queue:3,output:3
  flow_id=17, cookie=0x0, duration=426.136s, table=0, n_packets=n/a, n_bytes=0,
  in_port=1,dl_src=22:11:11:11:11:11 actions=set_queue:0,output:3
root@PicOS-OVS$ 
root@PicOS-OVS$ovs-appctl pica/dump-flows
#10 normal permanent flow_id=17 in_port=1,dl_src=22:11:11:11:11:11, actions:3
#6 normal_d permanent flow_id=13 priority=0, actions:drop
#8 normal permanent flow_id=18 in_port=2,dl_src=22:11:11:11:11:12,
actions:set(skb_priority(0x3)),3
Total 3 flows in HW.
root@PicOS-OVS$
```

notice: queue 0 is the default queue, the HW flow don't display it.

1.3 Result

Firstly, both the queue 0 and queue 7 should provide a minimum bandwidth guarantee (200M) during congestion, then each queue distribute the remaining packets according to the queue weight.

2 Check qos

```
root@PicOS-OVS$ovs-vsctl list qos
_uuid          : b41dbaabb-dc60-40ff-8a11-7a241bfd5a5c
external_ids    : {}
other_config    : {}
queues          : {0=f7c5e8ff-f81f-4e69-9096-9b1251f29618,
3=5d57cffb-b9b7-4324-9402-034ab9fac050}
type           : PRONTO_WEIGHTED_ROUND_ROBIN
root@PicOS-OVS$
```

3 Check one queue according to its queue-uuid.

```
root@PicOS-OVS$ovs-vsctl list queue 5d57cffb-b9b7-4324-9402-034ab9fac050
_uuid          : 5d57cffb-b9b7-4324-9402-034ab9fac050
dscp           : []
external_ids    : {}
other_config    : {min-rate="200000000", weight="3"}
root@PicOS-OVS$
```

4 Check qos of a port from hardware

```
root@PicOS-OVS$ovs-appctl qos/show te-1/1/3
QoS: te-1/1/3 PRONTO_WEIGHTED_ROUND_ROBIN
Default:
    burst: 0
    max-rate: 0
    min-rate: 200000000
    weight: 3
    tx_packets: 0
    tx_bytes: 0
    tx_errors: 0
Queue 3:
    burst: 0
    max-rate: 0
    min-rate: 200000000
    weight: 3
    tx_packets: 0
    tx_bytes: 0
    tx_errors: 0
root@PicOS-OVS$
```

5 Reset one queue according to queue-uuid

```
root@PicOS-OVS$ovs-vsctl set queue f7c5e8ff-f81f-4e69-9096-9b1251f29618
other-config=min-rate=100000000,weight=2
root@PicOS-OVS$ 
root@PicOS-OVS$ovs-appctl qos/show te-1/1/3
QoS: te-1/1/3 PRONTO_WEIGHTED_ROUND_ROBIN
Default:
    burst: 0
    max-rate: 0
    min-rate: 100000000
    weight: 2
    tx_packets: 0
    tx_bytes: 0
    tx_errors: 0
Queue 3:
    burst: 0
    max-rate: 0
    min-rate: 200000000
    weight: 3
    tx_packets: 0
    tx_bytes: 0
    tx_errors: 0
root@PicOS-OVS$
```

6 Remove one queue from qos according qos-uuid and queue-key.

```
root@PicOS-OVS$ovs-vsctl remove qos b41dbaab-dc60-40ff-8a11-7a241bfd5a5c queue 3
root@PicOS-OVS$ 
root@PicOS-OVS$ovs-appctl qos/show te-1/1/3
QoS: te-1/1/3 PRONTO_WEIGHTED_ROUND_ROBIN
Default:
    burst: 0
    max-rate: 0
    min-rate: 100000000
    weight: 2
    tx_packets: 0
    tx_bytes: 0
    tx_errors: 0
root@PicOS-OVS$ovs-vsctl destroy queue 5d57cffb-b9b7-4324-9402-034ab9fac050
```

```
root@PicOS-OVS$ ovs-vsctl list queue
_uuid : f7c5e8ff-f81f-4e69-9096-9b1251f29618
_dscp : []
_external_ids : {}
_other_config : {min-rate="100000000", weight="2"}
```

7 Add one new queue to qos according qos-uuid

```
root@PicOS-OVS$ ovs-vsctl set qos b41dbaab-dc60-40ff-8a11-7a241bfd5a5c queues:1=@newqueue1 -- --id=@newqueue1 create queue other-config:weight=3
02273a2b-f556-454f-959f-96290dac7642
root@PicOS-OVS$
root@PicOS-OVS$ ovs-vsctl list qos
_uuid : b41dbaab-dc60-40ff-8a11-7a241bfd5a5c
_external_ids : {}
_other_config : {}
_queues : {0=f7c5e8ff-f81f-4e69-9096-9b1251f29618,
1=02273a2b-f556-454f-959f-96290dac7642}
_type : PRONTO_WEIGHTED_ROUND_ROBIN
```

8 Remove all queues from qos

```
root@PicOS-OVS$ ovs-vsctl clear qos b41dbaab-dc60-40ff-8a11-7a241bfd5a5c queues
root@PicOS-OVS$ ovs-vsctl --all destroy queue
root@PicOS-OVS$
root@PicOS-OVS$ ovs-vsctl list queue
root@PicOS-OVS$
root@PicOS-OVS$ ovs-appctl qos/show te-1/1/3
QoS: te-1/1/3 PRONTO_STRICT
root@PicOS-OVS$
```

9 Remove qos

```
root@PicOS-OVS$ ovs-vsctl clear port te-1/1/3 qos
root@PicOS-OVS$ ovs-vsctl --all destroy qos
root@PicOS-OVS$
```

WRED

WRED(short for Weighted Random Early Detection) is a congestion avoidance mechanism which makes use of the congestion control mechanism of TCP (Transmission Control Protocol). When an output queue begins to experience congestion, WRED starts dropping packets *selectively*. By dropping some packets earlier than the point when the queue is full, WRED prevents the situation where a large number of packets get dropped at once. The principle specifies as follows:

- (1) The length of the queue(measured with kbps or pkts) is calculated.
- (2) If the queue length is less than the minimum threshold, the packet is placed in the queue.
- (3) If the queue length is more than the maximum threshold, the packet is dropped.

(4) If the queue length is more than the minimum threshold but less than the maximum threshold, the packet is either dropped or queued, based on the packet drop probability. The greater the chance of the packet is dropped when the drop_probability value of its corresponding queue is greater. Usually, the drop_probability value of the lower priority queues with less importance are the greatest.

1 Create two wred_queues for interface te-1/1/35

```
root@PicOS-OVS$ovs-vsctl set interface te-1/1/35 wred_queues:0=@wred1 wred_queues:3=@wred2
-- --id=@wred1 create wred_queue enable=true min_thresh=100 max_thresh=200
drop_probability=50 -- --id=@wred2 create wred_queue enable=true min_thresh=200
max_thresh=400 drop_probability=40
52f26264-3656-4840-8f0e-fb7be8e52ef8
9b36b9dc-903a-4bb6-8e25-f6e2b1fd2b4d
root@PicOS-OVS$ovs-ofctl add-flow br0 in_port=34,dl_src=22:11:11:11:11:11,actions=set_queue:3,output=35
```

Notice: The drop_probability is according to the percentage to drop packets.

2 Check wred queues

```
Check all wred_queue:
root@PicOS-OVS$ovs-vsctl list wred_queue
_uuid : 52f26264-3656-4840-8f0e-fb7be8e52ef8
drop_probability : 50
enable : true
max_thresh : 200
min_thresh : 100
_uuid : 9b36b9dc-903a-4bb6-8e25-f6e2b1fd2b4d
drop_probability : 40
enable : true
max_thresh : 400
min_thresh : 200
root@PicOS-OVS$

Check one wred_queue:
root@PicOS-OVS$ovs-vsctl list wred_queue 9b36b9dc-903a-4bb6-8e25-f6e2b1fd2b4d
_uuid : 9b36b9dc-903a-4bb6-8e25-f6e2b1fd2b4d
drop_probability : 40
enable : true
max_thresh : 400
min_thresh : 200
root@PicOS-OVS$

Check interface wred_queue:
root@PicOS-OVS$ovs-vsctl list interface te-1/1/35
_uuid : db795f1d-71e9-448c-9ac4-f2304c4bf74d
.....
.....
status : {}
type : "pica8"
wred_queues : {0=f37f30cf-8470-4edd-925f-5c2abcd061eb,
3=b97fa496-57b2-40f1-a525-b978abc64403}
root@PicOS-OVS$
```

3 Reset one wred queue according to its uuid

```
root@PicOS-OVS$ovs-vsctl set wred_queue 9b36b9dc-903a-4bb6-8e25-f6e2b1fd2b4d max_thresh=300
root@PicOS-OVS$ovs-vsctl list wred_queue 9b36b9dc-903a-4bb6-8e25-f6e2b1fd2b4d
```

```

_uuid : 9b36b9dc-903a-4bb6-8e25-f6e2b1fd2b4d
drop_probability : 40
enable : true
max_thresh : 300
min_thresh : 200
root@PicOS-OVS$
```

4 Add one wred queue to wred_queues

```

root@PicOS-OVS$ovs-vsctl set interface te-1/1/35 wred_queue:1=@wrednew -- --id=@wrednew
create wred_queue enable=true min_thresh=100 max_thresh=500 drop_probability=40
d60801bd-827b-45b7-aa5d-1a2db9f1e513
root@PicOS-OVS$
```

5 Remove one or all wred queue

```

Remove one wred queue:
root@PicOS-OVS$ovs-vsctl remove interface te-1/1/35 wred_queues 3
root@PicOS-OVS$  

root@PicOS-OVS$ovs-vsctl list wred_queue
_uuid : 52f26264-3656-4840-8f0e-fb7be8e52ef8
drop_probability : 50
enable : true
max_thresh : 200
min_thresh : 100
_uuid : d60801bd-827b-45b7-aa5d-1a2db9f1e513
drop_probability : 40
enable : true
max_thresh : 500
min_thresh : 100
root@PicOS-OVS$  
  

Delete all wred queues:
root@PicOS-OVS$ovs-vsctl clear interface te-1/1/35 wred_queues
root@PicOS-OVS$  

root@PicOS-OVS$ovs-vsctl list wred_queue
root@PicOS-OVS$
```

Result

1. In the current version, OVS is not support WRED.
2. In the PicOS switch, there are 7 queues in ASIC with priority 0~7 (In the current version, P3922/P3780/3920, only support queues 0~3). Queue 7 has the highest priority and queue 0 has the lowest priority. When user configure a flow that will be forwarded in queue 3 (set_queue:3), all packets matching this flow will be forwarded in physical queue 3. “PRONTO_STRICT” means the scheduler is based on Strict priority between queues. In other words, if the high priority queue has packets, the scheduler will never send packets from the low priority queues.

Configuring OpenFlow Meter

PicOS OVS supports meter in Openflow 1.3

Create meter

In the following configuration, users can create a meter. The valid meter ID is from 1 to 100. Define the meter before using it.

type=drop,without burst_size

30M bits per second will be forward to port 2.

```
root@PicOS-OVS# ovs-ofctl add-meter br0 meter=2,rate=30000
root@PicOS-OVS# ovs-ofctl add-flow br0 in_port=1,actions=meter:2,output:2
```

type=drop,with burst_size

30M bits per second will be forward to port 2.

```
root@PicOS-OVS# ovs-ofctl add-meter br0
meter=2,rate=30000,burst_size=30000
root@PicOS-OVS# ovs-ofctl add-flow br0 in_port=1,actions=meter:2,output:2
```

type=dscp_remark,without burst_size

30M bits per second dscp value is changed as 14.

```
root@PicOS-OVS# ovs-ofctl add-meter br0
meter=2,rate=30000,prec_level=14
```

type=dscp_remark,with burst_size

30M bits per second dscp value is changed as 14.

```
root@PicOS-OVS# ovs-ofctl add-meter br0
meter=2,rate=30000,prec_level=14,burst_size=30000
```

Modify meter

In following configuration, users can modify the meter

```
root@PicOS-OVS# ovs-ofctl mod-meter br0
meter=2,rate=30000,prec_level=12
root@PicOS-OVS# ovs-ofctl mod-meter br0
meter=2,rate=10000,burst_size=30000
```

Delete meter

In following configuration, the user deletes the meter

```
root@PicOS-OVS# ovs-ofctl del-meters br0
root@PicOS-OVS# ovs-ofctl del-meter br0 meter=1
```

Display the information of meter

The user can display the information of all meters

```
root@PicOS-OVS# ovs-ofctl meter-features br0
root@PicOS-OVS# ovs-ofctl dump-meters br0
root@PicOS-OVS# ovs-ofctl meter-stats br0
```

Configuring QinQ

PicOS OVS supports QinQ. (3290,3295,3296 do not support set inner pcp)

Push tag, Push <tag:2000>

```
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,actions=push_vlan:0x8100,set_field:2000->vlan_vid,output:2
```

Push <tag:2000 pcp:3>

```
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,actions=push_vlan:0x8100,set_field:2000->vlan_vid,set_field:3->vlan_pcp,output:2
```

Push <tag:3000 tag:4094>

```
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,actions=push_vlan:0x8100,set_field:3000->vlan_vid,push_vlan:0x8100,set_field:4094->vlan_vid,output:2
```

Push <tag:3000 tag:4094 pcp:3>

```
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,actions=push_vlan:0x8100,set_field:3000->vlan_vid,push_vlan:0x8100,set_field:4094->vlan_vid,set_field:3->vlan_pcp,output:2
```

Push <tag:3000 pcp:3 tag:4094 pcp:7>

```
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,actions=push_vlan:0x8100,set_field:3000->vlan_vid,set_field:3->vlan_pcp,push_vlan:0x8100,set_field:4094->vlan_vid,set_field:7->vlan_pcp,output:2
```

Pop tag, Pop one header

```
root@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=pop_vlan,output:2
```

Pop two header

```
root@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=pop_vlan,pop_vlan,output:2
```

User can also use the strip_vlan to achieve pop VLAN tagged, for example:

```
root@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,priority=100,actions=strip_vlan,output:2
```

In hardware ASIC, implementation of "strip_vlan" is: change the packet's tag to "4095" and strip the vlan tag of 4095 in the egress. Thus, the above flow will be split into two flows in ingress and egress respectively, as follows:

Ingress "in_port=1, priority=100, action=set_field:2000->vlan_vid"

Egress "in_port=1, priority=100,action=strip_vlan,output:2"

In this case, maybe other traffic which matches the egress flow will be stripped vlan and forwarded to port-3. Users can install the other flow with higher priority to avoid this problem.

i Hardware limitation of Pushing Two Tags

There is a limitation in pushing two tags; hardware ASIC can only identify two tags.

- a. If primary packet is untagged, do two push_vlan (add tagA, tagB), output packets is tagged with two vlans. (tagA, tagB)
- b. If primary packet is tagged with one vlan (tag0), do two push_vlan (add tagA, tagB), output packets is tagged with two vlans. (tag0, tagB),
- c. If primary packet is tagged with more than one vlan (outer vlan is tag0), do two push_vlan (add tagA, tagB), output packets is tagged with vlans. (tag0 and other tag of primary packets, tagB),
- d. For Platform 3290, 3295, 3296, 3297, 4804, these don't support modifying inner vlan pcp.

Configuring OpenFlow Provider Backbone Bridge

push

Push pbb_isid,eth_src,eth_dst

Outer src mac is set as 00:00:00:11:11:11, and dsc mac is set as 00:00:00:22:22:22, Vlan is set as 4094, pbb isid is set as 23.

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=11,dl_type=0x800,dl_src=22:11:11:11:11:11,dl_dst=22:22:22:22:22:22,actions=push_pb
b:0x88e7,set_field:23->pbb_isid,set_field:00:00:00:11:11:11->eth_src,set_field:00:00:00:2
2:22:22->eth_dst,push_vlan:0x8100,set_field:4094->vlan_vid,output:12
```

Push pbb without pbb_isid,eth_src,eth_dst

Outer src mac is set as 22:11:11:11:11:11, and dsc mac is set as 22:22:22:22:22:22, Vlan is set as 4094, pbb isid is set as 0.

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=11,dl_type=0x800,dl_src=22:11:11:11:11:11,dl_dst=22:22:22:22:22:22,actions=push_pb
b:0x88e7,push_vlan:0x8100,set_field:4094->vlan_vid,output:12
```

Push pbb_isid,eth_src,eth_dst for pbb packets

Outer src mac is set as 00:00:00:11:11:11, and dsc mac is set as 00:00:00:22:22:22, Vlan is set as 4094, pbb isid is set as 21. (isid of primary pbb packet should not be 21)

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=11,dl_type=0x88e7,actions=push_pbb:0x88e7,set_field:21->pbb_isid,set_field:00:00:0
0:11:11:11->eth_src,set_field:00:00:00:22:22:22->eth_dst,push_vlan:0x8100,set_field:4094-
\>vlan_vid,output:12
```

pop

Pop pbb packets tagged with vlan 1 (Primary pbb packets should be tagged with vlan 1) Pbb packets are popped.

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=11,dl_type=0x88e7,dl_src=00:00:00:11:11:11,dl_dst=00:00:00:22:22:22,actions=pop_vla
n,pop_pbb0x88e7,output:12
```

Pop pbb packets tagged with vlan 2000 (Primary pbb packets should be tagged with vlan 2000) Pbb packets are popped.

```
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=11,dl_type=0x88e7,pbb_isid=23,dl_vlan=2000,dl_src=00:00:00:11:11:11,dl_dst=00:00:00
:22:22:22,actions=pop_vlan,pop_pbb:0x88e7,output:12
```

Important Things to Know

- Push pbb should be done with push_vlan,
- When do push pbb, primary src mac, and dst mac will be used if no config of eth_src , eth_dst
- Do push pbb for pbb packet, primary pbb isid should be not same as the push pb isid.
- When do pop pbb, primary packets should include vlan, and actions should include pop_vlan.
- The switches support pbb:

2632XL2948_6XL3780

392039223924

393051015401

AS5712_54XAS6701_32XAS6712_32X

Configuring Loopback

Loop the traffic which into switch back to ingress.

It is possible to configure the egress interface to be the ingress interface.

PicOS supports loopback interface in hardware. By default, a packet coming into an interface cannot be sent back to the same interface via Openflow. That means the user cannot configure a flow whose output port is the "in_port". For example, the following flow will not work in hardware by default:

```
root@PicOS-OVS# ovs-ofctl add-flow br0 in_port=1,actions=in_port
```

This behavior can be changed with the following commands:

```
ovs-appctl loopback/enable true
```

This is supported starting in PicOS 2.2. It should only be used for specific traffic as it can be dangerous to send broadcast traffic back on the same port on a Layer 2 network.

Except this one, also supports another command to enable loopback. This one will replace above command.

```
root@PicOS-OVS#ovs-vsctl set-loopback-enable true
```

With the above configuration, hardware can allow the flow output port to be the same as in_port. The user can disable the loopback interface with the following command:

```
ovs-vsctl set-loopback-enable false
```

Users should know the limitation of the loopback interface in hardware. In the OpenFlow Specification, there are some actions (Flood, Group table, for example) that are for broadcasting. The packet should not be forwarded back to the **in_port** port. Be cautious using the enable loopback interface, so that the packet is not forwarded back to the in_port port.

Example1, loop the traffic back to in_port.

```
admin@PicOS-OVS$ovs-vsctl set-loopback-enable true
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=in_port
```

Example2, loop the traffic back to in_port and same time send out from others.

```
admin@PicOS-OVS$ovs-vsctl set-loopback-enable true
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=in_port,all
```

 1, The port can be physical port, lag interface.

Loop the traffic which out from switch back to switch again.

From version 2.8.1, PicOS also support enable loop in interface, and the feature is different from above.

User can configure one or more ports as loopback port. Thus, traffic can be outputed and loopback to itself. The port can be plugin module or not.

Command:

ovs-vsctl set interface <port> options:loopback=true|false

<port>: can be physical port, lag port and GRE/L2GRE/VxLAN tunnel port.

Configure port te-1/1/1 as a loopback port:

```
ovs-vsctl set interface te-1/1/1 options:loopback=true
```

Example1, Modify traffic dl_dst=22:22:22:11:11:11 and out from port14, then modify dl_dst=22:22:22:22:22:22 and out from port25.

Need configure two port as loopback port, use port2 and port3 here.

```
admin@PicOS-OVS$ovs-vsctl set interface te-1/1/2 options:loopback=true
admin@PicOS-OVS$ovs-vsctl set interface te-1/1/3 options:loopback=true
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=output:2,3
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=2,actions=set_field:22:22:22:11:11:11->eth_dst,output:14
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=3,actions=set_field:22:22:22:22:22->eth_dst,output:25
```

Example2, traffic after encap vxlan then go through L3 flow table.

Configure loopback in physical port.

```
admin@PicOS-OVS$ovs-vsctl set-l2-mode true
admin@PicOS-OVS$ovs-vsctl set-l3-mode true
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac=C8:0A:A9:04:49:1A
options:dst_mac=C8:0A:A9:9E:14:A5 options:egress_port=te-1/1/2
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-vsctl set interface te-1/1/2 options:loopback=true
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=output:4097
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=251,dl_vlan=1,dl_dst=C8:0A:A9:9E:14:A5,actions=normal
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=252,ip,nw_dst=10.10.10.2,actions=set_field:100->vlan_vid,set_field:22:22:22:00:00:11
->eth_src,set_field:22:22:22:00:00:22->eth_dst,output:3
```

Example2, traffic go through L3 flow table then discap vxlan.

Configure loopback in physical port.

```
admin@PicOS-OVS$ovs-vsctl set-l2-mode true
admin@PicOS-OVS$ovs-vsctl set-l3-mode true
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac=C8:0A:A9:04:49:1A
options:dst_mac=C8:0A:A9:9E:14:A5 options:egress_port=te-1/1/2
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-vsctl set interface te-1/1/2 options:loopback=true
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=251,dl_vlan=1,dl_dst=C8:0A:A9:04:49:1A,actions=normal
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=252,ip,nw_dst=10.10.10.1,actions=set_field:1->vlan_vid,set_field:C8:0A:A9:9E:14:A5->eth_src,set_field:C8:0A:A9:04:49:1A->eth_dst,output:2
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=4097,actions=output:1
```



1. The port will be always be "linkup" because it is a loopback port
2. Add loopback config in interface options for physical ports and lag ports at ovs mode.

3. Due to different switch's chip are different, so some switch loopback port can send out packets, and others can not.
4. This function in crossflow mode should be added by xorplus.

Optimizing TCAM Usage

1, extend-group

TCAM flow table support add arp/mpls and other normal flows. Due to ASIC limitation these flows cannot use one same slice, so we use group to devide these flows. In other words, arp and mpls flows use different groups from normal flow, we said them extend groups.

Before PicOS2.8.1, user can add these different groups flow at same time. From PicOS2.8.1, user also can add these flows at same time, but also can control disable or enable extend group for arp/mpls flows. The command as below,

ovs-vsctl disable-extend-group <TRUE|FALSE>

Once disable it, user will not allowed to add arp flow which include arp_tpa or arp_spa, and not allowed to add mpls flow include mpls_label. But support add dl_type=0x0806 and dl_type=0x8847/0x8848.

For example,

```
admin@PicOS-OVS$ovs-vsctl disable-extend-group TRUE
Please reboot for the change to take effect!
admin@PicOS-OVS$sudo service picos restart
[....] Stopping web server: lighttpd.
[....] Stopping: PicOS Open vSwitch/OpenFlowdevice ovs-pica8 left promiscuous mode
device br0 left promiscuous mode
.
[....] Stopping enhanced syslogd: rsyslogd.
[....] Starting enhanced syslogd: rsyslogd.
[....] Stopping internet superserver: xinetd.
[....] Restarting OpenBSD Secure Shell server: sshd.
[....] Starting: PicOS Open vSwitch/OpenFlow.
admin@PicOS-OVS$ovs-pica8 entered promiscuous mode
device br0 entered promiscuous mode
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-vsctl show-extend-group
extend group: disabled
admin@PicOS-OVS$
```

2, match-mode

By default, 2 TCAM entries are used to support all matching tuples for all flows even if the flow does not use all matching tuples.

PicOS allows users to configure the switch in short flow TCAM match mode to optimize the TCAM usage, in this mode, each flow will only consume 1 TCAM entry (doubling the flows capacity in the TCAM).

When this mode is enabled (with the set-match-mode command), only specific fields can be used in the priority range defined by the command.

The flows must use the exact fields described below:

mac mode: "in_port, dl_src, dl_dst, vlan_vid, dl_type"

ip mode: "in_port, nw_proto, nw_src, nw_dst, dl_type=0x0800"

arp_tpa mode: "in_port, arp_tpa, dl_type=0x0806"

ipv6_full mode: "in_port,dl_vlan,ipv6_src,ipv6_dst,nw_proto,dl_type=0x86dd"

ipv6_src mode: "in_port,dl_src,dl_dst,dl_vlan,ipv6_src,nw_proto,dl_type=0x86dd"

ipv6_dst mode: "in_port,dl_src,dl_dst,dl_vlan,ipv6_dst,nw_proto,dl_type=0x86dd"

For example, if mac mode is enabled, all the flows must only use one or more fields defined in the mac mode. If mac and ip modes are enabled, then user can configure either mac flows or ip flows based on the fields described above. However, user cannot mix the fields from mac and ip (that is, dl_src and nw_src).

Each mode is configured with a priority range. This range can only be used by the specific type of flows configured.

In the example below, all the flows between priority "10 and 1000" have to be Mac flows. All flows between 2000 and 20000 have to be IP flows and all flows between 30000 and 60000 have to be ARP flows, all flows between 50000 and 50001 have to be ipv6_full, all flows between 50002 and 50003 have to be ipv6_dst, all flows between 50004 and 60000 have to be ipv6_src.

```
ovs-vsctl set-match-mode
mac=10-1000,ip=2000-20000,arp_tpa=30000-40000,ipv6_full=50000-50001,ipv6_dst=50002-50003,ip
v6_src=50004-60000
```

User can display this configuration with the following command:

```
ovs-vsctl show-match-mode
```

User can remove this configuration with the following command:

```
ovs-vsctl set-match-mode default
```

Once the mode is reconfigured to the default mode or another mode, the current flow table is flushed and the table starts clean.

Flow configuration examples are as follow.

```
ovs-ofctl add-flow br0
priority=900,dl_src=22:11:11:11:11:11,dl_dst=22:00:00:00:00:00,actions=output:2
ovs-ofctl add-flow br0
priority=2000,nw_src=192.168.1.1,nw_dst=192.168.100.100,nw_proto=6,dl_type=0x0800,actions=o
```

```

utput:2
ovs-ofctl add-flow br0 priority=30000,arp_tpa=192.168.2.2,dl_type=0x0806,actions=output:2
ovs-ofctl add-flow br0
priority=50000,dl_vlan=1,ipv6_src=2001:2:0:0:0:0:0:0,ipv6_dst=2001:1:0:0:0:0:0:0,nw_proto=6
,dl_type=0x86dd,actions=output:2
ovs-ofctl add-flow br0
priority=50002,dl_dst=22:00:00:00:00:00,dl_src=22:11:11:11:11:11,dl_vlan=1,ipv6_dst=2001:1:
0:0:0:0:0:0,nw_proto=6,dl_type=0x86dd,actions=output:2
ovs-ofctl add-flow br0
priority=50004,dl_dst=22:00:00:00:00:00,dl_src=22:11:11:11:11:11,dl_vlan=1,ipv6_src=2001:2:
0:0:0:0:0:0,nw_proto=6,dl_type=0x86dd,actions=output:2

```

From picos2.4, ipv6_full, ipv6_src, ipv6_dst match mode are supported.

Configuring Layer 2 over GRE on Trident-2 based switches

Only switches 5101 and 5401 support Layer 2 over Generic Routing Encapsulation (L2GRE); the port number of L2GRE ranges from 5121 to 6143. GRE is an encapsulated mechanism that encapsulates packet IPs; L2GRE is an encapsulated mechanism that encapsulates the entire packet. To resolve the problem that pushes the interface PVID to the untagged packets before encapsulation by the L2GRE header, use the command **ovs-vsctl set interface <interface> type=pica8 options:access-vport=true**. Like this, the untagged packets can be encapsulated by L2GRE header with no VLAN; that is, the PVID of ingress port. And the tagged packets are encapsulated by L2GRE header, the inner VLAN is the VLAN tag of the packets that are received by ingress port. See the example below.

```

admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10 options:vlan=1
options:l2gre_key=1234 options:src_mac=C8:0A:A9:9E:49:1A options:dst_mac=C8:0A:A9:9E:14:A5
options:egress_port=qe-1/1/2

```

Description

1. br0: bridge name
2. remote_ip=10.10.61.10: the IP address of the peer L2GRE tunnel interface; this IP address will be the destination IP of the encapsulated L2GRE packets
3. local_ip=10.10.60.10: the IP address of this L2GRE tunnel interface; this IP address will be the source IP of the encapsulated L2GRE packets
4. src_mac==C8:0A:A9:9E:49:1A: the logical MAC address of the L2GRE tunnel interface; this MAC address will be the source MAC of the encapsulated L2GRE packets to next-hop
5. dst_mac=C8:0A:A9:9E:14:A5: the next-hop MAC address; this MAC address will be the destination MAC the encapsulated L2GRE packets to next-hop
6. egress_port=qe-1/1/2: the output port of the encapsulated L2GRE packets
7. l2gre_key=1234:the key value of L2GRE tunnel,different tunnel has different key.
8. vlan=1:the vlan of L2GRE tunnel.this vlan will be pop or not according to the pvid of the egress port.

Examples

push one L2GRE header

topology

Creating a L2GRE tunnel

(1) create a new bridge named br0.

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

(2) add ports to br0.

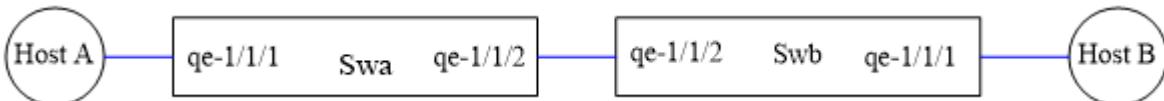
```
admin@PicOS-OVS$ovs-vsctl add-port br0 qe-1/1/1 vlan_mode=trunk tag=1 -- set Interface qe-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 qe-1/1/2 vlan_mode=trunk tag=1 -- set Interface qe-1/1/2 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10 options:vlan=1
options:l2gre_key=1234 options:src_mac=C8:0A:A9:9E:49:1A options:dst_mac=C8:0A:A9:9E:14:A5
options:egress_port=qe-1/1/2
```

User must configure a flow if user wants to send packets to a L2GRE port. And port number is 5121 for l2gre1 tunnel, different L2GRE tunnels must have different l2gre_key.

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=output:5121
```

Send packets (ARP, L2/L3 packets) to qe-1/1/1, then packets are encapsulated by L2GRE header. When the VLAN tag of Layer 2 GRE tunnel is the same with native VLAN-ID of output port, L2GRE VLAN of the packets are stripped when forwarded by egress port. When the VLAN tag of L2GRE tunnel is different from native VLAN-ID of output port, L2GRE VLAN of the packets are not stripped when forwarded by egress port.

strip L2GRE tunnel



configuration

configure the L2GRE tunnels named l2gre1 on qe-1/1/2 of swa and l2gre2 on qe-1/1/2 of swb.

swa:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set Interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10 options:vlan=2
options:l2gre_key=1234 options:src_mac=C8:0A:A9:04:49:1A options:dst_mac=C8:0A:A9:9E:14:A5
options:egress_port=qe-1/1/2
```

swb:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set Interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.60.10 options:local_ip=10.10.61.10 options:vlan=2
options:l2gre_key=1234 options:src_mac=C8:0A:A9:9E:14:A5 options:dst_mac=C8:0A:A9:04:49:1A
options:egress_port=qe-1/1/2
```

User must add the two flows below if user wants to push L2GRE header on qe-1/1/2 of swa and strip the Layer 2 GRE header on qe-1/1/2 of swb.

swa:

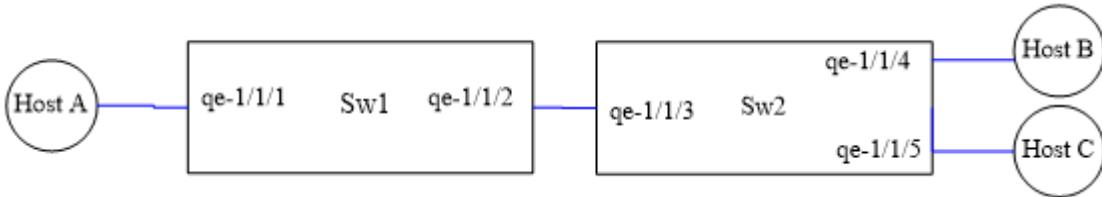
```
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=output:5121
```

swb:

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=5121,actions=output:1
```

qe-1/1/1 of swb will receive the original packets (the contents of packets are the same with packets that qe-1/1/1 of swa received).

configure two L2GRE tunnels on one physical port



Configuration

Configure two L2GRE tunnels on both qe-1/1/2(l2gre1,l2gre2) and qe-1/1/3(l2gre1,l2gre2). These two tunnels have different IP and l2gre_key so user must configure some flows.

Sw1:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set Interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.60.10 options:local_ip=10.10.61.10 options:vlan=2
options:l2gre_key=1234 options:src_mac=C8:0A:A9:9E:14:A5 options:dst_mac=C8:0A:A9:04:49:1A
options:egress_port=qe-1/1/2
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre2 -- set Interface l2gre2 type=pica8_l2gre
options:remote_ip="10.10.61.61" options:local_ip=10.10.60.60 options:vlan=10
options:l2gre_key=1235 options:src_mac=C8:0A:A9:04:49:1A options:dst_mac=88:88:88:88:88:88
options:egress_port=qe-1/1/2"
```

flows in sw1,

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_dst=22:66:66:66:66:66,actions=output:5121
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_dst=22:66:66:66:67:67,actions=output:5122
```

sw2:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set Interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10 options:vlan=2
options:l2gre_key=1234 options:src_mac=C8:0A:A9:04:49:1A options:dst_mac=C8:0A:A9:9E:14:A5
options:egress_port=qe-1/1/3
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre2 -- set Interface l2gre2 type=pica8_l2gre
options:remote_ip=10.10.60.60 options:local_ip=10.10.61.61 options:vlan=10
options:l2gre_key=1235 options:src_mac=88:88:88:88:88:88 options:dst_mac=C8:0A:A9:04:49:1A
options:egress_port=qe-1/1/3
```

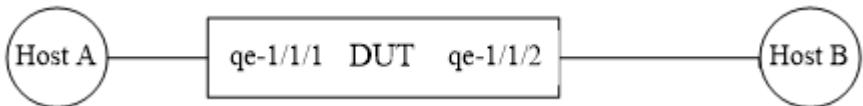
flows in sw2

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=5121,d1_dst=22:66:66:66:66:66,actions=output:4
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=5122,d1_dst=22:66:66:66:66:67,actions=output:5
```

send packets to qe-1/1/1, different packets will go to different L2GRE tunnels. When they are stripped L2GRE header on qe-1/1/3, they are forwarded to a different port.

Length of l2gre_key

In pica8 switch, the length of l2gre_key can be 16bit, 20bit, 24bit or 32bit; 24 bit is the default value.



configuration

configure the L2GRE tunnel on qe-1/1/2 .

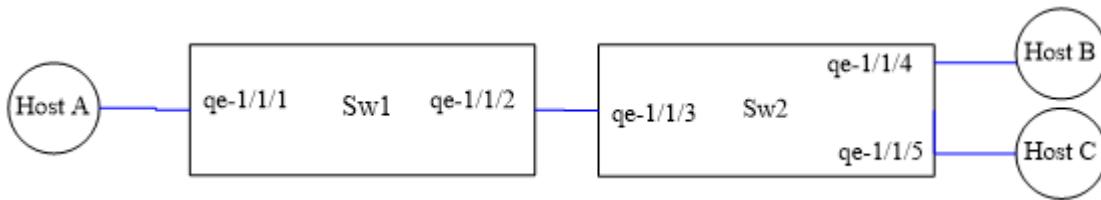
```
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set Interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10 options:vlan=1
options:l2gre_key=1234 options:src_mac=C8:0A:A9:04:49:1A options:dst_mac=C8:0A:A9:9E:14:A5
options:egress_port=qe-1/1/2
```

Add a flow to switch

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=output:5121
```

This key of L2GRE tunnel is 1234 here in decimal, 4d2 in hex. The default value of Layer 2 GRE key is 20, so the value of GRE key of packets is 0x004d2000. When user sets the l2gre_key value to 16 using the command **ovs-vsctl set-l2gre-key-length 16**, the value of the GRE key of packet is 0x04d20000. When user sets the l2gre_key value to 20 using the command **ovs-vsctl set-l2gre-key-length 24**, the value of GRE key packet is 0x0004d200. When user sets the l2gre_key value to 32 using the command **ovs-vsctl set-l2gre-key-length 32**, the value of GRE key packet is 0x000004d2.

Collaboration between nvGRE and VXLAN



Configuration

configure the L2GRE tunnel and VXLAN tunnel on qe-1/1/2 and qe-1/1/3.

```

admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set Interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10 options:vlan=1
options:l2gre_key=1234 options:src_mac=C8:0A:A9:04:49:1A options:dst_mac=C8:0A:A9:9E:14:A5
options:egress_port=qe-1/1/2
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set Interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.60.10 options:local_ip=10.10.61.10 options:vlan=1
options:l2gre_key=1234 options:src_mac=C8:0A:A9:9E:14:A5 options:dst_mac=C8:0A:A9:04:49:1A
options:egress_port=qe-1/1/3
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac=66:66:66:77:77:77
options:dst_mac=88:88:88:77:77:77 options:egress_port=qe-1/1/2
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.1 options:local_ip=10.10.10.2 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac=88:88:88:77:77:77
options:dst_mac=66:66:66:77:77:77 options:egress_port=qe-1/1/3

```

Flows in Switches

sw1:

```

admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_dst=22:22:22:22:22:23,actions=output:4097
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_dst=22:22:22:22:22:22,actions=output:5121

```

sw2:

```

admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=5122,dl_dst=22:22:22:22:22:22,actions=output:4
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=4098,dl_dst=22:22:22:22:22:23,actions=output:5

```

qe-1/1/4 should receive the de-capsulated packets with src_mac 22:22:22:22:22:22, qe-1/1/5 should receive the de-capsulated packets with src_mac 22:22:22:22:22:23. That is to say VXLAN and L2GRE do not affect each other.

Configuring VXLAN

Only switches with ASIC Trident-II (e.g. P5401 and P5101) can support VXLAN, and the port number of VXLAN ranges from 4097 to 5119. VXLAN mechanism is based on the limited number of VLANs(0-4094).VXLAN is used to provide more networks for switches or the host. To resolve the problem that pushing the interface' PVID to the untagged packets before encapsulated by the VXLAN header, the user must use this command "ovs-vsctl set interface <interface> type=pica8 options:access-vport=true ". Like this, the untagged packets can be encapsulated by VXLAN header with no VLAN that is pvid of ingress port. And the tagged packets are encapsulated by VXLAN header, the inner VLAN is the VLAN tag of packets that received by ingress port.

Command

```
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac=C8:0A:A9:04:49:1A
options:dst_mac=C8:0A:A9:9E:14:A5 options:egress_port=qe-1/1/2
```

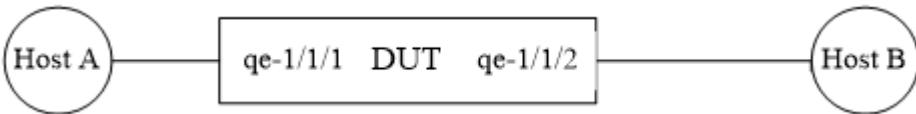
Description

1. br0: bridge name
2. remote_ip=10.10.10.2: the IP address of the peer VXLAN tunnel interface; this IP address will be the destination IP of the encapsulated VXLAN packets
3. local_ip=10.10.10.1: the IP address of this VXLAN tunnel interface; this IP address will be the source IP of the encapsulated VXLAN packets
4. src_mac==C8:0A:A9:9E:49:1A: the logical MAC address of the VXLAN tunnel interface; this MAC address will be the source MAC of the encapsulated VXLAN packets to next-hop
5. dst_mac=C8:0A:A9:9E:14:A5: the next-hop MAC address; this MAC address will be the destination MAC the encapsulated VXLAN packets to next-hop
6. egress_port=qe-1/1/2: the output port of the encapsulated VXLAN packets
7. vnid=1234:the key value of VXLAN tunnel,different tunnel has different vnid.
8. vlan=1:the vlan of VXLAN tunnel.this vlan will be pop or not according to the pvid of the egress port.
9. udp_dst_port=4789:the udp destination port of encapsulating packets by VXLAN tunnel.All the encapsulated packets has the this udp dst port.

Examples

configure a VXLAN tunnel

topology



configuration

(1)create a new bridge named br0.

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

(2)add ports to br0.

```
admin@PicOS-OVS$ovs-vsctl add-port br0 qe-1/1/1 vlan_mode=trunk tag=1 -- set Interface
qe-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 qe-1/1/2 vlan_mode=trunk tag=1 -- set Interface
qe-1/1/2 type=pica8
```

(3)add a VXLAN port named vxlan1 on qe-1/1/2

```
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac=C8:0A:A9:04:49:1A
options:dst_mac=C8:0A:A9:9E:14:A5 options:egress_port=qe-1/1/2
```

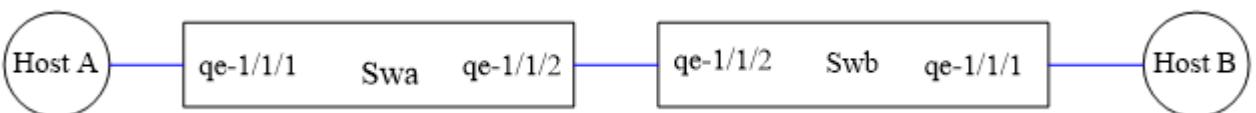
add a flow to switch

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=output:4097
```

Send packets to qe-1/1/1,qe-1/1/2 will receive packets that encapsulated by VXLAN header. When VLAN of VXLAN tunnel is the same with the pvid of qe-1/1/2,the packets from qe-1/1/2 will be stripped VLAN of VXLAN. Or, packets will have two VLANs (outer VLAN is vxlan-vlan, inner VLAN is the pvid of ingress port or original VLAN of packets)

strip a VXLAN header

topology



configuration

User must configure VXLAN port on qe-1/1/2 and qe-1/1/3, and add some flows to the switches so that packets can be encapsulated or decapsulated and forwarded correctly.

(1) create a new bridge named br0.

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

(2) add ports to br0.

SwA:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 qe-1/1/1 vlan_mode=trunk tag=1 -- set Interface
qe-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 qe-1/1/2 vlan_mode=trunk tag=1 -- set Interface
qe-1/1/2 type=pica8
```

SWb:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 qe-1/1/1 vlan_mode=trunk tag=1 -- set Interface
qe-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 qe-1/1/2 vlan_mode=trunk tag=1 -- set Interface
qe-1/1/2 type=pica8
```

(3) add VXLAN port vxlan1 on egress port qe-1/1/2 of switcha and switchb

SwA:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac=C8:0A:A9:04:49:1A
options:dst_mac=C8:0A:A9:9E:14:A5 options:egress_port=qe-1/1/2
```

flow in swa.

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,actions=output:4097
```

Swb:

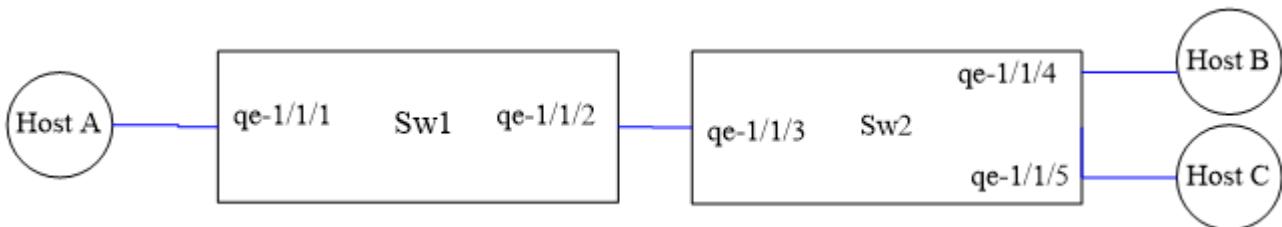
```
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.1 options:local_ip=10.10.10.2 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac= C8:0A:A9:9E:14:A5
options:dst_mac= C8:0A:A9:04:49:1A options:egress_port=qe-1/1/2
```

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=4097,actions= output:1
```

send packets to qe-1/1/1 of swa,qe-1/1/1 of switchb will receive the original packets(the contents of packets are the same with packets that qe-1/1/1 of swa received).

configure two VXLAN tunnels on a pair of physical port

topology



configuration

add two pairs of VXLAN ports on qe-1/1/2,qe-1/1/3

sw1:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac=C8:0A:A9:04:49:1A
options:dst_mac=C8:0A:A9:9E:14:A5 options:egress_port=qe-1/1/2
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan2 -- set interface vxlan2 type=pica8_vxlan
options:remote_ip=10.10.60.1 options:local_ip=10.10.60.2 options:vlan=2
options:vnid=1122869 options:udp_dst_port=4789 options:src_mac=22:22:22:04:49:1A
options:dst_mac=44:44:44:9E:14:A5 options:egress_port=qe-1/1/2
```

flows in sw1,

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_src=22:22:22:22:22:22,actions=output:4097
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_src=22:22:22:22:22:23,actions=output:4098
```

sw2:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.1 options:local_ip=10.10.10.2 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789 options:src_mac=C8:0A:A9:9E:14:A5
options:dst_mac=C8:0A:A9:04:49:1A options:egress_port=qe-1/1/3
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan2 -- set interface vxlan2 type=pica8_vxlan
options:remote_ip=10.10.60.2 options:local_ip=10.10.60.1 options:vlan=2
options:vnid=1122869 options:udp_dst_port=4789 options:src_mac=44:44:44:04:49:1A
options:dst_mac=22:22:22:9E:14:A5 options:egress_port=qe-1/1/3
```

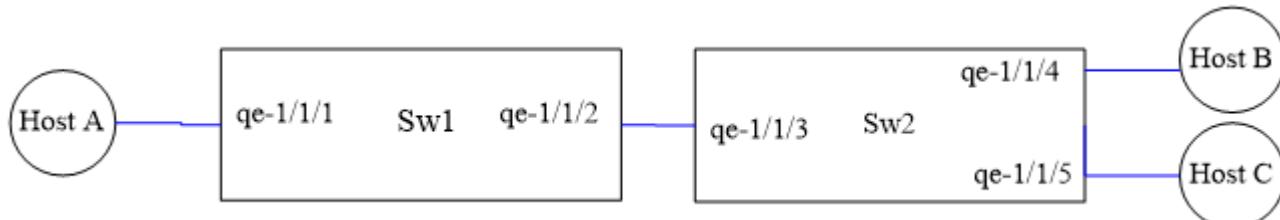
flows in sw2,

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=4097,dl_src=22:22:22:22:22:22,actions=output:4
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=4098,dl_src=22:22:22:22:22:23,actions=output:5
```

send packets to qe-1/1/1 of sw1,qe-1/1/4 should receive the packets with src_mac :22:22:22:22:22:22, and qe-1/1/5 should receive the packets with src_mac 22:22:22:22:22:23.

collaboration between L2GRE and VXLAN

topology



configuration

User must configure VXLAN port and l2gre port on qe-1/1/2 and qe-1/1/3. Add flows on both switches, so packets can be forwarded correctly.

sw1:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789  options:src_mac=C8:0A:A9:04:49:1A
options:dst_mac=C8:0A:A9:9E:14:A5  options:egress_port=qe-1/1/2
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set Interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10 options:vlan=1
options:l2gre_key=1234 options:src_mac=C8:0A:A9:22:22:22 options:dst_mac=C8:0A:A9:33:33:33
options:egress_port=qe-1/1/2
```

flows in sw1,

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_dst=22:22:22:22:22:22,actions=output:4097
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_dst=22:22:22:22:22:23,actions=output:5121
```

sw2:

```
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=pica8_vxlan
options:remote_ip=10.10.10.1 options:local_ip=10.10.10.2 options:vlan=1
options:vnid=1122867 options:udp_dst_port=4789  options:src_mac=C8:0A:A9:9E:14:A5
options:dst_mac=C8:0A:A9:04:49:1A options:egress_port=qe-1/1/3
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 -- set Interface l2gre1 type=pica8_l2gre
options:remote_ip=10.10.60.10 options:local_ip=10.10.61.10 options:vlan=1
options:l2gre_key=1234 options:src_mac=C8:0A:A9:33:33:33 options:dst_mac=C8:0A:A9:22:22:22
options:egress_port=qe-1/1/3
```

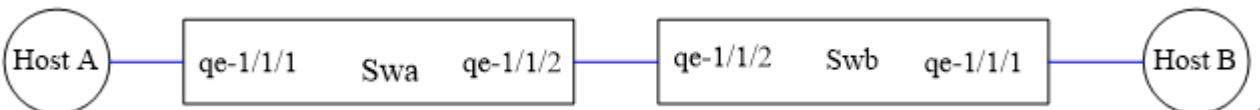
flows in sw2,

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=4097,dl_dst=22:22:22:22:22:22,actions=output:4
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=5121,dl_dst=22:22:22:22:22:23,actions=output:5
```

Vnid must be the same when the user wants to build a VXLAN tunnel between two ports. Different VXLAN tunnels must have different vnids. Besides, packets are not decapsulated when the vnid is different between the VXLAN tunnel. VXLAN can work together with GRE, L2GRE, VXLAN.

Option:

topology



Generally, untagged packets from Host A sent through Swa will be tagged by pvid in port qe-1/1/1. The new tagged packet adds VXLAN header and strip VXLAN header through VXLAN tunnel, and will keep the tag while forwarding on Swb qe-1/1/1 even though the tag equals the pvid of Swb qe-1/1/1. The result is that Host B receives a tagged packet which is different from the original packet.

To avoid the above issue, pica8 support packet keep untag through pica8 switch port. The following command is necessary.

```
ovs-vsctl set interface qe-1/1/1 options:access-vport=true
```

If the user adds the command on Swa, untagged packets that pass through Swa qe-1/1/1 will not be tagged by the pvid, then pass through VXLAN tunnel and stay untagged while forwarding to Swb qe-1/1/1. The result is Host B will receive untag packet.

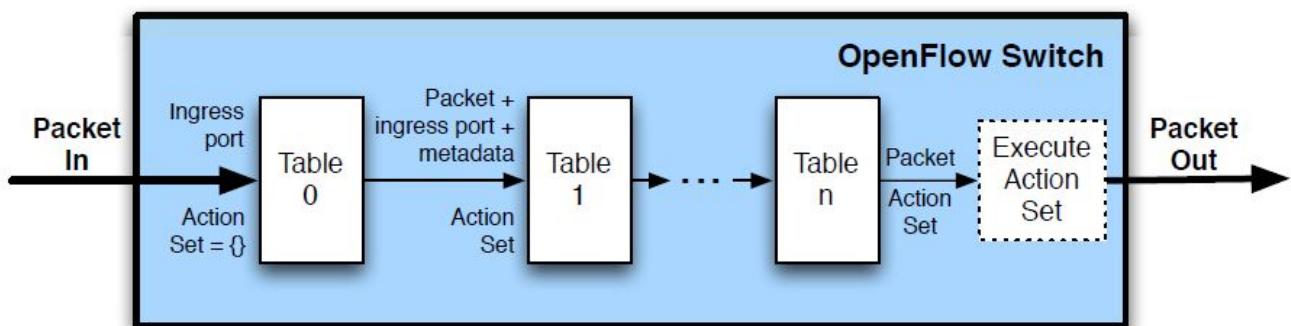
i vxlan numbers

User can create 1023 VXLAN ports at most (with the same mac, ip, vlan, only vnid is different) on one physical port. But the max flows number of VXLAN flow (with action=output:4097) is the minimum value of flows number that Vfilter table and Tcam table can support. That is 512 on P5401, P5101, AS6701, P5712,2632

Configuring Multi-Table

Hardware OpenFlow Multi-table Limitations

OpenFlow 1.1 and later versions have a concept of tables--independent lookup tables that can be chained in any way user wishes. This is a very useful concept to decrease the number of flows by segmenting those flows in multiple tables.



The implementation of those multiple tables is not difficult in a software based switch like OVS, but it is a substantial issue for a hardware based switch. This is because most ASIC's have a limited set of capacities and the ability to do multiple lookup on packets is severely limited.

The multi-table concept is very useful though to emulate an ASIC Pipeline. It allows the Openflow based solution to leverage a lot more of the Switch ASIC capacities like complex lookup or different types of memory available on the ASIC. A hardware based multi-table implementation must be limited to reflect the limitation of the underlying ASIC.

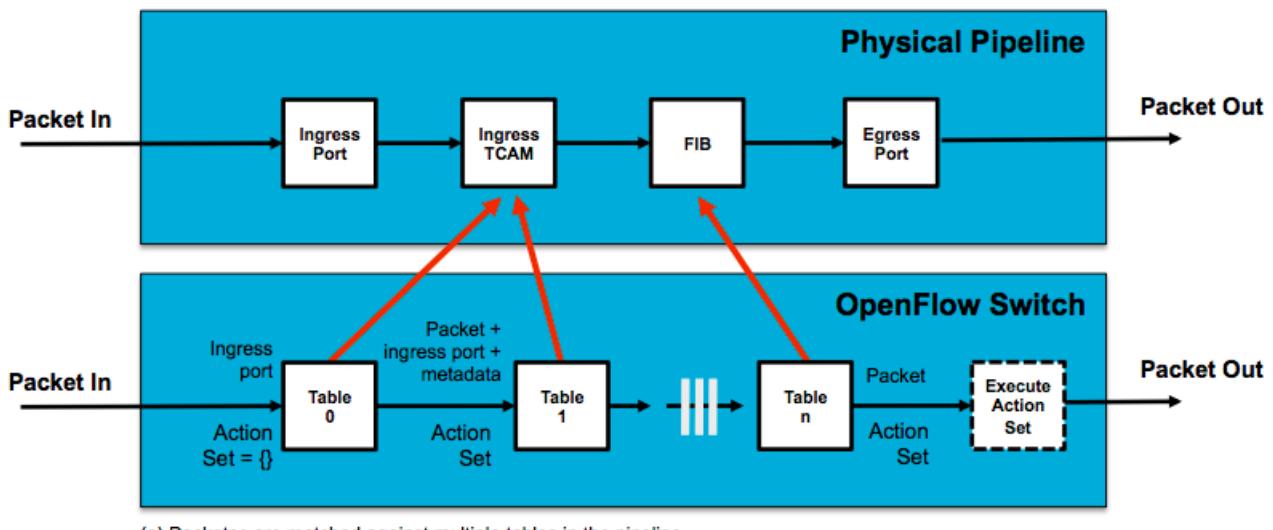
This means that the number of tables, the conflict between tables, the capacity of those tables and the link between them will be limited by the implementation. This is now defined more generically as a Table Typed Patterned by the ONF.

Multi-Tables in TCAM

Traditionally in a hardware based Openflow implementation, the flows are placed in the switch TCAM memory. This is because this memory is perfect for complex matching (can match on many parts of the packet header and the actions possible once the flow is matched are very diverse) and as such is a good match for most Openflow solutions.

In PicOS, by default, all Openflow tables are implemented in TCAM.

It is possible to create multiple tables, but because only one table is available, the OS is normalizing the flows into only one hardware table (table 0).



Because the normalizing process cannot simulate all the types of multi table logic, using this TCAM-only Multi-table implementation is typically not recommended. It is mainly used as a proof of concept or demonstration purpose. One way to be sure that the normalizing process will render the logic of the flows correctly, is to have only one of the tables with actions. All the other tables should only have drop or goto action.



PicOS 2.7.1 Note:

- If adding L2/L3 flow entry applied to one port which does not exist in the bridge, the flow cannot add correctly.
- Command of adding L3 flow modified. User must enable L2 mode first and add a system mac flow (the action is normal) to the L2 table. If user wants to use the L3 table to do the routing.
- The route flows are limited to 12000 by default.

Using the Forwarding Database instead of the TCAM

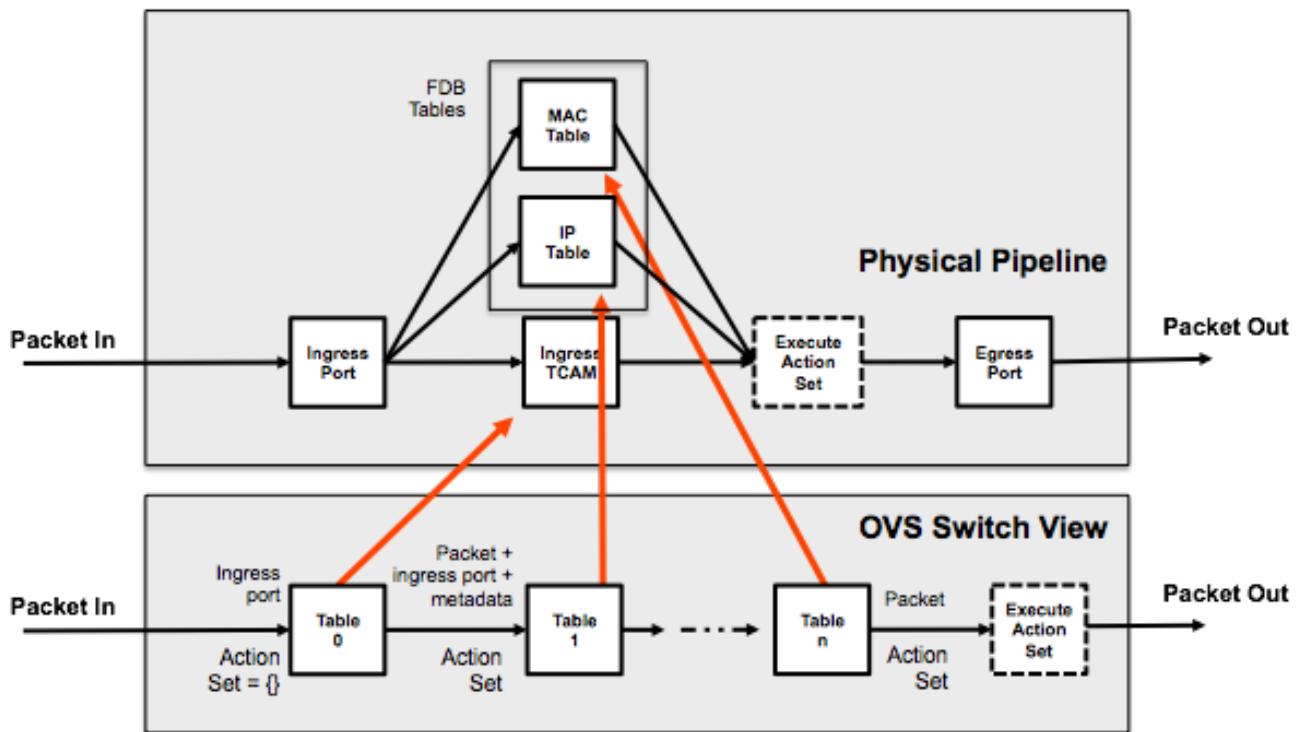
Starting in version 2.4, PicOS supports the FDB (forwarding database) table or ROUTE table like the traditional L2/L3 mode. That is to say the flows can be stored not only in TCAM table but also in FDB or ROUTE table. See the Switch Hardware Architecture for a description of the actual hardware pipeline.

This is very useful when the scaling of the solution is important and this allows the usage of more memories on the switch, as well as access to a more complex lookup.

The FDB tables consist of a MAC table (similar to a typical L2 Switch Mac lookup) and an IP Table (similar to a typical L3 Router IP lookup). User can select to download flows into the TCAM (default), the MAC table, or the IP table.

Every packet will match all those tables. Conflict between tables (different action in different tables) is managed by the table priority which can be configured.

! OpenFlow "goto" action is not supported between tables. In this hardware implementation, all tables will be used.



To Map a specific OpenFlow table to the MAC table, use the command:

set-l2-mode TRUE|FALSE [TABLE] command to enable the MAC table to store flows. **[TABLE]** is the table number user set as the FDB table. By default it is the table 251. The flow in the TCAM table should strictly match dl_dst,dl_vlan, (output port in action of flow).

To Map a specific OpenFlow table to the IP table, use the command:

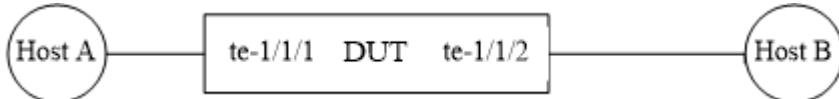
set-l3-mode TRUE|FALSE [TABLE] command to enable the ROUTE table to store flows. **[TABLE]** is the table number user set as the ROUTE table. By default, the ROUTE table number is 252. The flows to be stored in ROUTE must strictly match dl_type,nw_dst, (mod_dl_dst,mod_dl_vlan,output port in action of flow). But user should add a flow with normal action to FDB table first if user wants the L3 flow to work.

By default, TCAM matching has higher priority than L2/L3, and the priority is 0. User can use the command '**ovs-vsctl set-l2-l3-preference true**' to have the FIB/MAC table with a higher priority than the TCAM table.

By default, the ROUTE table is higher priority than the MAC table.

⚠️ It is possible to have a maximum of 3 hardware tables with flows in our current implementation simultaneously: 1 TCAM table, 1 ROUTE table and 1 MAC table.

Examples



FDB table configuration example

Step 1: Create a new bridge named br0.

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

Step 2: Add ports to br0.

```
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/1 vlan_mode=trunk tag=1 -- set Interface te-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/2 vlan_mode=trunk tag=1 -- set Interface te-1/1/2 type=pica8
```

Step 3: Set L2-mode true without table number

```
admin@PicOS-OVS$ovs-vsctl set-l2-mode true
```

Step 4: Add a flow with table = 251

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=251,dl_vlan=10,dl_dst=22:22:22:22:22:22,actions=output:2
```

Check the flows in hardware using command **ovs-appctl pica/dump-flows**. User will see that the flow is stored in L2 table. If user wants table 2 to be the FDB table, use the **ovs-vsctl set-l2-mode true 2** command.

ROUTE table configuration example

Step 1: Create a new bridge named br0.

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

Step 2: Add ports to br0.

```
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/1 vlan_mode=trunk tag=1 -- set Interface
te-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/2 vlan_mode=trunk tag=1 -- set Interface
te-1/1/2 type=pica8
```

Step 3: Set L3-mode true without table number

```
admin@PicOS-OVS$ovs-vsctl set-l3-mode true
```

Step 4: Add a flow with table = 252

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=251,dl_dst=22:22:22:22:22:22,dl_vlan=10,actions=normal
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=252,dl_type=0x0800,nw_dst=192.168.2.30,actions=set_field:44:44:44:22:22:22->dl_dst,se
t_field:40->vlan_vid,output:2
```

Check the flows in hardware using command ***ovs-appctl pica/dump-flows***. User will see that the flow is stored in L3 table. If user wants table 4 to be the FDB table, use the ***ovs-vsctl set-l3-mode true 4*** command.

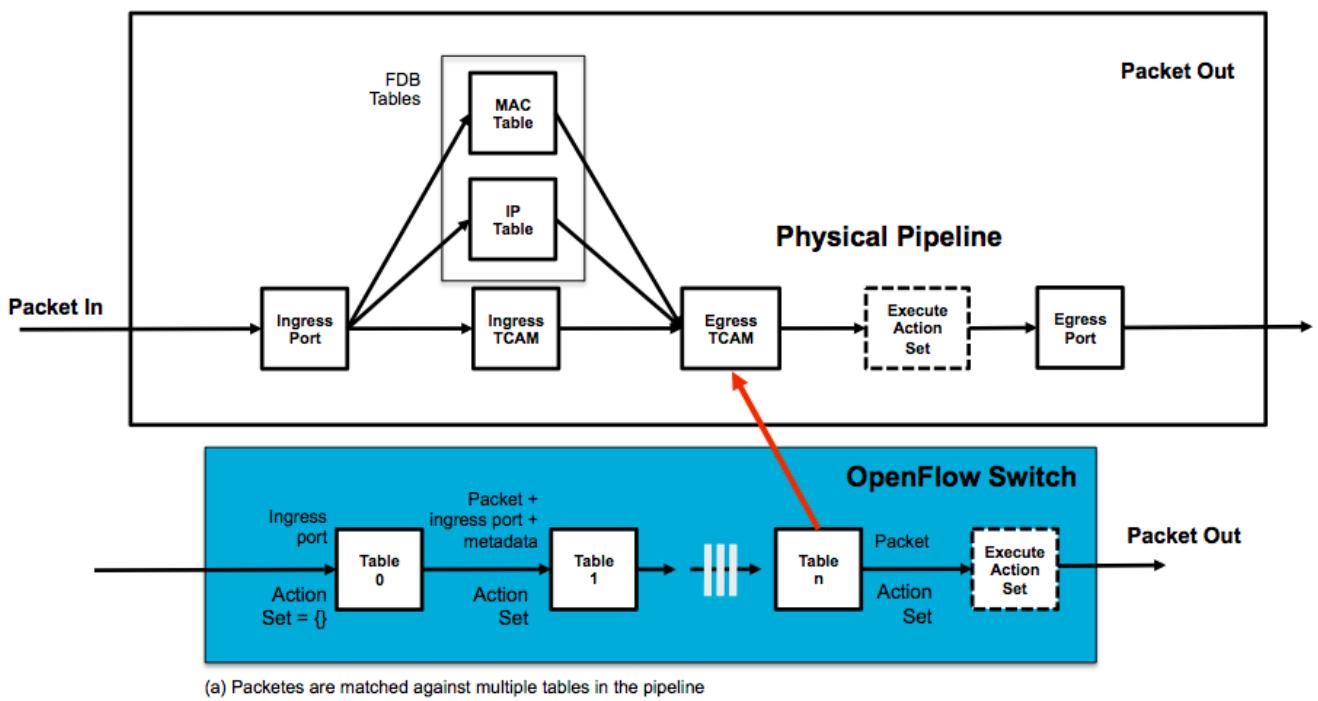
Egress Flow Table

The match in the egress openflow table is very similar to the ingress tcam table. Only some ipv6 fields cannot be matched.

The full match list is in_port, src_mac, dst_mac, ether_type, vlan_id, vlan_priority, ip_protocol, ipv4_src_addr, ipv4_dst_addr, ipv4_tos, tcpudp_src_port, tcpudp_dst_port.

The size of egress table is 512 flows for most platforms (one exception is the P-3290 limited to only 256 flows).

For a full description of the command usage to configure the egress TCAM, see: Egress-mode Command



- Multitable Resources

Multitable Resources

From PicOS version 2.7.1, some new modifications have been made to the multitable feature with the addition of the "resource" feature. Different platforms have different resources.

(All platforms which support l2-l3-buffer-mode are as follows: as5712_54x, as6701_32x, as6712_32x, dcs7032q28, es4654bf, niagara2632xl, niagara2948_6xl, pronto5101, pronto5401, as4610_54p, as4610_54t, as4610_30p, as4610_30t)

Platforms	Buffer-mode Resources
as5712_54x	"0" "l2-size 294912, host-route 12000, route 12000"; "1" "l2-size 229376, host-route 44800, route 12000"; "2" "l2-size 163840, host-route 70400, route 12000"; "3" "l2-size 98304, host-route 96000, route 12000 "; "4" "l2-size 32768, host-route 12000, route 115200"; "5" "l2-size 32768, host-route 12000, route 12000";
as6701_32x	"0" "l2-size 294912, host-route 12000, route 12000"; "1" "l2-size 229376, host-route 44800, route 12000"; "2" "l2-size 163840, host-route 70400, route 12000"; "3" "l2-size 98304, host-route 96000, route 12000 "; "4" "l2-size 32768, host-route 12000, route 115200"; "5" "l2-size 32768, host-route 12000, route 12000";

as6712_32x	"0" "l2-size 294912, host-route 12000, route 12000"; "1" "l2-size 229376, host-route 44800, route 12000"; "2" "l2-size 163840, host-route 70400, route 12000"; "3" "l2-size 98304, host-route 96000, route 12000 "; "4" "l2-size 32768, host-route 12000, route 115200"; "5" "l2-size 32768, host-route 12000, route 12000";
dcs7032q28	"0" "l2-size 294912, host-route 12000, route 12000"; "1" "l2-size 229376, host-route 44800, route 12000"; "2" "l2-size 163840, host-route 70400, route 12000"; "3" "l2-size 98304, host-route 96000, route 12000"; "4" "l2-size 32768, host-route 12000, route 115200"; "5" "l2-size 32768, host-route 12000, route 12000";
es4654bf	"0" "l2-size 131072, host-route 13100, route 4800"; "1" "l2-size 98304, host-route 39300, route 4800"; "2" "l2-size 98304, host-route 13100, route 4800"; "3" "l2-size 32768, host-route 91700, route 4800"; "4" "l2-size 32768, host-route 13100, route 4800"; "5" "l2-size 32768, host-route 13100, route 4800";
niagara2632xl	"0" "l2-size 294912, host-route 12000, route 12000"; "1" "l2-size 229376, host-route 44800, route 12000"; "2" "l2-size 163840, host-route 70400, route 12000"; "3" "l2-size 98304, host-route 96000, route 12000"; "4" "l2-size 32768, host-route 12000, route 115200"; "5" "l2-size 32768, host-route 12000, route 12000";
niagara2948_6xl	"0" "l2-size 294912, host-route 12000, route 12000"; "1" "l2-size 229376, host-route 44800, route 12000"; "2" "l2-size 163840, host-route 70400, route 12000"; "3" "l2-size 98304, host-route 96000, route 12000 "; "4" "l2-size 32768, host-route 12000, route 115200"; "5" "l2-size 32768, host-route 12000, route 12000";
pronto5101	"0" "l2-size 294912, host-route 12000, route 12000"; "1" "l2-size 229376, host-route 44800, route 12000"; "2" "l2-size 163840, host-route 70400, route 12000"; "3" "l2-size 98304, host-route 96000, route 12000 "; "4" "l2-size 32768, host-route 12000, route 115200"; "5" "l2-size 32768, host-route 12000, route 12000";
pronto5401	"0" "l2-size 294912, host-route 12000, route 12000"; "1" "l2-size 229376, host-route 44800, route 12000"; "2" "l2-size 163840, host-route 70400, route 12000";

	"3" "l2-size 98304, host-route 96000, route 12000"; "4" "l2-size 32768, host-route 12000, route 115200"; "5" "l2-size 32768, host-route 12000, route 12000";
as4610_54p, as4610_54t, as4610_30p, as4610_30t	"0" "l2-size 65536, host-route 8192, route 6000"; "1" : "l2-size 49152, host-route 16384, route 6000"; "2" "l2-size 49152, host-route 24576, route 6000"; "3" "l2-size 49152, host-route 8192, route 6000"; "4" "l2-size 32768, host-route 24576, route 6000"; "5" "l2-size 32768, host-route 16384, route 6000";
Others (not support buffer mode)	"l2-size 32768, host-route 12000, route 12000";

First, user should enable the L2 and L3 mode. Then, use the command "**ovs-appctl pica/show tables**" to check the L2 and L3 max flows in the switch.

Example

Step 1: Add bridge and port

```
ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
ovs-vsctl add-port br0 te-1/1/77 vlan_mode=trunk tag=1 -- set Interface te-1/1/77
type=pica8
ovs-vsctl add-port br0 te-1/1/78 vlan_mode=trunk tag=1 -- set Interface te-1/1/78
type=pica8
```

Step 2: Enable L2 and L3 mode

```
ovs-vsctl set-l2-mode true
ovs-vsctl set-l3-mode true
```

Step 3: Check the default resources

```
admin@PicOS-OVS$ovs-appctl pica/show tables
Pica Tables Statistics:
  Pica Tables          Max Limitation          Current Used
  -----
  ICAP Table           2044                      10
  ECAP Table           (null)                   (null)
  VCAP Table           512                       0
  L2 System Table     512                       0
  L2 FDB Table         32256                     0
  L3 Host Table        12000                    IPv4(0),IPv6(0*2)
  L3 Route Table       12000                    IPv4(0),IPv6(0*2)
  UDF Table            (null)                   (null)
```

Step 4: Add L2 flows

```
ssun@dev-42:~/ryu/ryu/app$ ryu-manager simple_switch_13_ssun_buffer_mode5.py
for i in range(4797, 5195):
    for k in ('0','1','2','3','4','5','6','7','8','9'):
        for z in ('0','1','2','3','4','5','6','7','8','9'):
            dl_dst="22:22:22:22:+k+z+:22"
```

```

        match=self.set_match(datapath, dl_dst=dl_dst, vlan_vid=i)
        output_port=78
        actions=self.set_action(datapath, out_port=output_port)
        self.add_flow(datapath, 251, 32768, match, actions)
    
```

Step 5: Check max L2 flow members

```

admin@PicOS-OVS$ovs-ofctl dump-flows br0|grep -c "table=251"
32256
admin@PicOS-OVS$ovs-appctl pica/show tables
Pica Tables Statistics:
  Pica Tables          Max Limitation          Current Used
  -----
  ICAP Table           2044                      8
  ECAP Table           (null)                   (null)
  VCAP Table           512                       0
  L2 System Table     512                       0
  L2 FDB Table         32256                     32256
  L3 Host Table        12000                     IPv4(0),IPv6(0*2)
  L3 Route Table       12000                     IPv4(0),IPv6(0*2)
  UDF Table            (null)                   (null)

```

Step 6: Delete L2 flows and add ipv4 net route (ipv4 l3 host ,ipv6 host ,ipv6 route)

```

admin@PicOS-OVS$ovs-ofctl del-flows br0
ssun@dev-42:~/ryu/ryu/app$ ryu-manager simple_switch_13_ssun_buffer_mode5.py
for j in range(0,51):
    for i in range(0,255):
        output_port=78
        tmp1='%d' %i
        tmp2 = '%d' %j
        ipv4_dst='192.168.' + tmp2 + '.' +tmp1
        dl_dst="22:22:22:22:22:22"
        vlan_vid=20
        match=self.set_match(datapath, dl_type=0x0800, ipv4_dst=ipv4_dst,
        ipv4_dst_mask="255.255.255.255")
        actions=self.set_action(datapath, dl_dst="44:44:44:88:88:88",
        vlan_vid=vlan_vid, out_port=output_port)
        self.add_flow(datapath, 252, 32768, match, actions)

```

Step 7: Check max ipv4 l3 flow members

```

admin@PicOS-OVS$ovs-ofctl dump-flows br0|grep -c "table=252"
12000
admin@PicOS-OVS$ovs-appctl pica/show tables
Pica Tables Statistics:
  Pica Tables          Max Limitation          Current Used
  -----
  ICAP Table           2044                      10
  ECAP Table           (null)                   (null)
  VCAP Table           512                       0
  L2 System Table     512                       1
  L2 FDB Table         32256                     0
  L3 Host Table        12000                     IPv4(0),IPv6(0*2)
  L3 Route Table       12000                     IPv4(12000),IPv6(0*2)
  UDF Table            (null)                   (null)

```

Configuring Network Address Translation

The Network Address Translation (NAT) process maps IP addresses from one address domain (or realm) to another to provide transparent routing to end hosts. Typically, NAT allows organizations to map public external addresses to private or unregistered addresses. Platforms with ASIC Trident2, Trident2Plus and Tomahawk support this function in OVS mode only. A flow with NAT actions (changing IP address or L4 port) can be hardware switched. Flows can be associated with the following actions: mod_nw_dst, mod_nw_src, mod_tp_dst and mod_tp_src.

Listed below is the minimal information needed to process the packet on hardware only (direct flow):

- 1) dl_dst(match field or action), dl_vlan(match field or action), mod_nw_src, mod_tp_src
- 2) dl_dst(match field or action), dl_vlan(match field or action), mod_nw_dst, mod_tp_dst
- 3) dl_dst(match field or action), dl_vlan(match field or action), tp_src(with or wthout in match field),mod_nw_src
- 4) dl_dst(match field or action), dl_vlan(match field or action), nw_src(match field),mod_tp_src
- 5) dl_dst(match field or action), dl_vlan(match field or action), tp_dst(with or wthout in match field),mod_nw_dst
- 6) dl_dst(match field or action), dl_vlan(match field or action), nw_dst(match field),mod_tp_dst

Example 1: SNAT

Step 1: Create a new bridge named br0.

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

Step 2: Add ports to br0.

```
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/1 vlan_mode=trunk tag=1-- set interface te-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/2 vlan_mode=trunk tag=1-- set interface te-1/1/2 type=pica8
```

Step 3: If user is inside network and wants to visit outside network, source IP needs to be modified:

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,tcp,delay,vlan=1999,dl_dst=22:22:22:22:22:22,actions=set_field:192.168.5.5->nw_src,
set_field:888->tp_src,output:2
```

Step 4: Check flow tables.

```
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  cookie=0x0, duration=264.202s, table=0, n_packets=n/a, n_bytes=0,
  tcp,in_port=1,dl_vlan=1999dl_dst=22:22:22:22:22:22
  actions=set_field:192.168.5.5->ip_src,set_field:888->tcp_src,output:2
admin@PicOS-OVS$ 
admin@PicOS-OVS$ovs-appctl pica/dump-flows
```

```
#7417 normal permanent
recirc_id=0,tcp,in_port=1,d1_vlan=1999,d1_dst=22:22:22:22:22:22,actions:set(ipv4(src=192.16
8.5.5)),set(tcp(src=888)),2
#7416 normal permanent priority=0,recirc_id=0, actions:drop
Total 2 flows in HW.
admin@PicOS-OVS$
```

Example 2: D NAT

Step 1: Create a new bridge named br0

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

Step 2: Add ports to br0

```
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/1 vlan_mode=trunk tag=1-- set interface
te-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/2 vlan_mode=trunk tag=1-- set interface
te-1/1/2 type=pica8
```

Step 3: If user is outside network and wants to visit inside network, destination IP needs to be modified:

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,tcp,d1_vlan=1999,d1_dst=22:22:22:22:22:22,actions=set_field:192.168.6.6->nw_dst,
set_field:800->tp_dst,output:2
```

Step 4: Check flow tables

```
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  cookie=0x0, duration=264.202s, table=0, n_packets=n/a, n_bytes=0,
  tcp,in_port=1,d1_vlan=1999d1_dst=22:22:22:22:22:22
  actions=set_field:192.168.5.5->ip_dst,set_field:888->tcp_dst,output:2
admin@PicOS-OVS$ 
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#7417 normal permanent
recirc_id=0,tcp,in_port=1,d1_vlan=1999,d1_dst=22:22:22:22:22:22,actions:set(ipv4(dst=192.16
8.5.5)),set(tcp(dst=888)),2
#7416 normal permanent priority=0,recirc_id=0, actions:drop
Total 2 flows in HW.
admin@PicOS-OVS$
```

Example 3: Packet-driven-flow

If match field or actions cannot satisfy condition of direct flow, this flow will be packet-driven-flow, and it cannot be added to hardware table directly.

Establish br0 and add ports in br0 like above configuration. And add flow as follows:

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,d1_type=0x0800,tcp,actions=set_field:192.168.5.5->nw_src,set_field:0x456->tp_sr
c,output:2
admin@PicOS-OVS$
```

```
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  cookie=0x0, duration=3.442s, table=0, n_packets=n/a, n_bytes=0, tcp,in_port=1
  actions=set_field:192.168.5.5->ip_src,set_field:1110->tcp_src,output:2
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#7460 normal permanent recirc_id=0,tcp,in_port=1, actions:To_CPU(for_packet_driven)
#7458 normal permanent priority=0,recirc_id=0, actions:drop
Total 2 flows in HW.
admin@PicOS-OVS$
```

Sending increasing dst_mac packets to te-1/1/1, mac address from 22:22:22:22:22:22 to 22:22:22:22:22:22:2b, then check tables:

```
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  cookie=0x0, duration=135.680s, table=0, n_packets=n/a, n_bytes=1124112312, tcp,in_port=1
  actions=set_field:192.168.5.5->ip_src,set_field:1110->tcp_src,output:2
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#7479 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=22:22:22:22:22:22,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
#7478 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=22:22:22:22:22:2a,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
#7477 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=22:22:22:22:22:29,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
#7475 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=22:22:22:22:22:27,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
#7471 normal
priority=1048560,recirc_id=0,tcp,in_port=1,nw_src=192.168.5.5,nw_frag=no,tp_src=1110,
actions:2
#7476 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=22:22:22:22:22:28,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
#7472 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=00:00:00:00:00:00,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
#7484 normal permanent recirc_id=0,tcp,in_port=1, actions:To_CPU(for_packet_driven)
#7483 normal permanent priority=0,recirc_id=0, actions:drop
#7480 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=22:22:22:22:22:23,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
#7482 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=22:22:22:22:22:26,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
#7481 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=22:22:22:22:22:24,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
#7474 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=22:22:22:22:22:2b,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
#7473 normal
priority=1048560,recirc_id=0,tcp,in_port=1,vlan_tci=0x0000/0xffff,d1_dst=22:22:22:22:25,
nw_src=2.0.0.2,nw_frag=no,tp_src=18,tp_dst=35,
```

```
actions:set(ipv4(src=192.168.5.5)),set(tcp(src=1110)),2
Total 14 flows in HW.
admin@PicOS-OVS$
```

1. And te-1/1/2 receive packets with the src_ip 192.168.5.5 and src_port 1110.

Due to ASIC limitation, a flow can not modify l4_src_port without modifying SIP or modify l4_dst_port without modifying DIP.

If only modifying SIP(DIP) or SIP+L4_SRC_PORT(DIP+L4_DST_PORT), up to 2k flow can be configured. If modifying both SIP[|L4_SRC_PORT] and DIP[|L4_DST_PORT], the flow supported is 1k.

2. If set_dl_src is included in actions, the packets will be stamped with set_dl_src (as before). If set_dl_src is not included in actions, the packets will be stamped with the original dl_src . That is to say, keep the original source mac address.

```
Eg1:
ovs-ofctl add-flow br0
in_port=1,ip,tcp,dl_vlan=2,dl_src=00:11:22:33:44:55,dl_dst=00:01:02:03:04:05,actions=set_field:0x456->tp_src,set_field:192.168.5.5->nw_src,output:2
send packets
result:

MAC: ----- MAC Header -----
MAC: Destination Address : 00 01 02 03 04 05
MAC: Source Address : 00 11 22 33 44 55

Eg2:
ovs-ofctl add-flow br0
in_port=1,ip,tcp,dl_vlan=2,dl_dst=00:01:02:03:04:05,actions=set_field:0x456->tp_src,set_field:192.168.5.5->nw_src,output:2
send packets
result:

MAC: ----- MAC Header -----
MAC: Destination Address : 00 01 02 03 04 05
MAC: Source Address : 00 11 22 33 44 55

Eg3:
ovs-ofctl add-flow br0
in_port=1,ip,tcp,dl_vlan=2,dl_src=00:11:22:33:44:55,dl_dst=00:01:02:03:04:05,actions=set_field:0x456->tp_src,set_field:192.168.5.5->nw_src,set_field:22:22:22:22:22:22->dl_src,output:2
send packets
result:

MAC: ----- MAC Header -----
MAC: Destination Address : 00 01 02 03 04 05
MAC: Source Address : 22 22 22 22 22 22

Eg4:
ovs-ofctl add-flow br0
in_port=1,ip,tcp,dl_vlan=2,dl_dst=00:01:02:03:04:05,actions=set_field:0x456->tp_src,set_field:192.168.5.5->nw_src,set_field:22:22:22:22:22:>dl_src,output:2
send packets
result:
```

```
MAC: ----- MAC Header -----
MAC: Destination Address : 00 01 02 03 04 05
MAC: Source Address : 22 22 22 22 22 22
```

ASIC Limitation

Because some limitation of ASIC, some flow installed hardware can not work as expected. User should refer to this chapter before user starts to trouble-shoot the issues.

udp/ip, tcp/ip

When user adds flows with the same priority, and one flow's match fields includes another flow's match fields, the action of flow is at random. For example:

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
priority=10000,ip,in_port=14,dl_vlan=2,actions=push_vlan:0x8100,set_field:2503-\>vlan_vid,o
utput:15
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#40 permanent priority=10000,ip,in_port=14,dl_vlan=2,
actions:push_vlan(vid=2503),mod_vlan_pcp(pcp=0),15
Total 1 flows in TCAM.
admin@PicOS-OVS$ovs-ofctl add-flow br0
priority=10000,udp,in_port=14,dl_vlan=2,tp_dst=2123,actions=push_vlan:0x8100,set_field:2500
-\>vlan_vid,output:15
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#41 permanent priority=10000,udp,in_port=14,dl_vlan=2,tp_dst=2123,
actions:push_vlan(vid=2503),mod_vlan_pcp(pcp=0),15
#40 permanent priority=10000,ip,in_port=14,dl_vlan=2,
actions:push_vlan(vid=2503),mod_vlan_pcp(pcp=0),15
Total 2 flows in TCAM.
```

If user doesn't want this result, user should modify the two flows' priorities.

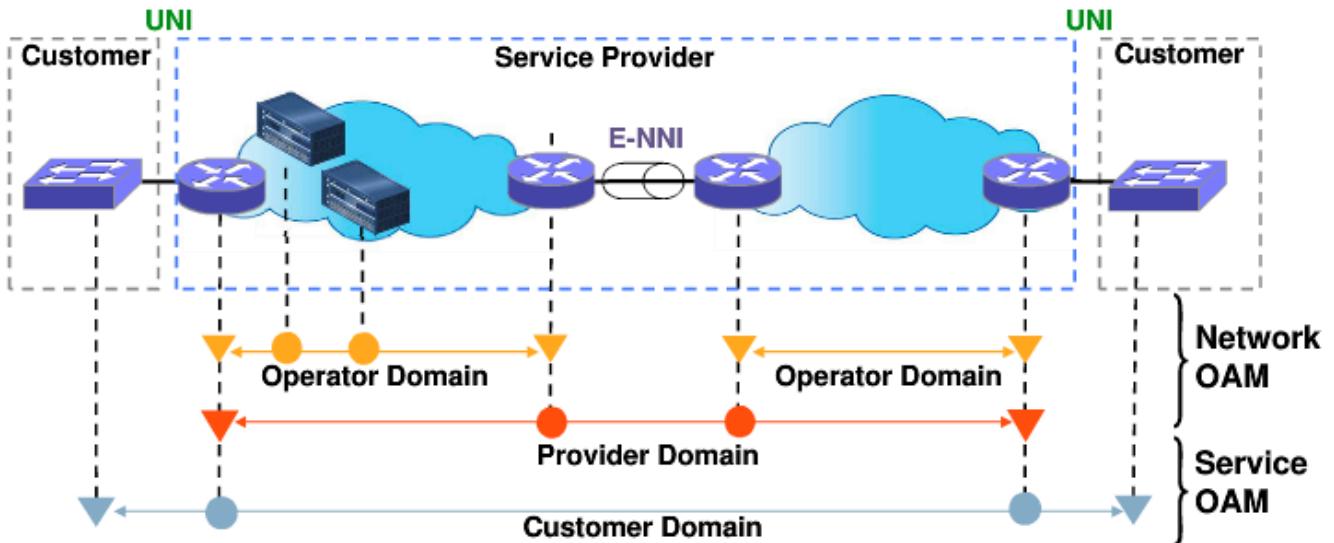
```
admin@PicOS-OVS$ovs-ofctl add-flow br0
priority=12000,udp,in_port=14,dl_vlan=2,tp_dst=2123,actions=push_vlan:0x8100,set_field:2500
-\>vlan_vid,output:15
ovs-ofctl add-flow br0
priority=10000,ip,in_port=14,dl_vlan=2,actions=push_vlan:0x8100,set_field:2503-\>vlan_vid,o
utput:15
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#42 permanent priority=12000,udp,in_port=14,dl_vlan=2,tp_dst=2123,
actions:push_vlan(vid=2500),mod_vlan_pcp(pcp=0),15
#40 permanent priority=10000,ip,in_port=14,dl_vlan=2,
actions:push_vlan(vid=2503),mod_vlan_pcp(pcp=0),15
Total 2 flows in TCAM.
```

Vlan Isolation

At the moment, the vlan isolation only works on the egress of a port, not on the ingress direction.

Configuring CFM

Connectivity Fault Management (CFM) is an IEEE standard, 802.1ag, which specifies protocols, procedures, and managed objects to support transport fault management. CFM is used for detecting link connectivity fault, confirming the fault and locating the fault in the network.



The standard, 802.1ag defines:

- Maintenance domains (MD), their constituent maintenance points (MIP, MEP), and the managed objects (MA) required to create and administer them.
- The relationship between maintenance domains and the services offered by VLAN-aware bridges and provider bridges.
- The description of protocols and procedures used by maintenance points to maintain and diagnose connectivity faults within a maintenance domain.
- Means for future expansion of the capabilities of maintenance points and their protocols.

As illustrated in the above figure, each domain (operator, provider, customer) is allocated. Maintenance association Intermediate endPoint (MIP) and Maintenance domain End Point (MEP) essentially define the domain boundaries and the intermediate elements. Effectively, these end-points define a fault-domain which aids in building a flexible fault-management framework. Network and Service OAM shown above, are typically used to represent faults to a network operator and a service consumer.

IEEE 802.1ag Ethernet CFM (Connectivity Fault Management) protocols comprise three protocols that work together to help administrators debug Ethernet networks. They are:

Continuity Check Protocol (CCP) - Heartbeat messages for CFM. The Continuity Check Message (CCM) provides a means to detect connectivity failures in an MA. CCMs are multicast messages. CCMs are confined to a domain (MD). These messages are unidirectional and do not solicit a response. Each MEP transmits a periodic multicast Continuity Check Message inward towards the other MEPs.

Link Trace (LT) - Link Trace messages otherwise known as Mac Trace Route, are Multicast frames that a MEP transmits to track the path (hop-by-hop) to a destination MEP, which is similar in concept to User Datagram Protocol(UDP) Trace Route. Each receiving MEP sends a Trace Route Reply directly to the Originating MEP, and regenerates the Trace Route Message.

Loop-back (LB) - Loop-back messages otherwise known as MAC ping are Unicast frames that a MEP transmits, they are similar in concept to an Internet Control Message Protocol (ICMP) Echo (Ping) messages, sending Loopback to successive MIP's can determine the location of a fault. Sending a high volume of Loopback Messages can test bandwidth, reliability, or jitter of a service, which is similar to flood ping. An MEP can send a Loopback to any MEP or MIP in the service. Unlike CCMs, Loop back messages are administratively initiated and stopped.

Monitor connectivity to a remote maintenance point on ge-1/1/1

Set the MPID of CFM:

```
admin@PicOS-OVS$ ovs-vsctl set Interface ge-1/1/1 cfm_mpid=2333
```

A Maintenance Point ID (MPID) uniquely identifies each endpoint within a Maintenance Association. According to the 802.1ag specification, MPIDs can only range between [1, 8191].

Set extended mode:

```
admin@PicOS-OVS$ovs-vsctl set Interface ge-1/1/1 other_config:cfm_extended=true
```

Extended mode increases the accuracy of the cfm_interval configuration parameter by breaking wire compatibility with 802.1ag compliant implementations. An extended mode allows **eight byte MPIDs**.

Set demand mode:

```
admin@PicOS-OVS$ovs-vsctl set Interface ge-1/1/1 other_config:cfm_demand=true
```

When true, and cfm_extended is true, the CFM module operates in demand mode. By default it is set to false. When in demand mode, traffic received on the Interface is used to indicate liveness. CCMs are still transmitted and received. At least one CCM must be received every 100 * cfm_interval amount of time. Otherwise, even if traffic is received, the CFM module will trigger the connectivity fault. Demand mode disables itself when there are multiple remote maintenance points.

Set the requested transmission interval:

```
admin@PicOS-OVS$ovs-vsctl set Interface ge-1/1/1 other_config:cfm_interval=10000
```

In standard mode supports intervals of 3, 10, 100, 1000, 10000, 60000, or 600000 ms are supported. Extended mode supports any interval up to 65535 ms and default is 1000 ms. However, we do not recommend intervals less than 100 ms.

Set CCM VLAN tag:

```
admin@PicOS-OVS$ovs-vsctl set Interface ge-1/1/1 other_config:cfm_ccm_vlan=2000
admin@PicOS-OVS$ovs-vsctl set Interface ge-1/1/1 other_config:cfm_ccm_vlan=random
```

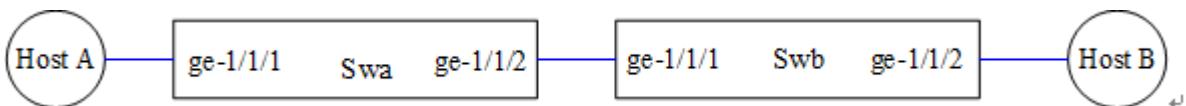
When set, the CFM module will apply a VLAN tag to all CCMs it generates with the given value.

Set CCM Priority:

```
admin@PicOS-OVS$ovs-vsctl set Interface ge-1/1/1 other_config:cfm_ccm_pcp=7
```

When set, the CFM module will apply a VLAN tag to all CCM's, it generates with the given PCP value, the VLAN ID of the tag is governed by the value of "cfm_ccm_vlan". If "cfm_ccm_vlan" is unset, a VLAN ID of zero is used.

CFM Example



Step 1: Basic configuration

DUT1:

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set Interface
ge-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1 -- set Interface
ge-1/1/2 type=pica8
```

DUT2:

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set Interface
ge-1/1/1 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1 -- set Interface
ge-1/1/2 type=pica8
```

Step 2: Configure cfm:

DUT1:

```
admin@PicOS-OVS$ovs-vsctl set interface ge-1/1/2 cfm-mpid=8999
admin@PicOS-OVS$ovs-vsctl set interface ge-1/1/2 other_config:cfm_extended=true
```

DUT2:

```
admin@PicOS-OVS$ovs-vsctl set interface ge-1/1/1 cfm-mpid=9000
admin@PicOS-OVS$ovs-vsctl set interface ge-1/1/1 other_config:cfm_extended=true
```

Step 3: Check packets

DUT1:

Check list interface:

```
admin@PicOS-OVS$ovs-vsctl list interface ge-1/1/2
_uuid : 94942d57-d9a8-4030-ad3b-483dadbd7926
admin_state : up
bfd : {}
bfd_status : {}
cfm_fault : false
cfm_fault_status : []
cfm_flap_count : 2
cfm_health : []
cfm_mpid : 8999
cfm_remote_mpids : [9000]
cfm_remote_opstate : up
duplex : full
external_ids : {}
ifindex : 13
ingress_policing_burst: 0
ingress_policing_rate: 0
lacp_current : []
link_resets : 0
link_speed : 1000000000
link_state : up
mac : []
mac_in_use : "00:e0:ec:25:2d:5e"
mtu : 9212
name : "ge-1/1/2"
ofport : 13
ofport_request : []
options : {}
other_config : {cfm_extended="true"}
statistics : {collisions=0, rx_bytes=3255, rx_crc_err=0, rx_dropped=28, rx_errors=0,
rx_frame_err=0, rx_over_err=0, rx_packets=35, tx_bytes=1395, tx_dropped=0, tx_errors=0,
tx_packets=15}
status : {}
type : "pica8"
wred_queues : {}
```

Check cfm/show:

```
admin@PicOS-OVS$ovs-appctl cfm/show
---- ge-1/1/2 ----
MPID 8999: extended
    average health: undefined
    opstate: up
    remote_opstate: up
    interval: 1000ms
    next CCM tx: 481ms
    next fault check: 973ms
Remote MPID 9000
    recv since check: true
    opstate: up
admin@PicOS-OVS$
```

Check hardware table:

```
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#168 normal permanent priority=18000000,in_port=2,d1_dst=01:23:20:00:00:30,d1_type=0x8902,
```

```
actions:userspace(pid=0,slow_path(cfm))
#167 normal permanent priority=0, actions:drop
Total 2 flows in HW.
```

DUT2:**Check list interface:**

```
admin@PicOS-OVS$ovs-vsctl list interface ge-1/1/1
_uuid : 61bb8ef5-30f9-4855-8cfa-flee0bc5b154
admin_state : up
bfd : {}
bfd_status : {}
cfm_fault : false
cfm_fault_status : []
cfm_flap_count : 0
cfm_health : []
cfm_mpid : 9000
cfm_remote_mpids : [8999]
cfm_remote_opstate : up
duplex : full
external_ids : {}
ifindex : 11
ingress_policing_burst: 0
ingress_policing_rate: 0
lacp_current : []
link_resets : 0
link_speed : 1000000000
link_state : up
mac : []
mac_in_use : "08:9e:01:a8:00:49"
mtu : 9212
name : "ge-1/1/11"
ofport : 11
ofport_request : []
options : {}
other_config : {cfm_extended="true"}
statistics : {collisions=0, rx_bytes=1302, rx_crc_err=0, rx_dropped=8, rx_errors=0,
rx_frame_err=0, rx_over_err=0, rx_packets=14, tx_bytes=558, tx_dropped=0, tx_errors=0,
tx_packets=6}
status : {}
type : "pica8"
wred_queues : {}
admin@PicOS-OVS$
```

Check cfm/show:

```
admin@PicOS-OVS$ovs-appctl cfm/show
---- ge-1/1/1 ----
MPID 9000: extended
    average health: undefined
    opstate: up
    remote_opstate: up
    interval : 1000ms
    next CCM tx: 802ms
    next fault check: 1254ms
Remote MPID 8999
    recv since check: true
    opstate: up
admin@PicOS-OVS$
```

Check hardware table:

```
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#168 normal permanent priority=18000000,in_port=1,d1_dst=01:23:20:00:00:30,d1_type=0x8902,
actions:userspace(pid=0,slow_path(cfm))
#167 normal permanent priority=0, actions:drop
Total 2 flows in HW.
```

 Standard mode, dl_mac is 01:80:c2:00:00:30; when extended mode, the dst mac is 01:23:20:00:00:30.

Configure cfm on port ge-1/1/13, if delete cfm

```
admin@PicOS-OVS$ovs-vsctl clear interface ge-1/1/13 cfm_mpid
admin@PicOS-OVS$ovs-vsctl remove interface ge-1/1/13 other_config cfm_interval="10000"
admin@PicOS-OVS$ovs-vsctl remove interface ge-1/1/13 other_config cfm_extended="true"
admin@PicOS-OVS$ovs-vsctl remove interface ge-1/1/13 other_config cfm_ccm_vlan=random
```

Open vSwitch Configuration File

The PicOS switch stores the OVS (Open vSwitch) configuration in **/ovs/ovs-vswitchd.conf.db**.

```
admin@SpineA-OVS$cd /ovs
admin@SpineA-OVS$ls
bin  etc  lib  ovs-vswitchd.conf.db  sbin  share  var
```

OVS LLDP

The **Link Layer Discovery Protocol (LLDP)** is a vendor-neutral link layer protocol used by network devices for advertising their identity, capabilities, and neighbors on ethernet networks . The protocol is formally referred to by the IEEE as *Station and Media Access Control Connectivity Discovery* specified in standards document **IEEE 802.1AB**

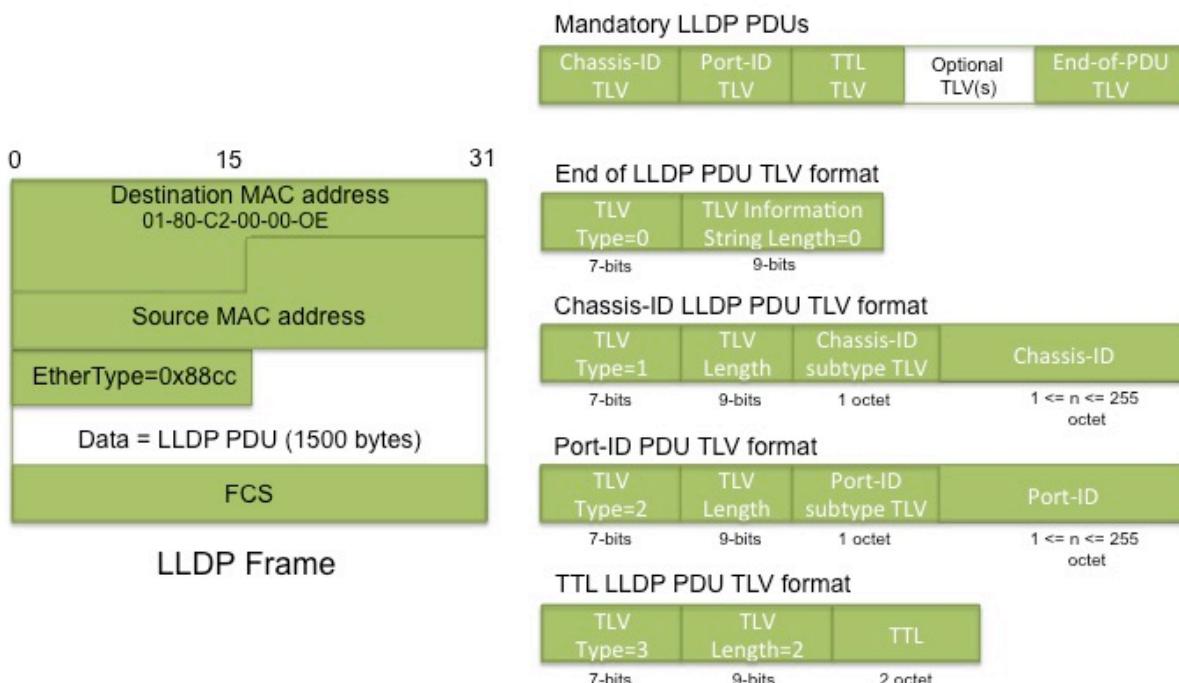


Figure1: LLDP packet format and the mandatory LLDP PDUs

The LLDP frame consists of the following:

1. a well known destination multicast address (01:80:C2:00:00:OE)
2. ether type of 0x88cc
3. LLDP PDU consisting of the following mandatory LLDP TLVs:
 - a. Chassis TLV: Identifies the 802 LAN device's chassis.
 - b. Port-ID TLV: Identifies the port from which the LLDPDU is transmitted.
 - c. TTL TLV: Indicates how long (in seconds) the LAN device's information received in the LLDPDU is to be treated as valid information. Non-zero information indicates the device's information is to be updated in the LLDP remote system MIB. A value of 0 indicates the information associated with the LAN device is no longer valid and should be removed.
 - d. End-of-LLDP TLV: Indicates the end of TLVs in the LLDPDU.

Typically, an open flow controller uses LLDP messages to discover neighbor devices, active links and hosts connected to the open flow switches. Following represents a typical flow used to relay LLDP messages from an open flow switch to the controller

dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc, actions:CONTROLLER

Alternatively LLDP can be enabled in PicOS. PicOS supports the following LLDP TLVs:

1. *Port Description TLV*
2. *System Name TLV*
3. *System Description TLV*
4. *System Capabilities TLV*
5. *Management Address TLV* V.802.1

6. Organizationally Specific TLV includes:

- a. *PVID TLV*(The default value is false)
- b. *MAC PHY TLV*(The default value is false)
- c. *MDI TLV*(The default value is false)
- d. Max Frame Size TLV (The default value is false)



PicOS supports LLDP from version 2.6.

LLDP commands

Enable or disable ll dp on bridge

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 lldp_enable=true
admin@PicOS-OVS$ovs-vsctl clear bridge br0 lldp_enable
```

LLDP admin status

```
admin@PicOS-OVS$ovs-vsctl set interface ge-1/1/1 lldp_admin_status=RxTx
admin@PicOS-OVS$ovs-vsctl set interface ge-1/1/1 lldp_admin_status=RxOnly
admin@PicOS-OVS$ovs-vsctl set interface ge-1/1/1 lldp_admin_status=TxOnly
admin@PicOS-OVS$ovs-vsctl set interface ge-1/1/1 lldp_admin_status=disabled
```

LLDP transmit parameters

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-msg-tx-hold=4
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-msg-tx-interval=30
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-reinit-delay=2
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tx-delay=2
```

LLDP transmit parameters on a per interface

```
ovs-vsctl set interface te-1/1/25 lldp_tx_interval=4
ovs-vsctl set interface te-1/1/25 lldp_tx_delay=2

clear commands:
ovs-vsctl clear interface te-1/1/14 lldp_tx_interval
ovs-vsctl clear interface te-1/1/14 lldp_tx_delay
```

Optional TLVs in ll dpdu

Port Description TLV (The default value is true)

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-port-desc=true
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-port-desc=false
```

System Name TLV (The default value is true)

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-sys-name=true
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-sys-name=false
```

System Description TLV (The default value is true)

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-sys-desc=true
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-sys-desc=false
```

System Capabilities TLV (The default value is true)

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-sys-cap=true
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-sys-cap=false
```

Management Address TLV (The default value is true)

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-mgmt-addr=true
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-mgmt-addr=false
```

PVID TLV (802.1 Organizationally Specific TLV) (The default value is false)

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-pvid=true
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-pvid=false
```

MAC PHY TLV (802.3 Organizationally Specific TLV) (The default value is false)

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-mac-phy=true
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-mac-phy=false
```

Power Via MDI TLV (802.3 Organizationally Specific TLV) (The default value is false)

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-power-via-mdi=true
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-power-via-mdi=false
```

Link Aggregation TLV (802.3 Organizationally Specific TLV) (The default value is false)

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-link-aggregation=true
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-link-aggregation=false
```

Max Frame Size TLV (802.3 Organizationally Specific TLV) (The default value is false)

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-max-frame-size=true
admin@PicOS-OVS$ovs-vsctl set bridge br0 other_config:lldp-tlv-max-frame-size=false
```

show configuration

```
admin@PicOS-OVS$ovs-vsctl list bridge br0
admin@PicOS-OVS$ovs-vsctl list interface ge-1/1/1
```

show LLDP remote neighbors

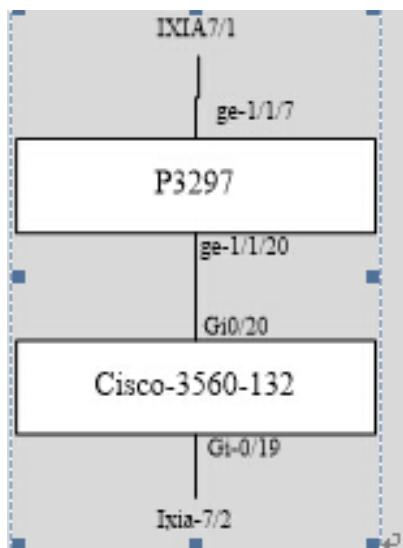
```
admin@PicOS-OVS$ovs-vsctl list lldp_neighbor
```

show LLDP using "ovs-appctl"

```
admin@PicOS-OVS$ovs-appctl lldp/show br0
admin@PicOS-OVS$ovs-appctl lldp/show br0 ge-1/1/1
```

Examples

topology



default port status test

Cisco3560 (mac 00:22:BE:96:F5:94):

```
Switch#configure terminal
Switch(config)#lldp run(default status is TXRx)
Switch(config)#monitor session 1 source interface gigabitEthernet 0/20 rx
Switch(config)#monitor session 1 destination interface gigabitEthernet 0/19
```

P3297 (mac 00:e0:ec:25:2d:5e):

```
admin@PicOS-OVS$ovs-vsctl set bridge br0 lldp_enable=true
not set lldp_admin_status(default value is RxTx)
admin@PicOS-OVS$ovs-vsctl list lldp_neighbor
_uuid : dc8a0cba-aefd-4b54-b02f-e53b315fd783
```

```

local_id          : "<00:e0:ec:25:2d:5e>:<ge-1/1/20>"
mgmt_address     : {"10.10.51.132":0}
port_description : "GigabitEthernet0/20"
remote_id        : "<00:22:be:96:f5:80>:<Gi0/20>"
system_description : "Cisco IOS Software, C3560E Software (C3560E-UNIVERSALK9-M), Version 15.0(2)SE2, RELEASE SOFTWARE (fc1)\nTechnical Support:\nhttp://www.cisco.com/techsupport\nCopyright (c) 1986-2013 by Cisco Systems, Inc.\nCompiled Tue 05-Feb-13 12:04 by prod_rel_team"
system_name       : Switch
admin@PicOS-OVS$
```

Result:

- 7/1: receive the lldp from both Cisco3560 and P3297
- 7/2: only receiving the lldp with src mac **00:e0:ec:25:2d:5e**

ⓘ note

1. LLDP only works on physical ports
2. PicOS does not support power-via-mdi TLV;
3. $1 \leq \text{lldp-tx-delay} \leq 0.25 \times \text{msg-tx-interval}$;

Abstract

The goal is to be able to add flows which utilizes the udf (User defined filter) function of the hardware.

As hardware cannot parse the L3 protocol of an mpls packet, we can not match both mpls labels and transport layer ports. But we can do this by utilizing the udf function of hardware. The udf allow us to match 4 bytes of content at the given offset of the L2 header of the L3 header. The L2 header refers to the mac header of frame, and the L3 header refers to the first mpls label of mpls frame or IP header of the non-mpls frame (in other words the header next to the inner VLAN tag).

Due to the limitation of the hardware, we only support 4 fields of udf. The max number of udf flows share the same limitation of normal tcam flows.

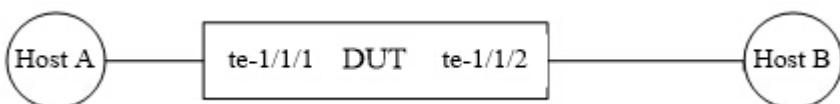
ⓘ Note:

Only untag packets can be matched when configure udf base L2 and L3 at the same time.

Example 1

Send IP packets with dl_dst 22:22:22:22:22:22, dl_src 22:11:11:11:11:11, tag 1000.

And we want to match this tag.



Step1: Create a new bridge named br0

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

Step 2: Add ports to br0

```
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/1 vlan_mode=trunk tag=1 -- set Interface  
te-1/1/1 type=pica8  
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/2 vlan_mode=trunk tag=1 -- set Interface  
te-1/1/2 type=pica8
```

Step 3: Configure udf and add flow

```
ovs-vsctl set-udf-mode "udf0(12,offset=12,length=4)"  
ovs-ofctl add-flow br0 table=250,in_port=1,udf0=0x810003E8/0x0000ffff,actions=2
```

Step 4: Check table

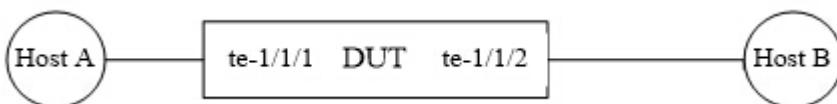
```
admin@PicOS-OVS$ovs-vsctl show-udf-mode  
    udf mode is udf0(12,offset=12,length=4)  
admin@PicOS-OVS$  
admin@PicOS-OVS$ovs-ofctl dump-flows br-vi  
OFPST_FLOW reply (OF1.4) (xid=0x2):  
  cookie=0x0, duration=25.315s, table=250, n_packets=n/a, n_bytes=0,  
  in_port=4,udf0=0x3e8/0xffff actions=output:5  
admin@PicOS-OVS$  
admin@PicOS-OVS$ovs-appctl pica/dump-flows  
#1 udf permanent recirc_id=0,in_port=4,udf0=0x3e8/0xffff, actions:5  
#0 normal permanent priority=0,recirc_id=0, actions:drop  
Total 2 flows in HW.  
admin@PicOS-OVS$
```

Example 2

Send IP packets with dl_dst 22:22:22:22:22:22, dl_src

22:11:11:11:11:11,ip_src=192.168.200.100,ip_dst=192.168.100.100,tp_src=2002,tp_dst=3003,mpls_label

And we want to match mpls_label and tp_src:



Step1: Create a new bridge named br0

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

Step 2: Add ports to br0

```
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/1 vlan_mode=trunk tag=1 -- set Interface  
te-1/1/1 type=pica8  
admin@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/2 vlan_mode=trunk tag=1 -- set Interface  
te-1/1/2 type=pica8
```

Step 3: Configure udf and add flow

```
ovs-vsctl set-udf-mode "udf0(13,offset=0,length=4), udf1(13,offset=24,length=2)"
ovs-ofctl add-flow br0 table=250,
in_port=1,udf0=0x0029a140/0xfffff000,udf1=0x07d2,actions=2
```



Note:

1. Offset needs to be aligned on 4 bytes and less than or equal to 124, length is less than or equal to 4.
2. (base, offset, length) defines one field of udf. Base refers to L2 or L3. Due to hardware limitations, we only support 4 fields of udf. But the max number of udf flows share the same limitation of normal tcam flows. L2 fields must be placed before L3 fields, and fields are in ascending order of offset with no overlapping of each other.
3. In the udf table, we can only use udf match format. We can not use both OXM (OpenFlow Extensible Match) and udf format in the same flow.
4. Flows in other tables can not use goto action with a udf table, and udf tables do not support goto action either, supporting output and drop actions, and from version 2.6.4, udf actions also support set_queue, meter and normal.
5. All udf flows will have higher priority than all the other tcam flows.
6. Adding udf flows will return errors, when there are arp or mpls flows in hardware already; also arpor mpls flows will return an error when there are udf flows in the hardware table already.
7. When adding udf flows, table=250 in match field is needed.
8. User cannot enable udf mode and match mode at the same time.

UDF L4

From PicOS2.8.0, picos support UDF offset from L4. The L4 header refers to L4 protocol ports, such as tcp or udp.

For example:

Send packets with

```
dl_dst=22:22:22:22:22:22,dl_src=22:11:11:11:11:11,
dl_vlan=199,nw_dst=1.1.1.1,nw_src=2.2.2.2,udp,udp_src=1234,udp_dst=5678
```

And we will match udp source port and destination port with udf.

Configure udf:

```
admin@PicOS-OVS$ovs-vsctl set-udf-mode "udf0(14,offset=0,length=4)"
```

Add flow.

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,table=250,udf0=0x04d2162e,actions=2
```

Macro UDF

From PicOS2.8.0, support a new format to add udf flow entry. Different from previous use (base, offset, length) to configure udf, user can use udf fields to configure it now.

1)Show all udf field.

```
admin@PicOS-OVS$ovs-vsctl show-udf-field
Udf support fields:
```

Match field	layer	Offset	Length
<hr/>			
dl_type	12	12	2
vlan_inner	12	16	4
mpls_outermost	13	0	4
ip_src	13	12	4
ip_dst	13	16	4
l2gre_key	13	24	4
l2gre_ip_src_high	13	52	4
l2gre_ip_src_low_dst_high	13	56	4
l2gre_ip_dst_low	13	60	2
l2gre_ip_src_high_tag	13	56	4
l2gre_ip_src_low_dst_high_tag	13	60	4
l2gre_ip_dst_low_tag	13	64	2
l4_port	14	0	4
vxlan_vni	14	12	3
vxlan_ip_src_high	14	40	4
vxlan_ip_src_low_dst_high	14	44	4
vxlan_ip_dst_low	14	48	2
vxlan_ip_src_high_tag	14	44	4
vxlan_ip_src_low_dst_high_tag	14	48	4
vxlan_ip_dst_low_tag	14	52	2

admin@PicOS-OVS\$

2)configure macro udf.

admin@PicOS-OVS\$ovs-vsctl set-macro-udf dl_type,ip_src,ip_dst

The match field can be any field in show-udf-field.

3)show udf mode.

admin@PicOS-OVS\$ovs-vsctl show-udf-mode

or:

admin@PicOS-OVS\$ovs-vsctl show-macro-udf

For example:

We will match vxlan vni for sending vxlan packets with vni=1122867.

1)Configure macro udf.

```
admin@PicOS-OVS$ovs-vsctl set-macro-udf vxlan_vni
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-vsctl show-udf-mode
    udf mode is udf0(14,offset=12,length=3)
admin@PicOS-OVS$ovs-vsctl show-macro-udf
    macro udf mode is
    vxlan_vni : udf0(14,offset=12,length=3)
admin@PicOS-OVS$
```

2)Add flow to match vni.

admin@PicOS-OVS\$ovs-ofctl add-flow br0 table=250,udf0=0x112233,actions=3



- 1.I4 can not work with I2/I3.
- 2.If you want to configure I2 and I3 udf at the same time, the matched packets must be untag packets.
- 3.It cannot match the tag value which packets have one tag using macro udf.

Combinated Mode

Our switches support such a command "*ovs-vsctl set-combined-mode true / false*". This command is for resolving the packets forwarding problem of push_vlan. Sometimes, the packets forwarding is not as expected.

Egthere are two flows in our switch below:

```
priority=100, in_port=1, dl_vlan=1, actions=push_vlan:100, output:2
```

```
priority=110,in_port=1,dl_vlan=100,actions=output:3
```

Sending packets match to the first flow: We expect the packets to forward to port 2 after pushing vlan 100, but the actual result is that port 3 receives these packets with outer vlan 100.

In order to solve this kind of problem, we add a new command "ovs-vsctl set-combined-mode true|false". When enabling this combined mode, each flow can have a new match field class-id added. Each flow has a different class-id, so the packets can be forwarded correctly. In hardware, each flow is separated with two flows with class-id as following:

When user adds such two flows to the switch:

```
priority=100, in_port=1, dl_vlan=1, actions=push_vlan:100, output:2
```

```
priority=110,in_port=1,dl_vlan=100,actions=output:3
```

In hardware, this flow can be sepeared to two parts in icap and vcap:

```
vcap: priority=100, class-id=1, in_port=1, dl_vlan=1, and push_vlan 100
```

```
icap: priority=100, class-id=1, in_port=1, dl_vlan=100, and output:2
```

```
priority=110, class-id=2, in_port=1, dl_vlan=100, and output:3
```

Commands

enable combinetedmode

```
admin@PicOS-OVS$ovs-vsctl set-combined-mode true
```

disable combined mode(by default)

```
admin@PicOS-OVS$ovs-vsctl set-combined-mode false
```

Example

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set interface
ge-1/1/1 type=pica8
```

```
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1 -- set interface
ge-1/1/2 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/3 vlan_mode=trunk tag=1 -- set interface
ge-1/1/3 type=pica8
admin@PicOS-OVS$ovs-vsctl set-combinated-mode true
admin@PicOS-OVS$ovs-ofctl add-flow br0
priority=100,in_port=1,dl_vlan=1,actions=push_vlan:100,output:2
admin@PicOS-OVS$ovs-ofctl add-flow br0 priority=110,in_port=1,dl_vlan=100,actions=output:3
```

When sending packets match the first flow with dl_vlan=1 to ge-1/1/1, then ge-1/1/2 will forward packets with outer vlan 100, not ge-1/1/3. This will provide our expected result.

Note

1. When enabling or disabling combined mode, all the flows in table 0 are evicted from the switch, except the default **drop** flow;
2. When enabling combined mode, the max flow numbers including push_vlan and other normal flows is different in different platforms:
P5401&&P5101&&P3922&&P3930&&AS6701&&P3780: the max number is 512.
P3290&&P3297&&P3295: the max number is 1024

Enabling Radius in PicOS OVS Mode

Perform the following steps to enable Radius in PicOS OVS mode:

1. Enable Radius on the PicOS switch.

```
admin@PicOS-OVS$sudo /pica/bin/shell/radius_disable.sh false
```

2. Configure the IP address of the external Radius server on the PicOS switch. In the following example, IP address of the Radius server is 1.1.5.41 and the shared key is *abc*.

```
admin@XorPlus$cat /etc/pam_radius_auth.conf
1.1.5.41:1812 abc 1
```

3. Users on the Radius server must be super users. The following example shows the Radius record for user *test8*.

```
test8 Cleartext-Password := "pica8"
Service-Type = Framed-User,
Framed-Protocol = PPP,
Framed-IP-Address = 172.16.3.33,
Framed-IP-Netmask = 255.255.255.0,
Framed-Routing = Broadcast-Listen,
Framed-Filter-Id = "std.ppp",
Framed-MTU = 1500,
Framed-Compression = Van-Jacobsen-TCP-IP,
Class = "super-user"
```

4. Login to the switch via SSH and execute Linux shell commands.

```
build@dev-18:$ ssh 10.10.51.145 -l test8
test8@10.10.51.145's password:
test8@PicOS-OVS:~$ version
Copyright (C) 2009-2014 Pica8, Inc.
```

```
=====
Hardware Model : P-3290
Linux System Version/Revision : 1.1/20809
Linux System Released Date : 03/21/2015
L2/L3 Version/Revision : 1.1/20809
L2/L3 Released Date : 03/21/2015
OVS/OF Version/Revision : 1.1/20809
OVS/OF Released Date : 03/21/2015
test8@PicOS-OVS:~$ pwd
/home/test8
test8@PicOS-OVS:~$
```

5. The Radius user must use the full path to execute OVS commands, as shown below:

```
test8@PicOS-OVSS$ sudo /ovs/bin/ovs-ofctl show br0
OFPT_FEATURES_REPLY (OF1.4) (xid=0x2): dpid:5e3ee89a8f503d30
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS
OFPST_PORT_DESC reply (OF1.4) (xid=0x3):
 1(ge-1/1/1): addr:e8:9a:8f:50:3d:30
    config: 0
    state: LINK_DOWN
    current: COPPER AUTO_NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    speed: 0 Mbps now, 1000 Mbps max
  LOCAL(br0): addr:e8:9a:8f:50:3d:30
    config: 0
    state: LINK_UP
    current: 10MB-FD COPPER
    supported: 10MB-FD COPPER
    speed: 10 Mbps now, 10 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.4) (xid=0x5): frags=normal miss_send_len=0

test8@PicOS-OVS:~$ sudo ovs-vsctl show
395575df-5939-45af-8e4c-d99da4c442dc
  Bridge "br0"
    Port "ge-1/1/1"
      tag: 1
      Interface "ge-1/1/1"
        type: "pica8"
    Port "br0"
      Interface "br0"
        type: internal
```

6. User can also modify the path by editing the `/etc/sudoers` file, so that Radius users can execute OVS commands directly. Change the value of **Defaults secure_path** from `"/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/ovs/bin:/ovs/sbin"` to another desired value.

```
test8@PicOS-OVS:~$ sudo cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      mail_badpass
Defaults
secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/ovs/bin:/o
vs/sbin"
```

```

# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL:ALL) ALL
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
%xorp  ALL=(ALL) NOPASSWD: NOPASSWD: ALL
%root   ALL=(ALL) NOPASSWD: NOPASSWD: ALL
# See sudoers(5) for more information on "#include" directives:
#include /etc/sudoers.d

```

Creating SSL Connection to a Controller

This section describes the procedure to create an SSL connection with the controller.

PicOS Switch

The following steps need to be completed on the PicOS switch:

```

root@PicOS-OVS#apt-get install openssl
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  ca-certificates
The following NEW packages will be installed:
  openssl
0 upgraded, 1 newly installed, 0 to remove and 17 not upgraded.
Need to get 696 kB of archives.
After this operation, 1070 kB of additional disk space will be used.
WARNING: The following packages cannot be authenticated!
  openssl
Authentication warning overridden.
Get:1 http://ftp.debian.org/debian/ stable/main openssl powerpc 1.0.1e-2 [696 kB]
Fetched 696 kB in 5s (131 kB/s)
Selecting previously unselected package openssl.
(Reading database ... 17049 files and directories currently installed.)
Unpacking openssl (from .../openssl_1.0.1e-2_powerpc.deb) ...
Processing triggers for man-db ...
Setting up openssl (1.0.1e-2) ...

root@PicOS-OVS#ovs-pki init
/ovs/bin/ovs-pki: /ovs/var/lib/openvswitch/pki already exists and --force not specified

root@PicOS-OVS#ovs-pki init --force
Creating controllerca...
Creating switchca...

root@PicOS-OVS#cd /ovs/var/lib/openvswitch/pki/controllerca

root@PicOS-OVS#ovs-pki req+sign ctl controller
ctl-req.pem      Mon Jan 13 03:26:05 UTC 2014
               fingerprint 1cbf63b21301f33d9b4aa30540bff492f15bcfd3

root@PicOS-OVS#ls
ca.cnf          careq.pem    crl          ctl-cert.pem    ctl-req.pem    index.txt.attr
index.txt.old    private     serial.old
cacert.pem      certs       crlnumber    ctl-privkey.pem  index.txt     index.txt.attr.old
newcerts        serial

root@PicOS-OVS#ls ctl-privkey.pem ctl-cert.pem
ctl-cert.pem   ctl-privkey.pem

```

```

root@PicOS-OVS#cd /ovs/var/lib/openvswitch/pki/switchca
root@PicOS-OVS#ovs-pki req+sign sc switch
sc-req.pem      Mon Jan 13 03:26:54 UTC 2014
               fingerprint 65ed449bee94b8e7b8ba7da6f6584af2f9cc2fb

root@PicOS-OVS#ls sc-privkey.pem sc-cert.pem
sc-cert.pem    sc-privkey.pem

root@PicOS-OVS#
root@PicOS-OVS#scp /ovs/var/lib/openvswitch/pki/controllerca/ctl-cert.pem
10.10.50.41:/home/build
The authenticity of host '10.10.50.41 (10.10.50.41)' can't be established.
ECDSA key fingerprint is e6:04:3b:c8:24:36:c7:dd:c1:06:6a:69:e2:3b:82:2f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.50.41' (ECDSA) to the list of known hosts.
root@10.10.50.41's password:
ctl-cert.pem

100% 4063      4.0KB/s  00:00
root@PicOS-OVS#scp /ovs/var/lib/openvswitch/pki/controllerca/ctl-privkey.pem
10.10.50.41:/home/build
root@10.10.50.41's password:
ctl-privkey.pem

100% 1675      1.6KB/s  00:00
root@PicOS-OVS#scp /ovs/var/lib/openvswitch/pki/switchca/cacert.pem 10.10.50.41:/home/build
root@10.10.50.41's password:
cacert.pem

100% 4028      3.9KB/s  00:00
root@PicOS-OVS#ovs-vsctl set-ssl /ovs/var/lib/openvswitch/pki/switchca/sc-privkey.pem
/ovs/var/lib/openvswitch/pki/switchca/sc-cert.pem
/ovs/var/lib/openvswitch/pki/controllerca/cacert.pem

root@PicOS-OVS#ovs-vsctl del-br br0
ovs-vsctl: no bridge named br0
root@PicOS-OVS#ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
root@PicOS-OVS#ovs-vsctl set-controller br0 ssl:10.10.50.41:6633
root@PicOS-OVS#

```

Controller

The following steps need to be completed on the controller:

```

root@dev-41:/home/build# ryu-manager --ctl-privkey ./ctl-privkey.pem --ctl-cert
./ctl-cert.pem --ca-certs ./cacert.pem --verbose
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
BRICK ofp_event
  CONSUMES EventOFPPortDescStatsReply
  CONSUMES EventOFPSwitchFeatures
  CONSUMES EventOFPErrorMsg
  CONSUMES EventOFPEchoRequest
  CONSUMES EventOFPHello
connected socket:<eventlet.green.ssl.GreenSSLocket object at 0x9f1ebfc>
address:(('10.10.50.155', 48508)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x9ecf1ec>
move onto config mode
switch features ev version: 0x4 msg_type 0x6 xid 0xa2f1cf23
OFPSwitchFeatures(auxiliary_id=0,capabilities=7,datapath_id=7461368339596857098L,n_buffers=
256,n_tables=254)
move onto main mode

```

Related articles

- Page: How to configure GRE in OVS mode
- Page: How to configure IGMP snooping
- Page: How to create inband connection with controller in OVS
- Page: How to create SSL connection with controller in OVS

Configuring L2MPLS

PUSH L2 MPLS

Pushing an L2 MPLS with one label:

```
root@PicOS-OVS$ovs-ofctl add-flow br0
"in_port=17,ip,dl_vlan=11,actions=push_l2mpls:0x8847,set_field:22:11:11:11:11:10->dl_dst,se
t_field:22:00:00:00:00:01->dl_src,set_field:16->mpls_label,output:18"
```

Pushing an L2MPLS with two labels:

```
root@PicOS-OVS$ovs-ofctl add-flow br0
"in_port=17,ip,dl_vlan=11,actions=push_l2mpls:0x8847,set_field:22:11:11:11:11:10->dl_dst,se
t_field:22:00:00:00:00:01->dl_src,set_field:20->mpls_label,push_mpls:0x8847,set_field:16->m
pls_label,output:18"
root@PicOS-OVS$ovs-ofctl add-flow br0
"in_port=17,dl_type=0x8847,dl_vlan=11,actions=push_mpls:0x8847,set_field:22:00:00:00:00:01->
dl_dst,set_field:22:11:11:11:11:10->dl_src,set_field:20->mpls_label,output:18"
```

SWAP L2 MPLS Label

In the following configuration example, specify a flow which has two labels, and in this situation user wants to swap the outer label. The output flow is L2MPLS packet with outer label 30 and inner label 20.

```
root@PicOS-OVS$ovs-ofctl add-flow br0
"in_port=18,dl_type=0x8847,dl_dst=22:00:00:00:00:01,dl_vlan=11,mpls_label=16,mpls_label2=20
,actions=set_field:30->mpls_label,output:17"
```

POP L2MPLS Label

Popping an MPLS Label of the flow:

```
root@PicOS-OVS$ovs-ofctl add-flow br0
"in_port=17,dl_vlan=11,dl_dst=22:00:00:00:00:01,mpls,mpls_label=16,actions=pop_l2mpls:0x884
7,output:18"
```

Popping one MPLS label for flows with two MPLS labels:

Two labels then pop one outer label.

```
root@PicOS-OVS$ovs-ofctl add-flow br0
"in_port=17,d1_vlan=11,d1_dst=22:00:00:00:00:01,mpls,mpls_label=16,mpls_label2=20,actions=pop_mpls:0x8847,output:18"
```

Popping two MPLS labels for flows with two MPLS labels

In the following configuration, specify a flow which has two labels to pop. The output flow is IP packet. Configure two pop entries to pop the flow.

```
root@PicOS-OVS#ovs-ofctl add-flow br0
"in_port=17,d1_vlan=11,d1_dst=22:00:00:00:00:01,mpls,mpls_label=16,mpls_label2=30,actions=pop_12mpls,output:18"
```

PUSH L2MPLS Label and VLAN

Pushing one L2MPLS label and one VLAN:

In the following configuration, specify flows to push one L2MPLS label and one VLAN.

```
root@PicOS-OVS#ovs-ofctl add-flow br0
"in_port=17,ip,actions=push_12mpls:0x8847,set_field:22:22:22:22:22:22->d1_dst,set_field:22:00:00:00:00:00->d1_src,set_field:16->mpls_label,push_vlan:0x8100,set_field:11->vlan_vid,output:18"
root@PicOS-OVS#ovs-ofctl add-flow br0
in_port=2,d1_type=0x8847,d1_vlan=2999,d1_dst=22:22:22:22:22:22,mpls_label=66,actions=push_mpls:0x8847,set_field:333->mpls_label,push_vlan:0x8100,set_field:1999->vlan_vid,output:4
```

Pushing two L2MPLS Labels and one VLAN:

In the following configuration, specify flows to push two MPLS labels and one VLAN.

```
root@PicOS-OVS#ovs-ofctl add-flow br0
"in_port=17,ip,actions=push_12mpls:0x8847,set_field:22:11:11:11:11:10->d1_dst,set_field:22:00:00:00:01->d1_src,set_field:20->mpls_label,push_mpls:0x8847,set_field:16->mpls_label,push_vlan:0x8100,set_field:11->vlan_vid,output:18"
```

Pushing One MPLS Label and Pop One VLAN

```
root@PicOS-OVS#ovs-ofctl add-flow br0
"in_port=18,d1_type=0x8847,mpls_label=30,d1_dst=22:22:22:22:22:22,d1_src=44:44:44:22:22:22,d1_vlan=11,actions=push_mpls:0x8847,set_field:20->mpls_label,pop_vlan,output:17"
```

Pushing One L2MPLS Label and Pop One VLAN

```
root@PicOS-OVS#ovs-ofctl add-flow br0
"in_port=18,d1_vlan=100,actions=push_L2mpls:0x8847,set_field:22:22:22:22:22->d1_dst,set_field:44:44:44:44:44->d1_src,set_field:20->mpls_label,pop_vlan,output:17"
```

Pushing Two L2MPLS Headers

```
root@PicOS-OVS#ovs-ofctl add-flow br0
"in_port=18,dl_type=0x8847,actions=push_l2mpls:0x8847,set_field:22:00:00:00:00:01->dl_dst,
set_field:22:11:11:11:11:10->dl_src,set_field:20->mpls_label,output:17"
```

Configuring Inner VLAN

Push L2MPLS, push one inner VLAN:

```
ovs-ofctl add-flow br0
"in_port=17,ip,actions=push_vlan:0x8100,set_field:8->vlan_vid,push_l2mpls:0x8847,set_field:
22:22:22:22:22:22->dl_dst,set_field:22:00:00:00:00:00->dl_src,set_field:1000->mpls_label,pu
sh_vlan:0x8100,set_field:13->vlan_vid,output:18"

ovs-ofctl add-flow br0
"in_port=17,dl_vlan=2,ip,actions=push_vlan:0x8100,set_field:8->vlan_vid,push_l2mpls:0x8847,
set_mpls_ttl:64,set_field:22:22:22:22:22:22->dl_dst,set_field:22:00:00:00:00:00->dl_src,se
t_field:1000->mpls_label,push_vlan:0x8100,set_field:11->vlan_vid,output:18"
```

Push L2MPLS, modify inner VLAN:

```
ovs-ofctl add-flow br0
"in_port=17,ip,actions=set_field:8->vlan_vid,push_l2mpls:0x8847,set_mpls_ttl:64,set_field:2
2:22:22:22:22:22->dl_dst,set_field:22:00:00:00:00:00->dl_src,set_field:1000->mpls_label,pu
sh_vlan:0x8100,set_field:11->vlan_vid,output:18"

ovs-ofctl add-flow br0
"in_port=17,dl_vlan=2,ip,actions=set_field:8->vlan_vid,push_l2mpls:0x8847,set_mpls_ttl:64,s
et_field:22:22:22:22:22:22->dl_dst,set_field:22:00:00:00:00:00->dl_src,set_field:1000->mpls
_label,push_vlan:0x8100,set_field:11->vlan_vid,output:18"
```

Push L2MPLS, pop inner VLAN:

```
ovs-ofctl add-flow br0
"in_port=17,ip,actions=pop_vlan,push_l2mpls:0x8847,set_mpls_ttl:64,set_field:22:22:22:22:22
:22->dl_dst,set_field:22:00:00:00:00:00->dl_src,set_field:1000->mpls_label,push_vlan:0x8100
,set_field:11->vlan_vid,output:18"

ovs-ofctl add-flow br0
"in_port=17,dl_vlan=2,ip,actions=pop_vlan,push_l2mpls:0x8847,set_mpls_ttl:64,set_field:22:2
2:22:22:22:22->dl_dst,set_field:22:00:00:00:00:00->dl_src,set_field:1000->mpls_label,push_v
lan:0x8100,set_field:11->vlan_vid,output:18"
```

Pop inner VLAN after pop L2MPLS:

```
ovs-ofctl add-flow br0
"in_port=17,dl_vlan=11,dl_dst=22:22:22:22:22:22,mpls,mpls_label=1000,actions=pop_l2mpls,pop
_vlan,output:1"
```

Modify inner VLAN after popping L2MPLS:

```
ovs-ofctl add-flow br0
"in_port=18,d1_vlan=11,d1_dst=22:22:22:22:22:22,mpls,mpls_label=16,mpls_label2=20,actions=pop_12mpls,push_vlan:0x8100,set_field:100->vlan_vid,output:17"
```

i The L2MPLS packet must encapsulate outer VLAN and not be PVID, the original packets must encapsulate the VLAN tag.

Push inner VLAN after popping L2MPLS:

```
ovs-ofctl add-flow br0
"in_port=17,d1_vlan=11,d1_dst=22:22:22:22:22:22,mpls,mpls_label=1000,actions=pop_12mpls,push_vlan:0x8100,set_field:100->vlan_vid,output:18"
```

i The L2MPLS packet must encapsulate outer VLAN and not be PVID, the original packets shouldn't encapsulate VLAN tag.

Inventory Database

Introduction

Pica-OVS mode switch needs to store some basic switch information and some run-time information in the inventory database, this feature is running when Pica-OVS starts.

There are five kinds of information in this module as shown below, including basic Switch Information, SFP/QSFP Information, Counter Information, Alarm Information, and Hardware Flow Information.

Switch Information

Show function control information.

```
root@PicOS-OVS$ovs-invctl show
23aa0580-9578-471a-abae-a9c84b73affd
  sfp_enable: true
  sfp_query_interval: 10
  counter_enable: false
  hwflow_enable: false
  counter_query_interval: 30
  hwflow_query_interval: 30
  alarm_enable: false
  alarm_high_temp: 176
  alarm_low_temp: 32
  last_alarm_id: 0
```

Show basic switch information in the Switch Inventory table, including switch information, temperature information, port information and SFP/QSFP information.

i Before checking the switch information, the DUT must create a bridge and add ports into the bridge. Then make sfp enable=true.

```
root@PicOS-OVS$ovs-invctl enable-sfp true

root@PicOS-OVS$ovs-invctl show-switch-inventory
2ef72735-5c91-43a3-9c95-cb16dd8b2cd7
model: "P3295"
system_mac: "e8:9a:8f:50:3d:30"
vendor_name: "Pica8"
serial_number: "QTFAXI12400017"
ge_port_number: 48
te_port_number: 4
qe_port_number: 0
cpu_temperature: "100 F"
switch_temperature: "100 F"
system_temperature: "93 F"
7c40378e-751f-470f-a8ad-7176a9c16b2c
port_name: "te-1/1/50"
vendor_name: JESS-LINK
vendor_sn: "12344D0001"
wavelength: "256 nm"
temperature: "32 F"
supply_voltage: "0.00 V"
connector: "Copper pigtail"
length_copper: "5 m"
length_50m: "0 m"
length_625m: "0 m"
length_9m: "0 m"
length_9m_km: "0 m"
plugged_in: true
rx_receive_power: "-inf dBm"
tx_bias_current: "0.00 mA"
tx_optical_power: "-inf dBm"
```

SFP/QSFP Information

SFP/QSFP information is stored into the database if user enables this feature. The Administrator could enable/disable this function by using command "ovs-invctl enable-sfp {true | false}".

```
root@PicOS-OVS$ovs-invctl enable-sfp true

Administrator could show these information by command "ovs-invctl show-switch-inventory" or
"ovs-invctl show-sfp-qsf".

root@PicOS-OVS$ovs-invctl show-sfp-qsf
84cb20bd-4ddc-4094-aaaa-b36dc59332a9
port_name: "te-1/1/49"
vendor_name: JESS-LINK
vendor_sn: "12344D0001"
wavelength: "256 nm"
temperature: "32 F"
supply_voltage: "0.00 V"
connector: "Copper pigtail"
length_copper: "5 m"
length_50m: "0 m"
length_625m: "0 m"
length_9m: "0 m"
length_9m_km: "0 m"
plugged_in: true
rx_receive_power: "-inf dBm"
tx_bias_current: "0.00 mA"
tx_optical_power: "-inf dBm"
```

Administrator could control the refresh interval by command "ovs-invctl set-sfp-interval."

```
root@PicOS-OVS$ovs-invctl set-sfp-interval 50
root@PicOS-OVS$ovs-invctl show
23aa0580-9578-471a-abae-a9c84b73affd
  sfp_enable: true
  sfp_query_interval: 50
  counter_enable: false
  hwflow_enable: false
  counter_query_interval: 30
  hwflow_query_interval: 30
  alarm_enable: false
  alarm_high_temp: 176
  alarm_low_temp: 32
  last_alarm_id: 0
```

Counter Information

There are 4 types of counters: flow, group, meter, and port counter. As expected, the flow counter is the counter of each flow and the port counter is the counter of each port. The meter counter is the counter of each meter, associated with the band counter, and the group counter is the counter of each group, associated with the bucket counter.

 Note: this kind of information generates a lot of data. Turning this on will spend some CPU and memory. So if this information is not needed, please turn it off.

Administrator could enable/disable this function by command "**ovs-invctl enable-counter {true | false}**"

```
root@PicOS-OVS$ovs-invctl enable-counter true
```

Administrator could control the refresh interval by command "**ovs-invctl set-counter-interval**".

```
root@PicOS-OVS$ovs-invctl enable-counter trueroot@PicOS-OVS$ovs-invctl show
23aa0580-9578-471a-abae-a9c84b73affd
  sfp_enable: true
  sfp_query_interval: 50
  counter_enable: true
  hwflow_enable: false
  counter_query_interval: 20
  hwflow_query_interval: 30
  alarm_enable: false
  alarm_high_temp: 176
  alarm_low_temp: 32
  last_alarm_id: 0
```

The Administrator could show this information by command "**ovs-invctl show-port/meter/flow/group-counter**" or "**ovs-invctl show-all-counter**".

```
root@PicOS-OVS$ovs-invctl show-all-counter
Switch PicOS-OVS
  Bridge "br0"
    Port "Bridge br0 Local Port"
      statistics: {collisions=0, duration_nsec=3220239464, duration_sec=1208147968,
      multicast=0, rx_bytes=368, rx_crc_errors=0, rx_dropped=0, rx_errors=0, rx_fifo_errors=0,
      rx_frame_errors=0, rx_length_errors=0, rx_missed_errors=0, rx_over_errors=0, rx_packets=4,
      tx_aborted_errors=0, tx_bytes=0, tx_carrier_errors=0, tx_dropped=0, tx_errors=0,
      tx_fifo_errors=0, tx_heartbeat_errors=0, tx_packets=0, tx_window_errors=0}
    Port "Bridge br0 Port 50"
      statistics: {collisions=0, duration_nsec=3220239464, duration_sec=1208147968,
      multicast=0, rx_bytes=0, rx_crc_errors=0, rx_dropped=0, rx_errors=0, rx_fifo_errors=0,
```

```

rx_frame_errors=0, rx_length_errors=0, rx_missed_errors=0, rx_over_errors=0, rx_packets=0,
tx_aborted_errors=0, tx_bytes=0, tx_carrier_errors=0, tx_dropped=0, tx_errors=0,
tx_fifo_errors=0, tx_heartbeat_errors=0, tx_packets=0, tx_window_errors=0}
    Port "Bridge br0 Port 49"
        statistics: {collisions=0, duration_nsec=3220239464, duration_sec=1208147968,
multicast=0, rx_bytes=0, rx_crc_errors=0, rx_dropped=0, rx_errors=0, rx_fifo_errors=0,
rx_frame_errors=0, rx_length_errors=0, rx_missed_errors=0, rx_over_errors=0, rx_packets=0,
tx_aborted_errors=0, tx_bytes=0, tx_carrier_errors=0, tx_dropped=0, tx_errors=0,
tx_fifo_errors=0, tx_heartbeat_errors=0, tx_packets=0, tx_window_errors=0}

root@PicOS-OVS$ovs-invctl show-port-counter
Switch : PicOS-OVS
    Bridge : br0
        Port "Bridge br0 Local Port"
            statistics: {collisions=0, duration_nsec=3220239464, duration_sec=1208147968,
multicast=0, rx_bytes=368, rx_crc_errors=0, rx_dropped=0, rx_errors=0, rx_fifo_errors=0,
rx_frame_errors=0, rx_length_errors=0, rx_missed_errors=0, rx_over_errors=0, rx_packets=4,
tx_aborted_errors=0, tx_bytes=0, tx_carrier_errors=0, tx_dropped=0, tx_errors=0,
tx_fifo_errors=0, tx_heartbeat_errors=0, tx_packets=0, tx_window_errors=0}
        Port "Bridge br0 Port 49"
            statistics: {collisions=0, duration_nsec=3220239464, duration_sec=1208147968,
multicast=0, rx_bytes=0, rx_crc_errors=0, rx_dropped=0, rx_errors=0, rx_fifo_errors=0,
rx_frame_errors=0, rx_length_errors=0, rx_missed_errors=0, rx_over_errors=0, rx_packets=0,
tx_aborted_errors=0, tx_bytes=0, tx_carrier_errors=0, tx_dropped=0, tx_errors=0,
tx_fifo_errors=0, tx_heartbeat_errors=0, tx_packets=0, tx_window_errors=0}
        Port "Bridge br0 Port 50"
            statistics: {collisions=0, duration_nsec=3220239464, duration_sec=1208147968,
multicast=0, rx_bytes=0, rx_crc_errors=0, rx_dropped=0, rx_errors=0, rx_fifo_errors=0,
rx_frame_errors=0, rx_length_errors=0, rx_missed_errors=0, rx_over_errors=0, rx_packets=0,
tx_aborted_errors=0, tx_bytes=0, tx_carrier_errors=0, tx_dropped=0, tx_errors=0,
tx_fifo_errors=0, tx_heartbeat_errors=0, tx_packets=0, tx_window_errors=0}Alarm information

```

Alarm Information

Alarm stores the alarm information about: PSU, FAN, OPTICS, PORT, TEMP, MANAGEMENT_PORT_STATUE. This information is stored in the Alarm table.

Administrator could enable/disable this function by command "ovs-invctl enable-alarm {true | false}".

```

root@PicOS-OVS$ovs-invctl enable-alarm true

root@PicOS-OVS$ovs-invctl show
23aa0580-9578-471a-abae-a9c84b73affd
    sfp_enable: true
    sfp_query_interval: 50
    counter_enable: true
    hwflow_enable: false
    counter_query_interval: 20
    hwflow_query_interval: 30
    alarm_enable: true
    alarm_high_temp: 176
    alarm_low_temp: 32
    last_alarm_id: 9

```

The Administrator could control the temperature threshold by the command "**ovs-invctl set-high/low-temp**". When the switch temp id is higher or lower then the temperature threshold, it will produce an alarm message.

```

root@PicOS-OVS$ovs-invctl set-low-temp 10
root@PicOS-OVS$ovs-invctl set-high-temp 133

root@PicOS-OVS$ovs-invctl show

```

```
23aa0580-9578-471a-abae-a9c84b73affd
  sfp_enable: true
  sfp_query_interval: 50
  counter_enable: true
  hwflow_enable: false
  counter_query_interval: 20
  hwflow_query_interval: 30
  alarm_enable: true
  alarm_high_temp: 133
  alarm_low_temp: 10
  last_alarm_id: 9
```

The Administrator could show this information by the command "**ovs-invctl show-alarm**"

```
root@PicOS-OVS$ovs-invctl show-alarm
Alarm 1
  severity_code: cleared
  switch_name: PicOS-OVS
  entity_text: "te-1/1/50"
  text: "Optics inserted"
  time: "Tue Nov  3 13:49:28 2015"
Alarm 2
  severity_code: cleared
  switch_name: PicOS-OVS
  entity_text: "te-1/1/49"
  text: "Optics inserted"
  time: "Tue Nov  3 13:49:28 2015"
Alarm 3
  severity_code: critical
  switch_name: PicOS-OVS
  entity_text: "power 1"
  text: "power removed"
  time: "Tue Nov  3 13:49:39 2015"
Alarm 4
  severity_code: critical
  switch_name: PicOS-OVS
  entity_text: "PSU 1"
  text: "PSU removed"
  time: "Tue Nov  3 13:49:39 2015"
```

Hardware Flow Information

i Note: This kind of information is very large, turning this feature on will spend some CPU and memory. So if this information is not needed, please turn it off. For saving memory, we support max 20,000 hardware flows written into the database. If the count of hardware flows is exceeded, they are discarded.

Administrator could enable/disable this function by the command "**ovs-invctl enable-hwflow {true | false}**".

```
root@PicOS-OVS$ovs-invctl enable-hwflow true

root@PicOS-OVS$ovs-invctl show
23aa0580-9578-471a-abae-a9c84b73affd
  sfp_enable: true
  sfp_query_interval: 50
  counter_enable: true
  hwflow_enable: true
  counter_query_interval: 20
  hwflow_query_interval: 30
```

```

alarm_enable: true
alarm_high_temp: 133
alarm_low_temp: 10
last_alarm_id: 9

```

The Administrator could control the refresh interval by command "**ovs-invctl set-hwflow-interval INTEGER**".

```

root@PicOS-OVS$ovs-invctl enable-hwflow true

root@PicOS-OVS$ovs-invctl set-hwflow-interval 50
root@PicOS-OVS$ovs-invctl show
23aa0580-9578-471a-abae-a9c84b73affd
  sfp_enable: true
  sfp_query_interval: 50
  counter_enable: true
  hwflow_enable: true
  counter_query_interval: 20
  hwflow_query_interval: 50
  alarm_enable: true
  alarm_high_temp: 133
  alarm_low_temp: 10
  last_alarm_id: 9

```

The Administrator could get hardware flow information by the command "**ovs-invctl list Hardware_Flow**"

Others Hardware_Flow commands are available by using the command "**root@PicOS-OVS\$ovs-invctl --help**"

```

root@PicOS-OVS$ovs-invctl list Hardware_Flow
_uuid          : e6e61c9d-df38-43c6-abc0-2b0654af9713
hwflow         : "[\"#0 normal_d permanent priority=0,recirc_id=0, actions:drop\",
\"#1 normal permanent
recirc_id=0,tcp,in_port=49,d1_src=22:11:11:11:11:11,d1_dst=22:00:00:00:00:00,nw_src=1.1.2.1
00,nw_dst=2.2.2.100, actions:50\"]"
update_time    : "Tue Nov  3 15:11:19 2015"

```

Show Power/Fan Information

```

root@PicOS-OVS$ovs-invctl list power-status
_uuid          : 19af5a8c-9adc-4d92-8615-0e4a50aadabd
good_12v        : "false"
led             :
number          : 1
power_alert     :
present         : "false"

_uuid          : 95b3fa12-8f6e-4709-bf0a-363a9fda0c2f
good_12v        : "true"
led             :
number          : 2
power_alert     :
present         : "true"

root@PicOS-OVS$ovs-invctl list fan_status
_uuid : 43d903ec-7894-4df7-833b-5d5a1004b388
direction : "fan direction from front to back"
fault : "false"
input : " 11400 rpm"

```

```
number : 3
pwm : " 50 %"
```

 Currently only P-5712 supports Fan_status.

Configuring option-match-vlan-type

PicOS supports enable/disable option-match-mode-type from version 2.6.5 in Platform P-3922/3924. This function is used to enable or disable matching untagged packets.

Example:

```
admin@PicOS-OVS$ovs-vsctl set-option-match-vlan-type TRUE
Please reboot for the change to take effect!
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-vsctl set-option-match-vlan-type FALSE
Please reboot for the change to take effect!
admin@PicOS-OVS$
```

If the type is false, PicOS cannot configure one flow entry only matching untagged packets, and cannot configure one flow entry match all tagged packets (tag from1 to 4094). Once option-match-vlan-type is enabled, the flow entry `vlan_tci=0x0000/0x1000` can match untagged packets, and flow entry `vlan_tci=0x1000/0x1000` can match all tagged packets.

Example:

```
admin@PicOS-OVS$ovs-vsctl set-option-match-vlan-type TRUE
Please reboot for the change to take effect!
admin@PicOS-OVS$

admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=11,vlan_tci=0x0000/0x1000,actions=output:12
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=12,vlan_tci=0x1000/0x1000,actions=output:11
```

Send untagged packets to port te-1/1/11, the packets can match the first flow entry and forward on port te-1/1/12. Send tagged packets to port te-1/1/11, the packets cannot match the first flow entry;

Send untagged packets to port te-1/1/12, the packets cannot match the second flow entry; Send tagged (1~4094) packets to port te-1/1/12, the packets can match the second flow entry and forward on port te-1/1/12.

Connecting to Manager

PicOS OVS supports connection to Manager.

Command to set manager as below,

```
admin@PicOS-OVS$ovs-vsctl set-manager ptcp:6640:10.10.51.49
or
admin@PicOS-OVS$ovs-vsctl set-manager tcp:10.10.51.49:6640
```

Before PicOS 2.6.5, manager is not supported by default. If user wants to connect manager, ovsdb-server must be restarted.

```
Kill following ovsdb-server:  
ovsdb-server /ovs/ovs-vswitchd.conf.db /ovs/inventory.conf.db --pidfile  
--remote=ptcp:6640:10.10.51.138 --remote=punix:/ovs/var/run/openvswitch/db.sock
```

```
Then run ovsdb-server with --remote=db:Open_vSwitch,Manager,target:  
ovsdb-server /ovs/ovs-vswitchd.conf.db /ovs/inventory.conf.db --pidfile  
--remote=ptcp:6640:10.10.51.138 --remote=punix:/ovs/var/run/openvswitch/db.sock  
--remote=db:Open_vSwitch,Manager,target
```

From PicOS 2.6.5, manager supported in ovsdb-server. Once user configures “ovs-vsctl set-manager tcp:10.10.51.49:6640”, user can connect manager by remote.

```
root@PicOS-OVS$ps aux | grep ovsdb-server  
root      9598  0.1  0.1    7256  2816 pts/0      S      18:00   0:00 ovsdb-server  
/ovs/ovs-vswitchd.conf.db /tmp/inventory.conf.db /ovs/function.conf.db --pidfile  
--remote=ptcp:6640:127.0.0.1 --remote=punix:/ovs/var/run/openvswitch/db.sock  
--remote=db:Open_vSwitch,Manager,target
```

Before PicOS 2.7.1, when using command “ovs-vsctl set-manager tcp:10.10.51.49:6640” to connect manager, the manager’s “is_connected” is false, such like:

```
root@PicOS-OVS$ovs-vsctl list manager  
_uuid          : 3cd2a235-6921-4580-a093-a3dd96431cc4  
connection_mode : []  
external_ids   : {}  
inactivity_probe : []  
is_connected   : false  
max_backoff    : []  
other_config    : {}  
status          : {}  
target          : "tcp:10.10.51.49:6640"
```

The behavior depends on the “remote” configuration.

If the “remote” is “db: Open_vSwitch,Manager, target”, the manager status will not be refreshed.

```
root@PicOS-OVS$ps aux | grep ovsdb-server  
root      9598  0.1  0.1    7256  2816 pts/0      S      18:00   0:00 ovsdb-server  
/ovs/ovs-vswitchd.conf.db /tmp/inventory.conf.db /ovs/function.conf.db --pidfile  
--remote=ptcp:6640:127.0.0.1 --remote=punix:/ovs/var/run/openvswitch/db.sock  
--remote=db:Open_vSwitch,Manager,target
```

If the “remote” is “db: Open_vSwitch, Open_vSwitch, manager_options”, the manager status will be displayed right.

We can change the boot argument:

```
ovsdb-server /ovs/ovs-vswitchd.conf.db /tmp/inventory.conf.db /ovs/function.conf.db  
--pidfile --remote=ptcp:6640:127.0.0.1 --remote=punix:/ovs/var/run/openvswitch/db.sock  
--remote=db:Open_vSwitch,Open_vSwitch,manager_options
```

Or we can use command to add this “remote”:

```
root@PicOS-OVS$ovs-appctl -t ovsdb-server ovsdb-server/add-remote  
db:Open_vSwitch,Open_vSwitch,manager_options
```

```
root@PicOS-OVS$ovs-vsctl list manager
_uuid          : 3cd2a235-6921-4580-a093-a3dd96431cc4
connection_mode : []
external_ids   : {}
inactivity_probe : []
is_connected   : true
max_backoff    : []
other_config   : {}
status         : {sec_since_connect="6", state=ACTIVE}
target         : "tcp:10.10.51.49:6640"
```

We can delete manager using command:

```
admin@PicOS-OVS$ovs-vsctl del-manager
```

From PicOS 2.7.1, user can list manager status correctly without re-configure ovsdb-server.

OVS LFS

Abstract

Link fault signaling operates between the remote RS and the local RS. Faults detected between the remote RS and the local RS are received by the local RS as Local Fault. Only an RS originates Remote Fault signals.

Sub-layers within the PHY are capable of detecting faults that render a link unreliable for communication. Upon recognition of a fault condition, a PHY sub-layer indicates Local Fault status on the data path. When this Local Fault status reaches an RS, the RS stops sending MAC data, and continuously generates a Remote Fault status on the transmit data path (possibly truncating a MAC frame being transmitted). When Remote Fault status is received by an RS, the RS stops sending MAC data, and continuously generates Idle control characters. When the RS no longer receives fault status messages, it returns to normal operation, sending MAC data.

The RS reports the fault status of the link. Local Fault indicates a fault detected on the receive data path between the remote RS and the local RS. Remote Fault indicates a fault on the transmit path between the local RS and the remote RS.

The fault status is as follows:

a) link_fault = OK

The RS shall send MAC frames as requested through the PLS service interface. In the absence of MAC frames, the RS shall generate Idle control characters.

b) link_fault = Local Fault

The RS shall continuously generate Remote Fault Sequence ordered_sets.

c) link_fault = Remote Fault

The RS shall continuously generate Idle control characters.

Link Fault Signaling

If ignore local fault is set as false. When link local fault is triggered, the RS shall continuously generate Remote Fault Sequence ordered_sets. Otherwise, the RS will not generate Remote Fault Sequence ordered_sets.

If ignore remote fault is set as false. When link remote fault is received, The RS shall continuously generate Idle control characters. Other, The RS shall send MAC frames as requested through the PLS service interface and generate Idle control characters in the absence of MAC frames.

LFS Commands

The following is the configuration command as sample:

```
ovs-vsctl set Interface te-1/1/1 options:link-fault-signaling=ignore-none
ovs-vsctl set Interface te-1/1/1 options:link-fault-signaling=ignore-local
ovs-vsctl set Interface te-1/1/1 options:link-fault-signaling=ignore-remote
ovs-vsctl set Interface te-1/1/1 options:link-fault-signaling=ignore-both
```

1. "ignore-local" means ignoring local signaling fault.
2. "ignore-remote" means ignoring local remote fault.
3. "ignore-both" means ignoring both local and remote signaling fault.
4. "ignore-none" means not ignoring neither local signaling fault nor remote signaling fault.

Up Mode

The force up command forcibly brings up a fiber Ethernet port and enables the port to forward packets uni-directionally over a single link. In this way, transmission links are well utilized.

Up mode commands.

The following is the configuration command as a sample:

```
ovs-vsctl set Interface te-1/1/1 options:up-mode=true
ovs-vsctl set Interface te-1/1/1 options:up-mode=false
```

- (i) 1. Disable port command has a higher priority than set port up command. If one port is disabled manually, it is not effective to set it up forcibly.
- 2. Up-mode true command should be configured together with ignore-local-fault true command. If user only configures up-mode true and not ignore-local-fault command, traffic can't transmit from TX link
- 3. LFS command must configurate on more than 10G port.

Configure TPID in port

PicOS support push/swap/pop S-VLAN in Q-in-Q, and the S-VLAN TPID can configure as 0x8100/0x88a8/0x9100 and 0x9200. But in the same port, PicOS only support configure one TPID value, that means the traffic through one port will can push/swap/pop one type of TPID.

- a, If one TPID configured in ingress port, switch can recognize it and deal the matched traffic;
- b, If one TPID configured in egress port, switch can design it in the traffic which send out from the port.

1, TPID used in Q-in-Q



If the actions include push-vlan and has no push_l2mpls:
 push_vlan action will use out_port's tpid.
 For example,
 a. untagged packet push one tag, the result will be: out_port's tpid | payload
 b. untagged packet push two tags, the result will be: out_port's tpid | 0x8100 | payload

Once configure TPID in port, need reboot switch.Command:

```
ovs-vsctl set interface <port> options:tpid=<value>
<value> can configure one value as 0x8100/0x88a8/0x9100/0x9200, for example if the S-VLAN
TPID is 0x88a8, the value must set 0x88a8.
By default, the value is 0x8100.

admin@PicOS-OVS$ sudo service picos restart
```

In Q-in-Q flow, only support match outer-vlan.

Push-vlan: Design TPID during push S-VLAN, need configure TPID in egress.

Example:

```
admin@PicOS-OVS$ ovs-vsctl set interface te-1/1/1 options:tpid=0x8100
admin@PicOS-OVS$ ovs-vsctl set interface te-1/1/2 options:tpid=0x88a8

Push one vlan:
admin@PicOS-OVS$ ovs-ofctl add-flow br0
in_port=1,actions=push_vlan=0x88a8,set_field:1234->vlan_vid,output:2

Push two vlans,
admin@PicOS-OVS$ ovs-ofctl add-flow br0
in_port=1,actions=push_vlan=0x8100,set_field:10->vlan_vid,push_vlan=0x88a8,set_field:100->vlan_vid,output:2
```

Swap-vlan: Match TPID in ingress port and design TPID in egress port, so need configure TPID in ingress and egress.

Example:

```
admin@PicOS-OVS$ ovs-vsctl set interface te-1/1/1 options:tpid=0x88a8
admin@PicOS-OVS$ ovs-vsctl set interface te-1/1/2 options:tpid=0x88a8
```

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1,dl_vlan=10,actions=set_field:100->vlan_vid,output:2
```

Pop-vlan: Match TPID in ingress port, need configure TPID in ingress port.

Example:

```
admin@PicOS-OVS$ovs-vsctl set interface te-1/1/1 options:tpid=0x88a8
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=1,dl_vlan=10,actions=pop_vlan,output:2
```

2, TPID used in push L2MPLS

This feature means push/swap/pop vlan before push L2MPLS. It is simmilar with Q-in-Q, but operation is different from Q-in-Q.



- 1, This feature only support in platform 3922 and 3924;
- 2, If user want use TPID before push L2MPLS, must configure TPID on ingress port.
- 3, push_vlan action before push_l2mpls will use in_port's tpid, and push_vlan action in push_l2mpls will use out_port's tpid. One case need to note is that if only one tag exist before l2mpls, its tpid should be 0x8100.
- For example,
- a. untagged packet push one tag and then push l2mpls, the result will be: l2mpls | 0x8100 | payload
- b. untagged packet push two tags and then push l2mpls, the result will be: l2mpls | in_port's tpid | 0x8100 | payload

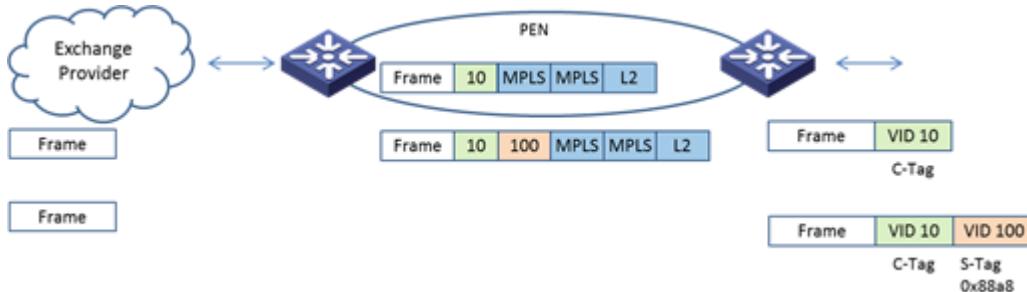
a, No matter push/swap/pop vlan before push L2MPLS, TPID must configure in ingress port.

Once configure TPID in port, need reboot switch.Command:

```
ovs-vsctl set interface <port> options:tpid=<value>
<value> can configure one value as 0x8100/0x88a8/0x9100/0x9200, for example if the S-VLAN
TPID is 0x88a8, the value must set 0x88a8.
By default, the value is 0x8100.
```

Push-vlan: Configure TPID in ingress port.

Example: The push_vlan and push_l2mpls are performed in network ingress side. In network egress side, PicOS switch just pop_l2mpls. In this example, we only care the network ingress side.



a, At the Ingress side, matching in_port and untag, actions: push_svlan (TPID 0x8100/0x88a8/0x9100/0x9200) push_cvlan(TPID 0x8100) then push l2mpls.

```
ovs-vsctl set interface te-1/1/11 options:tpid=0x88a8
ovs-ofctl add-flow br-iris
"in_port=11,vlan_vid=0x0000/0xffff,actions=push_vlan:0x88a8, set_field:100->vlan_vid,push_vl
```

```
an:0x8100, set_field:10->inner_vlan_vid, push_l2mpls:0x8847, set_field:22:11:11:11:11:10->dl_dst, set_field:22:00:00:00:00:01->dl_src, set_field:16->mpls_label, push_vlan:0x88a8, set_field:1000->vlan_vid, output:12"
```

b, At the Ingress side, cvlan (TPID=0x8100) then push_l2mpls.

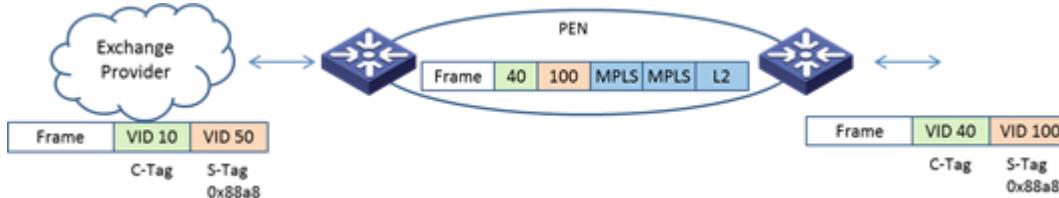
```
ovs-vsctl set interface te-1/1/11 options:tpid=0x88a8
ovs-ofctl add-flow br-iris
"in_port=11,vlan_vid=0x0000/0x1fff,actions=push_vlan:0x8100, set_field:10->vlan_vid, push_l2mpls:0x8847, set_field:22:11:11:11:11:10->dl_dst, set_field:22:00:00:00:00:01->dl_src, set_field:16->mpls_label, push_vlan:0x88a8, set_field:1000->vlan_vid, output:12"
```

⚠ If want match untag packets, the flow entry should include match vlan_vid=0x0000/0x1fff, and must set option-match-vlan-type is true.

```
admin@PicOS-OVS$ovs-vsctl set-option-match-vlan-type true
```

Swap-vlan: Configure TPID in ingress port.

Example: Swap and push_l2mpls are performed in network ingress side. In egress side, PicOS switch just pop_l2mpls. In this example, we only care the network ingress side.



a, At the **Ingress** side, matching in_port and S-VLAN/C-VLAN (C-VLAN TPID support 0x8100, S-VLAN TPID support 0x8100/0x88a8/0x9100/0x9200), actions: swap S-VLAN and C-VLAN;

```
ovs-vsctl set interface te-1/1/11 options:tpid=0x88a8
ovs-ofctl add-flow br-iris
"in_port=11,vlan_tci=0x1032,inner_vlan_tci=0x100a,actions=set_field:100->vlan_vid, set_field:40->inner_vlan_vid, push_l2mpls:0x8847, set_field:22:11:11:11:11:10->dl_dst, set_field:22:00:00:00:00:01->dl_src, set_field:16->mpls_label, push_vlan:0x8100, set_field:1000->vlan_vid, output:12"
```

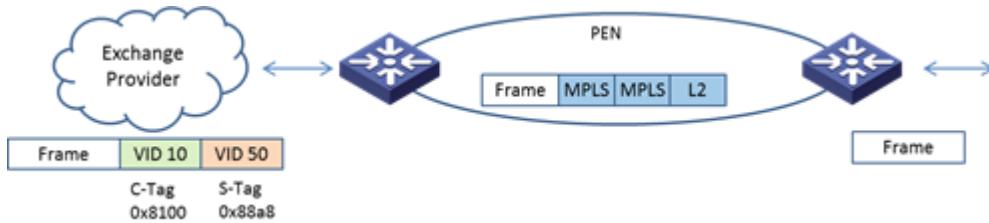
b, swap S-VLAN only or swap C-VLAN , then push_l2mpls.

```
ovs-vsctl set interface te-1/1/11 options:tpid=0x88a8
ovs-ofctl add-flow br-iris
"in_port=11,vlan_tci=0x1032,inner_vlan_tci=0x1000/0x1000,actions=set_field:100->vlan_vid, push_l2mpls:0x8847, set_field:22:11:11:11:10->dl_dst, set_field:22:00:00:00:00:01->dl_src, set_field:16->mpls_label, push_vlan:0x8100, set_field:1000->vlan_vid, output:12"

ovs-ofctl add-flow br-iris
"in_port=11,vlan_tci=0x1000/0x1000,inner_vlan_tci=0x1000/0x1000,actions=set_field:100->inner_vlan_vid, push_l2mpls:0x8847, set_field:22:11:11:11:10->dl_dst, set_field:22:00:00:00:00:01->dl_src, set_field:16->mpls_label, push_vlan:0x8100, set_field:1000->vlan_vid, output:12"
```

Pop_vlan: Configure TPID in ingress port.

Example: The pop_vlan and push_l2mpls are performed in network ingress side. In egress side, PicOS switch just pop_l2mpls. In this example, we only care the network ingress side.



At the Ingress side, matching in_port and S-VLAN/C-VLAN, actions: pop S-VLAN or pop S-VLAN and C-VLAN, then push_l2mpls.

```
ovs-vsctl set interface te-1/1/11 options:tpid=0x88a8
ovs-ofctl add-flow br-iris
"in_port=11,vlan_tci=0x1032,inner_vlan_tci=0x100a,actions=pop_vlan,pop_vlan,push_l2mpls:0x847,set_field:22:11:11:11:10->dl_dst,set_field:22:00:00:00:01->dl_src,set_field:16->mpls_label,push_vlan:0x8100,set_field:1000->vlan_vid,output:12"
```

Broadcom Chip Limitation in OVS

Limitation 1

Due to VCAP limitation, ARP flow entry will generate an error once combined-mode is enabled.

When combined mode is enabled, VCAP needs to match the same fields as ICAP. ARP flows use UDF in ICAP, but not all HW platforms support UDE in VCAP.

Example:

```
ovs-vsctl set-combined-mode true
admin@PicOS-OVS$ovs-ofctl dump-flows br2
OFPST_FLOW reply (OF1.4) (xid=0x2):
  cookie=0x0, duration=70.192s, table=0, n_packets=n/a, n_bytes=0, priority=500,arp
  actions=output:2
  cookie=0x0, duration=6.928s, table=0, n_packets=n/a, n_bytes=0, priority=200,in_port=7
  actions=push_vlan:0x8100,set_field:2001->vlan_vid,output:8
```

Actual result:

sent packets match both flows to ge-1/1/7,ge-1/1/2 will receive packets that with "push_vlan:0x8100,set_field:2001->vlan_vid", ge-1/1/8 will not receive any packets.

Goto_table

PicOS OVS supports goto_table on all platforms. When user dump-tables, tables 0 to 253 can be seen, but PicOS OVS only supports table 0. Also, flows will be merged as a flow to hardware when configuring flows with different tables.

Example 1. add flow entries with one goto action.

1) Add bridge

```
admin@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
```

2) Add ports to br0

```
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/13 vlan_mode=trunk tag=1 -- set interface
ge-1/1/13 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/15 vlan_mode=trunk tag=1 -- set interface
ge-1/1/15 type=pica8
admin@PicOS-OVS$ovs-vsctl add-port br0 ge-1/1/17 vlan_mode=trunk tag=1 -- set interface
ge-1/1/17 type=pica8
```

3) Add flows with actions=goto_table:

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 priority=200,in_port=13,actions=goto_table:1
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=1,priority=101,in_port=13,ip,nw_src=10.10.10.11,actions=15
```

After adding flows, check software table and hardware table:

```
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
 flow_id=10, cookie=0x0, duration=18.552s, table=0, n_packets=n/a, n_bytes=0,
priority=200,in_port=13 actions=goto_table:1
 flow_id=11, cookie=0x0, duration=4.648s, table=1, n_packets=n/a, n_bytes=0,
priority=101,ip,in_port=13,nw_src=10.10.10.11 actions=output:15
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#3 normal permanent flow_id=10 priority=200,ip,in_port=13,nw_src=10.10.10.11, actions:15
#0 normal_d permanent flow_id=2 priority=0, actions:drop
Total 2 flows in HW.
```

From the example above, two flow entries were merged as one flow entry in the hardware table, and we only use table=0. The priority of flow entry in hardware is decided by flow in table=0.

Example 2. add flow entries with two goto action.

Add flows:

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 priority=200,in_port=13,actions=goto_table:1
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=1,priority=101,in_port=13,ip,nw_src=10.10.10.11,actions=15
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=1,priority=102,ip,nw_src=10.10.10.12,actions=goto_table:2
admin@PicOS-OVS$ovs-ofctl add-flow br0
table=2,priority=123,d1_dst=00:11:22:33:44:55,actions=17
admin@PicOS-OVS$
```

After adding flows, check software table and hardware table:

```
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
 flow_id=6, cookie=0x0, duration=43.530s, table=0, n_packets=n/a, n_bytes=0,
priority=200,in_port=13 actions=goto_table:1
 flow_id=8, cookie=0x0, duration=15.120s, table=1, n_packets=n/a, n_bytes=0,
priority=102,ip,nw_src=10.10.10.12 actions=goto_table:2
 flow_id=7, cookie=0x0, duration=25.110s, table=1, n_packets=n/a, n_bytes=0,
priority=101,ip,in_port=13,nw_src=10.10.10.11 actions=output:15
 flow_id=9, cookie=0x0, duration=4.330s, table=2, n_packets=n/a, n_bytes=0,
priority=123,d1_dst=00:11:22:33:44:55 actions=output:17
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#0 normal_d permanent flow_id=2 priority=0, actions:drop
#1 normal permanent flow_id=6 priority=200,ip,in_port=13,nw_src=10.10.10.11, actions:15
#2 normal permanent flow_id=6
```

```
priority=201,ip,in_port=13,dl_dst=00:11:22:33:44:55,nw_src=10.10.10.12, actions:17
Total 3 flows in HW.
admin@PicOS-OVS$
```

This example shows that four flow entries in table0 and table1 have goto action, so these flows are merged as two flows in hardware.

Clear counter

Clear port counter

Command: ovs-appctl bridge/clear-counts br0

User can use above command to clear the port counter.

Example:

```
Check port counter:
ovs-ofctl dump-ports br0

Clear the port counter:
ovs-appctl bridge/clear-counts br0
```

Clear flow counter

Command: ovs-ofctl mod-flows br0 reset_counts,<match>,<actions>

User can use the command to clear flow counter.

Example:

```
ovs-ofctl add-flow br0 in_port=3,actions=4
ovs-ofctl mod-flows br0 reset_counts,in_port=3,actions=4
```

Ovs CLI Enhancements

PicOS ovs cli has some enhancements from version 2.7.1.

List ovs Configuration

PicOS ovs added a new command to display the current configuration by traversing the ovsdb tables.

The command is:

ovs-vsctl show-running-config

```
admin@PicOS-OVS$ovs-vsctl show-running-config
Open_vSwitch c645ee8a-34d5-4c64-a3dc-0c1a20f3c26e
  Bridge "br0"
    datapath_id: "1c48cc37ab254bc1"
    datapath_type: "pica8"
    Port "ge-1/1/19"
```

```

Interface "ge-1/1/19"
    type: "pica8"
    tag: 1
    vlan_mode: trunk
Port "br0"
    Interface "br0"
        mtu: 1500
        type: internal
Pica8 ecafe6f4-97ac-407b-b4df-c871b5cd9561
    hardware_type: "as4610_54t"
admin@PicOS-OVS$
```

List System Resources Usage

Adds two commands to display current system resources usage and interfaces.

When we show system resources usage, the current mode is cared, such as match-mode, udf-mode, L2-mode, L3-mode, combinante-mode and egress-mode.

The commands are:

ovs-appctl pica/show tables

ovs-appctl pica/show interfaces

```

admin@PicOS-OVS$ovs-appctl pica/show tables
Pica Tables Statistics:
  Pica Tables          Max Limitation      Current Used
  -----
  ICAP Table           2046                  9
  ECAP Table           (null)                (null)
  VCAP Table           1024                  0
  L2 System Table     (null)                (null)
  L2 FDB Table         (null)                (null)
  L3 Host Table        (null)                (null)
  L3 Route Table       (null)                (null)
  UDF Table            (null)                (null)
admin@PicOS-OVS$  

admin@PicOS-OVS$  

admin@PicOS-OVS$ovs-appctl pica/show interfaces
Valid Interfaces On Switch P3290:
Physical interfaces:
ge-1/1/1(1) ge-1/1/2(2) ge-1/1/3(3) ge-1/1/4(4)
ge-1/1/5(5) ge-1/1/6(6) ge-1/1/7(7) ge-1/1/8(8)
ge-1/1/9(9) ge-1/1/10(10) ge-1/1/11(11) ge-1/1/12(12)
ge-1/1/13(13) ge-1/1/14(14) ge-1/1/15(15) ge-1/1/16(16)
ge-1/1/17(17) ge-1/1/18(18) ge-1/1/19(19) ge-1/1/20(20)
ge-1/1/21(21) ge-1/1/22(22) ge-1/1/23(23) ge-1/1/24(24)
ge-1/1/25(25) ge-1/1/26(26) ge-1/1/27(27) ge-1/1/28(28)
ge-1/1/29(29) ge-1/1/30(30) ge-1/1/31(31) ge-1/1/32(32)
ge-1/1/33(33) ge-1/1/34(34) ge-1/1/35(35) ge-1/1/36(36)
ge-1/1/37(37) ge-1/1/38(38) ge-1/1/39(39) ge-1/1/40(40)
ge-1/1/41(41) ge-1/1/42(42) ge-1/1/43(43) ge-1/1/44(44)
ge-1/1/45(45) ge-1/1/46(46) ge-1/1/47(47) ge-1/1/48(48)
te-1/1/49(49) te-1/1/50(50) te-1/1/51(51) te-1/1/52(52)
LAG interfaces: ae1(1025) - ae1023(2047)
Bond interfaces: bond1(2049) - bond1023(3071)
GRE interfaces: gre1(3073) - gre1023(4095)
VXLAN interfaces: vxlan1(4097) - vxlan1023(5119)
L2GRE interfaces: l2gre1(5121) - l2gre1023(6143)
admin@PicOS-OVS$
```

Associate sw-flow with hw-flow

When a new flow entry is added, a unique 64bit flow-id is assigned to it. The flow-id can be used by ovs-ofctl and ovs-appctl to filter the flow entries and associate sw-flow with hw-flow.

The commands are:

ovs-ofctl dump-flows br0 [flow_id=n]

ovs-appctl pica/dump-flows [flow_id=n]

Example 1:

```
root@PicOS-OVS$ovs-ofctl add-flow br0 priority=200,in_port=1,actions=2
root@PicOS-OVS$ovs-ofctl add-flow br0
priority=124,in_port=1,ip,nw_src=10.10.10.10,actions=4
root@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  cookie=0x0, duration=36.591s, flow_id=1, table=0, n_packets=n/a, n_bytes=0,
priority=200,in_port=1 actions=output:2
  cookie=0x0, duration=6.393s, flow_id=2, table=0, n_packets=n/a, n_bytes=0,
priority=124,ip,in_port=1,nw_src=10.10.10.10 actions=output:4
root@PicOS-OVS$ovs-appctl pica/dump-flows
#4 normal permanent flow_id=1, priority=124,recirc_id=0,ip,in_port=1,nw_src=10.10.10.10,
actions:2
#3 normal permanent flow_id=1, priority=200,recirc_id=0,in_port=1, actions:2
#0 normal_d permanent priority=0,recirc_id=0, actions:drop
Total 3 flows in HW.
```

From the above example, if two flow entries in table 0 don't have the exact same match and priority, but the match of sw-flow which has higher priority can cover another sw-flow's match, then two hw-flows with the same flow-id of sw-flow which has the higher priority will be installed.

Example 2:

```
root@PicOS-OVS$ovs-ofctl add-flow br0 priority=200,in_port=1,actions=goto_table:1
root@PicOS-OVS$ovs-ofctl add-flow br0
table=1,priority=101,in_port=1,ip,nw_src=10.10.10.11,actions=2
root@PicOS-OVS$ovs-ofctl add-flow br0
table=1,priority=102,ip,nw_src=10.10.10.12,actions=goto_table:2
root@PicOS-OVS$ovs-ofctl add-flow br0
table=2,priority=123,d1_dst=00:11:22:33:44:55,actions=3
root@PicOS-OVS$ 
root@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  cookie=0x0, duration=184.078s, flow_id=1, table=0, n_packets=n/a, n_bytes=0,
priority=200,in_port=1 actions=goto_table:1
  cookie=0x0, duration=49.800s, flow_id=2, table=1, n_packets=n/a, n_bytes=0,
priority=102,ip,nw_src=10.10.10.12 actions=goto_table:2
  cookie=0x0, duration=107.575s, flow_id=3, table=1, n_packets=n/a, n_bytes=0,
priority=101,ip,in_port=1,nw_src=10.10.10.11 actions=output:2
  cookie=0x0, duration=7.991s, flow_id=4, table=2, n_packets=n/a, n_bytes=0,
priority=123,d1_dst=00:11:22:33:44:55 actions=output:3
root@PicOS-OVS$ovs-appctl pica/dump-flows
#6 normal permanent flow_id=1,
priority=201,recirc_id=0,ip,in_port=1,d1_dst=00:11:22:33:44:55,nw_src=10.10.10.12,
actions:3
#5 normal permanent flow_id=1, priority=200,recirc_id=0,ip,in_port=1,nw_src=10.10.10.11,
actions:2
#0 normal_d permanent priority=0,recirc_id=0, actions:drop
Total 3 flows in HW.
```

If a flow entry in table 0 has goto action, then the flow-id will be applied to all hw-flows generated by it. Only the flow-id's of sw-flows in first table are cared now.

Display dpid in both hex and decimal

When the status of bridge is shown, we add decimal numbers for dpid to display.

```
admin@PicOS-OVS$ovs-ofctl show br0
OFPT_FEATURES_REPLY (OF1.4) (xid=0x2):
dpid:0x1c48cc37ab254bc1(2038103370851765185)
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS
OFPST_PORT_DESC reply (OF1.4) (xid=0x3):
 13(ge-1/1/13): addr:cc:37:ab:25:4b:c1
    config: 0
    state: LINK_UP
    current: 100MB-FD COPPER AUTO_NEG
    advertised: 10MB-FD 100MB-FD 1GB-FD COPPER AUTO_NEG
    supported: 10MB-FD 100MB-FD 1GB-FD COPPER AUTO_NEG
    peer: 10MB-HD 10MB-FD 100MB-HD 100MB-FD COPPER
    speed: 100 Mbps now, 1000 Mbps max
LOCAL(br0): addr:cc:37:ab:25:4b:c1
    config: 0
    state: LINK_UP
    current: 10MB-FD COPPER
    supported: 10MB-FD COPPER
    speed: 10 Mbps now, 10 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.4) (xid=0x5): frags=normal miss_send_len=0
```

List Interface Details

A command is added to display the details of a specific interface or all interfaces in the bridge. A corresponding openflow multipart message should be added by using multipart experimenter type, the controller can then get these statistics as well.

The command is:

ovs-ofctl dump-interfaces <bridge> [interface]

Example:

Add the following flow:

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=13,actions=15
```

No packets:

```
admin@PicOS-OVS$ovs-ofctl dump-interfaces br0 ge-1/1/15
PXST_INTERFACE_STATS reply (OF1.4) (xid=0x4): 1 interfaces
  ge-1/1/15(port 15):
    Traffic statistics:
      Input Packets.....0
      Output Packets.....0
      Input Octets.....0
      Output Octets.....0
```

Port 13 send unicast packets to match the flow and check:

```
admin@PicOS-OVS$ovs-ofctl dump-interfaces br0 ge-1/1/13
PXST_INTERFACE_STATS reply (OF1.4) (xid=0x4): 1 interfaces
ge-1/1/13(port 13):
Traffic statistics:
  Input Packets.....3013397
  Output Packets.....0
  Input Octets.....771429632
  Output Octets.....0
Transmit:
  Unicast packets.....0
  Multicast packets.....0
  Broadcast packets.....0
  Packets 64 Octets.....0
  Packets 65-127 Octets.....0
  Packets 128-255 Octets.....0
  Packets 256-511 Octets.....0
  Packets 512-1023 Octets.....0
  Packets 1024-1518 Octets.....0
  Oversize Packets.....0
  Total Packets Without Errors.....0
  Discarded Packets.....0
  Total Packets With Errors.....0
  Single Collision Frames.....0
  Multiple Collision Frames.....0
  Deferred Frames.....0
  Late Collisions.....0
  Excessive Collisions.....0
  Pause Frames.....0
Receive:
  Unicast packets.....3013393
  Multicast packets.....0
  Broadcast packets.....0
  Packets 64 Octets.....0
  Packets 65-127 Octets.....0
  Packets 128-255 Octets.....0
  Packets 256-511 Octets.....3013396
  Packets 512-1023 Octets.....0
  Packets 1024-1518 Octets.....0
  Oversize Packets.....0
  Total Packets Without Errors.....3013397
  Discarded Packets.....0
  Total Packets With Errors.....0
  Alignment Errors.....0
  FCS Errors.....0
  Collisions.....0
  Pause Frames.....0
```

Table Type Pattern

Before version 2.8.0, the ttp file named as "vz_etech_ttp_v0.3.json." And from version 2.8.0, the ttp json file changed the name to "routing_ttp_v1.0.json" The ttp json file is as following:

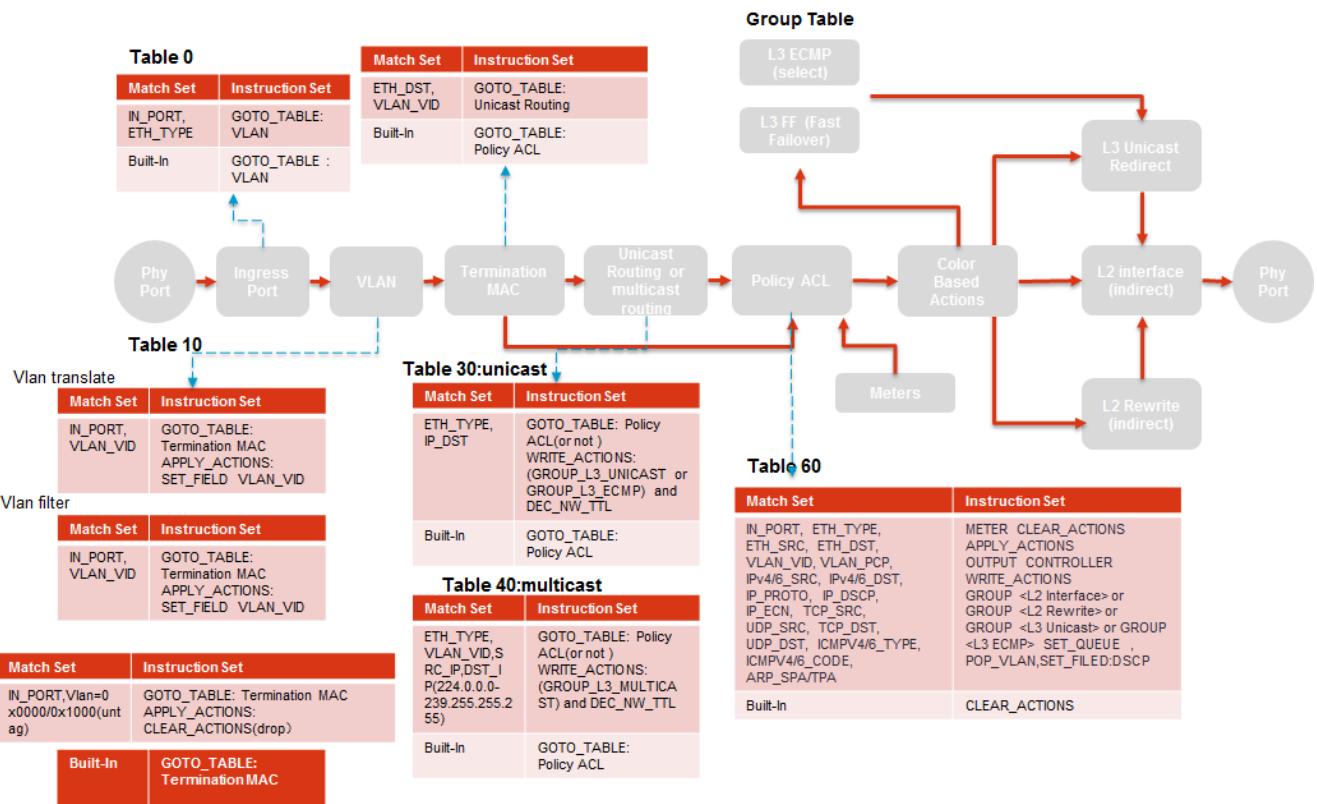


Figure 1:the json file analyse

- **TTP unicast**

- [TTP multicast](#)

TTP Multicast

We will include support for TTP multicast from version 2.8.0. The table number is 40 by default. The pipeline is as follows:

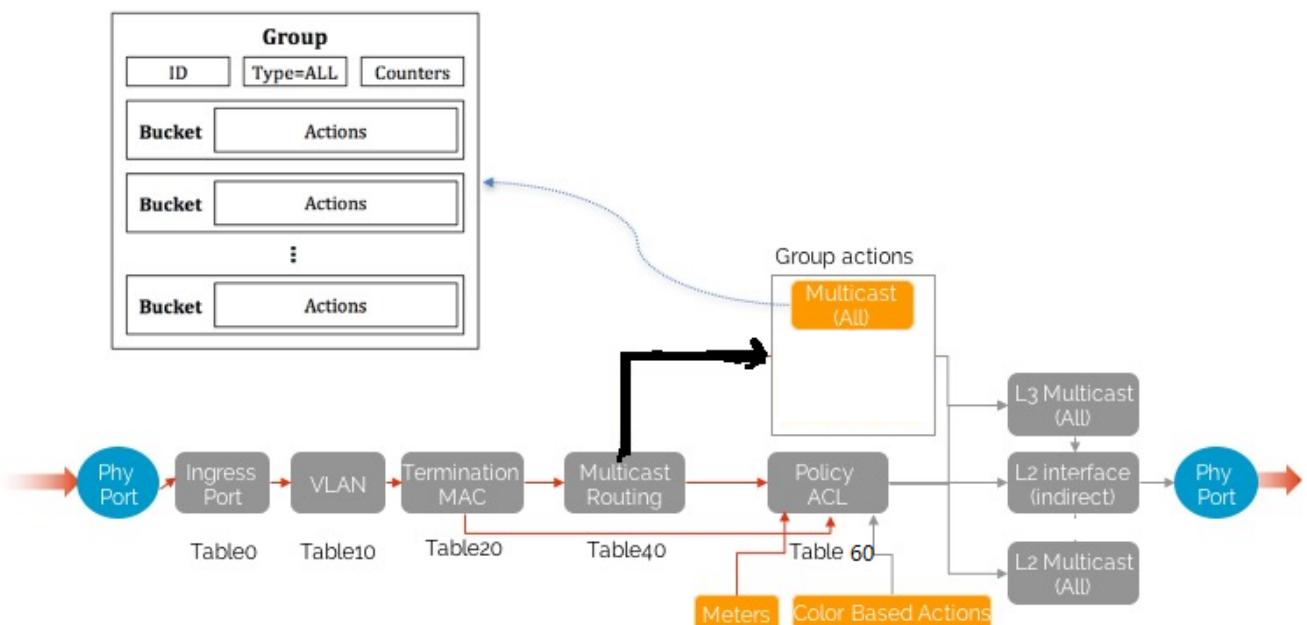


Fig 1: TTP multicast Pipeline

The process is different from adding a flow to a unicast routing table. If you add a flow to multicast routing table, you do not need to add the corresponding flows to the VLAN and terminal MAC tables. The switch will check whether the destination MAC of the packet is a multicast address or not. In case its a unicast MAC address, the switch will report an error message. The match fields are Ethernet type, vlan_vid, multicast destination IP address, source IP address. Source ip 0.0.0.0 means (*,g). And (s,g) has the higher priority.

By far, the policy ACL table has ecmp problems in processing the multicast packets coming from multicast routing table.

The multicast routing flow priority is 32768, the IPV4 multicast destination IP address ranges from 224.0.0.0 to 239.255.255.255. The IPV4 destination MAC must begin from 01:00:5e:0..., the IPV6 multicast destination IP address should begin with 0xFF..., the IPV6 multicast destination MAC begins with 33:33.....

i I2 multicast and I3 multicast

I2 multicast: the field "dec_nw_ttl" can not work, but in flow, dec_nw_ttl is necessary.

I3 multicast: del_nw_dec field is necessary and works as well

Examples

Example 1: IPv4 Multicast (I3 multicast)

Test Name	Test basic flow
Test Configuration	
Test Procedure	<p>Step1:configure ttp</p> <pre>ovs-vsctl set-tpp-enable true ovs-vsctl set-tpp-file vz_etech_tpp_v0.3.json ovs-vsctl show-tpp sudo /etc/init.d/picos restart</pre> <p>step2:configure bridge and port</p> <pre>ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8 ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set interface ge-1/1/1 type=pica8 ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1 -- set interface ge-1/1/2 type=pica8 ovs-vsctl add-port br0 ge-1/1/3 vlan_mode=trunk tag=1 -- set interface ge-1/1/3 type=pica8</pre> <p>step3:add group</p> <p>I2 interface:</p> <pre>ovs-ofctl add-group br0 group_id=1,type=indirect,bucket=output:2 ovs-ofctl add-group br0 group_id=2,type=indirect,bucket=output:3</pre>

I3 multicast interface:

```
ovs-ofctl add-group br0
group_id=3,type=indirect,bucket=set_field:22:22:22:00:00:00->dl_src,set_field:2
00->vlan_vid,group:1
ovs-ofctl add-group br0
group_id=4,type=indirect,bucket=set_field:22:22:22:11:11:11->dl_src,set_field:3
00->vlan_vid,group:2
```

I3 multicast group:

```
ovs-ofctl add-group br0 group_id=5,type=all,bucket=group:3,bucket=group:4
```

step4:add flow

Here,flows' action ***goto_table:60*** is optional.

```
ovs-ofctl add-flow br0
table=40,priority=32768,d1_type=0x0800,d1_vlan=10,nw_src=192.168.1.100,nw_dst=23
4.234.245.245,actions=write_actions\(group:5,dec_nw_ttl\),goto_table:60
```

step5:check the hardware flow

```
admin@PicOS-OVS$ovs-appctl pica/dump-flows
Multicast Routing Table: (Total 1 flows)
  ID=12 ip,d1_vlan=10,nw_src=192.168.1.100,nw_dst=234.234.245.245,
  actions:group(id=3,all,n=2,b0(group(id=2,indirect,n=1,b0(set(dl_src=88:88:88:00:
00:00),set(vlan_vid=200),group(id=1,indirect,n=1,b0(live,output:2)))),b1(live,g
roup(id=6,indirect,n=1,b0(live,set(dl_src=44:44:44:00:00:00),set(vlan_vid=300),g
roup(id=5,indirect,n=1,b0(live,output:3)))),goto(Policy ACL Table)
```

step5:Send packets to ge-1/1/1,

send packets contents:

MAC: Destination Address : 01 00 5E 07 88 88

MAC: Source Address : 22 11 11 11 11 11

VLAN: VLAN Identifier = 10 (0x00A)

IP: Destination Address = 234.234.245.245

ge-1/1/2 will transmit the packets with the following content:

MAC: Destination Address : 01 00 5E 07 88 88

MAC: Source Address : 88 88 88 00 00 00

VLAN: Tag Control Information = 0x00C8

VLAN: VLAN Identifier = 200 (0x0C8)

IP: Destination Address = 234.234.245.245

IP: Time to Live = 63 (0x3F)

ge-1/1/3 will transmit the packets with the following content:

MAC: Destination Address : 01 00 5E 07 88 88

MAC: Source Address : 44 44 44 00 00 00

VLAN: Tag Control Information = 0x012C

VLAN: VLAN Identifier = 300 (0x012C)

IP: Destination Address = 234.234.245.245

	IP: Time to Live = 63 (0x3F)
Expect results	pass
Actual results	pass

Example 2: IPv4 Multicast (I2 multicast)

Test Name	Test basic flow
Test Configuration	
Test Procedure	<p>Step1: Configure TTP</p> <pre>ovs-vsctl set-ttp-enable true ovs-vsctl set-ttp-file vz_etech_ttp_v0.3.json ovs-vsctl show-ttp sudo /etc/init.d/picos restart</pre> <p>step2: Configure bridge and port</p> <pre>ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8 ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set interface ge-1/1/1 type=pica8 ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1 -- set interface ge-1/1/2 type=pica8 ovs-vsctl add-port br0 ge-1/1/3 vlan_mode=trunk tag=1 -- set interface ge-1/1/3 type=pica8</pre> <p>step3: Add group</p> <p>I2 interface:</p> <pre>ovs-ofctl add-group br0 group_id=1,type=indirect,bucket=output:2</pre> <p>I3 multicast group:</p> <pre>ovs-ofctl add-group br0 group_id=3,type=all,bucket=group:1</pre> <p>step4: Add flow</p> <pre>ovs-ofctl add-flow br0 table=40,priority=32768,dl_type=0x0800,dl_vlan=10,nw_src=192.168.1.100,nw_dst=234.234.245.245,actions=write_actions\((group:3,dec_nw_ttl\)),goto_table:60</pre> <p>step5: Check the hardware flow</p> <pre>admin@PicOS-OVS\$ovs-appctl pica/dump-flows Multicast Routing Table: (Total 1 flows) ID=10 ip,dl_vlan=10,nw_src=192.168.1.100,nw_dst=234.234.245.245, actions:group(id=3,all,n=1,b0(live,group(id=1,indirect,n=1,b0(live,output:2))), goto(Policy ACL Table))</pre>

	<p>step5: Send packets to ge-1/1/1,</p> <p>send packets contents:</p> <p>MAC: Destination Address : 01 00 5E 07 88 88</p> <p>MAC: Source Address : 22 11 11 11 11 11</p> <p>VLAN: VLAN Identifier = 10 (0x00A)</p> <p>IP: Destination Address = 234.234.245.245</p> <p>ge-1/1/2 will transmit the packets with the following content:</p> <p>MAC: Destination Address : 01 00 5E 07 88 88</p> <p>MAC: Source Address : 22 11 11 11 11 11</p> <p>VLAN: Tag Control Information = 0x000A</p> <p>VLAN: VLAN Identifier = 10 (0x00A)</p> <p>IP: Destination Address = 234.234.245.245</p> <p>IP: Time to Live = 64 (0x40)</p>
Expect results	pass
Actual results	Pass

Example 3: IPv4 Multicast mod-groups

Test Name	Test basic flow
Test Configuration	
Test Procedure	<p>Step1: Configure TTP</p> <pre>ovs-vsctl set-ttp-enable true ovs-vsctl set-ttp-file vz_etech_ttp_v0.3.json ovs-vsctl show-ttp sudo /etc/init.d/picos restart</pre> <p>step2: Configure bridge and port</p> <pre>ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8 ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set interface ge-1/1/1 type=pica8 ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1 -- set interface ge-1/1/2 type=pica8 ovs-vsctl add-port br0 ge-1/1/3 vlan_mode=trunk tag=1 -- set interface ge-1/1/3 type=pica8 ovs-vsctl add-port br0 ge-1/1/4 vlan_mode=trunk tag=1 -- set interface ge-1/1/4 type=pica8</pre> <p>step3: Add group</p> <p>I2 interface:</p> <pre>ovs-ofctl add-group br0 group_id=1,type=indirect,bucket=output:1 ovs-ofctl add-group br0 group_id=2,type=indirect,bucket=output:2</pre>

I3 multicast interface:

```
ovs-ofctl add-group br0
group_id=5,type=indirect,bucket=set_field:88:88:88:00:00:00->dl_src,set_field:2
00->vlan_vid,group:1
ovs-ofctl add-group br0
group_id=6,type=indirect,bucket=set_field:44:44:44:44:44:44->dl_src,set_field:2
00->vlan_vid,group:2
```

I3 multicast group:

```
ovs-ofctl add-group br0 group_id=7,type=all,bucket=group:5
```

step4: Add flow

```
ovs-ofctl add-flow br0
table=40,priority=32768,dl_type=0x0800,dl_vlan=10,nw_src=192.168.1.100,nw_dst=23
4.234.245.245,actions=write_actions\group:7,dec_nw_ttl\),goto_table:60
```

step5: Check the hardware flow

```
admin@PicOS-OVS$ovs-appctl pica/dump-flows
Multicast Routing Table: (Total 1 flows)
ID=6 ip,dl_vlan=10,nw_src=192.168.1.100,nw_dst=234.234.245.245,
actions:group(id=7,all,n=1,b0(live,group(id=5,indirect,n=1,b0(live, set(dl_src=88
:88:88:00:00:00),set(vlan_vid=200),group(id=1,indirect,n=1,b0(live,output:1)))),)
, goto(Policy ACL Table)
```

step5: Send packets to ge-1/1/1,

send packets contents:

MAC: Destination Address : 01 00 5E 07 88 88

MAC: Source Address : 22 11 11 11 11 11

VLAN: VLAN Identifier = 10 (0x00A)

IP: Destination Address = 234.234.245.245

ge-1/1/1 will transmit the packets with the following content:

MAC: Destination Address : 01 00 5E 07 88 88

MAC: Source Address : 88 88 88 00 00 00

VLAN: Tag Control Information = 0x00C8

VLAN: VLAN Identifier = 200 (0x0C8)

IP: Destination Address = 234.234.245.245

IP: Time to Live = 63 (0x3F)

Step6: Modify the group

```
admin@PicOS-OVS$ovs-ofctl mod-group br0 group_id=7,type=all,bucket=group:6
admin@PicOS-OVS$ovs-appctl pica/dump-flows
```

Step7: Send packets to ge-1/1/1:

send packets contents:

MAC: Destination Address : 01 00 5E 07 88 88

MAC: Source Address : 22 11 11 11 11 11

	<p>VLAN: VLAN Identifier = 10 (0x00A)</p> <p>IP: Destination Address = 234.234.245.245</p> <p>ge-1/1/2 will transmit the packets with the following content:</p> <p>MAC: Destination Address : 01 00 5E 07 88 88</p> <p>MAC: Source Address : 44 44 44 44 44 44</p> <p>VLAN: Tag Control Information = 0x00C8</p> <p>VLAN: VLAN Identifier = 200 (0xC8)</p> <p>IP: Destination Address = 234.234.245.245</p> <p>IP: Time to Live = 63 (0x3F)</p>
Expect results	pass
Actual results	pass

Example 4: IPv6 Multicast

Test Name	Test basic flow
Test Configuration	
Test Procedure	<p>Step1: Configure ttp</p> <pre>ovs-vsctl set-tpp-enable true ovs-vsctl set-tpp-file vz_etech_tpp_v0.3.json ovs-vsctl show-tpp sudo /etc/init.d/picos restart</pre> <p>step2: Configure bridge and port</p> <pre>ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8 ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set interface ge-1/1/1 type=pica8 ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1 -- set interface ge-1/1/2 type=pica8 ovs-vsctl add-port br0 ge-1/1/3 vlan_mode=trunk tag=1 -- set interface ge-1/1/3 type=pica8</pre> <p>step3: Add group</p> <p>I2 interface:</p> <pre>ovs-ofctl add-group br0 group_id=1,type=indirect,bucket=output:2 ovs-ofctl add-group br0 group_id=5,type=indirect,bucket=output:3</pre> <p>I3 multicast interface:</p> <pre>ovs-ofctl add-group br0 group_id=2,type=indirect,bucket=set_field:88:88:88:00:00:00->dl_src,set_field:200->vlan_vid,group:1 ovs-ofctl add-group br0 group_id=6,type=indirect,bucket=set_field:44:44:44:00:00:00->dl_src,set_field:300->vlan_vid,group:5</pre>

	I3 multicast group:
<pre>ovs-ofctl add-group br0 group_id=3,type=all,bucket=group:2,bucket=group:6</pre>	
	step4: Add flow
<pre>ovs-ofctl add-flow br0 table=40,priority=32768,dl_type=0x86dd,dl_vlan=10,ipv6_src=2000::1,ipv6_dst=ff00 ::1,actions=write_actions_(group:3,dec_nw_ttl_),goto_table:60</pre>	
step5: Check the hardware flow	
<pre>admin@PicOS-OVS\$ovs-appctl pica/dump-flows Multicast Routing Table: (Total 1 flows) ID=5040 ipv6(dl_vlan=10,ipv6_src=2000::1,ipv6_dst=ff00::1, actions:group(id=3,all,n=2,b0(live,group(id=2,indirect,n=1,b0(live,set(dl_src=88 :88:88:00:00:00),set(vlan_vid=200),group(id=1,indirect,n=1,b0(live,output:2))))), b1(live,group(id=6,indirect,n=1,b0(live,set(dl_src=44:44:44:00:00:00),set(vlan_ vid=300),group(id=5,indirect,n=1,b0(live,output:3)))),goto(Policy ACL Table)</pre>	
Step6: Send packets to ge-1/1/1,	
send packets contents:	
MAC: Destination Address : 33 33 33 00 00 01	
MAC: Source Address : 22 11 11 11 11 11	
VLAN: VLAN Identifier = 10 (0x00A)	
IPv6: Source Address: 2000:0000:0000:0000:0000:0000:0001	
IPv6: Dest Address: FF00:0000:0000:0000:0000:0000:0001	
Hop limit:255	
ge-1/1/2 will transmit the packets with the following content:	
MAC: Destination Address : 33 33 33 00 00 01	
MAC: Source Address : 88 88 88 00 00 00	
VLAN: Tag Control Information = 0x00C8	
VLAN: VLAN Identifier = 200 (0x0C8)	
IPv6: Source Address: 2000:0000:0000:0000:0000:0000:0001	
IPv6: Dest Address: FF00:0000:0000:0000:0000:0000:0001	
Hop limit:254	
ge-1/1/3 will transmit the packets with the following content:	
MAC: Destination Address : 33 33 33 00 00 01	
MAC: Source Address : 44 44 44 00 00 00	
VLAN: Tag Control Information = 0x012C	
VLAN: VLAN Identifier = 300 (0x012C)	
IPv6: Source Address: 2000:0000:0000:0000:0000:0000:0001	
IPv6: Dest Address: FF00:0000:0000:0000:0000:0000:0001	
Hop limit:254	
Expect results	pass

Actual results	pass
----------------	------

Example 5: IPv6 Multicast (I2 multicast)

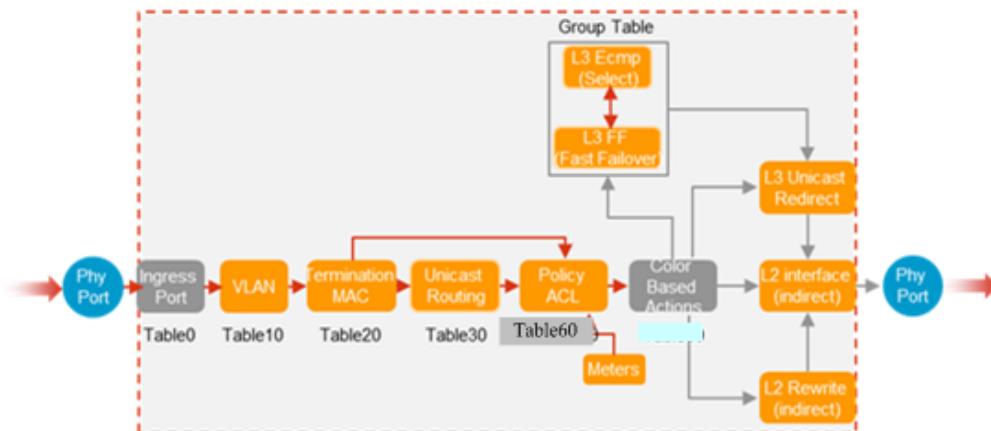
Test Name	Test basic flow
Test Configuration	
Test Procedure	<p>Step1:configure ttp</p> <pre>ovs-vsctl set-ttp-enable true ovs-vsctl set-ttp-file vz_etech_ttp_v0.3.json ovs-vsctl show-ttp sudo /etc/init.d/picos restart</pre> <p>step2:configure bridge and port</p> <pre>ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8 ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set interface ge-1/1/1 type=pica8 ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1 -- set interface ge-1/1/2 type=pica8 ovs-vsctl add-port br0 ge-1/1/3 vlan_mode=trunk tag=1 -- set interface ge-1/1/3 type=pica8</pre> <p>step3:add group</p> <p>I2 interface:</p> <pre>ovs-ofctl add-group br0 group_id=1,type=indirect,bucket=output:2 ovs-ofctl add-group br0 group_id=2,type=indirect,bucket=output:3</pre> <p>I3 multicast group:</p> <pre>ovs-ofctl add-group br0 group_id=3,type=all,bucket=group:1,bucket=group:2</pre> <p>step4:add flow</p> <pre>ovs-ofctl add-flow br0 table=40,priority=32768,dl_type=0x86dd,dl_vlan=10,ipv6_src=2000::1,ipv6_dst=ff00::1,actions=write_actions\((group:3,dec_nw_ttl\)),goto_table:60</pre> <p>step5:check the hardware flow</p> <pre>admin@PicOS-OVS\$ovs-appctl pica/dump-flows Multicast Routing Table: (Total 1 flows) ID=1 ipv6,dl_vlan=10,ipv6_src=2000::1,ipv6_dst=ff00::1, actions:group(id=3,all,n=2,b0(group(id=1,indirect,n=1,b0(output:2))),b1(live,gro up(id=2,indirect,n=1,b0(live,output:3))),goto(Policy ACL Table)</pre> <p>step5:Send packets to ge-1/1/1,</p> <p>first time send packets contents:</p> <p>MAC: Destination Address : 33 33 33 00 00 01</p>

	<p>MAC: Source Address : 22 11 11 11 11 11 VLAN: VLAN Identifier = 10 (0x00A) IPv6: Source Address: 2000:0000:0000:0000:0000:0000:0000:0001 IPv6: Dest Address: FF00:0000:0000:0000:0000:0000:0000:0001 Hop limit:255</p> <p>ge-1/1/2 will transmit the packets with the following content:</p> <p>MAC: Destination Address : 33 33 33 00 00 01 MAC: Source Address : 88 88 88 00 00 00 VLAN: Tag Control Information = 0x00C8 VLAN: VLAN Identifier = 200 (0x0C8) IPv6: Source Address: 2000:0000:0000:0000:0000:0000:0000:0001 IPv6: Dest Address: FF00:0000:0000:0000:0000:0000:0000:0001 Hop limit:255</p> <p>ge-1/1/3 will transmit the packets with the following content:</p> <p>MAC: Destination Address : 33 33 33 00 00 01 MAC: Source Address : 88 88 88 00 00 00 VLAN: Tag Control Information = 0x00C8 VLAN: VLAN Identifier = 200 (0x0C8) IPv6: Source Address: 2000:0000:0000:0000:0000:0000:0000:0001 IPv6: Dest Address: FF00:0000:0000:0000:0000:0000:0000:0001 Hop limit:255</p>
Expect results	pass
Actual results	pass

TTP Unicast

Introduction

This document describes the Table Type Pattern (TTP) and its usage.



TTP defines a template of tables that can be configured using open flow. It uses JSON notation to define the data structure for the pipeline defined in the TTP. Open flow instructions can use the parameters defined in the TTP file to program the flows. If a flow cannot be configured, the switch will report an error message.

Enabling TTP Mode

From PicOS version 2.7.1, supporting for TTP mode has been added in our switches. To use the TTP mode, user must first enable this mode. To enable the TTP mode, use the command "***ovs-vsctl set-tpp-enable true***". To add the TTP file, use the command "***ovs-vsctl set-tpp-file < TTP file>.json***". After adding the file, restart the switch and the TTP mode should be enabled on the switch.

json file

About .JSON FILE:

- Only "table_map" and "flow_tables" can be modified.
- "table_map": User can modify the table number of "VLAN," "Termination MAC," "Unicast Routing" and "Policy ACL". All of the four table id's must be increasing and different. For example, if user wants table 20 to be the VLAN table, write like this "VLAN:" 20 and so on.
- Flow_tables: Only the priority of all the flows tables can be changed. If user modifies vlan filter tables' priority to 1999, when user adds a flow, user must specify that the priority is 1999.
- path: /ovs/share/openvswitch

At present, users cannot modify the json file. If necessary, users may notify our team, and we may be able to change it.

Table map is:

- VLAN:10
- MAC:20
- ROUTE:30
- ACL:60

Notification

- In the acl table, if any layer 2 header content of packets is modified, previous actions in route table will be invalid.
- At present, the packets can ecmp in src ip by default.

VLAN Table

VLAN Filter Table

In this table, the user can configure the flow matching a vlan and in_port, and the action is dropped. That means that when packets arrive in this table, all the packets that are matching this flow will be dropped, others will pass through the vlan filter table. See the example configuration below:

```
ovs-ofctl add-flow br-s table=10,priority=32768,in_port=77,d1_vlan=20,actions=drop
```

VLAN Assignment Table

In this table, the user can configure a flow to decide the actions of the untagged packets. The flow matching fields are in_port, and vlan_vid=0x0000/0x1000, the action is set_field a vlan and goto terminal mac table. Refer to the example configuration below:

```
ovs-ofctl add-flow br-s
table=10,priority=32768,in_port=77,vlan_vid=0x0000/0x1000,actions=set_field:123->vlan_vid,
goto_table:20
```

VLAN Translate Single Tag Table

In this table, the user can configure a flow to decide the action of the tagged packets. The flow matching fields are in_port and a valid vlan vid, the actions is set_field a vlan and goto terminal mac table. As an example, see the code below:

```
ovs-ofctl add-flow br-s
table=10,priority=32768,in_port=77,dl_vlan=10,actions=set_field:123->vlan_vid, goto_table:2
0
```

Terminal MAC Table

In this table, the user can configure a flow to decide if the packets go to the route table or not. By default, packets will go to the acl table directly, instead of going to route table first. The match fields of the flow supported by this table are dl_dst, dl_vlan, action is goto_table:30. Multicast and broadcast MAC are not supported here.

```
ovs-ofctl add-flow br-s
table=20,priority=32768,dl_dst=00:11:22:33:44:56,dl_vlan=2015,actions=goto_table:30
```

L3 Table

In this table, users can direct the packets' route through several different kind of groups by means of writing actions. The matching fields are dl_type, nw_dst, actions are several kinds of groups and dec_nw_ttl, then go to ACL table. According our .json file we can add this kind of flow. Broadcast and multicast are not supported in this table. By default, all packets pass through this table and goto acl table.

Examples

(1) Unicast

L2 interface group:

```
ovs-ofctl add-group br-s group_id=1,type=indirect,bucket=output:79
```

L3 unicast group:

```
ovs-ofctl add-group br-s
group_id=2,type=indirect,bucket=set_field:66:66:66:00:00:00->dl_src, set_field:66:66:66:11:
11:11->dl_dst, set_field:200->vlan_vid, group:1
```

```
ovs-ofctl add-flow br-s
table=30,priority=32768,dl_type=0x0800,nw_dst=192.168.1.100,actions=write_actions\(
group:2,
dec_nw_ttl\), goto_table:60
```

(2) Ecmp

User can also complete the ecmp through a select group.

L2 interface group:

```
ovs-ofctl add-group br-s group_id=1,type=indirect,bucket=output:77
ovs-ofctl add-group br-s group_id=5,type=indirect,bucket=output:79
```

L3 UNICAST group:

```
ovs-ofctl add-group br-s
group_id=2,type=indirect,bucket=set_field:66:66:66:00:00:00->dl_src,set_field:66:66:66:11:
11:11->dl_dst,set_field:200->vlan_vid,group:1
ovs-ofctl add-group br-s
group_id=6,type=indirect,bucket=set_field:22:11:11:11:11->dl_src,set_field:44:44:44:11:
11:11->dl_dst,set_field:200->vlan_vid,group:5
```

L3 ff group:

```
ovs-ofctl add-group br-s group_id=3,type=fast_failover,bucket=group:2,watch_port:77
ovs-ofctl add-group br-s group_id=7,type=fast_failover,bucket=group:6,watch_port:79
```

L3 ecmp:

```
ovs-ofctl add-group br-s group_id=4,type=select,bucket=group:2,bucket=group:7
```

Add flow and check:

```
ovs-ofctl add-flow br-s
table=30,priority=32768,dl_type=0x0800,nw_dst=192.168.1.100,actions=write_actions\_(group:4,
dec_nw_ttl\_),goto_table:60
admin@PicOS-OVS$ovs-ofctl dump-flows br-s
OFPST_FLOW reply (OF1.4) (xid=0x2):
  cookie=0x0, duration=76.544s, table=30, n_packets=n/a, n_bytes=n/a,
  ip,nw_dst=192.168.1.100 actions=write_actions(group:9,dec_ttl),goto_table:60
admin@PicOS-OVS$ovs-appctl pica/dump-flows
Ingress Port Table: (Total 0 flows)

VLAN Table: (Total 0 flows)

Termination MAC Table: (Total 0 flows)

Unicast Routing Table: (Total 1 flows)
  ID=1 ip,nw_dst=192.168.1.100,
  actions:group(id=9,select,n=2,b0(live,group(id=3,ff,n=1,b0(live,group(id=2,indirect,n=1,b0(
  live,set(dl_src=66:66:66:00:00:00,dl_dst=66:66:66:11:11:11),set(vlan_vid=200),group(id=1,in
  direct,n=1,b0(live,output:77)))))),b1(live,group(id=7,ff,n=1,b0(live,group(id=6,indirect,n
  =1,b0(live,set(dl_src=22:11:11:11:11:11,dl_dst=44:44:44:11:11:11),set(vlan_vid=200),group(i
  d=5,indirect,n=1,b0(live,output:79)))))),goto(Policy ACL Table)

Policy ACL Table: (Total 0 flows)
```

Policy ACL Table

In this table, users can add a flow with any match field supported by our switch. The actions could be meter, group, controller, drop, set_queue.

Example:

```
ovs-ofctl add-flow br-s
table=60,priority=65535,in_port=78,dl_type=0x86dd,dl_src=22:22:22:22:22:22,dl_dst=44:44:44:11:11:11,ipv6_src=2015::1,dl_vlan=100,dl_vlan_pcp=3,ip_proto=6,ip_dscp=128,ip_ecn=0,tp_src=1111,tp_dst=1444,actions=meter:1
```

```
ovs-ofctl add-flow br-s table=60,actions=write_actions\:(group:4\)
```

Check TTP flows

The “ovs-appctl pica/dump-flows [tables]” accept table numbers or strings from version 2.8.0.

1)Ingress Port Table:

```
root@PicOS-OVS$ovs-appctl pica/dump-flows 0
Ingress Port Table: (Total 0 flows)
root@PicOS-OVS$ovs-appctl pica/dump-flows port
Ingress Port Table: (Total 0 flows)
```

2)VLAN Table:

```
admin@PicOS-OVS$ovs-appctl pica/dump-flows 10
VLAN Table: (Total 2 flows)
flow_id=6 in_port=1,dl_vlan=20, actions:drop
flow_id=7 in_port=1,dl_vlan=21, actions:drop
admin@PicOS-OVS$ovs-appctl pica/dump-flows vlan
VLAN Table: (Total 2 flows)
flow_id=6 in_port=1,dl_vlan=20, actions:drop
flow_id=7 in_port=1,dl_vlan=21, actions:drop
```

3)Termination MAC Table:

```
admin@PicOS-OVS$ovs-appctl pica/dump-flows 20
Termination MAC Table: (Total 3 flows)
flow_id=6 dl_vlan=2015,dl_dst=00:11:22:33:44:56, actions:goto(Unicast Routing Table)
flow_id=7 dl_vlan=205,dl_dst=22:22:22:33:44:56, actions:goto(Unicast Routing Table)
admin@PicOS-OVS$ovs-appctl pica/dump-flows mac
Termination MAC Table: (Total 3 flows)
flow_id=6 dl_vlan=2015,dl_dst=00:11:22:33:44:56, actions:goto(Unicast Routing Table)
flow_id=7 dl_vlan=205,dl_dst=22:22:22:33:44:56, actions:goto(Unicast Routing Table)
```

4)Unicast Routing Table:

```
admin@PicOS-OVS$ovs-appctl pica/dump-flows 30
Unicast Routing Table: (Total 1 flows)
flow_id=6 ip,nw_dst=192.168.1.0/24,
actions:group(id=300,indirect,n=1,b0(live,set(dl_src=66:66:66:00:00:00,dl_dst=66:66:66:11:11:11),set(vlan_vid=200),group(id=100,indirect,n=1,b0(live,output:2))),goto(Policy ACL
```

```
Table)
admin@PicOS-OVS$ovs-appctl pica/dump-flows unicast
Unicast Routing Table: (Total 1 flows)
  flow_id=6 ip,nw_dst=192.168.1.0/24,
actions:group(id=300,indirect,n=1,b0(live,set(dl_src=66:66:66:00:00:00,dl_dst=66:66:66:11:1:11),set(vlan_vid=200),group(id=100,indirect,n=1,b0(live,output:2))),goto(Policy ACL Table)
```

5)Policy ACL Table:

```
root@PicOS-OVS$ovs-appctl pica/dump-flows 60
Policy ACL Table: (Total 1 flows)
  flow_id=4009 priority=40000,ip,dl_vlan=199,
actions:meter(id=2,drop),group(id=5,indirect,n=1,b0(live,output:4))
root@PicOS-OVS$ovs-appctl pica/dump-flows acl
Policy ACL Table: (Total 1 flows)
  flow_id=4009 priority=40000,ip,dl_vlan=199,
actions:meter(id=2,drop),group(id=5,indirect,n=1,b0(live,output:4))
```

List TTP System Resources Usage

From version 2.8.0, when ttp is enabled, we can use command ovs-appctl pica/show tables to check max limitation or current used of different tables.

Pica Tables	Max Limitation	Current Used
VLAN Table (vlan-translate)	512	512
VLAN Table (vlan-vfilter)	512	512
Termination MAC Table	512	510
Host Unicast Routing Table	12000	IPv4(0), IPv6(6000*2)
Route Unicast Routing Table	12000	IPv4(4261), IPv6(3870*2)
Policy ACL Table	2046	2046

Flow_id in TTP flows

Flow_ids can match for TTP based flow configuration, and you can check or delete the flow using flow_id.

```
admin@PicOS-OVS$ovs-ofctl add-flow br0 table=10,priority=32768,in_port=1,dl_vlan=199,actions=set_field:123->vlan_vid,goto_table:20
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  flow_id=6, cookie=0x0, duration=8.424s, table=10, n_packets=n/a, n_bytes=n/a,
  in_port=1,dl_vlan=199 actions=set_field:123->vlan_vid,goto_table:20
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-appctl pica/dump-flows
Ingress Port Table: (Total 0 flows)

VLAN Table: (Total 1 flows)
  flow_id=6 in_port=1,dl_vlan=199, actions:set(vlan_vid=123),goto(Termination MAC Table)
```

```
Termination MAC Table: (Total 0 flows)
Unicast Routing Table: (Total 0 flows)
Policy ACL Table: (Total 0 flows)
admin@PicOS-OVS$
```

Check flows using flow_id:

```
admin@PicOS-OVS$ovs-ofctl dump-flows br0 flow_id=6
OFPST_FLOW reply (OF1.4) (xid=0x2):
  flow_id=6, cookie=0x0, duration=23.068s, table=10, n_packets=n/a, n_bytes=n/a,
  in_port=1,dl_vlan=199 actions=set_field:123->vlan_vid,goto_table:20
admin@PicOS-OVS$ovs-appctl pica/dump-flows flow_id=6
VLAN Table: (Total 1 flows)
  flow_id=6 in_port=1,dl_vlan=199, actions:set(vlan_vid=123),goto(Termination MAC Table)
admin@PicOS-OVS$
```

Delete flows using flow_id:

```
admin@PicOS-OVS$ovs-ofctl del-flows br0 flow_id=6
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
admin@PicOS-OVS$
```

Flow Handling Mode

PicOS supports set-flow-handling-mode from version 2.7.1. If the flow is integrated and can be installed directly, there is no need for set-flow-handling-mode, and whatever mode it is, it should be installed. But if the hardware does not support the actions, or the flow is not an exact match, it will not be installed directly. This is where we need set-flow-handling-mode, to decide what to do with the flow, whether to install the flow and how to install it.

The command is:

```
admin@PicOS-OVS$ovs-vsctl set-flow-handling-mode [mode]
```

Mode value: [enable_packet_driven, hardware_flow_only, software_flow_allowed], and default value is software_flow_allowed.

After setting a particular flow handling mode, the switch should be rebooted to make the mode effective.

```
admin@PicOS-OVS$ovs-vsctl show-flow-handling-mode
```

Direct Flows

These flows can be installed directly to the hardware table whatever mode it is.

Example 1

```
admin@XorPlus$ovs-ofctl add-flow br0
in_port=9,ip,actions=push_vlan:0x8100,set_field:19->vlan_vid,output:11
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#96 normal permanent flow_id=18 ip,in_port=9,
actions:push_vlan(vid=19),mod_vlan_pcp(pcp=0),11
normal_d permanent internal flow_id=7 priority=0, actions:drop
Total 1 flows in HW.
admin@PicOS-OVS$
```

Example 2

```
admin@XorPlus$ovs-ofctl add-flow br0
in_port=9,dl_vlan=19,dl_dst=22:22:22:22:22:22,tcp,actions=set_field:3.3.3.3->nw_dst,set_fi
eld:800->tp_dst,11
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-appctl pica/dump-flows
normal_d permanent internal flow_id=7 priority=0, actions:drop
#98 normal permanent flow_id=20 tcp,in_port=9,dl_vlan=19,dl_dst=22:22:22:22:22:22,
actions:set(ipv4(dst=3.3.3.3)),set(tcp(dst=800)),11
Total 1 flows in HW.
admin@PicOS-OVS$
```

Regardless of whatever flow handling mode is currently in effect, direct flows will always be installed to the hardware table directly.

Packet-driven-flows

The packet-driven status is as follows:

set-flow-handling-mode = hardware_flow_only

This mode forbids the flow installation if the hardware does not support the actions or the flow is not an exact match.

Example

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=9,tcp,dl_vlan=19,actions=set_field:192.168.1.1->nw_src,set_field:0x123->tp_src,11
OFPT_ERROR (OF1.4) (xid=0x2): OFPBAC_MATCH_INCONSISTENT
OFPT_FLOW_MOD (OF1.4) (xid=0x2):
(**truncated to 64 bytes from 136**)
00000000 05 0e 00 88 00 00 00 02-00 00 00 00 00 00 00 00 | .....
00000010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 80 00 | .....
00000020 ff ff ff ff ff ff ff-ff ff ff ff 00 00 00 00 | .....
00000030 00 01 00 1d 80 00 00 04-00 00 00 09 80 00 0a 02 | .....
admin@PicOS-OVS$
```

In hardware_flow_only mode, configuring packet-driven flows will return an error.

set-flow-handling-mode = software_flow_allowed

This mode forbids the flow installation if the hardware does not support the actions. This mode permits the flow installation TO_CPU if the hardware supports the action but the flow is not exact an match. The item will be submitted to the CPU for further actions .

Example

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=9,ip,d1_vlan=199,actions=push_mpls:0x8847,set_field:299->mpls_label,11
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#1 normal_d permanent flow_id=12 priority=0, actions:drop
#2 normal_u permanent flow_id=16 ip,in_port=9,d1_vlan=199,
actions:To_CPU(for_packet_driven)
Total 2 flows in HW.
admin@PicOS-OVS$
```

Packets which match this flow will be submitted to cpu, and it will not be forwarded at line-speed.

set-packet-driven-mode = enable_packet_driven

This mode forbids the flow installation if the hardware does not support the actions. This mode permits the flow installation TO_CPU if the hardware supports the actions but the flow is not an exact match. The item will be submitted to the CPU. The CPU will modify the match and make it exact by parsing, then install the exact flow.

Example

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=9,d1_dst=22:22:22:22:22:22,udp,actions=set_field:192.168.100.2->nw_dst,set_field:2
00->udp_dst,11
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#1 normal_d permanent flow_id=12 priority=0, actions:drop
#3 normal_u permanent flow_id=17 udp,in_port=9,d1_dst=22:22:22:22:22:22,
actions:To_CPU(for_packet_driven)
Total 2 flows in HW.
admin@PicOS-OVS$
```

The packets matching this flow are checked in the hardware table, if there is no exact match, the packets are forwarded to the CPU. The CPU will modify the match and install the exact match flow on to the hardware table.

```
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#2 normal flow_id=11
priority=1048560,udp,in_port=9,vlan_tci=0x0000/0x1fff,d1_dst=22:22:22:22:22:22,nw_dst=2.2.2
.2,nw_frag=no,tp_src=63,tp_dst=63,
actions:set(ipv4(dst=192.168.100.2)),set(udp(dst=200)),11
#0 normal_d permanent flow_id=2 priority=0, actions:drop
#1 normal_u permanent flow_id=11 udp,in_port=9,d1_dst=22:22:22:22:22:22,
actions:To_CPU(for_packet_driven)
Total 3 flows in HW.
admin@PicOS-OVS$
```

Packets can then be transmitted at line-speed since there is now an exact match flow entry in the hardware table.



In case of direct flows, the set-packet-driven-mode is irrelevant and can be ignored. It is only when the flows cannot be installed directly, that the set-packet-driven-mode becomes necessary. The underlying hardware support is crucial to adding flows, if there is no support for the flow in the hardware, the system will generate an error message.

Drop counter

Pica8 has added support for drop counter from PicOS version 2.7.2. This feature is useful in cases where we need to count the dropped packets. For this counter, we use rdbgc4 register to count the dropped packets.

The command is:

```
ovs-vsctl set-rdbgc4 [TYPE]
```

TYPE value: **rfldr; ripd4; ripd6; riphe6; vlandr**

Default value: **rfldr**

Use the command below to display rdbgc4 setting.

```
ovs-vsctl show-rdbgc4
```

The command below is used get drop statistics on a given port.

```
ovs-ofctl dump-drop br0 [PORT]
```

1. rdbgc4 drop counter values:

1) ovs-vsctl set-rdbgc4 rfldr

Setting the rdbgc4 value to rfldr will count the packets which are dropped by a drop action (actions=drop) flow in TCAM. Packets matching this flow are dropped and drop statistics are recorded in the drop counter.

Example:

```
ovs-vsctl add-br br0
ovs-vsctl add-port br0 te-1/1/9 vlan_mode=trunk tag=1 -- set Interface te-1/1/9 type=pica8
ovs-vsctl add-port br0 te-1/1/10 vlan_mode=trunk tag=1 -- set Interface te-1/1/10
type=pica8
```

Packets will match default drop flow without configuring flows.

```
admin@PicOS-OVS$ovs-ofctl dump-drop br0 9
PXST_DROP_STATS reply (OF1.4) (xid=0x2): 1 interfaces
te-1/1/9(port 9):
    Statistics type in rdbgc4: rfldr
        Discarded Packets.....10000
    Statistics type : congest drop bytes
        Discarded Octets.....0
    Statistics type : congest drop packets
        Discarded Packets.....0
    Statistics type : ingress drop packets
        Discarded Packets..... 10000
```

2) ovs-vsctl set-rdbgc4 vlandr

Count the packets whose tag doesn't match the ingress port's VLAN ID.

Example:

```
ovs-vsctl add-port br0 te-1/1/9 vlan_mode=access tag=10 -- set Interface te-1/1/9
type=pica8
ovs-vsctl add-port br0 te-1/1/10 vlan_mode=trunk tag=1 -- set Interface te-1/1/10
type=pica8
```

add flow:

```
ovs-ofctl add-flow br0 in_port=9,actions=10
```

Send packets with vlan=199 to port 10 and check drop counter.

```
admin@PicOS-OVS$ovs-ofctl dump-drop br0 9
PXST_DROP_STATS reply (OF1.4) (xid=0x2): 1 interfaces
te-1/1/9(port 9):
    Statistics type in rdbgc4: vlandr
        Discarded Packets.....10000
    Statistics type : congest drop bytes
        Discarded Octets.....0
    Statistics type : congest drop packets
        Discarded Packets.....0
    Statistics type : ingress drop packets
        Discarded Packets..... 10000
```

3) ovs-vsctl set-rdbgc4 ripd4

Count the ipv4 packets which send to route table with VLAN **CFI=SET or TTL=0**.

Example:

```
ovs-vsctl set-l2-mode true
ovs-vsctl set-l3-mode true
ovs-ofctl add-flow br0 table=251, dl_vlan=199,d1_dst=22:22:22:22:22:22,actions=normal
ovs-ofctl add-flow br0 table=252, ip,nw_dst=2.2.2.2,actions=set_field:
00:00:00:00:22:22->d1_dst,set_field:1999->vlan_vid,10
```

Send packets with **dl_dst=22:22:22:22:22:22, dl_dst=22:11:11:11:11:11,dl_vlan=199,CFI=SET, nw_dst=2.2.2.2, nw_src=1.1.1.1** to port 10.

Check drop counter:

```
admin@PicOS-OVS$ovs-ofctl dump-drop br0 9
PXST_DROP_STATS reply (OF1.4) (xid=0x2): 1 interfaces
te-1/1/9(port 9):
    Statistics type in rdbgc4: ripd4
        Discarded Packets.....10000
    Statistics type : congest drop bytes
        Discarded Octets.....0
    Statistics type : congest drop packets
        Discarded Packets.....0
    Statistics type : ingress drop packets
        Discarded Packets..... 0
```

4) ovs-vsctl set-rdbgc4 ripd6

Count the ipv6 packets which send to route table with **VLAN CFI=set or Hop=0**.

Example:

```
ovs-vsctl set-l2-mode true
ovs-vsctl set-l3-mode true
ovs-ofctl add-flow br0 table=251,dl_vlan=199,dl_dst=22:22:22:22:22:22,actions=normal
ovs-ofctl add-flow br0 table=252,ipv6,ipv6_dst=2001::1,actions=set_field:
00:00:00:00:22:22->dl_dst,set_field:1999->vlan_vid,10
```

Send packets with dl_dst=22:22:22:22:22:22, dl_dst=22:11:11:11:11:11,dl_vlan=199,**CFI=RESET, hop=0**, ipv6_dst=2001::1,ipv6_src=2002::1 to port 10:

Check drop counter:

```
admin@PicOS-OVS$ovs-ofctl dump-drop br0 9
PXST_DROP_STATS reply (OF1.4) (xid=0x2): 1 interfaces
te-1/1/9(port 9):
    Statistics type in rdbgc4: ripd6
        Discarded Packets.....10000
    Statistics type : congest drop bytes
        Discarded Octets.....0
    Statistics type : congest drop packets
        Discarded Packets.....0
    Statistics type : ingress drop packets
        Discarded Packets..... 0
```

5) ovs-vsctl set-rdbgc4 riphe6

Count the ipv6 packets which send to route table with **ipv6_dst=::**

Example:

```
ovs-vsctl set-l2-mode true
ovs-vsctl set-l3-mode true
ovs-ofctl add-flow br0 table=251,dl_vlan=199,dl_dst=22:22:22:22:22:22,actions=normal
ovs-ofctl add-flow br0
table=252,ipv6,ipv6_dst=2001::1,actions=set_field:00:00:00:00:22:22->dl_dst,set_field:1999
->vlan_vid,10
```

Send packets with dl_dst=22:22:22:22:22:22,dl_dst=22:11:11:11:11:11,dl_vlan=199, ipv6_dst=0::0,ipv6_src=2002::1

Check drop counter:

```
admin@PicOS-OVS$ovs-ofctl dump-drop br0 9
PXST_DROP_STATS reply (OF1.4) (xid=0x2): 1 interfaces
te-1/1/9(port 9):
    Statistics type in rdbgc4: riphe6
        Discarded Packets.....10000
    Statistics type : congest drop bytes
        Discarded Octets.....0
    Statistics type : congest drop packets
        Discarded Packets.....0
    Statistics type : ingress drop packets
        Discarded Packets..... 0
```

2.Congest drop counter:

Count packets which dropped because of congestion.

```
admin@PicOS-OVS$ovs-ofctl dump-drop br0 9
PXST_DROP_STATS reply (OF1.4) (xid=0x2): 1 interfaces
te-1/1/9(port 9):
    Statistics type in rdbgc4: ripd6
        Discarded Packets.....0
    Statistics type : congest drop bytes
        Discarded Octets..... 22567715016
    Statistics type : congest drop packets
        Discarded Packets..... 331876187
    Statistics type : ingress drop packets
        Discarded Packets..... 0
```

3.ingress drop counter:

the packets which dropped by the actions=drop flow in tcam.

Example

```
ovs-ofctl add-flow br0 in_port=9,ip,d1_vlan=199,nw_dst=192.168.100.100,actions=drop
```

Send packets to match this flow and check drop counter:

```
admin@PicOS-OVS$ovs-ofctl dump-drop br0 9
PXST_DROP_STATS reply (OF1.4) (xid=0x2): 1 interfaces
te-1/1/9(port 9):
    Statistics type in rdbgc4: ripd6
        Discarded Packets.....0
    Statistics type : congest drop bytes
        Discarded Octets.....0
    Statistics type : congest drop packets
        Discarded Packets.....0
    Statistics type : ingress drop packets
        Discarded Packets..... 10000
```



Drop counter only works in ingress.

ecmp-select and lag-select group

From PicOS 2.8.0, Pica8 switch supports the traffic which has processed in TCAM, still can hash as ecmp interface or lag interface.

In prior PicOS version, the traffic in TCAM table only could hash as select group, refer to chapter 'Creating a Group Table,' the hash algorithm is defined by the designer. But from the new version, user can use ecmp or lag to process traffic.



- 1, 'dec_ttl' only can be used in table=252 route table and ecmp-select-group, others will neglect it.
- Once configure ecmp-select-group, L2 table (table=251) cannot work.
- 2, Platforms P-3290/P-3295/P-3297 do not support configure ecmp-group.

In default, ecmp and lag hash cannot work in TCAM table, user needs to use group table to define them. 'lag-select-groups=0-0' means the group ID range is 0-0, that is disable lag-select-groups. Same as others.

The command to show the ranges of ecmp-select-groups and lag-select-groups.

```
admin@PicOS-OVS$ovs-vsctl show-group-ranges
group_ranges: default
  lag-select-groups=0-0
  ecmp-select-groups=0-0
  ingress-mirror-groups=0-0
  egress-mirror-groups=0-0
```

ECMP Select Group

Select group uses ecmp, create an ecmp interface for each select group, and the ecmp interface contains egress interfaces which correspond to buckets in the group.

The command to configure ecmp-select-group ID range:

ovs-vsctl set-group-ranges ecmp-select-groups=<min_value>-<max_value>

The group IDs may be divided into different areas: ecmp-select-groups and the original groups.

Ecmp-select-groups ranges is <min_value>-<max_value> and the rest of group IDs for original groups. The max range of group is 1-4294967039.

After configuring the command, user needs to reboot switch.

Use command **ovs-vsctl set-group-ranges** to back default group ranges.

```
admin@PicOS-OVS$ovs-vsctl set-group-ranges ecmp-select-groups=1-10000
admin@PicOS-OVS$ovs-vsctl show-group-ranges
group_ranges:
  lag-select-groups=0-0
  ecmp-select-groups=1-10000
  ingress-mirror-groups=0-0
  egress-mirror-groups=0-0
```

In ecmp select group can modify dl_src, dl_src, dl_vlan, dec_ttl and set_queue. And the actions must be same in different buckets, the value can be the same or not.

If dec_ttl is set or set_field with vlan_vid,dl_src or dl_dst is set in an ecmp select group flow's actions, the ecmp will modify packet's corresponding field. If the fields are not set in an ecmp select group flow, the value also will be modified via L3 ecmp interface, and dl_src and dl_dst will be switch system mac and ttl will decrease.

User can use command 'set-l3-egress-keep-fields' to keep the field as original value.

ovs-vsctl set-l3-egress-keep-fields /FIELDS/

/FIELDS/ allows one or more value of {dl_vlan, dl_src, dl_dst, nw_ttl}. If not special configure /FIELDS/, will back default value and the fields will be modified by L3 ecmp interface.



During add flow entry whose actions include ecmp-select-group, other actions fields only support above fields: dl_src, dl_src, dl_vlan, dec_ttl and set_queue.

ovs-vsctl set-l3-ecmp-hash-fields /FIELDS/

<FIELDS> allowed values in_port, nw_dst, nw_proto, nw_src, port_dst, port_src, vlan. If not special configure <FIELDS>, the default value is nw_src. If user want configure multiple values, should use blank.

```
admin@PicOS-OVS$ovs-vsctl set-l3-ecmp-hash-fields in_port nw_dst
```

Example 1

Set group 1 as an ecmp select group, modify dl_vlan, dl_src and dl_dst, dec ttl.

```
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=set_field:10->vlan_vid,set_field:00:00:00:22:22:22->dl_src,
set_field:00:00:00:33:33:33->dl_dst,dec_ttl,output:33,bucket=set_field:20->vlan_vid,set_field:00:00:00:44:44:44->dl_src,set_field:00:00:00:55:55:55->dl_dst,dec_ttl,output:34
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=31,actions=group:1

admin@PicOS-OVS$ovs-appctl pica/dump-flows
#0 normal_d permanent flow_id=2 priority=0, actions:drop
#1 normal permanent flow_id=6 (ecmp select group) in_port=31,
actions:egress_dec_ttl,set(eth(src=00:00:00:22:22:22,dst=00:00:00:33:33:33)),set_vlan_id(vlan_id=10),33,egress_dec_ttl,set(eth(src=00:00:00:44:44:44,dst=00:00:00:55:55:55)),set_vlan_id(vlan_id=20),34
Total 2 flows in HW.
```

Example 2

Set group 1 as an ecmp select group, do not change dl_vlan, dl_src, dl_dst and ttl.

vlan_vid, dl_src and dl_dst are not set in an ecmp select group flow, and packet's corresponding fields stay unchanged.

```
admin@PicOS-OVS$ovs-vsctl set-l3-egress-keep-fields dl_vlan,dl_src,dl_dst,nw_ttl

admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:33,bucket=output:34
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=31,dl_vlan=30,dl_dst=00:00:00:11:11:11,actions=group:1

admin@PicOS-OVS$ovs-appctl pica/dump-flows
#0 normal_d permanent flow_id=2 priority=0, actions:drop
#2 normal permanent flow_id=8 (ecmp select group)
in_port=31,dl_vlan=30,dl_dst=00:00:00:11:11:11, actions:33,34
Total 2 flows in HW.
```

Example 3

Set group 1 as an ecmp select group, modify dl_vlan, dl_src only and other packet's corresponding fields stay unchanged.

```
admin@PicOS-OVS$ovs-vsctl set-l3-egress-keep-fields dl_dst,nw_ttl

admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=set_field:10->vlan_vid,set_field:00:00:00:22:22:22->dl_src,
output:33,bucket=set_field:20->vlan_vid,set_field:00:00:00:44:44:44->dl_src,output:34
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=31,actions=group:1

admin@PicOS-OVS$ovs-appctl pica/dump-flows
```

```
#0 normal_d permanent flow_id=2 priority=0, actions:drop
#3 normal permanent flow_id=9 (ecmp select group) in_port=31,
actions:set(eth_src((src=00:00:00:22:22:22))),set_vlan_id(vid=10),33,set(eth_src((src=00:00
:00:44:44:44))),set_vlan_id(vid=20),34
Total 2 flows in HW.
```

LAG Select Group

If user wants the traffic load balance like lag interface and doesn't want special configuration lag interface, lag select group is one choice.

Select group uses lag, create an lag interface for each select group, and the lag interface contains egress interfaces which corresponds to buckets in the group.



PICOS reserve ae1 to ae20 for static lag configuration, ae21 to ae48 use the internal lag for select group. PICOS will create internal lag when the user creates a lag select group. It only uses the first bucket actions except the output. It means that we can support 24 select groups. The physical port which has added in lag-select-group cannot be used in other groups and cannot work as one physical port.

Before add lag select group, user need configure group range.

The command to configure lag-select-group ID range:

ovs-vsctl set-group-ranges lag-select-groups=<min_value>-<max_value>

The group IDs may be divided into different areas: lag-select-groups and the original groups.

Lag-select-groups ranges is <min_value>-<max_value> and the rest of group IDs for original groups.

After configure the command, user needs to reboot switch.

```
admin@PicOS-OVS$ovs-vsctl set-group-ranges lag-select-groups=1-10000
admin@PicOS-OVS$ovs-vsctl show-group-ranges
group_ranges:
lag-select-groups=1-10000
ecmp-select-groups=0-0
ingress-mirror-groups=0-0
egress-mirror-groups=0-0
```

In lag select group can modify any fields switch support. And the actions must be configured in first bucket, other buckets only include output.

Example 1

Set group 1 as a lag select group, modify dl_vlan and dl_dst.

```
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=select,bucket=set_field:1000->vlan_vid,set_field:00:00:00:11:11:11->eth_dst,output:32,bucket=output:33,bucket=output:34
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=31,actions=group:1
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#0 normal_d permanent flow_id=2 priority=0, actions:drop
#1 normal permanent flow_id=6 (lag select group) in_port=31,
actions:set(eth_dst((dst=00:00:00:11:11:11))),set_vlan_id(vid=1000),1045
Total 2 flows in HW.
```

Close all Group-ranges

If user want close the group-ranges, please use command as below.

ovs-vsctl set-group-ranges

```
admin@PicOS-OVS$ovs-vsctl set-group-ranges
admin@PicOS-OVS$ovs-vsctl show-group-ranges
group_ranges:
    lag-select-groups=0-0
    ecmp-select-groups=0-0
    ingress-mirror-groups=0-0
    egress-mirror-groups=0-0
```

ingress-mirror-group and egress-mirror-group

From PicOS 2.8.0, Pica8 switch support ingress-mirror-group and egress-mirror-group.

- Ingress-mirror-group: Packets can modify corresponding fields and send out from one port, at the same time the packets also can keep original and send out from other ports.
- Egress-mirror-group: Packets can modify corresponding fields send out from one port, also can send out from another port at same time.

Using configure type=all group to create ingress-mirror-group or egress-mirror-group.

- Ingress-mirror-group: Only support 2 buckets, the first bucket process packets as normal, use the second bucket to mirror original packets at ingress and send out from one physical port or lag interface.
- Egress-mirror-group: Only support 2 buckets, the first bucket process packets as normal, use the second bucket to mirror modified packets at egress and send out from one physical port or lag interface.

We said the output in second bucket as monitor port.



1. ingress/Egress mirror group only support configure 2 buckets; Monitor port can be physical port/lag port/lag-select-group.
2. lag-select-group can include multiple ports, but if lag-select-group as monitor port only first eight ports can work.
3. only support add 4 monitor port (include ingress+egress) at most.
4. ingress/egress mirror group only can configure in table=0 flow entry.
5. ingress/egress mirror group cannot configure dec_ttl in bucket.

In default, ingress/egress-mirror-groups is disabled, user need use command to configure them.

'ingress-mirror-groups=0-0' means the group ID range is 0-0, that is disable ingress-mirror-groups.

The command to show the ranges of ingress-mirror-groups and egress-mirror-groups.

```
admin@PicOS-OVS$ovs-vsctl show-group-ranges
group_ranges: default
    lag-select-groups=0-0
    ecmp-select-groups=0-0
    ingress-mirror-groups=0-0
    egress-mirror-groups=0-0
```

The command to configure ingress-mirror-group and egress-mirror-group ID range:

```
ovs-vsctl set-group-ranges ingress-mirror-groups=<min_value>-<max_value>
```

or

```
ovs-vsctl set-group-ranges egress-mirror-groups=< min_value >-< max_value >
```

or

```
ovs-vsctl set-group-ranges ingress-mirror-groups=< min_value1 >-< max_value1 >,  
egress-mirror-groups=< min_value2 >-< max_value2 >
```

The group IDs may be divided into different areas: ingress-mirror-groups, egress-mirror-group and the original groups. The max range of group is 1-4294967039, and if configure different group the group range cannot conflict.

After configure the command, user need reboot switch.

Use command 'ovs-vsctl set-group-ranges' to back default group ranges.

```
admin@PicOS-OVS$ovs-vsctl set-group-ranges  
ingress-mirror-groups=1-10000,egress-mirror-groups=20000-30000  
  
admin@PicOS-OVS$ovs-vsctl show-group-ranges  
group_ranges:  
    lag-select-groups=0-0  
    ecmp-select-groups=0-0  
    ingress-mirror-groups=1-10000  
    egress-mirror-groups=20000-30000
```

Example 1

And ingress-mirror-group, first bucket modify packet's fields dl_vlan and dl_dst.

```
admin@PicOS-OVS$ovs-ofctl add-group br0  
group_id=100,type=all,bucket=set_field:100->vlan_vid,set_field:22:00:00:00:00:00->eth_dst  
,output:33,bucket=output:34  
admin@PicOS-OVS$  
admin@PicOS-OVS$ovs-ofctl add-flow br0  
in_port=31,dl_dst=22:11:11:11:11:11,actions=group:100  
admin@PicOS-OVS$  
admin@PicOS-OVS$ovs-appctl pica/dump-flows  
#0 normal_d permanent flow_id=2 priority=0, actions:drop  
#1 normal permanent flow_id=6 (ingress mirror group) in_port=31,dl_dst=22:11:11:11:11:11,  
actions:set(eth_dst((dst=22:00:00:00:00:00))),set_vlan_id(vid=100),33,mirror_nest_start,34,  
mirror_nest_end  
Total 2 flows in HW.  
admin@PicOS-OVS$
```

Example 2

And egress-mirror-group, first bucket modify packet's fields dl_src and dl_dst, second bucket is lag interface.

```
admin@PicOS-OVS$ovs-vsctl add-port br0 ae1 vlan_mode=trunk tag=1 -- set interface ae1  
type=pica8_lag options:members=te-1/1/32,te-1/1/33
```

```

admin@PicOS-OVS$ 
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=20010,type=all,bucket=set_field:00:00:00:11:11:11->eth_src, set_field:00:00:00:22:
22:22->eth_dst,output:34,bucket=output:1025
admin@PicOS-OVS$ 
admin@PicOS-OVS$ovs-ofctl add-flow br0 in_port=31,actions=group:20010
admin@PicOS-OVS$ 
admin@PicOS-OVS$ovs-appctl pica/dump-flows
#0 normal_d permanent flow_id=2 priority=0, actions:drop
#2 normal permanent flow_id=7 (egress mirror group) in_port=31,
actions:set(eth(src=00:00:00:11:11:11,dst=00:00:00:22:22:22)),34,mirror_nest_start,1025,mir
ror_nest_end
Total 2 flows in HW.

```

Close all Group-ranges

If user want close the group-ranges, please use command as below.

[ovs-vsctl set-group-ranges](#)

```

admin@PicOS-OVS$ovs-vsctl set-group-ranges
admin@PicOS-OVS$ovs-vsctl show-group-ranges
group_ranges:
  lag-select-groups=0-0
  ecmp-select-groups=0-0
  ingress-mirror-groups=0-0
  egress-mirror-groups=0-0

```

Configuring Meter

Summary

If do not limit the user ability to send traffic, then a large number of users constantly breaking data will make the network more crowded. In order to make the network resources able to be used fairly and efficiently, user traffic must be limited. For example, limit the flow of a stream at each time interval to assign it part of the resource, to prevent network congestion caused by excessive burst.

In the ovs mode, user can configure meter to achieve interface and set the speed-limit including ingress and egress. The meter uses the token bucket to evaluate the specifications of the traffic, then makes a policy for the traffic, such as through, remark or drop.

Now we support 1R2C and 2R3C for the token bucket in the ingress meter and egress meter.

Configuration

In different hardware models, the maximum count of meters that PicOS OVS supports is different. User can use the command to check the maximum count.

```

admin@PicOS-OVS$ovs-ofctl meter-features br0
OFPST_METER_FEATURES reply (OF1.4) (xid=0x2):
max_meter:2048 max_bands:1 max_color:3

```

```
band_types: drop dscp_remark
capabilities: kbps burst stats
```

Ingress Meter

1R2C: Add a meter, the type=drop.

Without burst size. Limit the rate as 300000 kbps.

```
root@PicOS-OVS$ovs-ofctl add-meter br0 meter=1, kbps, band=type=drop, rate=300000
```

With burst size. Limit the rate as 300000 kbps and burst 30000 kbit

```
root@PicOS-OVS$ovs-ofctl add-meter br0
meter=2, kbps, burst, band=type=drop, rate=300000, burst_size=30000
```

1R2C: Add a meter, the type=dscp_remark.

Without burst_size. The prec_level=14.

```
root@PicOS-OVS$ovs-ofctl add-meter br0
meter=2, kbps, band=type=dscp_remark, rate=300000, prec_level=14
```

With burst_size. The prec_level=14.

```
root@PicOS-OVS$ovs-ofctl add-meter br0
meter=2, kbps, burst, band=type=dscp_remark, rate=300000, prec_level=14, burst_size=30000
```

Check the meter configuration

```
root@LNOS-OVS$ovs-ofctl dump-meters br0
OFPST_METER_CONFIG reply (OF1.4) (xid=0x2):
meter=2 kbps burst bands=
type=dscp_remark rate=300000 burst_size=30000 prec_level=14
root@LNOS-OVS$
```

Modify one meter

```
root@LNOS-OVS$ovs-ofctl mod-meter br0 meter=2, kbps, burst, band=type=dscp_remark,
rate=400000, prec_level=1, burst_size=10000
root@LNOS-OVS$ovs-ofctl dump-meters br0
OFPST_METER_CONFIG reply (OF1.4) (xid=0x2):
meter=2 kbps burst bands=
type=dscp_remark rate=400000 burst_size=10000 prec_level=1
root@LNOS-OVS$
```

Dump meter stats

```
admin@PicOS-OVS$ovs-ofctl meter-stats br0
```

Delete one meter or all meters

```
root@LNOS-OVS$ovs-ofctl del-meter br0 meter=2
root@LNOS-OVS$
root@LNOS-OVS$ovs-ofctl del-meters br0
root@LNOS-OVS$
```

Notice: If one meter applies to multiple flow entries, all the flow entries will share the meter rate.

Example:

```
root@PicOS-OVS$ovs-ofctl add-meter br0 meter=1, kbps, band=type=drop,rate=300000
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=11,d1_dst=22:00:00:00:00:00,d1_src=22:11:11:11:11:11,d1_type=0x0800,actions=meter:1
,output:13
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=12,d1_dst=22:00:00:00:00:00,d1_src=22:11:11:11:11:22,d1_type=0x0800,actions=meter:1
,output:13
```

1. Each in_port are sent 200M flow to one output port.

The result: the port te-1/1/13, receive 300M flow, all of flow are green, the 100M red flow are dropped.

2R3C:

Without burst size.

The first bucket limit the rate as 300000 kbps, if the rate more than it and less than 600000 kbps, the packets dscp value will be remarked 14.

The second bucket limit the rate as 600000 kbps, if the rate more than it, then beyond the part will be dropped.

```
root@PicOS-OVS$ovs-ofctl add-meter br0
meter=2, kbps, bands=type=dscp_remark,rate=300000,prec_level=14,type=drop,rate=600000
```

With burst size. Limit the rate as 300000 kbps, 600000 kbps and burst as 10000 kbit, 20000 kbit.

```
root@PicOS-OVS$ovs-ofctl add-meter br0
meter=2, kbps, burst, bands=type=dscp_remark,rate=300000,burst_size=10000,prec_level=14,type=drop,rate=600000,burst_size=20000
```

Example:

```
root@PicOS-OVS$ovs-ofctl add-meter br0
meter=2,kbps,bands=type=dscp_remark,rate=300000,prec_level=14,type=drop,rate=500000
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=11,dl_dst=22:00:00:00:00:00,dl_src=22:11:11:11:11:11,dl_type=0x0800,actions=meter:2
,output:13
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=12,dl_dst=22:00:00:00:00:00,dl_src=22:11:11:11:11:22,dl_type=0x0800,actions=meter:2
,output:13
```

1. Each in_port are sent 200M flow to one output port.

The result: the port te-1/1/13 is receive 400M flow,including 300M green flow, and 100M yellow,no red flow.

2. Each in_port are sent 300M flow to one output port.

The result: the port te-1/1/13 is receive 500M flow,including 300M green flow, 200M yellow flow, and 100M red flow are dropped.

Egress Meter

Egress meter need redistribute meter id, and it can't be used by ingress meter. After configuring,need restart the ovs service.

```
root@LNOS-OVS$ovs-vsctl set-meter-ranges egress-meter=3-4
Please reboot for the change to take effect!
root@LNOS-OVS$/etc/init.d/picos restart
```

Check egress meter configuration

```
root@LNOS-OVS$ovs-vsctl show-meter-ranges
meter_ranges:
  egress-meter=3-4
root@LNOS-OVS$
```

The egress meter configuration is the same as ingress meter. And egress meter is also support 1R2C and 2R3C.

```
root@PicOS-OVS$ovs-ofctl add-meter br0
meter=3,kbps,bands=type=dscp_remark,rate=300000,prec_level=14,type=drop,rate=500000
```

Then put a flow need applying egress meter.

```
root@PicOS-OVS$ovs-ofctl add-flow br0
in_port=14,ip,nw_src=1.1.1.2,actions=egress_meter:3,output:38
```

Notice:

- 1.The meter configure at most two bands now in a meter flow. If only one band is set, 1R2C is effective; otherwise, 2R3C is used, and the rate configured in band1 should be smaller than rate in band2.
- 2.If the type of band1 is drop, the type of band2 is invalid.
- 3.ingress meter use color-blind mode which means the bucket don't care the color of incoming packets.
- 4.egress meter use color-aware mode which means the bucket can aware the color of incoming packets.

Application

```

root@LNOS-OVS$ovs-vsctl set-meter-ranges egress-meter=3-4
Please reboot for the change to take effect!
root@LNOS-OVS$/etc/init.d/picos restart

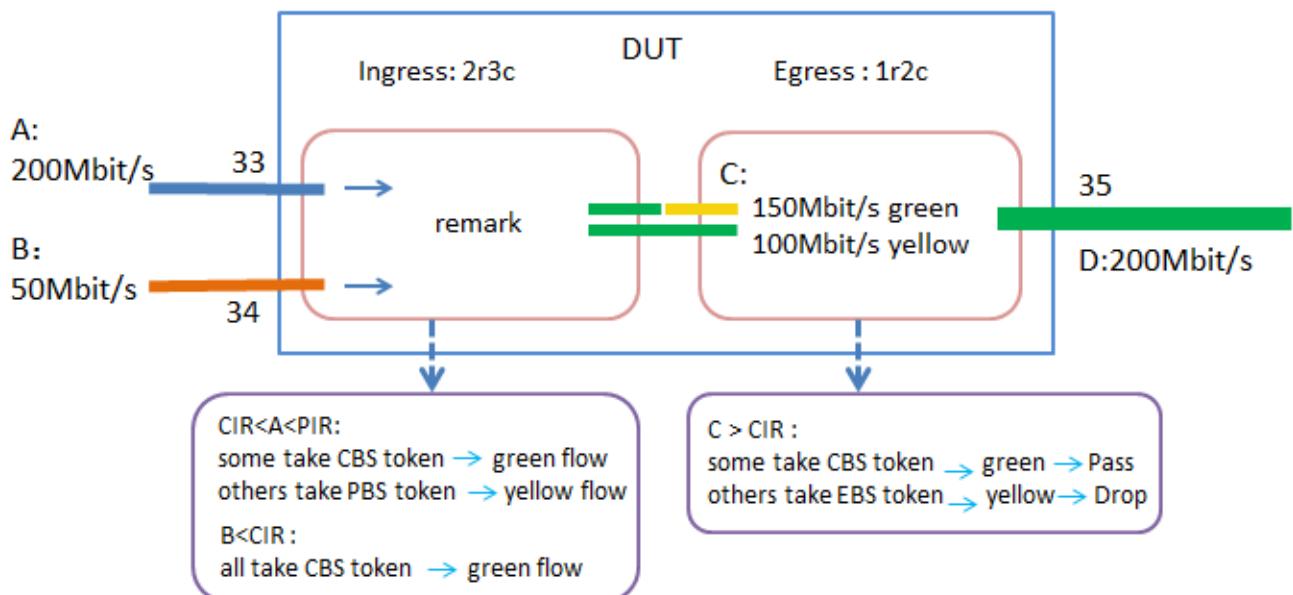
Ingress meter:
root@LNOS-OVS$ovs-ofctl add-meter br0 meter=1,rate=100000,prec_level=1,type=drop
root@LNOS-OVS$ovs-ofctl add-meter br0 meter=2,rate=100000,prec_level=14,type=drop

Egress meter:
ovs-ofctl
add-meter br0 meter=3,rate=200000

Add flow:
root@LNOS-OVS$ovs-ofctl add-flow br0 in_port=33,ip,d1_src=22:11:11:11:11:11,actions=meter:1,egress_meter:3,output:35
root@LNOS-OVS$ovs-ofctl add-flow br0 in_port=34,ip,d1_src=22:11:11:11:11:12,actions=meter:2,egress_meter:3,output:35
root@LNOS-OVS$ovs-appctl pica/dump-flows
#0 normal_d permanent flow_id=2 priority=0, actions:drop
#2 normal permanent flow_id=6 ip,in_port=33,d1_src=22:11:11:11:11:11,
actions:meter(id=1,band(dscp_remark=1),band(drop)),egress_meter(id=3,band(drop)),35
#3 normal permanent flow_id=7 ip,in_port=34,d1_src=22:11:11:11:11:12,
actions:meter(id=2,band(dscp_remark=14),band(drop)),egress_meter(id=3,band(drop)),35
Total 3 flows in HW.
root@LNOS-OVS$
```

Process

sent 200M flow to port 33,50M flow to 34.



Result: the port te-1/1/35 receive 200M flow including 150M green flow and 50M yellow flow.

Other Result:

1. Each in_port are sent 90M flow to one output port.

The result: the port te-1/1/35 receive 180M flow, all of flow are green.

2. sent 150M flow to port 33, 40M flow to 34.

The result: the port te-1/1/35 receive 190M flow including 50M yellow flow and 140M green flow.

3. Each in_port are sent 150M flow to one output port.

The result: the port te-1/1/35 receive 200M flow including 200M green flow.

4. sent 250M flow to port 33, 70M flow to 34.

The result: the port te-1/1/35 receive 200M flow including 170M green flow and 30M yellow flow.

5. sent 350M flow to port 33, 150M flow to 34.

The result: the port te-1/1/35 receive 200M flow including 200M green flow.

6. sent 350M flow to port 33, 350M flow to 34.

The result: the port te-1/1/35 receive 200M flow including 200M green flow.

NOTICE:

1.

Simply put, the setting of the Meter parameter depends on the speed-limit of the actual traffic.

In principle, the token bucket capacity needs to be greater than or equal to the length of the largest packet and the normal burst of traffic in the network.

For the PICA8 switches, we have a summary of empirical formulas:

1. bandwidth 100Mbit/s, token bucket capacity (kbits) = bandwidth (kbit/s) * 1000 (s)

2. bandwidth > 100Mbit/s, token bucket capacity (kbits) = 100000 (kbit/s) * (s)

2.

Now, Ingress meter are supported all platform, but egress meter just part of platform are supported,as shown below:

NIAGARA2632XL // NIAGARA2948_6XL // S4048
 PRONTO5101 // PRONTO5401 // AS6712_32X
 AS5712_54X // AS6701_32X // ARCTICA4806XP
 AS5812_54T

Configuration saving

Pica8 support saving configuration of meters,groups or flows which need to configure again previously after restart the switch system from version 2.8.1.

There are two scripts on /ovs/bin, ovs-pica-save and ovs-pica-load. When you have configured meters,groups or flows, please run ovs-pica-save before restart switch system. And run ovs-pica-load after restart.

Path of ovs-pica-save and ovs-pica-load: /ovs/bin

1.Configure meters.

```
admin@PicOS-OVS$ovs-ofctl add-meter br0
meter=2, kbps, band=type=dscp_remark, rate=300000, prec_level=14
admin@PicOS-OVS$ovs-ofctl add-meter br0 meter=1, kbps, band=type=drop, rate=300000
admin@PicOS-OVS$ovs-ofctl add-meter br0
meter=3, kbps, burst, band=type=dscp_remark, rate=300000, prec_level=14, burst_size=30000
```

2.Configure groups.

```
admin@PicOS-OVS$ovs-ofctl add-group br0 group_id=1, type=indirect, bucket=output:3
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=2, type=indirect, bucket=set_field:66:66:66:00:00:00->dl_src, set_field:66:66:66:11:11:11->dl_dst, set_field:200->vlan_vid, group:1
```

3.Add flows.

```
admin@PicOS-OVS$ovs-ofctl add-flow br0
in_port=1, ip, dl_dst=22:22:22:22:22:22, actions=meter:1, group:1
admin@PicOS-OVS$ovs-ofctl add-flow br0 arp, arp_tpa=192.168.100.100, actions=2
```

4.Run ovs-pica-save.

```
admin@PicOS-OVS$ovs-pica-save
Meter configurations are saved to : /ovs/config/meters
Group configurations are saved to : /ovs/config/groups
Flow configurations are saved to : /ovs/config/flows
```

```
Saved successfully!
admin@PicOS-OVS$
```

5.Restart the switch.

```
admin@XorPlus$ sudo /etc/init.d/picos restart
```

6.Load configuration.

```
admin@PicOS-OVS$ ovs-pica-load
Your configuration has been loaded successfully!
admin@PicOS-OVS$
```

If load failed, user should check files which has been saved in /ovs/config and then modify it.

7.Check after loading configuration.

```
admin@PicOS-OVS$ ovs-ofctl dump-meters br0
OFPST_METER_CONFIG reply (OF1.4) (xid=0x2):
meter=1 kbps bands=
type=drop rate=300000

meter=2 kbps bands=
type=dscp_remark rate=300000 prec_level=14

meter=3 kbps burst bands=
type=dscp_remark rate=300000 burst_size=30000 prec_level=14
admin@PicOS-OVS$
admin@PicOS-OVS$
admin@PicOS-OVS$ ovs-ofctl dump-groups br0
OFPST_GROUP_DESC reply (OF1.4) (xid=0x2):
group_id=1,type=indirect,bucket=actions=output:3
group_id=2,type=indirect,bucket=actions=set_field:66:66:66:00:00:00->eth_src,set_field:66:
66:66:11:11:11->eth_dst,set_field:200->vlan_vid,group:1
admin@PicOS-OVS$
admin@PicOS-OVS$ ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
flow_id=6, cookie=0x0, duration=16.841s, table=0, n_packets=n/a, n_bytes=0,
arp,arp_tpa=192.168.100.100 actions=output:2
flow_id=7, cookie=0x0, duration=16.840s, table=0, n_packets=n/a, n_bytes=0,
ip,in_port=1,dl_dst=22:22:22:22:22:22 actions=meter:1,group:1
admin@PicOS-OVS$
```

Note:

1) Related configuration in br0 will save to :

Meters: /ovs/config/meters/br0

Groups: /ovs/config/groups/br0

Flows: /ovs/config/flows/br0

User can modify above files to add or delete configuration and then load new configuration.

2) If there is error information when load configuration, user can modify above files too.

3) Load meters firstly and then groups and flows lastly.

Troubleshooting PicOS OVS

This section details basic procedures to troubleshoot PicOS switches in OVS (Open vSwitch) mode.

Verifying PicOS Mode

Verify if PicOS is actually running in OVS (Open vSwitch) mode, as described in Checking PicOS Mode.

When PicOS is running in the OVS mode, two processes should be running: **ovsdb-server** and **ovs-vswitchd**.

```
admin@XorPlus$ps -ef | grep ovs
root      1356      1  0 Jan26 ?          00:00:10 /ovs/sbin/ovsdb-server
/ovs/ovs-vswitchd.conf.db --pidfile --remote=punix:/ovs/var/run/openvswitch/db.sock
root      1358      1  0 Jan26 ?          00:19:07 /ovs/sbin/ovs-vswitchd
--enable-shared-lcmgr
```

In CrossFlow mode, the router stack must have been initialized in addition to having **ovsdb-server** and **ovs-vswitchd** processes running.

```
admin@XorPlus$ps -ef | grep pica
root      12430      1  0 Jan07 ?          00:05:49 pica_cardmgr
root      12432      1  0 Jan07 ?          01:03:19 pica_sif
root      12439      1  0 Jan07 ?          00:08:45 pica_lacp
root      12441      1  19 Jan07 ?         4-10:50:14 pica_lcmgr
root      12447      1  0 Jan07 ?          00:09:58 pica_login
root      13218      1  0 Jan07 ?          00:20:47 pica_mstp
root      13236      1  0 Jan07 ?          01:25:30 /pica/bin/xorp_rtrmgr -d -L local0.info -P
/var/run/xorp_rtrmgr.pid
```

Verifying Bridge Configuration

For the bridge and ports to forward frames in hardware, the **datapath_type** configured for each entity must be set to **pica8**.

```
admin@PicOS-OVS$ovs-vsctl show
ac9e5b1e-4234-4158-9214-5660b9343779
  Bridge east
    Controller "tcp:172.16.0.142:6653"
      is_connected: true
      fail_mode: standalone
      Port "ae1"
        tag: 1
        Interface "ae1"
          type: "pica8_lag"
          options: {lacp-mode=active, lacp-system-priority="32768", lacp-time=slow,
lag_type=lACP, link_speed=auto, members="qe-1/1/2"}
      Port "te-1/1/2"
        tag: 1
        Interface "te-1/1/2"
          type: "pica8"
          options: {flow_ctrl=none, link_speed=auto}
      Port "te-1/1/1"
        tag: 1
```

```

Interface "te-1/1/1"
    type: "pica8"
    options: {flow_ctrl=none, link_speed=auto}

admin@PicOS-OVS$ovs-ofctl show east
OFPT_FEATURES_REPLY (OF1.4) (xid=0x2): dpid:1deb0ae61be44040
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS
OFPST_PORT_DESC reply (OF1.4) (xid=0x4):
1(te-1/1/1): addr:ff:ff:ff:ff:ff:00
    config: 0
    state: LINK_UP
    current: 1GB-FD COPPER
    advertised: 1GB-FD 10GB-FD FIBER
    supported: 10MB-FD 100MB-FD 1GB-FD 10GB-FD FIBER AUTO_NEG
    speed: 1000 Mbps now, 10000 Mbps max
2(te-1/1/2): addr:ff:ff:ff:ff:ff:00
    config: 0
    state: LINK_DOWN
    current: 1GB-FD COPPER
    advertised: 1GB-FD 10GB-FD FIBER
    supported: 10MB-FD 100MB-FD 1GB-FD 10GB-FD FIBER AUTO_NEG
    speed: 1000 Mbps now, 10000 Mbps max
1025(ael): addr:ff:ff:ff:ff:ff:00
    config: 0
    state: LINK_UP
    current: 1GB-FD COPPER
    advertised: 1GB-FD 10GB-FD FIBER
    supported: 10MB-FD 100MB-FD 1GB-FD 10GB-FD FIBER AUTO_NEG
    speed: 1000 Mbps now, 10000 Mbps max
LOCAL(east): addr:0a:e6:1b:e4:40:40
    config: 0
    state: LINK_UP
    current: 10MB-FD COPPER
    supported: 10MB-FD COPPER
    speed: 10 Mbps now, 10 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.4) (xid=0x6): frags=normal miss_send_len=0
admin@PicOS-OVS$
```

Once the ports are configured and verified, flows can be managed in OVS.

Checking Flow Discrepancies

Check **ovs-vswitchd** flow discrepancies between the control plane and hardware:

```

admin@PicOS-OVS$ovs-ofctl dump-tables br0 | grep -v active=0:
0: active=4, lookup=n/a, matched=n/a

admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
  cookie=0x0, duration=1449.903s, table=0, n_packets=n/a, n_bytes=0,
  in_port=1,dl_src=00:00:3d:a6:c8:f2 actions=output:2
  cookie=0x0, duration=1444.537s, table=0, n_packets=n/a, n_bytes=0,
  in_port=1,dl_src=00:00:3d:a6:c9:14 actions=output:1
  cookie=0x0, duration=71723.842s, table=0, n_packets=n/a, n_bytes=0,
  mpls,in_port=1,dl_vlan=1,mpls_label=10 actions=output:3
  cookie=0x0, duration=74839.581s, table=0, n_packets=n/a, n_bytes=923443200, in_port=1
  actions=output:2
```

Display hardware flows as shown below:

```

admin@PicOS-OVS$ovs-appctl pica/dump-flows
#24 normal permanent priority=32769,in_port=1,dl_src=00:00:3d:a6:c8:f2, actions:2
```

```
#23 normal permanent priority=32769,in_port=1,d1_src=00:00:3d:a6:c9:14, actions:1
#22 normal permanent priority=32769,mpls,in_port=1,d1_vlan=1,mpls_label=10, actions:3
#21 normal permanent priority=32769,in_port=1, actions:2
#20 normal permanent priority=0, actions:drop
Total 5 flows in HW.
```

Displaying OVSDB

Display the full OVSDB (Open vSwitch Database) as shown below:

```
admin@Leaf1$ovsdb-client dump
Bridge table
_uuid controller datapath_id
datapath_type external_ids fail_mode flood_vlans flow_tables ipfix lldp_enable mirrors name
netflow other_config ports
protocols sflow status stp_enable
-----
-----
-----
-----
c880536a-b614-41bf-9870-2d0bdab3664f [bedb4af7-2125-4346-8c89-bf61bd21f63b]
"4c3e486e730203da" "pica8" {} [] [] {} [] false
[] "ECODE3" [] {} [31605950-d9be-40b2-9ccb-bc4fd09991f0,
61ac5778-554f-4553-83ae-3bbc19ccf715, 62b35f47-e8ca-4496-8b37-f9bfbb7e80b0,
6dee5c6a-e9b8-41f7-87ef-b9379637a7c4, 99ac75b7-9fa1-4583-85f7-66d3145e7fa4] ["OpenFlow13"]
[] {} false
<Some output omitted>
```

Debug Packet-In Messages

To debug the protocol messages between the switch and the controller, use the **ovs-ofctl snoop** command in the OVS mode. The following commands debug the protocol messages exchanged between the **br0** bridge and the controller:

```
admin@Switch$ovs-ofctl snoop br0
```

Issue 1, During add port error.

```
admin@PicOS-OVS$ovs-vsctl add-port br0 qe-1/1/49 vlan_mode=trunk tag=1 -- set interface qe-1/1/49 type=pica8
ovs-vsctl: Error detected while setting up 'qe-1/1/49'. See ovs-vswitchd log for details.
admin@PicOS-OVS$
```

Please check license installed or not.

```
admin@PicOS-OVS$license -s
```

If no license installed, please install license first; If license has installed but the port also add error, please check the port whether is valid in the switch.

```
admin@PicOS-OVS$ovs-appctl pica/show
```

Issue 2, Port cannot up.

```
admin@PicOS-OVS$ovs-ofctl show br0
OFPT_FEATURES_REPLY (OF1.4) (xid=0x2):
dpid:0x5e3e047d7b6293ff(6790870225108767743)
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS
OFPST_PORT_DESC reply (OF1.4) (xid=0x3):
1(te-1/1/1): addr:04:7d:7b:62:93:ff
    config:      0
    state:      LINK_DOWN
    current:     FIBER
    advertised: 1GB-FD 10GB-FD FIBER
    supported:   1GB-FD 10GB-FD FIBER AUTO_NEG
    speed: 0 Mbps now, 10000 Mbps max
```

1, Please verify the connection, make sure the connect is correct.

2, Use CLI check module plugin or not.

```
admin@PicOS-OVS$ovs-invctl enable-sfp true
admin@PicOS-OVS$ovs-invctl show-sfp-qsfq
```

3, If there is no module informations about the port, please repeat plugin module to check log messages.

```
admin@PicOS-OVS$ovs-appctl vlog/set hwlog
admin@PicOS-OVS$ovs-appctl hwlog/set-type config true
admin@PicOS-OVS$ovs-appctl hwlog/set-level all debug

Jan  5 2017 14:32:41 XorPlus daemon.warning : 14:32:41.220|ovs|00009|pica_private|WARN|SFP
te-1/1/1 is plugged in, optical module type 10G_BASE_SR, vendor: OEM, serial number:
A85351050280
```

4, If there is no log message as above. This is bug, please contact us.

5, If there have log message, and can read module use command ' ovs-invctl show-sfp-qsfq'. But port still cannot up, this also one bug, please contract us.

```
ec869713-92af-406e-abb4-f60efdec59f3
  port_name: "te-1/1/1"
  vendor_name: OEM
  vendor_sn: "A85351050280"
  wavelength: "850 nm"
  temperature: "82 F"
  supply_voltage: "3.00 V"
  connector: LC
  length_copper: "0 m"
  length_50m: "80 m"
  length_625m: "20 m"
  length_9m: "300 m"
  length_9m_km: "0 m"
```

```

plugged_in: true
rx_receive_power: "-2.68 dbm"
tx_bias_current: "6.00 mA"
tx_optical_power: "-3.26 dbm"

```

6, If there have plug-in log message, but the module is unknown. Like below.

```

Jan  5 2017 14:41:14 XorPlus daemon.warning : 14:41:14.631|ovs|00014|pica_private|WARN|SFP
te-1/1/1 is plugged in, optical module type unknown, vendor: FINISAR, serial number:
P117EWY

admin@PicOS-OVS$ovs-invctl show-sfp-qsf
d664a803-a6a1-4582-b162-6b9e617df2ea
  port_name: "te-1/1/1"
  vendor_name: FINISAR
  vendor_sn: "P117EWY"
  wavelength: "850 nm"
  temperature: "117995 F"
  supply_voltage: "65535.00 V"
  connector: LC
  length_copper: "0 m"
  length_50m: "300 m"
  length_625m: "150 m"
  length_9m: "0 m"
  length_9m_km: "0 m"
  plugged_in: true
  rx_receive_power: "65535.00 dBm"
  tx_bias_current: "65535.00 mA"
  tx_optical_power: "65535.00 dBm"

```

7, Please check the speed of module is same as the max speed of port supported. If it is, this is a bug, please contact us;

If the speed of module less than the max speed of port, please configure the port speed as the module. If the port also cannot up this time, please contact us.

```

admin@PicOS-OVS$ovs-vsctl set interface te-1/1/1 options:link_speed=1G

admin@PicOS-OVS$ovs-ofctl show br0
OFPT_FEATURES_REPLY (OF1.4) (xid=0x2):
dpid:0x5e3e047d7b6293ff(6790870225108767743)
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS
OFPST_PORT_DESC reply (OF1.4) (xid=0x3):
  1(te-1/1/1): addr:04:7d:7b:62:93:ff
    config: 0
    state: LINK_UP
    current: 1GB-FD FIBER
    advertised: 1GB-FD FIBER
    supported: 1GB-FD 10GB-FD FIBER AUTO_NEG
    speed: 1000 Mbps now, 10000 Mbps max

```

Examples and Topologies

This chapter provides configuration examples for 802.1Q.

- 802.1Q VLAN
- ECMP
- GRE Tunnel
- MPLS Network
- Multiple Virtual Bridges
- SSL Connection to Controller

802.1Q VLAN

In the following topology, user needs to configure 2 VLANs in switches A and B.

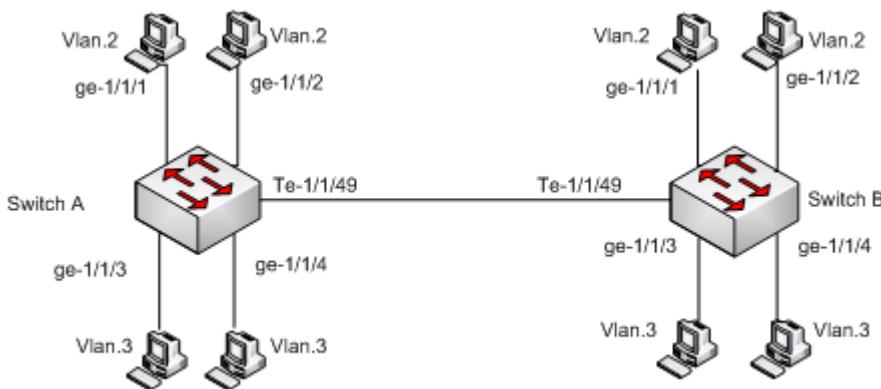


Figure 4-1. 802.1Q network configuration

Configure Switch-A

In Switch-A, user needs to configure ge-1/1/1~ge-1/1/4 as access port while configuring te-1/1/49 as trunk port because the 10Gbit link will trunk the traffic of VLAN-2 and VLAN-3.

```
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/1 vlan_mode=access tag=2 -- set Interface te-1/1/1 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/2 vlan_mode=access tag=2 -- set Interface te-1/1/2 type=pica8
root@PicOS-OVS#
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/3 vlan_mode=access tag=3 -- set Interface te-1/1/3 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/4 vlan_mode=access tag=3 -- set Interface te-1/1/4 type=pica8
root@PicOS-OVS#
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/49 vlan_mode=trunk trunk=2,3 -- set Interface te-1/1/49 type=pica8
root@PicOS-OVS#
```

Configure Switch-B

In Switch-B, user needs to configure ge-1/1/1~ge-1/1/4 as access port while te-1/1/49 as trunk port because the 10Gbit link will trunk the traffic of VLAN-2 and VLAN-3.

```
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/1 vlan_mode=access tag=2 -- set Interface te-1/1/1 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/2 vlan_mode=access tag=2 -- set Interface te-1/1/2 type=pica8
root@PicOS-OVS#
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/3 vlan_mode=access tag=3 -- set Interface te-1/1/3 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/4 vlan_mode=access tag=3 -- set Interface te-1/1/4 type=pica8
root@PicOS-OVS#
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/49 vlan_mode=trunk trunk=2,3 -- set Interface te-1/1/49 type=pica8
root@PicOS-OVS#
```

ECMP

```
root@PicOS-OVS#ovs-vsctl del-br br0
root@PicOS-OVS#ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
root@PicOS-OVS#ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1
trunks=1000,2000,3000,4094 -- set Interface ge-1/1/1 type=pica8
root@PicOS-OVS#ovs-vsctl add-port br0 ge-1/1/2 vlan_mode=trunk tag=1
trunks=1000,2000,3000,4094 -- set Interface ge-1/1/2 type=pica8
root@PicOS-OVS#ovs-vsctl add-port br0 ge-1/1/3 vlan_mode=trunk tag=1
trunks=1000,2000,3000,4094 -- set Interface ge-1/1/3 type=pica8
root@PicOS-OVS#ovs-vsctl add-port br0 ge-1/1/4 vlan_mode=trunk tag=1
trunks=1000,2000,3000,4094 -- set Interface ge-1/1/4 type=pica8
root@PicOS-OVS#ovs-ofctl del-flows br0
root@PicOS-OVS#ovs-ofctl add-group br0
group_id=1,type=select,bucket=output:2,bucket=output:3,bucket=output:4
root@PicOS-OVS#ovs-ofctl add-flow br0 dl_type=0x0800,nw_dst=192.168.2.0/24,actions=group:1
```

Send packets (nw_dst incr number is 200) to Port 1,

Packets whose nw_dst= 192.168.2.0/255.255.255.3 will be forwarded to Port 2.

Packets whose nw_dst= 192.168.2.1/255.255.255.3 will be forward to Port 3.

Packets whose nw_dst= 192.168.2.2/255.255.255.3 will be forward to Port 4.

Packets whose nw_dst= 192.168.2.3/255.255.255.3 will be forward to Port 2.

GRE Tunnel

In the following topology, the GRE tunnel between Switches A and B needs to be configured. The IP address of the GRE tunnel is 10.10.61.10/24 and 10.10.60.10/24.

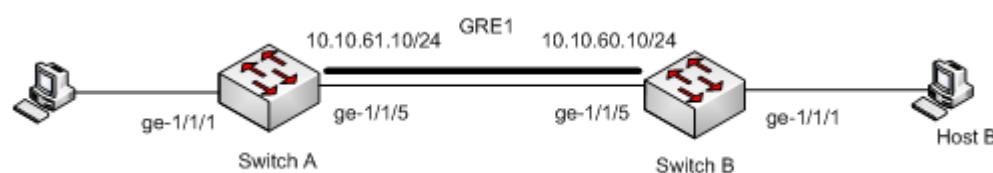


Figure 4-2. GRE tunnel configuration

Configure Switch-A

In Switch-A, user needs to configure a GRE tunnel and two flows as follows:

```
root@PicOS-OVS# ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set Interface
ge-1/1/1 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 ge-1/1/5 vlan_mode=trunk tag=1 -- set Interface
ge-1/1/5 type=pica8
root@PicOS-OVS#
root@PicOS-OVS# ovs-vsctl add-port br0 gre1 -- set Interface gre1 type=pica8_gre
options:remote_ip=10.10.60.10 options:local_ip=10.10.61.10 options:vlan=1
options:src_mac=00:11:11:11:11:11 options:dst_mac=00:22:22:22:22:22
options:egress_port=ge-1/1/5
root@PicOS-OVS#
root@PicOS-OVS# ovs-ofctl add-flow br0 in_port=1,actions=output:109
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=5,actions=mod_dl_src:00:11:11:11:11:11,mod_dl_dst:00:33:33:33:33:33,output:1
```

Configure Switch-B

In Switch-A, user also needs to configure a GRE tunnel and two flows as follows:

```
root@PicOS-OVS# ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 -- set Interface
ge-1/1/1 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 ge-1/1/5 vlan_mode=trunk tag=1 -- set Interface
ge-1/1/5 type=pica8
root@PicOS-OVS#
root@PicOS-OVS# ovs-vsctl add-port br0 gre1 -- set Interface gre1 type=pica8_gre
options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10 options:vlan=1
options:src_mac=00:22:22:22:22:22 options:dst_mac=00:11:11:11:11:11
options:egress_port=ge-1/1/5
root@PicOS-OVS#
root@PicOS-OVS# ovs-ofctl add-flow br0 in_port=1,actions=output:91
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=5,actions=mod_dl_src:00:22:22:22:22:22,mod_dl_dst:00:66:66:66:66:66,output:1
```

MPLS Network

In the following topology, a simple MPLS network is configured.

Traffic (Red) from host-A to host-B is forward by the MPLS network with Label 10. The traffic (Blue) from host-C to host-D is forwarded by the MPLS network with Label 20. All the flows will only push one MPLS header.

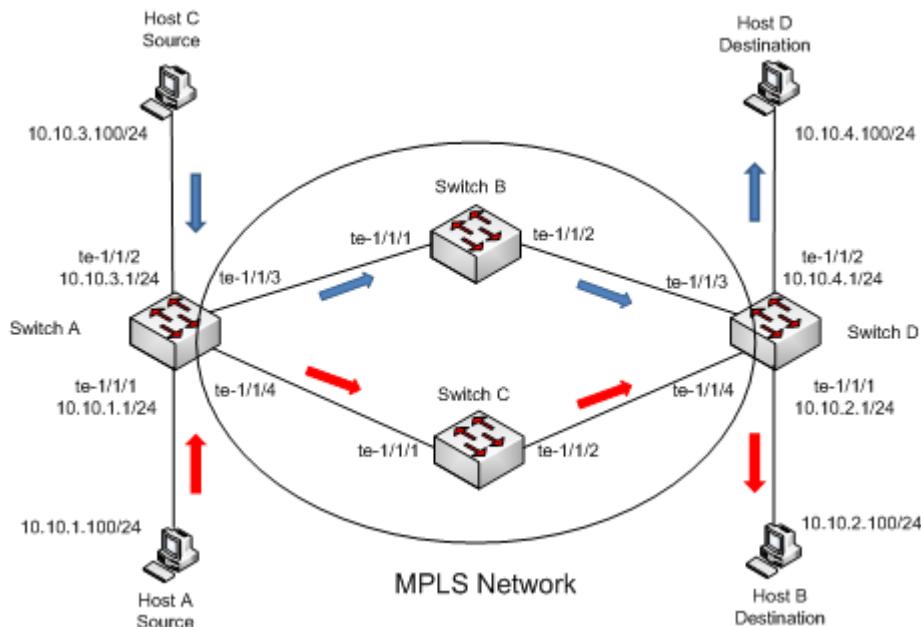


Figure 4-2. MPLS network configuration

Configure Switch-A

In Switch-A, user needs to configure two flows, which will push the MPLS Label 10 and 20 for traffic RED and BLUE, respectively.

```
root@PicOS-OVS# ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
device br0 entered promiscuous mode
root@PicOS-OVS#
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/1 vlan_mode=access tag=1 -- set Interface
te-1/1/1 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/2 vlan_mode=access tag=1 -- set Interface
te-1/1/2 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/3 vlan_mode=access tag=1 -- set Interface
te-1/1/3 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/4 vlan_mode=access tag=1 -- set Interface
te-1/1/4 type=pica8
root@PicOS-OVS#
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=1,dl_type=0x0800,nw_src=10.10.1.100,nw_dst=10.10.2.100,dl_vlan=1,actions=push_mpls:
0x8847,set_field:10->mpls_label,output:4
root@PicOS-OVS#
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=2,dl_type=0x0800,nw_src=10.10.3.100,nw_dst=10.10.4.100,dl_vlan=1,actions=push_mpls:
0x8847,set_field:20->mpls_label,output:3
root@PicOS-OVS#
```

The received packet format in port te-1/1/1 and te-1/1/2 is shown as follows (ingress):

Ethernet	IP Header
----------	-----------

The transmitted packet format to port te-1/1/3 and te-1/1/4 is shown as follows (egress):

Ethernet	MPLS label 10	IP Header
Ethernet	MPLS label 20	IP Header

Configure Switch-B

In Switch-B, user needs to configure one flow, which will SWAP the MPLS Label 20 to 200 for traffic BLUE.

```
root@PicOS-OVS# ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
device br0 entered promiscuous mode
root@PicOS-OVS#
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/1 vlan_mode=access tag=1 -- set Interface
te-1/1/1 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/2 vlan_mode=access tag=1 -- set Interface
te-1/1/2 type=pica8
root@PicOS-OVS#
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=1,dl_type=0x8847,nw_src=10.10.3.100,nw_dst=10.10.4.100,dl_vlan=1,mpls_label=20,act
ions=set_field:200->mpls_label,output:2
root@PicOS-OVS#
```

The transmitted packet format to port te-1/1/2 is shown as follows (egress):

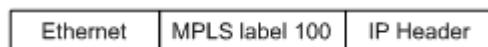


Configure Switch-C

In Switch-C, user needs to configure one flow which will SWAP the MPLS Label 10 to 100 for traffic RED.

```
root@PicOS-OVS# ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
device br0 entered promiscuous mode
root@PicOS-OVS#
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/1 vlan_mode=access tag=1 -- set Interface
te-1/1/1 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/2 vlan_mode=access tag=1 -- set Interface
te-1/1/2 type=pica8
root@PicOS-OVS#
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=1,dl_type=0x8847,nw_src=10.10.1.100,nw_dst=10.10.2.100,dl_vlan=1,mpls_label=10,act
ions=set_field:100->mpls_label,output:2
root@PicOS-OVS#
```

The transmitted packet format to port te-1/1/2 is shown as follows (egress):



Configure Switch-D

In Switch-D, user needs to configure two flows, which will POP the MPLS Label 100 and 200 for traffic RED and BLUE, respectively.

```
root@PicOS-OVS# ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
device br0 entered promiscuous mode
root@PicOS-OVS#
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/1 vlan_mode=access tag=1 -- set Interface
te-1/1/1 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/2 vlan_mode=access tag=1 -- set Interface
te-1/1/2 type=pica8
```

```

root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/3 vlan_mode=access tag=1 -- set Interface
te-1/1/3 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/4 vlan_mode=access tag=1 -- set Interface
te-1/1/4 type=pica8
root@PicOS-OVS#
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=4,dl_type=0x08847,nw_src=10.10.1.100,nw_dst=10.10.2.100,dl_vlan=1,actions=pop_mpls:
0x8847,output:1
root@PicOS-OVS#
root@PicOS-OVS# ovs-ofctl add-flow br0
in_port=3,dl_type=0x08847,nw_src=10.10.3.100,nw_dst=10.10.4.100,dl_vlan=1,actions=pop_mpls:
0x8847,output:2
root@PicOS-OVS#

```

The transmitted packet format to port te-1/1/1 and te-1/1/2 is shown as follows (egress):



Multiple Virtual Bridges

In PicOS OVS, user can create multiple virtual bridges that are independent of one another. One physical port is able to add into only one virtual bridge. Each virtual bridge can be configured as a controller, respectively.

```

root@PicOS-OVS# ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
other-config=datapath-id=0000d80aa99aaaaaa
device br0 entered promiscuous mode
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/1 vlan_mode=access tag=1 -- set Interface
te-1/1/1 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br0 te-1/1/2 vlan_mode=access tag=1 -- set Interface
te-1/1/2 type=pica8
root@PicOS-OVS# ovs-vsctl set-controller br0 tcp:10.10.50.1:6633
root@PicOS-OVS# ovs-vsctl add-br br1 -- set bridge br1 datapath_type=pica8
other-config=datapath-id=0000d80bb99bbbbbb
device br0 entered promiscuous mode
root@PicOS-OVS# ovs-vsctl add-port br1 te-1/1/3 vlan_mode=access tag=1 -- set Interface
te-1/1/3 type=pica8
root@PicOS-OVS# ovs-vsctl add-port br1 te-1/1/4 vlan_mode=access tag=1 -- set Interface
te-1/1/4 type=pica8
root@PicOS-OVS# ovs-vsctl set-controller br1 tcp:10.10.50.2:6633

```

SSL Connection to Controller

If user wants to create an SSL connection with the controller in PicOS switch, please follow the following steps:

Switch

```
root@PicOS-OVS#sudo apt-get install openssl
```

Reading package lists... Done

Building dependency tree

Reading state information... Done

Suggested packages:

ca-certificates

The following NEW packages will be installed:

openssl

0 upgraded, 1 newly installed, 0 to remove and 17 not upgraded.

Need to get 696 kB of archives.

After this operation, 1070 kB of additional disk space will be used.

WARNING: The following packages cannot be authenticated!

openssl

Authentication warning overridden.

Get:1 http://ftp.debian.org/debian/ stable/main openssl powerpc 1.0.1e-2 [696 kB]

Fetched 696 kB in 5s (131 kB/s)

Selecting previously unselected package openssl.

(Reading database ... 17049 files and directories currently installed.)

Unpacking openssl (from .../openssl_1.0.1e-2_powerpc.deb) ...

Processing triggers for man-db ...

Setting up openssl (1.0.1e-2) ...

root@PicOS-OVS#ovs-pki init

/ovs/bin/ovs-pki: /ovs/var/lib/openvswitch/pki already exists and --force not specified

root@PicOS-OVS#**ovs-pki init --force**

Creating controllerca...

Creating switchca...

root@PicOS-OVS#cd /ovs/var/lib/openvswitch/pki/controllerca

root@PicOS-OVS#**ovs-pki req+sign ctl controller**

ctl-req.pem Mon Jan 13 03:26:05 UTC 2014

fingerprint 1cbf63b21301f33d9b4aa30540bff492f15bcfd3

root@PicOS-OVS#ls

ca.cnf careq.pem crl ctl-cert.pem ctl-req.pem index.txt.attr index.txt.old private serial.old

cacert.pem certs crlnumber ctl-privkey.pem index.txt index.txt.attr.old newcerts serial

root@PicOS-OVS#ls ctl-privkey.pem ctl-cert.pem

ctl-cert.pem ctl-privkey.pem

root@PicOS-OVS#cd /ovs/var/lib/openvswitch/pki/switchca

root@PicOS-OVS#**ovs-pki req+sign sc switch**

sc-req.pem Mon Jan 13 03:26:54 UTC 2014

fingerprint 65ed449bee94b8e7b8ba7da6f6584afdf2f9cc2fb

root@PicOS-OVS#ls sc-privkey.pem sc-cert.pem

```

sc-cert.pem sc-privkey.pem
root@PicOS-OVS#
root@PicOS-OVS#scp /ovs/var/lib/openvswitch/pki/controllerca/ctl-cert.pem
10.10.50.41:/home/build

The authenticity of host '10.10.50.41 (10.10.50.41)' can't be established.
ECDSA key fingerprint is e6:04:3b:c8:24:36:c7:dd:c1:06:6a:69:e2:3b:82:2f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.50.41' (ECDSA) to the list of known hosts.

root@10.10.50.41's password:
ctl-cert.pem 100% 4063 4.0KB/s 00:00

root@PicOS-OVS#scp /ovs/var/lib/openvswitch/pki/controllerca/ctl-privkey.pem
10.10.50.41:/home/build

root@10.10.50.41's password:
ctl-privkey.pem 100% 1675 1.6KB/s 00:00

root@PicOS-OVS#scp /ovs/var/lib/openvswitch/pki/switchca/cacert.pem 10.10.50.41:/home/build
root@10.10.50.41's password:
cacert.pem 100% 4028 3.9KB/s 00:00

root@PicOS-OVS#ovs-vsctl set-ssl /ovs/var/lib/openvswitch/pki/switchca/sc-privkey.pem
/ovs/var/lib/openvswitch/pki/switchca/sc-cert.pem
/ovs/var/lib/openvswitch/pki/controllerca/cacert.pem

root@PicOS-OVS#ovs-vsctl del-br br0
ovs-vsctl: no bridge named br0

root@PicOS-OVS#ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
root@PicOS-OVS#ovs-vsctl set-controller br0 ssl:10.10.50.41:6633
root@PicOS-OVS#

# Controller

root@dev-41:/home/build# ryu-manager --ctl-privkey ./ctl-privkey.pem --ctl-cert ./ctl-cert.pem
--ca-certs ./cacert.pem --verbose
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
BRICK ofp_event
CONSUMES EventOFPPortDescStatsReply
CONSUMES EventOFPSwitchFeatures
CONSUMES EventOFPErrorMsg
CONSUMES EventOFPEchoRequest

```

CONSUMES EventOFPHello

connected socket:<eventlet.green.ssl.GreenSSLocket object at 0x9f1ebfc> address:('10.10.50.155', 48508)

hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x9ecf1ec>

move onto config mode

switch features ev version: 0x4 msg_type 0x6 xid 0xa2f1cf23

OFPSwitchFeatures(auxiliary_id=0,capabilities=7,datapath_id=7461368339596857098L,n_buffers=256,n

move onto main mode

PicOS OpenFlow Tutorials

Basic Bridge Configuration

- Basic Bridge Introduction
- Power On Configuration
- Configure Switch
- Configure Bridge
- Configure Port
- Default Bridge Behavior
- OVS Commands Reference

Basic Flow Configurations

- Flow Introduction
- Modify Default Flow
- Uni-directional Flow
- 1-to-Many Multicasting
- Many-to-One Aggregation
- OVS Commands Used in this Tutorial
- Packet Address File

Connection to a RYU Controller

- RYU Introduction
- Introduce RYU Open Flow Controller
- Configure OVS for RYU OpenFlow Controller
- Controller-OVS Interaction
- RYU Simple Switch Application
- Open Flow Message Type
- RYU Guide OVS Commands Reference
- Using TTP (router profile) with RYU Controller

Connection to OpenDaylight Controller

- OpenDaylight Introduction
- Introduction to the OpenDaylight OpenFlow Controller

- Configure OVS for OpenDaylight Open Flow Controller
- OpenDaylight Controller-OVS Interaction
- OpenDaylight Simple Switch Application
- Message Type of Open Flow
- OVS Commands Reference 04

Connection to a Floodlight Controller

- Floodlight Controller Introduction
- Floodlight Open Flow Controller
- Test Topology
- Configure OVS
- Launch Floodlight
- Floodlight REST Interface

Configuration Guide for Atrium Stack on ONOS Controller

- ONOS Introduction
- Installation Guide
- ONOS Configuration Guide
- Quagga Configuration Guide
- PicOS Configuration Guide

Basic Bridge Configuration

- Basic Bridge Introduction
- Power On Configuration
- Configure Switch
- Configure Bridge
- Configure Port
- Default Bridge Behavior
- OVS Commands Reference

Basic Bridge Introduction

This document provides instructions on how to configure open switches running Pica8 PicOS software in various application scenarios. This document assumes minimal to no knowledge of OVS (Open vSwitch) and OpenFlow.

After studying this document, user will be able to configure OpenFlow on PicOS switches, know how to optimize the configuration for an application environment, and understand more about Open vSwitch and OpenFlow.

This document provides instructions for completing the following tasks:

- Configuring a PicOS switch as an OVS/OpenFlow switch.
- Creating bridges, adding ports, displaying bridge and port status/statistics, and displaying the OVS database.
- Configuring flow tables for uni-directional traffic, bi-directional traffic, traffic switching, one-to-many multicasting, mirroring, filtering, many-to-one aggregation etc.
- Configuring PicOS switches to connect with the Ryu controller

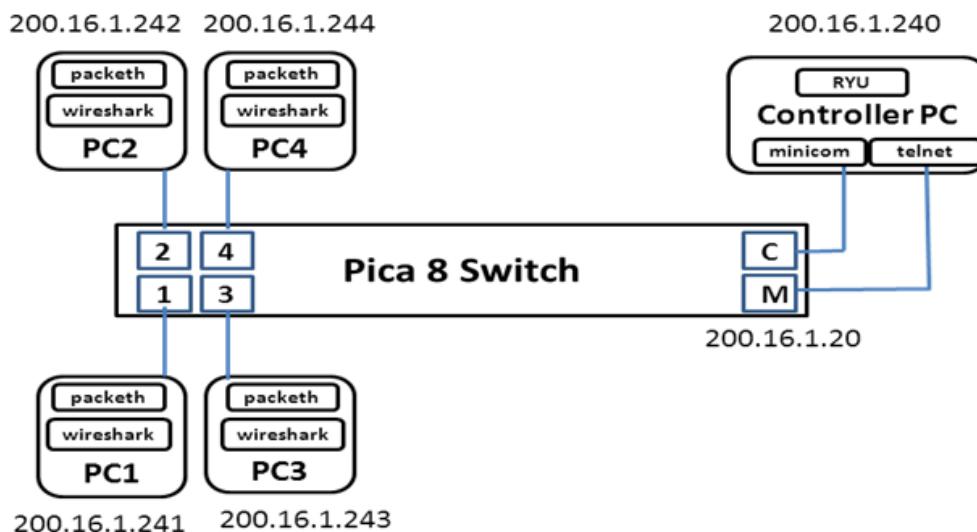


Figure 1 Test bed configuration

In this document, the system configuration depicted in Figure 1 includes:

- A Pica8 P-3295 open switch with 48 x 1GbE and 4 x 10GbE uplinks
- 5 Linux PCs running Ubuntu 12.4.1 LTS, one is connected to the management LAN port (RJ45) and console port (RJ45F); this PC is referred to the controller PC. The OpenFlow controller will be running on this PC. Four PCs are connected to 1GbE port 1 to 4 and serve as a data terminal for generating and monitoring traffic
- Tools installed on all the PCs are listed below. They can be installed through Linux installation utility *apt-get*
 - Terminal emulator minicom
 - Traffic monitoring tool Wireshark
 - Packet generator Packeth
 - ftp and ftppd
 - telnet and telnetd

Power On Configuration

Connect with the console port of the PicOS switch as described in Initial Switch Access . Once the console connection is set up, power on the switch.

Figure 2 shows the console output. Do not hit any keys until the booting choice menu appears. Enter **2** to boot into Open vSwitch mode. Next, the switch asks if the switch configuration should be done manually, enter **no** to enter the automatic mode. In this mode, the OVS processes will start automatically with default configuration such as log file, etc.

Next, the switch static IP address is entered. In this configuration, subnet 200.16.1.x is used. User can choose their own subnet address at this point. After the static IP address, a gateway IP address is entered. Next, an Open vSwitch configuration database name is required to store all the configuration information. In this example, *ovs-vswitchd.conf.db* database name is used. If the database name does not exist from previous configuration, it will be created in the default /ovs directory based on a database schema defined in /ovs/share/openvswitch/vswitch.ovsschema. Multiple databases can be created to provide different configurations; but only one database can be entered during this start up sequence. The OVS processes can be stopped and restarted manually, once the system is running. They can also be configured as cron processes. The database is persistent. The configuration stored in the database will be restored once the OVS processes start.

Figure 2 – Power on console output

```

david@ubuntu: ~
net.netfilter.nf_conntrack_acct = 1
net.ipv6.conf.all.forwarding = 1
System initiating...Please wait...
Please choose which to start: Pica8 XorPlus, OpenFlow, or System shell:
  (Will choose default entry if no input in 10 seconds.)
  [1] Pica8 XorPlus  * default
  [2] Open vSwitch
  [3] System shell
  [4] Boot menu editor
Enter your choice (1,2,3,4):2
Open vSwitch is selected.

Note: Defaultly, the OVS server is runned with static local management IP and port 6633.
      The default way of vswitch connecting to server is PTCP.
      If you do not want default configuration, choose manual start!

sh: you need to specify whom to kill
Do you want start the OVS by manual? (yes/no) no

Please set a static IP and netmask for the switch (e.g. 120.0.0.10/24) : 200.16.1.20/24
Please set the gateway IP (e.g 172.168.1.2):200.16.1.1
Specify the file name of database for server, if not exist, it will be created: ovs_vswitchd.conf.db
System have found the database file!

Waiting for eth0 up .....
Done!

Adding the gateway .....
Run the ovswitchd with 200.16.1.20 and port 6633 with ptcp .....
Waiting for ovswitchd .....
device br0 entered promiscuous mode
Done!

Run the ovs_vswitchd with 200.16.1.20 and port 6633 with ptcp .....
Waiting for ovs_vswitchd .....
device br0 entered promiscuous mode
Done!

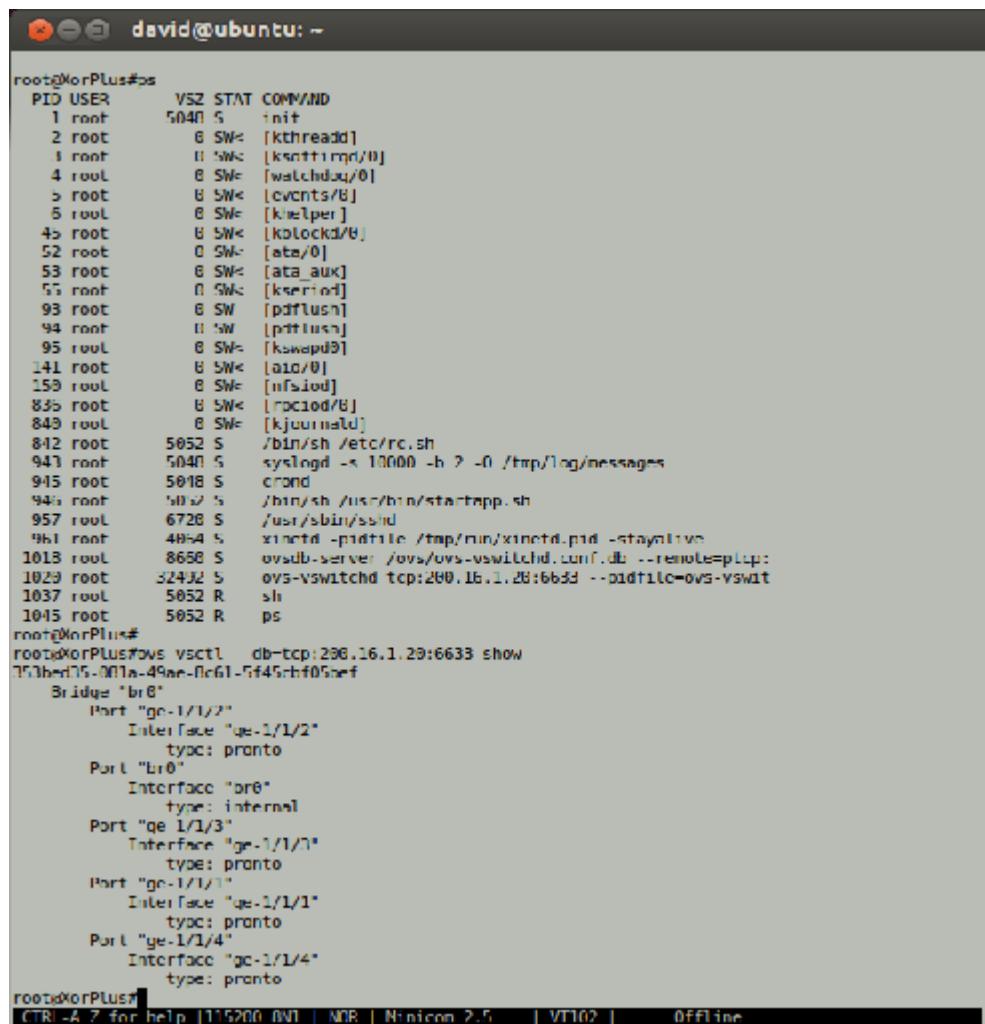
Startup finished!

```

In this example, the *ovs-vswitchd.conf.db* was used in a previous configuration. Therefore, the system found the database and created the initial configuration which will be shown later. In Figure 1, the management LAN port on the switch is *eth0*. *eth0* is connected to the *eth0* in the controller PC to allow the controller PC to *telnet* into the switch without the limitation of the console. In this configuration, all PCs are configured with static IP addresses to form an isolated environment for testing.

Next, the switch continues the boot sequence. Pay attention to the console messages regarding the *ovsdb-server* and *ovs-vswitchd*. They are the ovsdb server and ovs switch daemons. The IP address is the switch IP address and the 6633 is the default port number used to communicate with the ovs switch database server process. A different port number can be set through the manual configuration steps. Reference the PicOS Open vSwitch Configuration Guide for manual configuration steps.

Figure 3 – Switch processes and bridge information



```

root@NorPlus#ps
  PID USER      VSZ STAT COMMAND
    1 root      5040 S   init
    2 root      0 SW-  [kthreadd]
    3 root      0 SW-  [ksoftirqd/0]
    4 root      0 SW-  [watchdog/0]
    5 root      0 SW-  [cvcnts/0]
    6 root      0 SW-  [khelper]
   45 root      0 SW-  [kblockd/0]
   52 root      0 SW-  [ata/0]
   53 root      0 SW-  [ata_aux]
   55 root      0 SW-  [kservmon]
   93 root      0 SW  [pdflush]
   94 root      0 SW  [pdflush]
   95 root      0 SW-  [kswapd0]
 141 root      0 SW-  [aio/0]
 159 root      0 SW-  [nfssiod]
 835 root      0 SW-  [rprocod/0]
 840 root      0 SW-  [kjournald]
 842 root  5052 S /bin/sh /etc/rc.sh
 943 root  5040 S syslogd -s 10000 -h ? -o /tmp/log/messages
 945 root  5018 S crond
 946 root  5017 S /bin/sh /usr/bin/startapp.sh
 957 root  6720 S /usr/sbin/sshd
 961 root  4054 S xinetd -pidfile /tmp/min/xinetd.pid -stayalive
1018 root  8658 S ovsdb-server /ovs/ovs-vswitchd.conf.db --remote=plcp:
1029 root  52402 S ovs-vswitchd tcp:200.16.1.20:6633 --pidfile=ovs-vswit
1037 root  5052 R sh
1045 root  5052 R ps
root@NorPlus#
root@NorPlus#ovs-vsctl db=tcp:200.16.1.20:6633 show
75:herd5-011a-49ae-0c61-5f45:hf05:ef
  Bridge "br0"
    Port "ge-1/1/2"
      Interface "ge-1/1/2"
        type: portno
    Port "br0"
      Interface "br0"
        type: internal
    Port "qe-1/1/3"
      Interface "qe-1/1/3"
        type: portno
    Port "ge-1/1/1"
      Interface "ge-1/1/1"
        type: portno
    Port "ge-1/1/4"
      Interface "ge-1/1/4"
        type: portno
root@NorPlus#
```

The terminal window shows the output of the 'ps' command listing various system processes, and the 'ovs-vsctl show' command output. The 'ovs-vsctl show' output indicates that a bridge 'br0' has been created with four ports: 'ge-1/1/2', 'br0', 'qe-1/1/3', and 'ge-1/1/1'. The 'ge-1/1/4' port is also listed but appears to be inactive or not yet connected.

The IP address and the port number is often used in the *ovs-vsct*/and *ovs-ofct*/commands discussed in later sections. The *ovs-vswitchd.conf.db* was used in a previous configuration, which contains a bridge *br0* with 4 1GbE ports. After the *ovs-vswitchd*process started, a message on *device br0* is shown to indicate the bridge has been created. At this point, the switch is up and running. The root level *root@PicOS-OVS#shell* prompt is shown and ready for user input. Multiple telnet windows can be started from the controller PC to login to the switch. The user id is *root* and the default password is *pica8*.

Configure Switch

Use Linux command ***ps -A*** to show the running processes. The *ovsdb-server* and *ovs-vswitchd* are there to indicate the ovs switch is ready for operation. Next, print the content of the switch database by using the ***ovs-vsctl show*** command to dump the switch configuration. It shows the database id, a bridge named *br0* with four 1GbE ports, and an internal port.

In most start up cases, a new database name of administrator's choice will be entered. As a result, an empty database is created. The show command will just show the database id. If a new database is created, the next step should be skipped, so move on to the *add-br* command. The following example will demonstrate how to delete the old bridge by utilizing the ***ovs-vsctl del-br br0*** command. Check the database content by using the show command which should just show the database id. The following example shows how to create a bridge and add ports for the bridge.

```
root@PicOS-OVS$ 
root@PicOS-OVS$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8

device br0 entered promiscuous mode
root@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/1 vlan_mode=trunk tag=1 -- set interface
te-1/1/1 type=pica8
root@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/2 vlan_mode=trunk tag=1 -- set interface
te-1/1/2 type=pica8
root@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/3 vlan_mode=trunk tag=1 -- set interface
te-1/1/3 type=pica8
root@PicOS-OVS$ovs-vsctl add-port br0 te-1/1/4 vlan_mode=trunk tag=1 -- set interface
te-1/1/4 type=pica8
root@PicOS-OVS$ 
root@PicOS-OVS$ovs-vsctl show
d4d12890-c07a-4303-80cc-c6f79cf3af7
    Bridge "br0"
        Port "te-1/1/3"
            tag: 1
            Interface "te-1/1/3"
                type: "pica8"
        Port "te-1/1/4"
            tag: 1
            Interface "te-1/1/4"
                type: "pica8"
        Port "br0"
            Interface "br0"
                type: internal
        Port "te-1/1/2"
            tag: 1
            Interface "te-1/1/2"
                type: "pica8"
        Port "te-1/1/1"
            tag: 1
            Interface "te-1/1/1"
                type: "pica8"
root@PicOS-OVS$
```

Configure Bridge

To create a new bridge, issue ***ovs-vsctl add-br br0 – set bridge br0 datapath_type=pica8*** command. In the following example, the bridge needs four 1GbE ports. To add each 1GbE port to the bridge, issue ***ovs-vsctl add-port br0 ge-1/1/1 – set interface ge-1/1/1 type=pica8*** command 4 times to add ge-1/1/1

to ge1/1/4 to the bridge. To verify the configuration, use ***ovs-vsctl show*** to show the database content. As shown in the screen shot, the bridge should have four 1GbE ports and an internal port. Next, monitor the port status and examine the port configuration with ***ovs-ofctl show br0*** command. The following commands are to show the configuration of the bridge.

```
root@PicOS-OVS$ ovs-ofctl show br0
OFPT_FEATURES_REPLY (OF1.4) (xid=0x2): dpid:5e3ec80aa9ae0a66
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS
OFPST_PORT_DESC reply (OF1.4) (xid=0x4):
1(te-1/1/1): addr:c8:0a:a9:ae:0a:66
config: 0
state: LINK_DOWN
current: FIBER
advertised: 1GB-FD FIBER
supported: 1GB-FD 10GB-FD FIBER AUTO_NEG
speed: 0 Mbps now, 10000 Mbps max
2(te-1/1/2): addr:c8:0a:a9:ae:0a:66
config: 0
state: LINK_DOWN
current: FIBER
advertised: 1GB-FD FIBER
supported: 1GB-FD 10GB-FD FIBER AUTO_NEG
speed: 0 Mbps now, 10000 Mbps max
3(te-1/1/3): addr:c8:0a:a9:ae:0a:66
config: 0
state: LINK_DOWN
current: FIBER
advertised: 1GB-FD FIBER
supported: 1GB-FD 10GB-FD FIBER AUTO_NEG
speed: 0 Mbps now, 10000 Mbps max
4(te-1/1/4): addr:c8:0a:a9:ae:0a:66
config: 0
state: LINK_DOWN
current: FIBER
advertised: 1GB-FD FIBER
supported: 1GB-FD 10GB-FD FIBER AUTO_NEG
speed: 0 Mbps now, 10000 Mbps max
LOCAL(br0): addr:c8:0a:a9:ae:0a:66
config: 0
state: LINK_UP
current: 10MB-FD COPPER
supported: 10MB-FD COPPER
speed: 10 Mbps now, 10 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.4) (xid=0x6): frags=normal miss_send_len=0
root@PicOS-OVS$ 
root@PicOS-OVS$ 
root@PicOS-OVS$ ovs-ofctl dump-ports br0
OFPST_PORT reply (OF1.4) (xid=0x2): 5 ports
port 1: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=0, bytes=0, drop=0, errs=0, coll=0
duration=228.085s
port 2: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=0, bytes=0, drop=0, errs=0, coll=0
duration=216.224s
port 3: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=0, bytes=0, drop=0, errs=0, coll=0
duration=208.941s
port 4: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=0, bytes=0, drop=0, errs=0, coll=0
duration=199.026s
port LOCAL: rx pkts=7, bytes=746, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=0, bytes=0, drop=0, errs=0, coll=0
duration=246.761s
root@PicOS-OVS$
```

In the example provided, port state is *LINKDOWN* because in the example set up, the cable has not been connected yet. The Pica8 1GbE port supports RJ45 copper connector and auto negotiation from 10 MB to 1 GB speed range. Next, examine the port statistics, using the `_ovs-ofctl dump-ports br0` command. It shows the RX and TX statistics: since the link is down, no packets are sent or received, and all counters should be zeros.

Configure Port

A port can be added, deleted, turned up, or turned down, dynamically. The `add-port` command has been tested. To delete a port, use `ovs-vsctl del-port br0 ge-1/1/1`. Port state can also be modified with the `modport` command `ovs-ofctl mod-port br0 ge-1/1/1 action`. The keyword `action` can be one of the following parameters:

- Up or down

Enable or disable the interface. This is equivalent to `ifconfig up` or `ifconfig down` on a Linux system.

- Stp or nostp

Enable or disable 802.1D spanning tree protocol (STP) on the interface. OpenFlow implementations that don't support STP will refuse to enable it.

- Receive or noreceive/receivestp or noreceivestp

Enable or disable OpenFlow processing of packets received on this interface. When packet processing is disabled, packets will be dropped instead of being processed through the OpenFlow table. The receive or noreceive setting applies to all packets except 802.1D spanning tree packets, which are separately controlled by receivestp or noreceivestp.

- Forward or noforward

Allow or disallow forwarding of traffic to this interface. By default, forwarding is enabled.

- Flood ornoflood

Controls whether an OpenFlow flood action will send traffic out this interface. By default, flooding is enabled. Disabling flooding is primarily useful to prevent loops when a spanning tree protocol is not in use.

- packetin or nopacketin

Controls whether packets received on this interface that do not match a flow table entry generate a "packet in" message to the OpenFlow controller. By default, "packet in" messages are enabled.

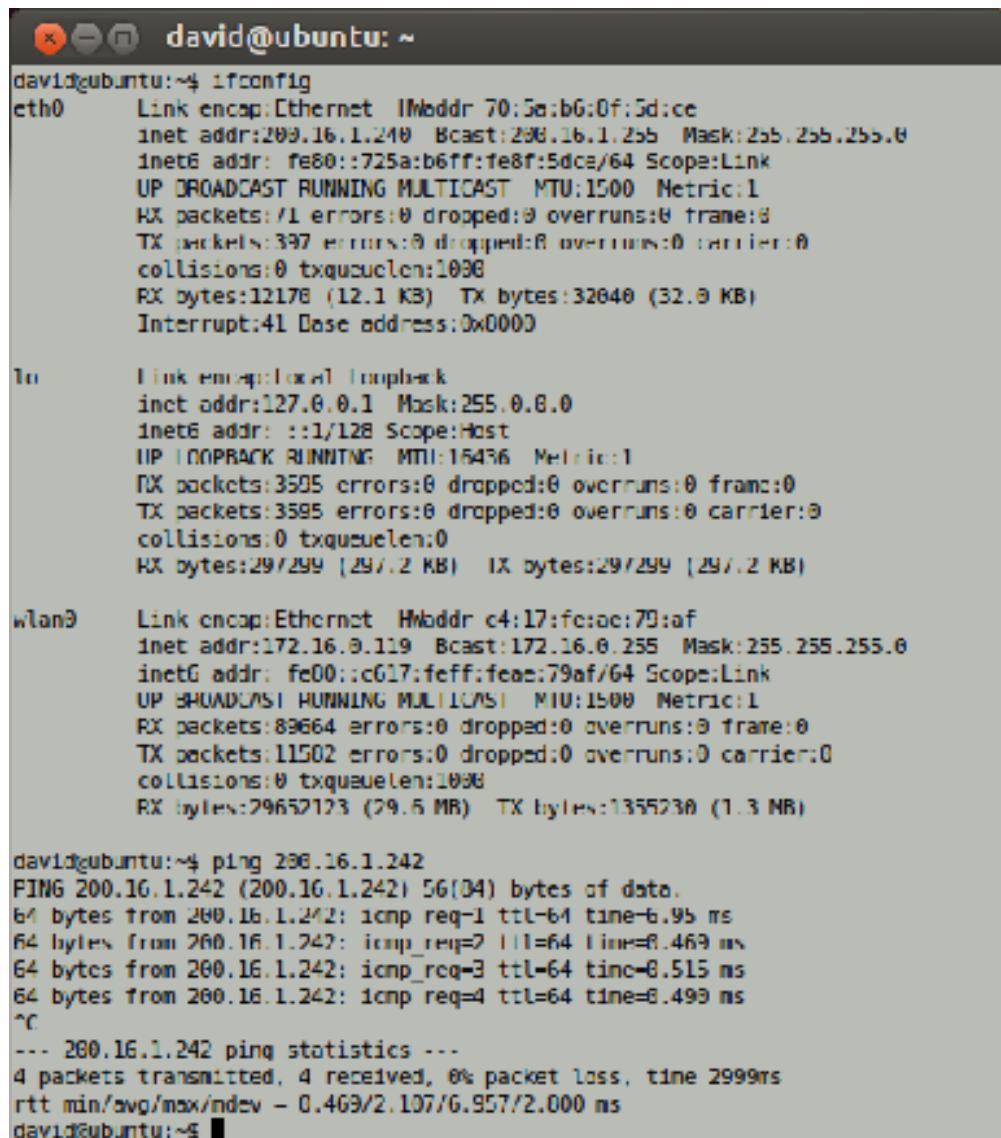
Again, the show command displays (among other information) the configuration that `modport` changes.

Default Bridge Behavior

If the newly created bridge does not connect to the OpenFlow controller, it will behave as a simple L2 switch, which floods packets received from a port to all other ports. This behavior is implemented with a default low priority flow added at bridge creation time. The flow can be shown by using the `ovs-ofctl dump-flows br0` command. The flow will be shown as priority 0 and actions=NORMAL. Action NORMAL

means the packet is subject to the device's normal L2/L3 processing. This action is not implemented by all OpenFlow switches. Next, connect 2 PCs to switch port 1 and port 2 with an Ethernet cable. Once the PCs are connected, the port state should be changed to LINK_UP soon after the cable is connected. Once both links are up, use ping to test the connectivity.

Figure 4 – Ping test



```
david@ubuntu:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 70:5e:b6:0f:5d:ce
          inet addr:200.16.1.240 Bcast:200.16.1.255 Mask:255.255.255.0
          inet6 addr: fe80::725a:b6ff:fe8f:5dce/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:397 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12170 (12.1 KB) TX bytes:32640 (32.0 KB)
          Interrupt:41 Base address:0x0000

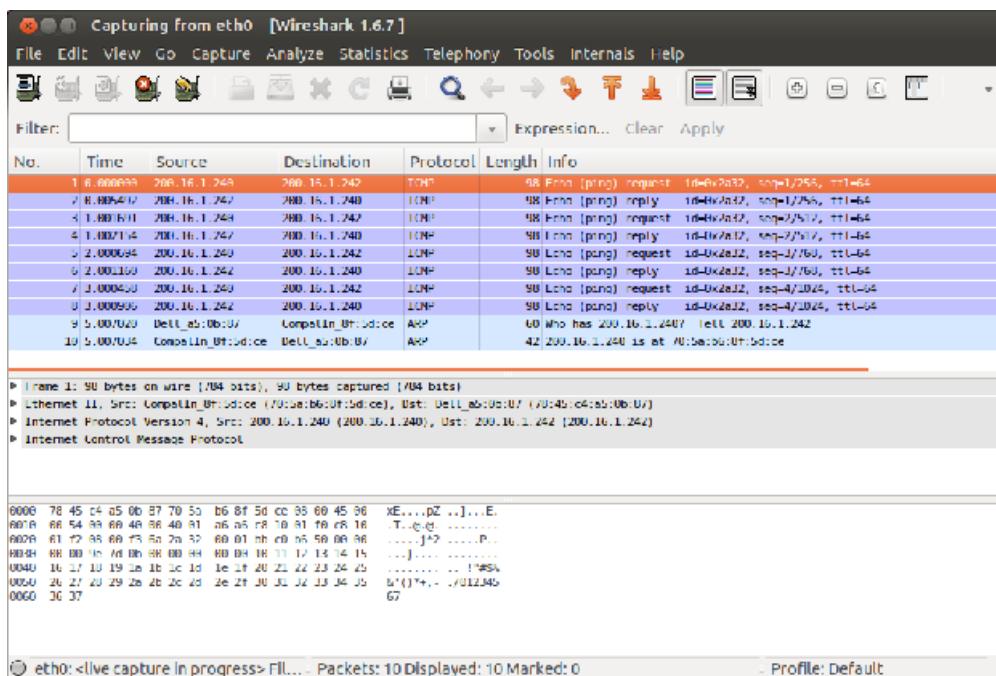
lo      Link encap:loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:3505 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3505 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:29/299 (29/.2 KB) TX bytes:29/299 (29/.2 KB)

wlan0     Link encap:Ethernet HWaddr c4:17:fc:ac:79:af
          inet addr:172.16.0.119 Bcast:172.16.0.255 Mask:255.255.255.0
          inet6 addr: fe00::c017:feff:feac:79af/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:89664 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11502 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:29652123 (29.5 MB) TX bytes:1355230 (1.3 MB)

david@ubuntu:~$ ping 200.16.1.242
PING 200.16.1.242 (200.16.1.242) 56(44) bytes of data.
64 bytes from 200.16.1.242: icmp_req=1 ttl=64 time=6.95 ms
64 bytes from 200.16.1.242: icmp_req=2 ttl=64 time=6.469 ms
64 bytes from 200.16.1.242: icmp_req=3 ttl=64 time=6.515 ms
64 bytes from 200.16.1.242: icmp_req=4 ttl=64 time=6.499 ms
^C
--- 200.16.1.242 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/stdv - 0.469/2.107/6.957/2.000 ms
david@ubuntu:~$
```

In this example, another Linux tool, wireshark, is also used to capture the packets sent and received on eth0. On the wireshark screen, a total of 4 pairs ping requests/replies are captured along with some arp packets. We can connect other PCs to the switch now, and ping should work for all PCs. In our set up, telnetd and ftpd are installed in our linux PC. User can try the telnet and ftp sessions to test the connectivity and bridge functionalities.

Figure 5 – ICMP request/reply



At this point, the switch is powered on, and the initial switch configuration without an open flow controller is completed. Proceed to Open SDN: Started Kit – Configure flows for flow manipulation.

OVS Commands Reference

```

ovs-vsctl show
ovs-ofctl show br0
ovs-ofctl dump-ports br0
ovs-vsctl list-ports br0
ovs-vsctl list-ifaces br0
ovs-ofctl dump-flows br0
ovs-vsctl list-br
ovs-vsctl add-br br0 - set bridge br0 datapath_type=pica8
ovs-vsctl del-br br0
ovs-vsctl set Bridge br0 stp_enable=true
ovs-vsctl add-port br0 ge-1/1/1 - set interface ge-1/1/1 type=pica8
ovs-vsctl add-port br0 ge-1/1/2 - set interface ge-1/1/2 type=pica8
ovs-vsctl add-port br0 ge-1/1/3 - set interface ge-1/1/3 type=pica8
ovs-vsctl add-port br0 ge-1/1/4 - set interface ge-1/1/4 type=pica8
ovs-vsctl add-port br0 ge-1/1/1 type=pronto options:link_speed=1G
ovs-vsctl del-port br0 ge-1/1/1
ovs-ofctl del-flows br0

```

Basic Flow Configurations

- Flow Introduction
- Modify Default Flow
- Uni-directional Flow
- 1-to-Many Multicasting
- Many-to-One Aggregation

- OVS Commands Used in this Tutorial
- Packet Address File

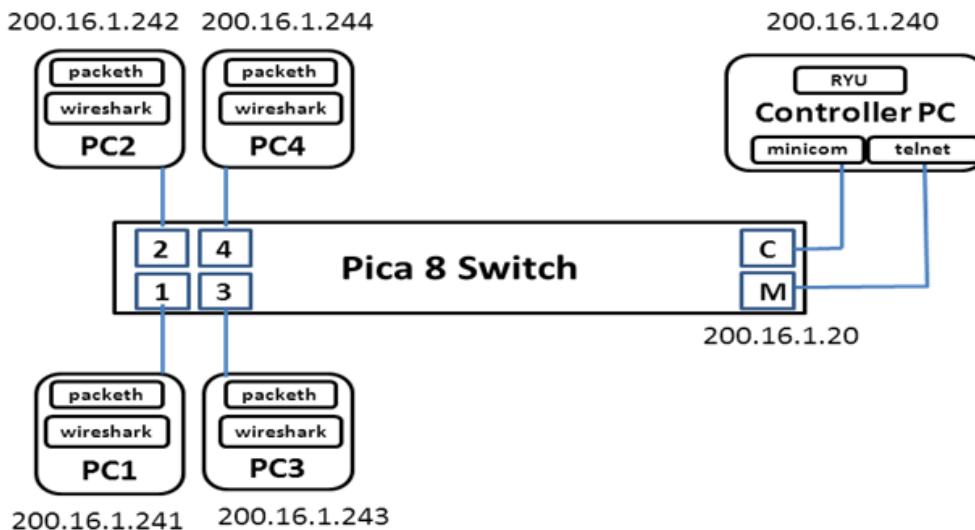
Flow Introduction

This document provides instructions on how to configure Pica8's open switches to work in various application scenarios. This document assumes the reader with minimal to no knowledge of the Open Virtual Switch (OVS) implementation defined by <http://openvswitch.org/> or the OpenFlow protocol defined by <https://www.opennetworking.org/>. After studying this guide, user will have the tools needed to configure Pica8's open switches as an OpenFlow switch. User will also gain insights on how to optimize the configuration to work in an application environment, while also learning about OVS and the OpenFlow protocol.

This starter kit provides screen shots and a list of off-the-shelf applications needed to complete the configuration, as well as tips on problems user may encounter during setup. More documents or cookbooks on other subjects will be published periodically. This document provides a tutorial on how to:

- Configure a Pica8 switch as an OVS OpenFlow switch
- Create bridges; add ports; show bridge, port statistics, and status; introduction to the OVS database
- Configure flow tables for uni-directional, bi-directional, traffic switching, one-to-many multi-casting, mirroring, filtering, many-to-one aggregation, etc.
- Configure Pica8 OpenFlow switches to interface with the RYU OpenFlow Controller

Figure 1 – Test bed configuration



In this document, the system configuration depicted in Figure 1 includes:

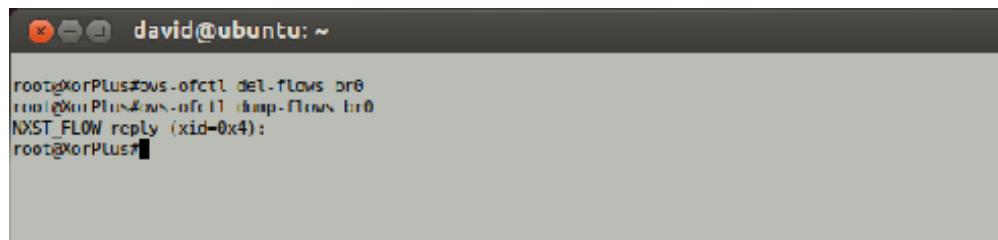
- A Pica8 P-3295 open switch with 48 x 1 GbE and 4 x 10 GbE uplinks
- 5 Linux PCs running Ubuntu 12.4.1 LTS, one is connected to the management LAN port (RJ45) and console port (RJ45F). This PC is referred to the controller PC. The OpenFlow controller will be running on this PC. Four PCs are connected to 1GbE port 1 to 4 and serve as a data terminal for generating and monitoring traffic

- Tools installed on all the PCs are listed below. They can be installed through Linux installation utility *apt-get*
 - Terminal emulator minicom
 - Traffic monitoring tool Wireshark
 - Packet generator Packeth
 - ftp and ftptd
 - telnet and telnetd

Modify Default Flow

Next, the process of disabling flooding behavior, starting configuration of flow table, and manipulating packet flows will be shown. Use the *ovs-ofctl del-flows br0* command to delete the default flow. Dump the flow table, and no flow entry is shown.

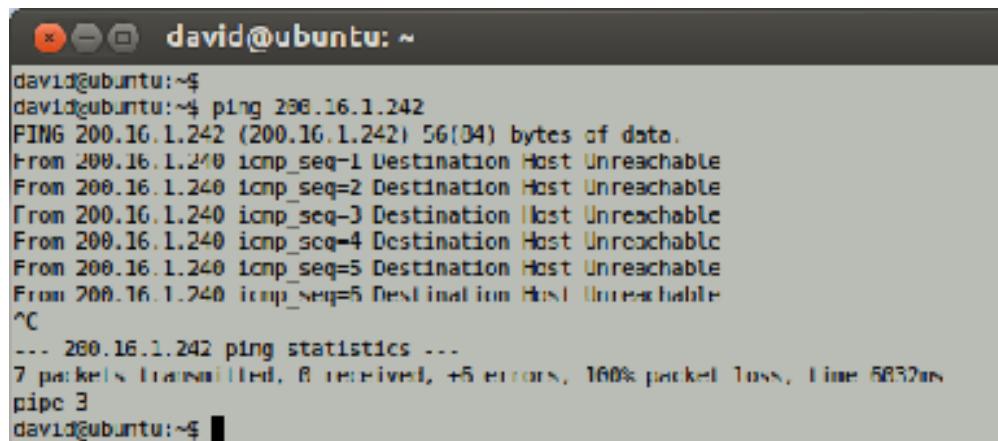
Figure 2 – Delete flows and dump flows



```
david@ubuntu: ~
root@XorPlus:~# ovs-ofctl del-flows br0
root@XorPlus:~# ovs-ofctl dump-flows br0
NXST_FLOW reply (xid=0x4):
root@XorPlus:~#
```

At this point, the ping should stop working because the flooding has been disabled. If interested, user can delete the bridge and re-create it with four ports. Then, the ping should work again.

Figure 3 – Pings



```
david@ubuntu: ~$ ping 200.16.1.242
PING 200.16.1.242 (200.16.1.242) 56(44) bytes of data.
From 200.16.1.240 icmp_seq=1 Destination Host Unreachable
From 200.16.1.240 icmp_seq=2 Destination Host Unreachable
From 200.16.1.240 icmp_seq=3 Destination Host Unreachable
From 200.16.1.240 icmp_seq=4 Destination Host Unreachable
From 200.16.1.240 icmp_seq=5 Destination Host Unreachable
From 200.16.1.240 icmp_seq=6 Destination Host Unreachable
^C
--- 200.16.1.242 ping statistics ---
7 packets transmitted, 0 received, +6 errors, 100% packet loss, time 5832ms
pipe 3
david@ubuntu: ~$
```

Uni-directional Flow

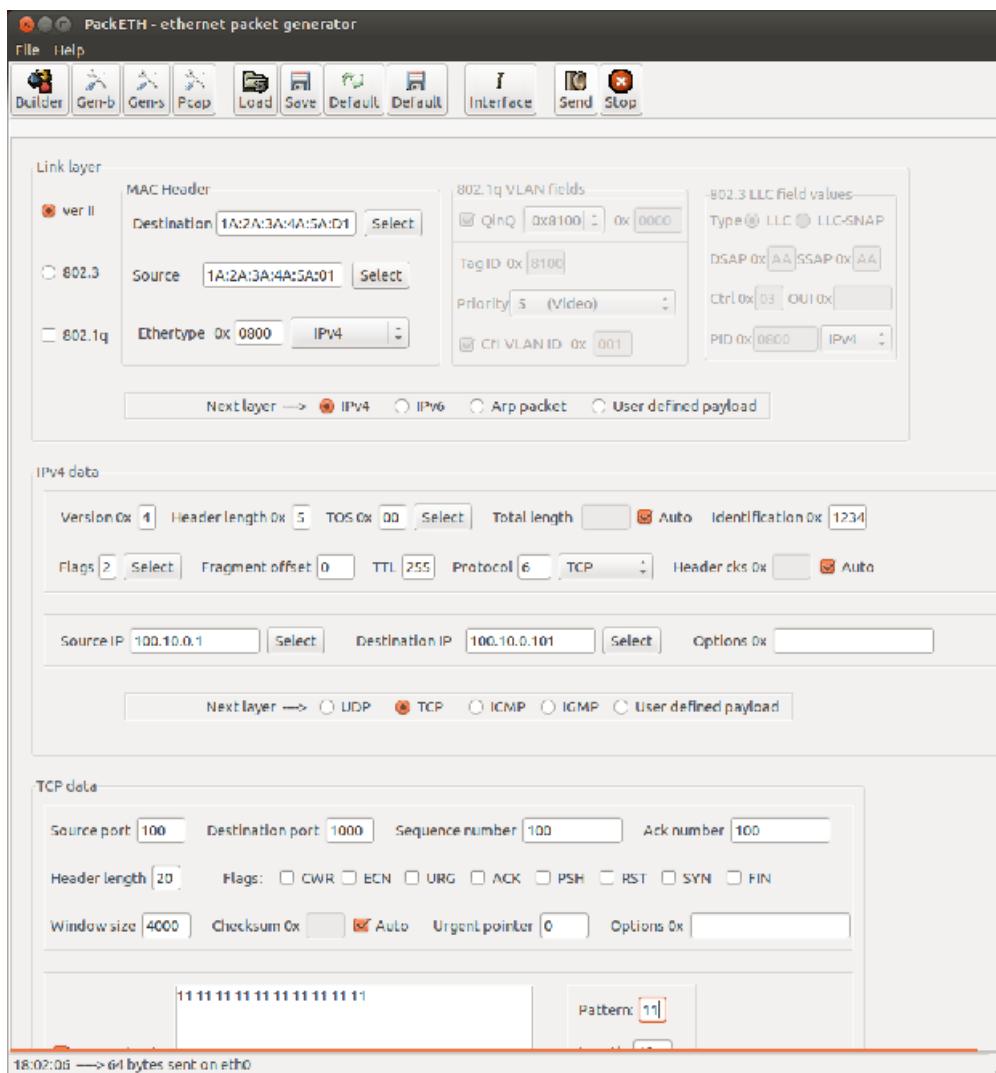
Before running uni-directional flow, user needs a packet generator to work with *wireshark* for packet generation and capturing. In this starter kit, a linux tool *packeth* is used for packet generation. The *packeth* can be installed via the linux command `sudo apt-get install packeth`. To use the *packeth*, an address file needs to be created as the address database for packet creation. The format is <IP address>:<MAC address>:<Names>. A sample address file is provided in the appendix (<http://intranet.pica8.com/display/picos25sp/Packet+address+file>).

Next, user can create some packets to be used in the later test scenarios. Start the *packeth*, click the *builder* button to enter the tab for creating a packet. The next screen shows the test packet we built for this test. Fill in each field using the *select* button or by entering the value. Each packet includes information in link layer, IP layer, and TCP payload. Once the packet is built, click the *interface* button then select *eth0* as the interface.

Next, create a uni-directional flow from port 1 to port 2 using ***ovs-ofctl add-flow br0 in_port=1,actions=output:2*** command. Then, use the dump flow command to show the flow. The command added a flow into *br0* to forward all packets coming in from *in_port=1* to *out_port=2*. Next, start the *wireshark* application to capture all packets on *eth0* for both PC1 and PC2.

Next, return to the *packeth* screen and click the *send* button. At the bottom of the *packeth* screen, there should be a time stamp and the number of bytes sent to *eth0* shown. User can verify the packet content on *wireshark* on both the sending and receiving PCs. This verifies the flow entry entered via the *add-flow* command works as expected. To test this flow further, follow the next *packeth* screen to create another packet with different information and send it through the *eth0*.

Figure 3 – Packeth

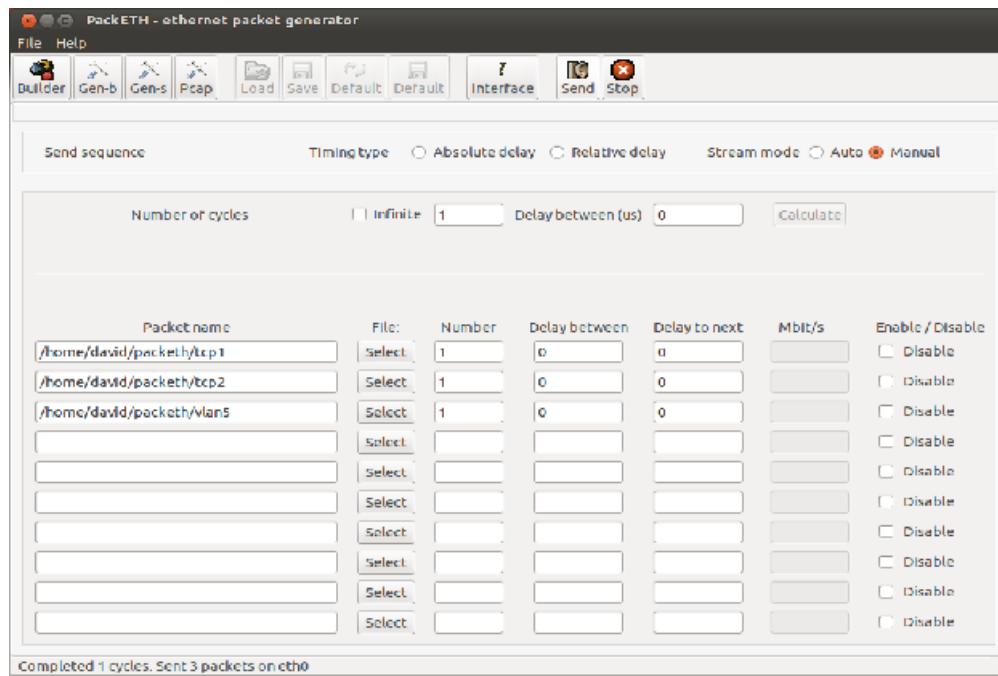


Next, use `ovs-ofctl del-flows br0` command to delete the flow. Then, use `ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_src=100.10.0.2,actions=output:2` to add a new flow that filters all packets received from port 1 and only forwards the packets with the matching IP address to port 2.

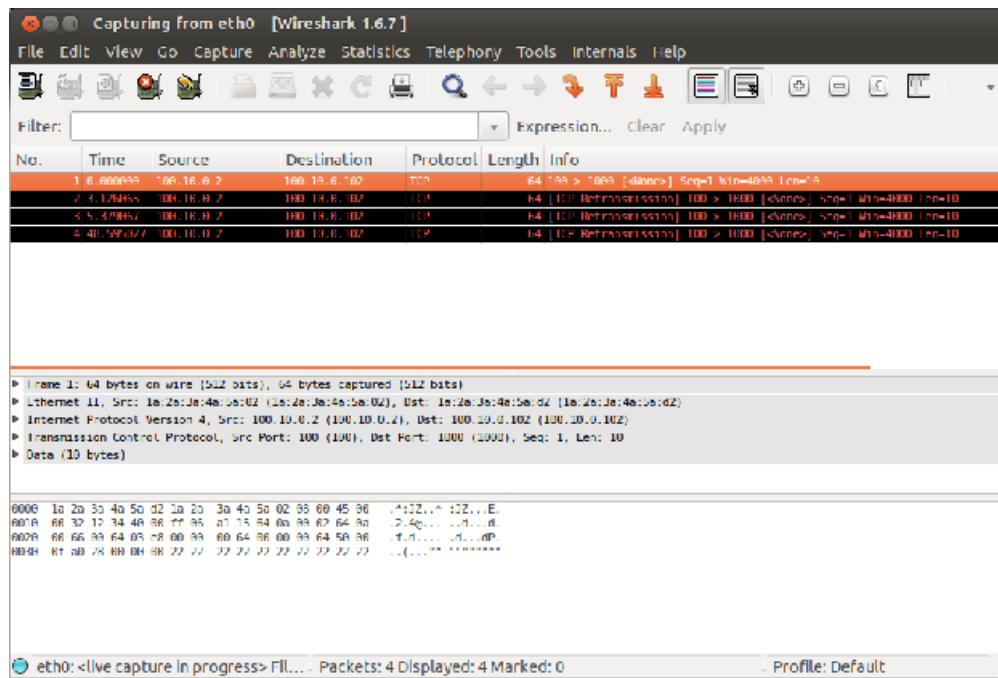
Figure 4 – Add flow with source IP matching field

```
david@ubuntu: ~
root@XenPlus4:~# ovs-ofctl del-flows br0
root@XenPlus4:~# ovs-ofctl mod-flows br0 in_port=1,dl_type=0x0800,nw_src=100.10.0.2,
actions=output:2
root@XenPlus4:~# ovs-ofctl dump-flows br0
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=4.616s, table=0, n_packets=0, n_bytes=0, ip,in port=1,nw src=100.10.0.2 actions2
root@XenPlus4:~# ovs-ofctl dump-flows br0
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=22.72s, table=0, n_packets=1, n_bytes=68, ip,in port=1,nw src=100.10.0.2 action2
root@XenPlus4:~# ovs-ofctl dump-flows br0
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=51.24s, table=0, n_packets=2, n_bytes=136, ip,in port=1,nw src=100.10.0.2 action2
root@XenPlus4:~#
```

On the `packet` menu, click the `gen-s` button to mix packets into one output stream. Select different packets built with different IP addresses to form one packet stream. Specify the delay and number of iteration. Then, select the manual operation to send the stream. Use `wireshark` to examine the result.

Figure 5 – Use packeth to generate mixed packet stream

As shown in the screen shot, the packet stream sent using *packeth*, with 3 different types of the packet and 3 different source IP addresses, is filtered by the flow and only the packet with source IP address 100.10.0.2 is forwarded to output port 2. With the *packeth* and *wireshark*, many of the fields can be tested in the uni-directional flow configuration.

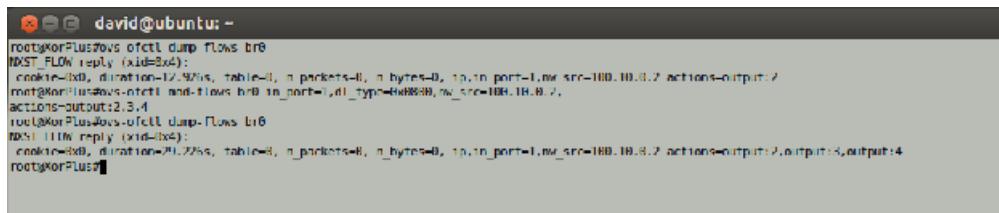
Figure 6 – Packet filtering for uni-directional traffic

1-to-Many Multicasting

After the unidirectional flow from one port to another, modify the flow entry to configure a 1-to-3 ports multicasting scenario where packets matching the flow entry are duplicated and forwarded to 3 output ports. This time, use the *mod-flows* command ***ovs-ofctl mod-flows br0 in_port=1,dl_type=0x0800,nw_src=100.10.0.2,actions=output:2,3,4***

in_port=1,dl_type=0x0800,nw_src=100.10.0.2,actions=output:2,3,4. Then, use the dump flow command to verify the flow is set up correctly.

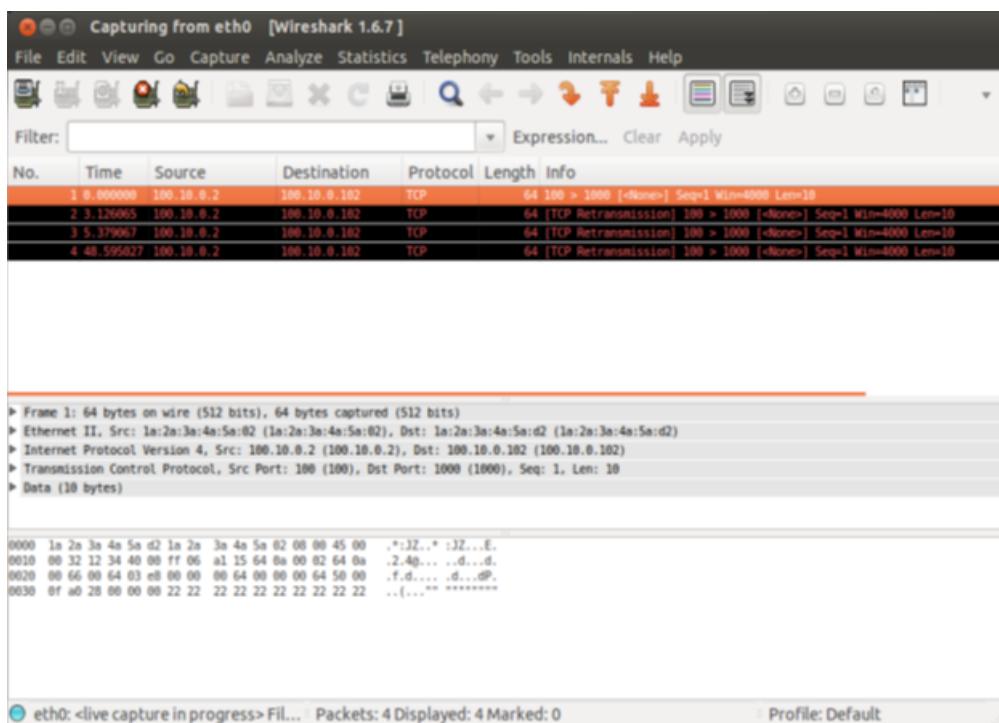
Figure 7 – 1 to 3 port packet duplication and multicasting



```
david@ubuntu: ~
root@korPlus:~# ovs-ofctl dump flows br0
ovs流表回复 (id=0x4):
ovs流表，duration=12.275s, table=0, n_packets=0, n_bytes=0, in_port=1,out_src=100.10.0.2 actions=output:2
root@korPlus:~# ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_src=100.10.0.2,
actions=output:2,3,4
root@korPlus:~# ovs-ofctl dump flows br0
ovs流表回复 (id=0x5):
ovs流表，duration=21.225s, table=0, n_packets=0, n_bytes=0, in_port=1,out_src=100.10.0.2 actions=output:2,output:3,output:4
root@korPlus:~#
```

Configure the *packet* and *wireshark* on all PCs, send the packets into port 1, and examine the packets received on port 2, 3, and 4 to see if the action matches the flow specification.

Figure 8 – 1 to 3 port packet filtering, duplication, and multicasting



Next, use the *packet* to build packet with VLAN, priority, ARP, TCP, UDP, and ICMP packets to exercise various flow packet matching fields and use the *wireshark* to verify the output.

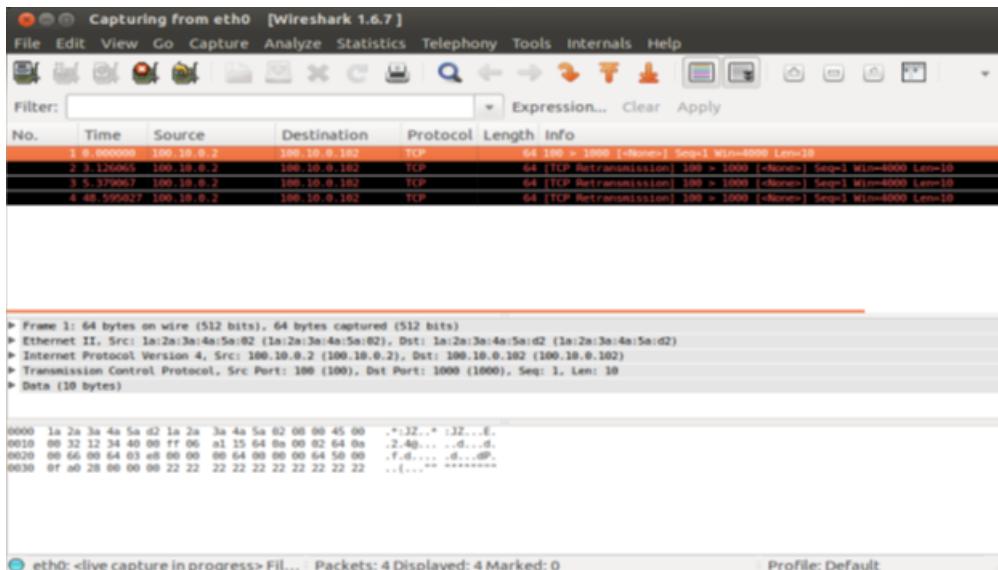
In addition to using filters in multicasting, port level duplication and multicasting is also supported. To configure this scenario, first clean up the flows in *br0* using *ovs-ofctl del-flows br0*. Then, use *ovs-ofctl add-flow br0 in_port=1,actions=output:2,3,4* to add a new multicasting flow.

Figure 9 – 1 to 3 port level multicasting

```
david@ubuntu: ~
root@korPlus:~# ovs-ofctl dump flows br0
NXT_FLOW reply (xid=0x4):
cookie=0xd0, duration=19.718s, table=0, n_packets=0, n_bytes=0, ip,in_port=1,src=100.10.0.2 actions=output:2,output:1,output:4
root@korPlus:~# ovs-ofctl dump flows br0
NXT_FLOW reply (xid=0x4):
cookie=0xd0, duration=19.718s, table=0, n_packets=0, n_bytes=0, ip,in_port=1,src=100.10.0.2 actions=output:2,output:1,output:4
root@korPlus:~# ovs-ofctl add-flow br0 in_port=1,actions=output:2,1,4
root@korPlus:~# ovs-ofctl dump-flows br0
NXT_FLOW reply (xid=0x4):
cookie=0xd0, duration=3.353s, table=0, n_packets=0, n_bytes=0, in_port=1 actions=output:2,output:3,output:4
root@korPlus:~#
```

The same traffic with source IP 100.10.0.2 is sent to port 1. The received traffic on port 4 is captured using *wireshark*. With the tools described in this document, various traffic patterns combined with different filters can be configured to test application scenarios.

Figure 10 – 1 to 3 port level multicasting



Many-to-One Aggregation

In this section, flow aggregation from multiple ports is examined. Two scenarios will be configured. The first scenario is to aggregate traffic from port 1, 2, and 3 without any filtering to port 4. The second scenario is to apply packet matching filter on each port to select specific traffic based on source IP address from each port for aggregation. For the first scenario, delete the existing flows using the *ovs-ofctl del-flows br0* command. Then, use the following commands to add 3 flows to the flow table:

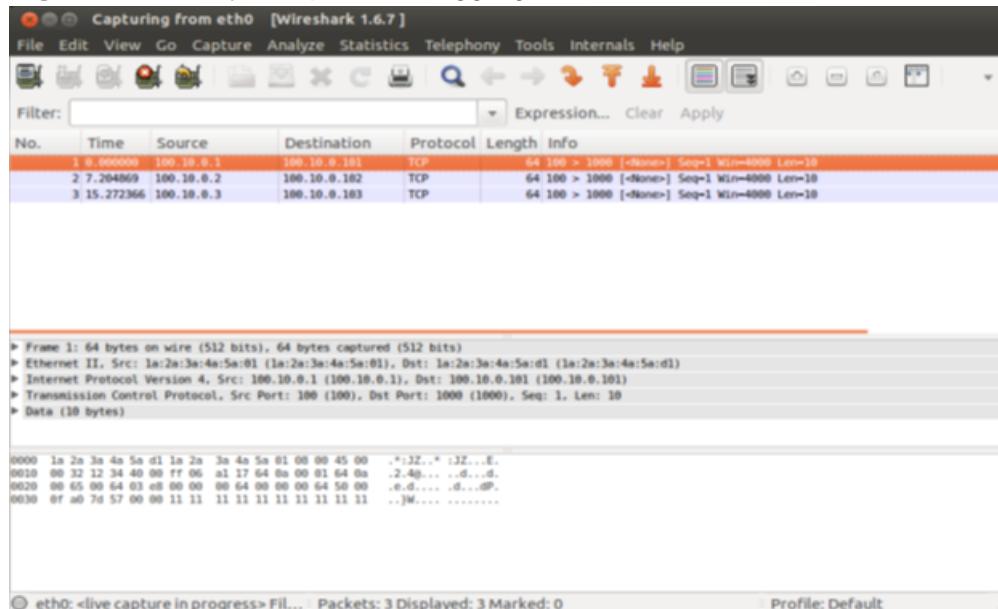
- *ovs-ofctl add-flow br0 in_port=1,actions=output:4*
- *ovs-ofctl add-flow br0 in_port=2,actions=output:4*
- *ovs-ofctl add-flow br0 in_port=3,actions=output:4*

Figure 11 – Many to 1 port level aggregation

```
david@ubuntu: ~
root@korPlus:~# ovs-ofctl del flows br0
root@korPlus:~# ovs-ofctl dump-flows br0
NXT_HL reply (xid=0x4):
root@korPlus:~# ovs-ofctl add-flow br0 in_port=1,actions=output:4
root@korPlus:~# ovs-ofctl add-flow br0 in_port=2,actions=output:4
root@korPlus:~# ovs-ofctl add-flow br0 in_port=3,actions=output:4
root@korPlus:~# ovs-ofctl dump-flows br0
NXT_HL reply (xid=0x4):
cookie=0xd0, duration=19.317s, table=0, n_packets=0, n_bytes=0, in_port=1 actions=output:4
cookie=0xd0, duration=12.645s, table=0, n_packets=0, n_bytes=0, in_port=2 actions=output:4
cookie=0xd0, duration=4.477s, table=0, n_packets=0, n_bytes=0, in_port=3 actions=output:4
root@korPlus:~#
```

Configure *packeth* on each PC to send packets from port 1 to 3, with source IP 100.10.0.1 from port1, source IP 100.10.0.2 from port 2, and 100.10.0.3 from port 3. All the packets should be forwarded to port 4.

Figure 12 – Many to 1 port level aggregation



In this many-to-one aggregation configuration, if the aggregated traffic is more than 1 Gbps, the over-subscribed packets will be dropped. The over-subscription scenario could not be demonstrated in this set up because the PC only has a 100 Mbps Ethernet port. But if user can create the scenario, the dropped packets can be shown via the **ovs-ofctl dump-ports br0** command as part of the port counters. To engineer the traffic aggregation, the filters described in *Open SDN: Starter kit – Power on and configure switch* can be applied to identify and select traffic for aggregation. In the scenario below, three flows are created with a filter on separate IP address on each port. The previous flows need to be deleted first, then use the following commands to set up the new flows:

- **ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_src=100.10.0.1,actions=output:4**
- **ovs-ofctl add-flow br0 in_port=2,dl_type=0x0800,nw_src=100.10.0.2,actions=output:4**
- **ovs-ofctl add-flow br0 in_port=3,dl_type=0x0800,nw_src=100.10.0.3,actions=output:4**

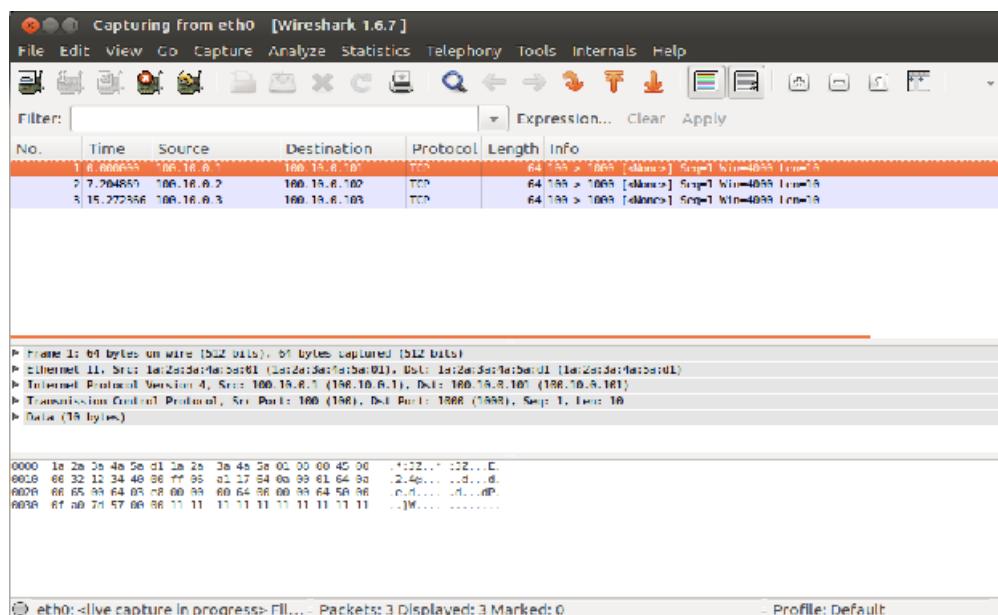
Figure 13 – Many to 1 port level aggregation

```
david@ubuntu:~$ ovs-ofctl del-flows br0
root@Kor1us:~#ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_src=100.10.0.1,actions=output:4
root@Kor1us:~#ovs-ofctl add-flow br0 in_port=2,dl_type=0x0800,nw_src=100.10.0.2,actions=output:4
root@Kor1us:~#ovs-ofctl add-flow br0 in_port=3,dl_type=0x0800,nw_src=100.10.0.3,actions=output:4
root@Kor1us:~#ovs-ofctl dump-flows br0
ovsdump: br0, duration=5.57s, table=0, n_packets=9, n_bytes=8, ip,in_port=3,nw_src=100.10.0.3 actions=output:1
ovsdump: br0, duration=5.57s, table=0, n_packets=0, n_bytes=0, ip,in_port=1,nw_src=100.10.0.1 actions=output:4
ovsdump: br0, duration=5.57s, table=0, n_packets=0, n_bytes=0, ip,in_port=2,nw_src=100.10.0.2 actions=output:4
root@Kor1us:~#
```

The *packeth* is configured to generate traffic with mixed source IP addresses. Try this with a manual option first to send a small amount of traffic through each port. Then, monitor the traffic on *wireshark* to see if the packets are filtered and forwarded correctly.

The *dump-flows* command is handy to verify the number of packets matched by the filtering rule. The *ovs-ofctl dump-ports br0* command is also very useful to show all the port statistics. Flows can be modified dynamically based on traffic conditions to throttle traffic during over-subscription, provide load balance, and re-distribute traffic. In deployment scenario, flows are managed via Open Flow controller with Open Flow protocols. In the next Starter kit, the RYU Open Flow controller will be discussed to show the controller-switch interaction.

Figure 14 – Many to 1 port level aggregation with filter



OVS Commands Used in this Tutorial

```

ovs-vsctl show
ovs-ofctl show br0
ovs-ofctl dump-ports br0
ovs-vsctl list-br
ovs-vsctl list-ports br0
ovs-vsctl list-ifaces br0
ovs-ofctl dump-flows br0
ovs-ofctl snoop br0
ovs-vsctl add-br br0 - set bridge br0 datapath_type=pica8
ovs-vsctl del-br br0
ovs-vsctl set-controller br0 tcp:172.16.1.240:6633
ovs-vsctl del-controller br0
ovs-vsctl set Bridge br0 stp_enable=true
ovs-vsctl add-port br0 ge-1/1/1 - set interface ge-1/1/1 type=pica8
ovs-vsctl add-port br0 ge-1/1/2 - set interface ge-1/1/2 type=pica8
ovs-vsctl add-port br0 ge-1/1/3 - set interface ge-1/1/3 type=pica8
ovs-vsctl add-port br0 ge-1/1/4 - set interface ge-1/1/4 type=pica8
ovs-vsctl add-port br0 ge-1/1/1 type=pronto options:link_speed=1G
ovs-vsctl del-port br0 ge-1/1/1
ovs-ofctl add-flow br0 in_port=1,actions=output:2
ovs-ofctl mod-flows br0 in_port=1,dl_type=0x0800,nw_src=100.10.0.1,actions=output:2
ovs-ofctl add-flow br0 in_port=1,actions=output:2,3,4
ovs-ofctl add-flow br0 in_port=1,actions=output:4
ovs-ofctl del-flows br0
ovs-ofctl mod-port br0 1 no-flood
ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_src=192.168.1.241,actions=output:3
ovs-ofctl add-flow br0
in_port=4,dl_type=0x0800,dl_src=60:eb:69:d2:9c:dd,nw_src=198.168.1.2,nw_dst=124.12.123.55,a

```

```
actions=output:1
ovs-ofctl mod-flows br0 in_port=4,dl_type=0x0800,nw_src=192.210.23.45,actions=output:3
ovs-ofctl del-flows br0 in_port=1
```

Packet Address File

```
100.10.0.1,1a:2a:3a:4a:5a:01,stream1
100.10.0.2,1a:2a:3a:4a:5a:02,stream2
100.10.0.3,1a:2a:3a:4a:5a:03,stream3
100.10.0.4,1a:2a:3a:4a:5a:04,stream4
100.10.0.5,1a:2a:3a:4a:5a:05,stream5
100.10.0.6,1a:2a:3a:4a:5a:06,stream6
100.10.0.7,1a:2a:3a:4a:5a:07,stream7
100.10.0.8,1a:2a:3a:4a:5a:08,stream8
100.10.0.9,1a:2a:3a:4a:5a:09,stream9
100.10.0.10,1a:2a:3a:4a:5a:0a,stream10
100.10.0.101,1a:2a:3a:4a:5a:d1,stream101
100.10.0.102,1a:2a:3a:4a:5a:d2,stream102
100.10.0.103,1a:2a:3a:4a:5a:d3,stream103
100.10.0.104,1a:2a:3a:4a:5a:d4,stream104
100.10.0.105,1a:2a:3a:4a:5a:d5,stream105
100.10.0.106,1a:2a:3a:4a:5a:d6,stream106
100.10.0.107,1a:2a:3a:4a:5a:d7,stream107
100.10.0.108,1a:2a:3a:4a:5a:d8,stream108
100.10.0.109,1a:2a:3a:4a:5a:d9,stream109
100.10.0.110,1a:2a:3a:4a:5a:da,stream110
```

Connection to a RYU Controller

- RYU Introduction
- Introduce RYU Open Flow Controller
- Configure OVS for RYU OpenFlow Controller
- Controller-OVS Interaction
- RYU Simple Switch Application
- Open Flow Message Type
- RYU Guide OVS Commands Reference
- Using TTP (router profile) with RYU Controller

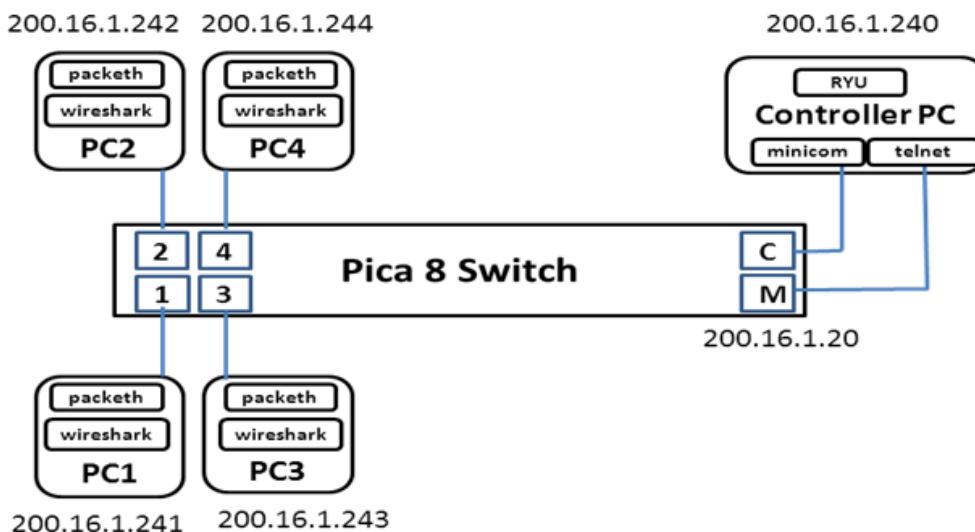
RYU Introduction

This document provides instructions on how to configure Pica8's open switches to work in various application scenarios. This document assumes the reader with minimal to no knowledge of the Open Virtual Switch (OVS) implementation defined by <http://openvswitch.org/> or the OpenFlow protocol defined by <https://www.opennetworking.org/>. After studying this guide, user will have the tools needed to configure Pica8's open switches as an OpenFlow switch. User will also gain insights on how to optimize the configuration to work in an application environment while also learning about OVS and the OpenFlow protocol.

This starter kit provides screen shots and a list of off-the-shelf applications needed to complete the configuration, as well as tips regarding problems user may encounter during the setup. More documents or cookbooks on other subjects will be published periodically. This document provides a tutorial on how to:

- Configure Pica8 as an OVS OpenFlow switch
- Create bridges, add ports, show bridge and port statistics, status, as well as the OVS database
- Configure flow tables for uni-directional, bi-directional, traffic switching, one-to-many multi-casting, mirroring, filtering, many-to-one aggregation, etc.,
- Configure Pica8 OVS OpenFlow switches to interface with the RYU OpenFlow Controller

Figure 1 – Test bed configuration



In this document, the system configuration depicted in Figure 1 includes:

- A Pica8 P-3295 open switch with 48 x 1 GbE and 4 x 10 GbE uplinks
- 5 Linux PCs running Ubuntu 12.4.1 LTS, one is connected to the management LAN port (RJ45) and console port (RJ45F); this PC is referred to the controller PC. The OpenFlow controller will be running on this PC. Four PCs are connected to 1GbE port 1 to 4 and serve as a data terminal for generating and monitoring traffic
- Tools installed on all the PCs are listed below. They can be installed through Linux installation utility *apt-get*
 - Terminal emulator minicom
 - Traffic monitoring tool Wireshark
 - Packet generator Packeth
 - ftp and ftppd
 - telnet and telnetd

Introduce RYU Open Flow Controller

RYU is an open flow controller that has been integrated with the Pica8 open switch with OVS 1.10 implementation that supports Open Flow v1.3. Additional RYU information can be found at the RYU website <http://osrg.github.com/ryu/>. The purpose of Pica8 RYU integration is to provide an open source SDN platform so that the SDN community can prototype, test, and develop applications in an open source environment with an open flow switching platform for real traffic testing. With the configuration provided in this starter kit, user should be able to have real traffic running in a week to test out the application scenarios using OVS commands. Both OVS and RYU are open source with Apache licenses that developers can access easily.

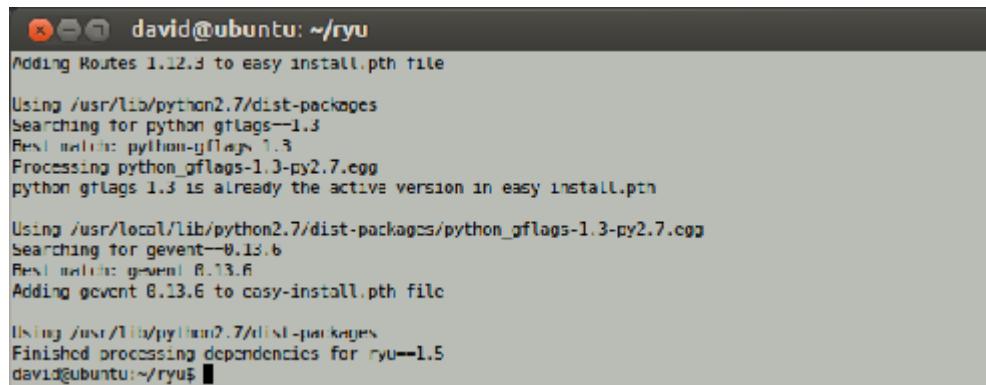
To clone the RYU directory, open a shell window from \$home directory. Then, use `git clone git://github.com/osrg/ryu.git` to copy the RYU code base. It will create an `ryu` directory in \$home.

Figure 2 – Clone RYU. Then `cd ryu` and `sudo python ./setup.py install` to complete the installation.



```
david@ubuntu:~/ryu
david@ubuntu:~$ git clone git://github.com/osrg/ryu.git
Cloning into 'ryu'...
remote: Counting objects: 2070, done.
remote: Compressing objects: 100% (901/901), done.
remote: Total 2878 (deletions 2118), reused 2689 (delta 1959)
Receiving objects: 100% (2070/2070), 9.00 MiB | 462 KiB/s, done.
Resolving deltas: 100% (2118/2118), done.
david@ubuntu:~$ cd ryu
david@ubuntu:~/ryu$ sudo python ./setup.py install
```

Figure 3 – Install RYU. The installation installs the `ryu-manager` and `ryu-client` to the \$home/`ryu/bin` and `/usr/local/bin` directories. Now we are ready to run the test applications.



```
david@ubuntu:~/ryu
Adding Routes 1.12.3 to easy_install.pth file
Using /usr/lib/python2.7/dist-packages
Searching for python_gflags==1.3
Best match: python_gflags 1.3
Processing python_gflags-1.3-py2.7.egg
python_gflags 1.3 is already the active version in easy_install.pth

Using /usr/local/lib/python2.7/dist-packages/python_gflags-1.3-py2.7.egg
Searching for gevent==0.13.6
Best match: gevent 0.13.6
Adding gevent 0.13.6 to easy-install.pth file

Using /usr/lib/python2.7/dist-packages
Finished processing dependencies for ryu==1.5
david@ubuntu:~/ryu$
```

Figure 4 – RYU-manager and RYU-client

```
david@ubuntu:~/ryu/bin
david@ubuntu:~/ryu$ ls
bin  dist  etc  MANIFEST.in  rmu_deserialize.py  ryu.egg-info  setup.py  tools
build  doc  LICENSE  README.rst  ryu      setup.cfg  SubmittingPatches.rst
david@ubuntu:~/ryu$ cd bin
david@ubuntu:~/ryu/bin$ ls
-rwxr-xr-x 1 root  root  163 Dec  4 11:58 /usr/local/bin/ryu-client
-rwxr-xr-x 1 root  root  165 Dec  4 11:58 /usr/local/bin/ryu-manager
david@ubuntu:~/ryu/bin$
```

Configure OVS for RYU OpenFlow Controller

In OVS, the controller property of the bridge created earlier needs to be added to include the controller IP address and port number. The command `ovs-vsctl set-controller br0 tcp:200.16.1.240:6633` sets the controller address for bridge `br0` on port 6633. The command `ovs-vsctl show` would then show the bridge information. The connection status is not shown because the controller has not been started yet.

Figure 5 – Set RYU controller IP address

```
david@ubuntu: ~
root@KaliPlus:~# ovs-vsctl db=lp:200.16.1.20:6633 set controller br0 tcp:200.16.1.240:6633
root@KaliPlus:~# ovs-vsctl db=lp:200.16.1.20:6633 show
353bea35 081e 49ae 8c61 5f45cbf65bef
  Bridge "br0"
    Controller "tcp:200.15.1.240:6633"
    Port "pc-1/1/2"
      Interface "pc-1/1/2"
        type: pionta
    Port "g0-1/1/1"
      Interface "g0-1/1/1"
        type: pionta
    Port "br0"
      Interface "br0"
        type: internal
    Port "qe-1/1/0"
      Interface "qe-1/1/0"
        type: pionta
    Port "qe-1/1/1"
      Interface "qe-1/1/1"
        type: pionta
root@KaliPlus:~#
```

The RYU controller will be running on the controller PC with IP address 200.16.1.240 using the default port 6633. The port number can be changed. For this exercise, the controller will be started with the `-verbose` mode.

Figure 6 – Start RYU-manager with verbose option

The `--verbose` mode helps us understand the *RYU controller-OVS* interaction. Use the command `ryu-manager -verbose` to start the controller. The TCP connection is established first and the connection information is printed with peer (OVS) IP address 10.10.50.20. Once the controller is started, the connection status will change to `is_connected: true`. The controller port information can also be shown using the command `ovs-ofctl show br0`.

Figure 7 – Show controller connection status

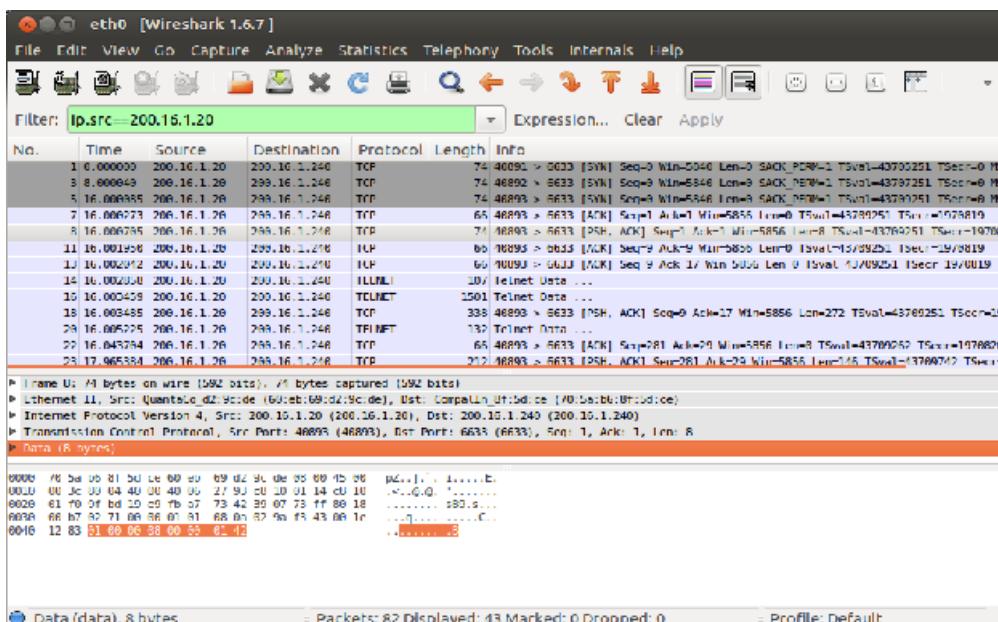
```
david@ubuntu: ~
root@XorPlus7ovs:~# vscctl db-tcp:200.16.1.20:6633 show
3531a-4135-881a-494e-8c61-5f45bf651bf
  Bridge "br0"
    Controller "tcp:200.16.1.210:6633"
      Is_connected: true
    Port "qc-1/1/2"
      Interface "ge-1/1/2"
        type: pronto
    Port "qc-1/1/1"
      Interface "ge-1/1/1"
        type: pronto
    Port "br0"
      Interface "br0"
        type: internal
    Port "ge 1/1/3"
      Interface "ge-1/1/3"
        type: pronto
    Port "ge 1/1/4"
      Interface "ge-1/1/4"
        type: pronto
root@XorPlus7ovs:~#
```

Controller-OVS Interaction

Once the controller and OVS are connected, a set of messages will be exchanged. For example, the OVS sends an OFPT_HELLO message to the controller. The hello message is captured on the *wireshark* screen. The first byte of the message is the version number and the second byte is the OFPT_TYPE. OFTP_HELLO message is type 0.

After the hello message from the switch, the controller sends OFPT_FEATURES_REQUEST (type=5) to retrieve the switch capabilities, including the supported open flow version, switch configuration, port hardware address, etc. The switch sends OFPT_FEATURES_REPLY (type=6) to provide the feature information. The message is shown on both the controller console and switch console.

Figure 8 – OFPT_HELLO message



The switch console information is provided by the snoop option of the *ovs-ofctl*/command. The command is ***ovs-ofctl snoop br0***. It shows the feature request from the controller and the feature reply with the bridge information. User can compare the switch console information with the controller console information to get a better understanding of the message exchange.

Figure 9 – ovs-ofctl snoop br0

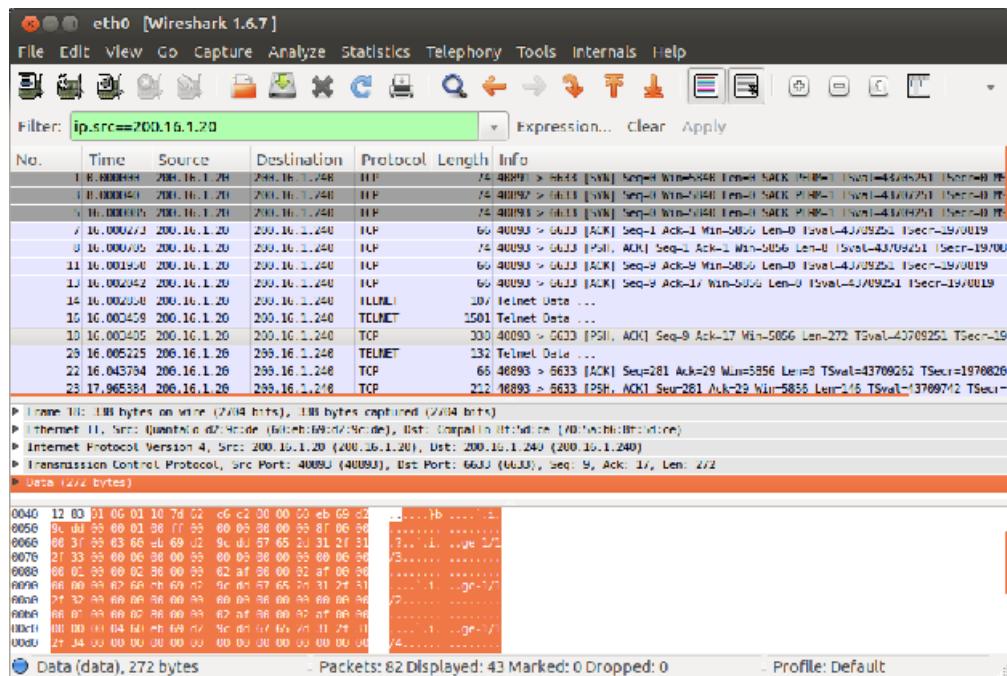
```
david@ubuntu: ~
root@XorPlus# ovs-ofctl snoop br0
OFPT_FEATURES_REQUEST (xid=0x272de88e):
OFPT_FEATURES_REPLY (xid=0x272de00e): ver:0x1, dpid:000000eb69d29cdd
n_tables:255, n_buffers:256
features: capabilitiess:0x1f, actions:0x3f
1:qg-1/1/1: addr:69:cb:50:d2:9c:dd
    config: 0
    state: LINK_DOWN
    current: COPPER AUTO NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO NEG
    supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
2:qg-1/1/2: addr:69:eb:89:d2:9c:dd
    config: 0
    state: LINK_DOWN
    current: COPPER AUTO NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO NEG
    supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO NEG
3:qg-1/1/3: addr:69:eb:89:d2:9c:dd
    config: 0
    state: LINK_DOWN
    current: COPPER AUTO NEG
    advertised: 10MB HD 10MB HD 100MB HD 100MB FD 1GB FD COPPER AUTO NEG
    supported: 10MB HD 10MB FD 100MB HD 100MB FD 1GB FD COPPER AUTO NEG
4:qg-1/1/4: addr:60:cb:08:d2:9c:dd
    config: 0
    state: LINK_DOWN
    current: COPPER AUTO NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO NEG
    supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO NEG
L2ONL(br0): addr:60:eb:69:d2:9c:dd
    config: PORT_DOWN
    state: LINK_DOWN
    current: 10MB-FD COPPER
OFPT_SET_CONFIG (xid=0x272de88f): flags=normal miss_send_len=128
ovs-ofctl: tcptdump exited with status 1
OFPT_PACKET_IN (xid=0x0): total_len=58 in_port=LOCAL data_len=58 buffer=0xffffffff
priority=0; tunnel0;in_portff:tc1(0) mac0:23:20:cf:91:63->ff:ff:ff:ff type05ff proto0 tos0 ttl0 ip0.0.0.0->0.0.0.0
ovs ofctl: tcptdump exited with status 1
```

The *wireshark* also captured the information. Notice the message type in the second byte is 6 representing the OFPT_FEATURES_REPLY. After the feature reply, the controller sends an OFPT_SET_CONFIG message to set the message parameters such as max length, etc. Once the controller is connected, the OVS changes its default behavior from a layer 2 switch to an OVS switch. It means the flooding is disabled and open flow packet processing starts. Each packet is processed based on the flow table entry. Unmatched packets are forwarded to the controller for analysis unless a rule is

defined to drop the packet.

During initial start up with the controller, the flow table is empty. Therefore, packets received from any port are forwarded to the controller. The next message from the switch is type OFPT_FEATURE_REPLY message (type=10/0x0a).

Figure 10 – OFPT_FEATURE_REPLY message



In this exercise, the *RYU-manager* does not have any application to receive and process the OFPT_PACKET_IN message. Therefore, on the controller screen, a bunch of *unhandled_events* are printed on the console. At this point, the RYU-OVS open flow session is established and ready for Open Flow application to take over the event handling and flow configuration.

RYU Simple Switch Application

With just the controller connected without any application, the ping between the PCs cannot work because the ARP requests are forwarded to the controller without any packet processing instructions in the flow tables. RYU code distribution comes with a set of applications to show how applications can be integrated. The following instructions will show how to run the simple switch application. The application processes the *packet_in* messages (e.g., ICMP_REQUEST) and instructs the bridge to flood all other ports with the packets once the destination host has received the request and has replied with its MAC address. This simple switch application sets up the flow table to forward traffic from source port to the correct destination port. This is the default switch behavior that we have tested before. The application is in the \$home/ryu/ryu/application directory. Run the *ryu-manager -verbose simple_switch.py* command to start the application.

Figure 11 – RYU-manager with simple switch application

```
david@ubuntu:~/ryu/ryu/app$ ryu-manager --verbose simple_switch.py
loading app ryu.controller.ofp_handler
loading app simple_switch.py
instantiating app simple_switch.py
instantiating app ryu.controller.ofp_handler
connected socket:<socket fileno=4 sock=200.16.1.240:6003 peer=200.16.1.20:56705> address:(('200.16.1.20', 56705)
unhandled event <ryu.controller.dispatcher.EventueueCreate object at 0x1f6e5b8>
Hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x1f6ff5b>
move onto config mode
unhandled event <ryu.controller.dispatcher.EventDispatcherChange object at 0x1f720d8>
switch features Ev-v1.0(xid: 0x1 msg_type:0x6 xid: 0x60993985 port: OFPPhyPort(port_no=1, hw_addr='\'\x00\x00\x00\x00\x00\x00', name='ge-1/1/1\x00\x00\x00\x00\x00\x00', config=0, state=0, curr=648, advertised=687, supported=687, pccr=655) OFPPhyPort(port_no=2, hw_addr='\'\x00\x00\x00\x00\x00\x00', name='ge-1/1/2\x00\x00\x00\x00\x00\x00', config=0, state=1, curr=649, advertised=687, supported=687, pccr=655) OFPPhyPort(port_no=3, hw_addr='\'\x00\x00\x00\x00\x00\x00', name='ge-1/1/3\x00\x00\x00\x00\x00\x00', config=0, state=0, curr=572, advertised=587, supported=587, pccr=637) OFPPhyPort(port_no=4, hw_addr='\'\x00\x00\x00\x00\x00\x00', name='ge-1/1/4\x00\x00\x00\x00\x00\x00', config=0, state=0, curr=572, advertised=587, supported=587, pccr=637) OFPPhyPort(port_no=5, hw_addr='\'\x00\x00\x00\x00\x00\x00', name='ge-1/1/5\x00\x00\x00\x00\x00\x00', config=0, state=0, curr=130, advertised=0, supported=0), config=1
move onto main mode
unhandled event <ryu.controller.dispatcher.EventDispatcherChange object at 0x1f720d8>
packet in 106554200991453 00:12:3f:20:a6:72 00:12:3f:20:7c:4b 4
packet in 106554200991453 00:12:3f:20:a6:72 00:12:3f:20:7c:4b 4
```

Once the simple switch application starts, the first part of the message output is the same as before, but instead of receiving unhandled events only as the previous RYU-manager only run, it sends an OFPT_PACKET_OUT message to the switch with FLOOD actions on first two packet_in messages. The first one is a probe message sent by the controller on local port.

Figure 12 – Switch responses with simple switch application

```
david@ubuntu: ~
supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GE-FD COPPER AUTO_NEG
peer: 10MD-ID 10MD-FD 100MD-ID 100MD-FD 1GD-FD COPPER AUTO_NEG
LOCAL(b=0): addr:50:cb:69:d2:9c:dd
config: PORT_DOWN
state: LINK_DOWN
current: 10MB HD COPPER
OFPT_SFT_CONFIG (xid=0x60993986): frags:normal miss_send_len=128
ovs-ofctl: tcptdump cxtcid with status 1
OFPT_PACKET_IN (xid=0x0): total_len=58 in_port=LOCAL data_len=58 buffer=0xffffffff80 priority=0:tunnel0:in_portffff:tc1(0) mac00:23:20:ad:59:76->ff:ff:ff:ff:ff type0x0ff proto0 tos0 ttl0 ip0.0.0.0->0.0.0.0
OFPT_PACKET_OUT (xid=0x60993987): in_port=LOCAL actions_len=8 actions=FLOOD buffer=0xffffffff80
ovs-ofctl: tcptdump exited with status 1
OFPT_PACKET_IN (xid=0x0): total_len=98 in_port=3 data_len=98 buffer=0x00000119 priority=0:tunnel10:in_port6003:tc1(0) mac00:12:3f:20:7c:4b->00:12:3f:20:7c:4b type0x0ff proto0 tos0 ttl164 ip10.1.0.43->10.1.0.44 port8->0
OFPT_PACKET_OUT (xid=0x60993988): in_port=3 actions_len=8 actions=FLOOD buffer=0x00000119
ovs-ofctl: tcptdump exited with status 1
OFPT_PACKET_IN (xid=0x0): total_len=98 in_port=4 data_len=98 buffer=0x0000011a priority=0:tunnel0:in_port6004:tc1(0) mac00:12:3f:20:a6:72->00:12:3f:20:7c:4b type0x0ff proto0 tos0 ttl64 ip10.1.0.44->10.1.0.43
OFPT_FLOW_MOD (xid=0x60993989): ADD in_port=4,dl_dst=00:12:3f:20:7c:4b->00:12:3f:20:a6:72 flags=0x1 actions=output:3
OFPT_PACKET_OUT (xid=0x6099398a): in_port=4 actions_len=8 actions=output:3 buffer=0x0000011a
ovs-ofctl: tcptdump cxtcid with status 1
OFPT_PACKET_IN (xid=0x0): total_len=98 in_port=3 data_len=98 buffer=0x0000011b priority=0:tunnel0:in_port0003:tc1(0) mac00:12:3f:20:7c:4b->00:12:3f:20:7c:4b type0x0ff proto0 tos0 ttl64 ip10.1.0.43->10.1.0.44 port8->0
OFPT_FLOW_MOD (xid=0x6099398b): ADD in_port=3,dl_dst=00:12:3f:20:7c:4b->00:12:3f:20:a6:72 flags=0x1 actions=output:4
OFPT_PACKET_OUT (xid=0x6099398c): in_port=3 actions_len=0 actions=output:4 buffer=0x0000011b
OFPT_ECHO_REQUEST (xid=0x0): 8 bytes of payload
```

The second message is an ICMP request that comes in from port 3 and has a target destination of port 4. This message is the result of a ping test running on the PC connected to port 3 to PC on port 4. Since the controller does not know which port has the PC 4 MAC address. It sends the OFPT_PACKET_OUT instruction to the switch to flood the message received on port 3. Once the PC on port 4 receives the ICMP request, it responds with its reply. The controller matches the reply destination MAC address with the PC3 on port3 and sends an OFPT_FLOW_MOD action to create a flow from port 4 to port 3 to forward the packets. The same process repeats again for setting up flows from port 3 and port 4.

Figure 13 – Flow tables set up by simple switch application

```
root@XenPlus:~# ovs-dump-flows br0
NXST_FLOW reply (xid=0x4):
root@XenPlus:~# ovs-dump-flows br0
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=19.989s, table=0, n_packets=0, n_bytes=318, in_port=3,dl_dst=00:12:3f:20:a6:72 actions=output:4
cookie=0x0, duration=11.999s, table=0, n_packets=10, n_bytes=1020, in_port=4,dl_dst=00:12:3f:20:a6:72 actions=output:3
root@XenPlus:~# ovs-del-flows br0
root@XenPlus:~# ovs-dump-flows br0
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=6.579s, table=0, n_packets=0, n_bytes=0, in_port=3,dl_dst=00:12:3f:20:a6:72 actions=output:4
cookie=0x0, duration=6.579s, table=0, n_packets=0, n_bytes=0, in_port=4,dl_dst=00:12:3f:20:a6:72 actions=output:3
root@XenPlus:~#
```

As a result, the *dump-flows* show 2 flows created by the simple switch application. To test it again, simply use *del-flows* command to delete the flow.

Figure 14 – Delete flow event from switch to controller

```
david@ubuntu:~/ryu/ryu/app$ ryu-manager --verbose simple_switch.py
loading app ryu.controller.ofp handler
loading app simple_switch.py
instantiating app simple_switch.py
instantiating app ryu.controller.ofp handler
connected socket <socket fd=4 sock=208.16.1.240:6633 peer=200.16.1.26:56785> address:(('200.16.1.26', 56785)
unhandled event <ryu.controller.dispatcher.EventOFPSwitchCreate object at 0x1f6fc5b>
hello ev <ryu.controller.ofp.event.EventOFPHello object at 0x1f6ff5e>
move onto main mode
unhandled event <ryu.controller.dispatcher.EventOFPSwitchChange object at 0x1f720d8>
switch features ev version: 0x1 msg type 0x1 vid 0x10991905 port 0x1 OFPPhyPort(port nn=1, hw_addr='>xehb1<x07<x9c<xd1', name='ge-1/1/1<x00>x00<x00>x00', config=0, state=618, curr=618, advertised=637, supported=637, peer=655) OFPPhyPort(part_no=2, hw_addr='>xccb1<x22<x9c<xd1', name='ge-1/1/2<x00>x00<x00>x00', config=0, state=1, curr=649, advrtiscd=607, supported=607, peer=0) OFPPhyPort(part_no=3, hw_addr='>xehb1<x07<x9c<xd1', name='ge-1/1/3<x00>x00<x00>x00', config=0, state=0, curr=548, adverLised=687, supported=687, peer=655) OFPPhyPort(part_no=4, hw_addr='>xehb1<x02<x9c<xd1', name='b-0<x00>x00<x00>x00', config=0, state=0, curr=542, advrtiscd=587, supported=587, pccr=6d1) OFPPhyPort(part_no=5, hw_addr='>xehb1<x01<x9c<xd1', name='b-1<x00>x00<x00>x00', config=1, state=1, curr=130, adverLised=0, supported=0, peer=0)
move onto main mode
unhandled event <ryu.controller.dispatcher.EventOFPSwitchChange object at 0x1f720d8>
packet_in 166554268991453 00:12:3f:20:a6:72 00:12:3f:20:a6:72 3
packet_in 166554268991453 00:12:3f:20:a6:72 00:12:3f:20:a6:72 3
packet_in 166554268991453 00:12:3f:20:a6:72 00:12:3f:20:a6:72 3
unhandled event <ryu.controller.ofp.event.EventOFPPFlowRemoved object at 0x1f5ff50>
unhandled event <ryu.controller.ofp.event.EventOFPPFlowRemoved object at 0x1f5ff10>
packet_in 166554268991453 00:12:3f:20:a6:72 00:12:3f:20:a6:72 3
packet_in 166554268991453 00:12:3f:20:a6:72 00:12:3f:20:a6:72 3
```

Once the flows are deleted, two OPEN_FLOW_REMOVED events are generated by the switch to notify the controller. The MAC learning process repeats itself again. The two flows will be created when the next round of the ICMP requests come into the controller. User can dump the flow table to verify its behavior. At this point, the starter kit has demonstrated the basic RYU controller integration with OVS and a simple application built on top of the RYU controller. Reader should be able to start testing and writing test applications using the SDN platform presented in this document.

Open Flow Message Type

```
# enum ofp_type
OFPT_HELLO = 0 # Symmetric message
OFPT_ERROR = 1 # Symmetric message
OFPT_ECHO_REQUEST = 2 # Symmetric message
OFPT_ECHO_REPLY = 3 # Symmetric message
OFPT_VENDOR = 4 # Symmetric message
OFPT_FEATURES_REQUEST = 5 # Controller/switch message
OFPT_FEATURES_REPLY = 6 # Controller/switch message
OFPT_GET_CONFIG_REQUEST = 7 # Controller/switch message
OFPT_GET_CONFIG_REPLY = 8 # Controller/switch message
OFPT_SET_CONFIG = 9 # Controller/switch message
OFPT_PACKET_IN = 10 # Async message
OFPT_FLOW_Removed = 11 # Async message
```

```

OFPT_PORT_STATUS = 12 # Async message
OFPT_PACKET_OUT = 13 # Controller/switch message
OFPT_FLOW_MOD = 14 # Controller/switch message
OFPT_PORT_MOD = 15 # Controller/switch message
OFPT_STATS_REQUEST = 16 # Controller/switch message
OFPT_STATS_REPLY = 17 # Controller/switch message
OFPT_BARRIER_REQUEST = 18 # Controller/switch message
OFPT_BARRIER_REPLY = 19 # Controller/switch message
OFPT_QUEUE_GET_CONFIG_REQUEST = 20 # Controller/switch message
OFPT_QUEUE_GET_CONFIG_REPLY = 21 # Controller/switch message

```

RYU Guide OVS Commands Reference

```

ovs-vsctl show
ovs-ofctl show br0
ovs-ofctl dump-ports br0
ovs-vsctl list-br
ovs-vsctl list-ports br0
ovs-vsctl list-ifaces br0
ovs-ofctl dump-flows br0
ovs-ofctl snoop br0
ovs-vsctl add-br br0 - set bridge br0 datapath_type=pica8
ovs-vsctl del-br br0
ovs-vsctl set-controller br0 tcp:172.16.1.240:6633
ovs-vsctl del-controller br0
ovs-vsctl set Bridge br0 stp_enable=true
ovs-vsctl add-port br0 ge-1/1/1 - set interface ge-1/1/1 type=pica8
ovs-vsctl add-port br0 ge-1/1/2 - set interface ge-1/1/2 type=pica8
ovs-vsctl add-port br0 ge-1/1/3 - set interface ge-1/1/3 type=pica8
ovs-vsctl add-port br0 ge-1/1/4 - set interface ge-1/1/4 type=pica8
ovs-vsctl add-port br0 ge-1/1/1 type=pronto options:link_speed=1G
ovs-vsctl del-port br0 ge-1/1/1
ovs-ofctl add-flow br0 in_port=1,actions=output:2
ovs-ofctl mod-flows br0 in_port=1,dl_type=0x0800,nw_src=100.10.0.1,actions=output:2
ovs-ofctl add-flow br0 in_port=1,actions=output:2,3,4
ovs-ofctl add-flow br0 in_port=1,actions=output:4
ovs-ofctl del-flows br0
ovs-ofctl mod-port br0 1 no-flood
ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_src=192.168.1.241,actions=output:3
ovs-ofctl add-flow br0
in_port=4,dl_type=0x0800,dl_src=60:eb:69:d2:9c:dd,nw_src=198.168.1.2,nw_dst=124.12.123.55,actions=output:1
ovs-ofctl mod-flows br0 in_port=4,dl_type=0x0800,nw_src=192.210.23.45,actions=output:3
ovs-ofctl del-flows br0 in_port=1

```

RYU Controller Configuration

RYU controller comes with sample applications that enable user to jumpstart SDN deployment. These sample applications are stored in the following directory:

```
ls -l /home/sudhirmadali/ryu/ryu/app
total 292
-rw-rw-r-- 1 sudhirmadali sudhirmadali 3647 Jul  5 00:12 bmpstation.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 1787 Jul  5 00:12 cbench.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 817 Jul  5 00:12 conf_switch_key.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 3903 Jul  5 00:12 example_switch_13.py
drwxrwxr-x 3 sudhirmadali sudhirmadali 4096 Jul  5 00:12 gui_topology
-rw-rw-r-- 1 sudhirmadali sudhirmadali 0 Jul  5 00:12 __init__.py
drwxrwxr-x 2 sudhirmadali sudhirmadali 4096 Jul  5 00:12 ofctl
-rw-rw-r-- 1 sudhirmadali sudhirmadali 26560 Jul  5 00:12 ofctl_rest.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 5943 Jul  5 00:12 rest_conf_switch.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 38839 Jul  5 00:12 rest_firewall.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 39625 Jul  5 00:12 rest_qos.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 70869 Jul  5 00:12 rest_router.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 3912 Jul  5 00:12 rest_topology.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 3294 Jul  5 00:12 simple_switch_12.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 4742 Jul  5 00:12 simple_switch_13.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 3485 Jul  5 00:18 simple_switch_13.pyc
-rw-rw-r-- 1 sudhirmadali sudhirmadali 3916 Jul  5 00:12 simple_switch_14.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 3889 Jul  5 00:12 simple_switch_igmp.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 4328 Jul  5 00:12 simple_switch_lacp.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 3755 Jul  5 00:12 simple_switch.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 4954 Jul  5 00:12 simple_switch_snort.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 4979 Jul  5 00:12 simple_switch_stp.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 3792 Jul  5 00:12 simple_switch_websocket_13.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 9379 Jul  5 00:12 wsgi.py
-rw-rw-r-- 1 sudhirmadali sudhirmadali 4396 Jul  5 00:12 ws_topology.py
```

Among all the applications, "ofctl_rest.py" (highlighted above) enables openflow REST interface for an open flow capable switch. This program supports OpenFlow versions 1.0 through 1.5. This provides a good framework to develop the APIs for configuring open flow rules.

```
pica8@ubuntu:~$ pip install ryu
pica8@ubuntu:~$ cd ryu
pica8@ubuntu:~/ryu$ ryu-manager ryu.app.ofctl_rest --verbose

loading app ryu.app.ofctl_rest
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu.app.ofctl_rest of RestStatsApi
instantiating app ryu.controller.ofp_handler of OFPHandler
(24694) wsgi starting up on http://0.0.0.0:8080
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7fc8a6756950>
address:('172.16.0.112', 39589)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7fc8a6623350>
move onto config mode
EVENT ofp_event->dpset EventOFPswitchFeatures
switch features ev
version=0x5,msg_type=0x6,msg_len=0x20,xid=0xb0590b8d,OFPSwitchFeatures(auxiliary_id=0,capabilities=15,datapath_id=6790866252693771931,n_buffers=256,n_tables=254)
move onto main mode
EVENT ofp_event->dpset EventOFPStateChange
DPSET: register datapath <ryu.controller.controller.Datapath object at 0x7fc8a668ca10>
```

Another important tool required to work with APIs is one of the following:

1. POSTman : (<https://www.getpostman.com>)
2. Swagger : (<http://swagger.io/swagger-ui/>)

The following example will demonstrate the use of POSTman to configure the open flow rules. Typically, these tools are used with a single switch to determine the REST APIs required to configure the switch. POSTman has a facility to run the set of APIs in an iterative process to perform the task multiple times if required.

The following is the configuration on the switch required to configure TTP:

```
admin@PicOS-OVS$ovs-vsctl show-running-config

Open_vSwitch ab7af11d-fc7a-4e9c-8165-36e67f92d043
  Bridge "br0"
    Controller fb5f8fd6-acaa8-49f9-add0-800cd6c4d840
      is_connected: true
      role: other
      status: {current_version="OpenFlow14", sec_since_connect="691372", state=ACTIVE}
      target: "tcp:172.16.0.123:6633"
    datapath_id: "5e3e00e09500169a"
    datapath_type: "pica8"
    Port "qe-1/1/1"
      Interface "qe-1/1/1"
        type: "pica8"
        tag: 1
        vlan_mode: trunk
    Port "br0"
      Interface "br0"
        mtu: 1500
        type: internal
    Port "qe-1/1/2"
      Interface "qe-1/1/2"
        type: "pica8"
        tag: 1
        vlan_mode: trunk
    protocols: [ "OpenFlow14" ]

Pica8 6d736702-1adb-44f8-93e6-998166ca813f
  hardware_type: "5401"
  pica_ttp_enable: true
  pica_ttp_name: RouterTTP.json
```

Step 1: Setting the following environment variables in Postman helps in managing the variables

Variable	Value
host	172.16.0.123:8080
dpid	6790866252693771930
MacTable	20
RoutingTable	30
PolicyTable	60
VLANTable	10
key	value

Table Type Patterns (TTP) are described in detail in the following blog (<http://www.pica8.com/pica8-deep-dive/scaling-up-sdns-using-ttts-table-type-patterns/>). The following is based on steps mentioned in the blog.

Step 2 : Creating a mac entry in the L2 table

The screenshot shows the Postman application interface. The left sidebar lists various collections and requests, including 'L3 Flow' and 'MAC table'. The main area shows a 'Builder' tab with a 'POST' request to 'http://{{host}}/stats/flowentry/add'. The 'Body' tab is selected, showing JSON code for creating a flow entry in the Mac table. The code defines a flow with a priority of 32768, matching a MAC address (dl_dst: 00:11:11:11:11), and sending it to a table (RoutingTable). The 'Send' button is highlighted.

```

1 {
2   "dpid": {{dpid}},
3   "table_id": {{MacTable}},
4   "priority": 32768,
5   "flags": 1,
6   "match": {
7     "dl_dst": "00:11:11:11:11",
8     "dl_vlan": 200
9   },
10  "instructions": [
11    {
12      "type": "GOTO_TABLE",
13      "table_id": {{RoutingTable}}
14    }
15  ]
16
17
18
19
}

```

Verify the switch has the flow configured

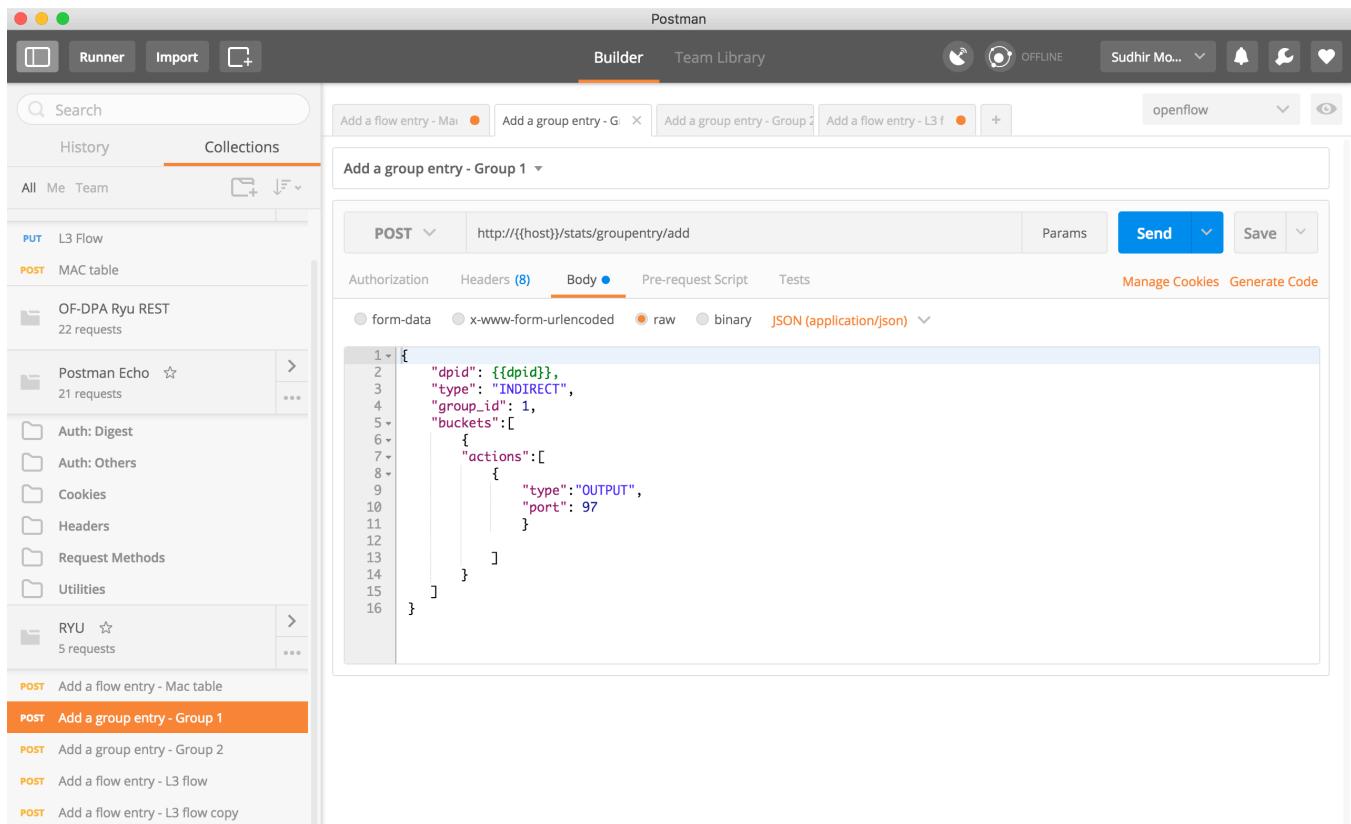
```

admin@PicOS-OVS$ovs-ofctl dump-flows br0

OFPST_FLOW reply (OF1.4) (xid=0x2):
  flow_id=3735291, cookie=0x0, duration=690201.683s, table=20, n_packets=n/a, n_bytes=n/a,
  dl_vlan=200,dl_dst=00:11:11:11:11 actions=goto_table:30

```

Step 3 : Creating a group entry (group_id = 1) for the egress L2 interface



Verify the group exists on the switch

```

admin@PicOS-OVS$ovs-ofctl dump-groups br0
OFPST_GROUP_DESC reply (OF1.4) (xid=0x2):
  group_id=1,type=indirect,bucket=weight:0,actions=output:97
  
```

Step 4 : Create a group entry (group_id = 2) for the egress L3 interface

The screenshot shows the Postman application interface. On the left, there's a sidebar with various collections and requests. In the main area, a specific request is being built:

- Method:** POST
- URL:** http://{{host}}/stats/groupentry/add
- Authorization:** (not explicitly shown)
- Headers:** (8) (not explicitly shown)
- Body:** (raw, JSON application/json)


```

1  [
2   "dpid": {{dpid}},
3   "type": "INDIRECT",
4   "group_id": 2,
5   "buckets": [
6     {
7       "actions": [
8         {
9           "type": "SET_FIELD",
10          "field": "eth_src",
11          "value": "00:01:01:01:01:01"
12        },
13        {
14          "type": "SET_FIELD",
15          "field": "eth_dst",
16          "value": "00:02:02:02:01:01"
17        },
18        {
19          "type": "SET_FIELD",
20          "field": "vlan_vid",
21          "value": 2016
22        },
23        {
24          "type": "GROUP",
25          "group_id": 1
26        }
27      ]
28    }
  ]
      
```
- Params:** (not explicitly shown)
- Tests:** (not explicitly shown)
- Send:** (button)
- Save:** (button)

Verify the switch has the configuration

```

admin@PicOS-OVS$ovs-ofctl dump-groups br0

OFPST_GROUP_DESC reply (OF1.4) (xid=0x2):
  group_id=1,type=indirect,bucket=weight:0,actions=output:97
  group_id=2,type=indirect,bucket=weight:0,actions=set_field:00:01:01:01:01:01->eth_src,set_
  field:00:02:02:02:01:01->eth_dst,set_field:2016->vlan_vid,group:1
  
```

Step 5 : Create a IP flow in the L3 table

The screenshot shows the Postman application interface. On the left sidebar, there are several collections and requests listed under categories like ODL, PUT, POST, and RYU. The 'POST Add a flow entry - L3 flow' request is highlighted with an orange background. The main workspace shows a 'Builder' tab with a 'POST' request to 'http://{{host}}/stats/flowentry/add'. The 'Body' tab is selected, showing a JSON payload for an L3 flow entry. The JSON code is as follows:

```

1  {
2     "dpid": "{{dpid}}",
3     "table_id": {{RoutingTable}},
4     "priority": 32768,
5     "flags": 1,
6     "match": {
7         "dl_type": 2048,
8         "nw_dst": "10.10.1.0/24"
9     },
10    "instructions": [
11        {
12            "type": "WRITE_ACTIONS",
13            "actions": [
14                {
15                    "type": "GROUP",
16                    "group_id": 2
17                },
18                {
19                    "type": "DEC_NW_TTL"
20                }
21            ]
22        },
23        {
24            "type": "GOTO_TABLE",
25            "table_id": {{PolicyTable}}
26        }
27    ]
28 }
29
30 }
31

```

Verify the configuration on the switch

```

admin@PicOS-OVS$ovs-ofctl dump-flows br0

OFPST_FLOW reply (OF1.4) (xid=0x2):
  flow_id=3735291, cookie=0x0, duration=690201.683s, table=20, n_packets=n/a, n_bytes=n/a,
  dl_vlan=200,dl_dst=00:11:11:11:11 actions=goto_table:30
  flow_id=3735292, cookie=0x0, duration=690201.538s, table=30, n_packets=n/a, n_bytes=n/a,
  ip,nw_dst=10.0.1.0/24 actions=write_actions(group:2,dec_ttl),goto_table:60

```

"ovs-ofctl dump-flows br0" shows the software flow table configured by the controller. "ovs-appctl pica/dump-flows" shows the ASIC tables.

```

admin@PicOS-OVS$ovs-appctl pica/dump-flows

Ingress Port Table: (Total 0 flows)

VLAN Table: (Total 0 flows)

Termination MAC Table: (Total 1 flows)
  ID=5 dl_vlan=200,dl_dst=00:11:11:11:11, actions:goto(Unicast Routing Table)

Unicast Routing Table: (Total 1 flows)
  ID=6 ip,nw_dst=10.0.1.0/24,
  actions:group(id=2,indirect,n=1,b0(set(dl_src=00:01:01:01:01:01,dl_dst=00:02:02:02:01:01),set(vlan_vid=2016),group(id=1,indirect,n=1,b0(output:97))),goto(Policy ACL Table))

Policy ACL Table: (Total 1 flows)
  ID=1 priority=18000009,tcp,dl_dst=00:e0:95:00:16:9a,nw_src=172.16.0.123,tp_src=6633,
  actions:set(queue=7,output:60000

```

Connection to OpenDaylight Controller

OpenDaylight is an open source platform for building programmable SDNs (software-defined networks). Learn more about OpenDaylight here.

- OpenDaylight Introduction
- Introduction to the OpenDaylight OpenFlow Controller
- Configure OVS for OpenDaylight Open Flow Controller
- OpenDaylight Controller-OVS Interaction
- OpenDaylight Simple Switch Application
- Message Type of Open Flow
- OVS Commands Reference 04

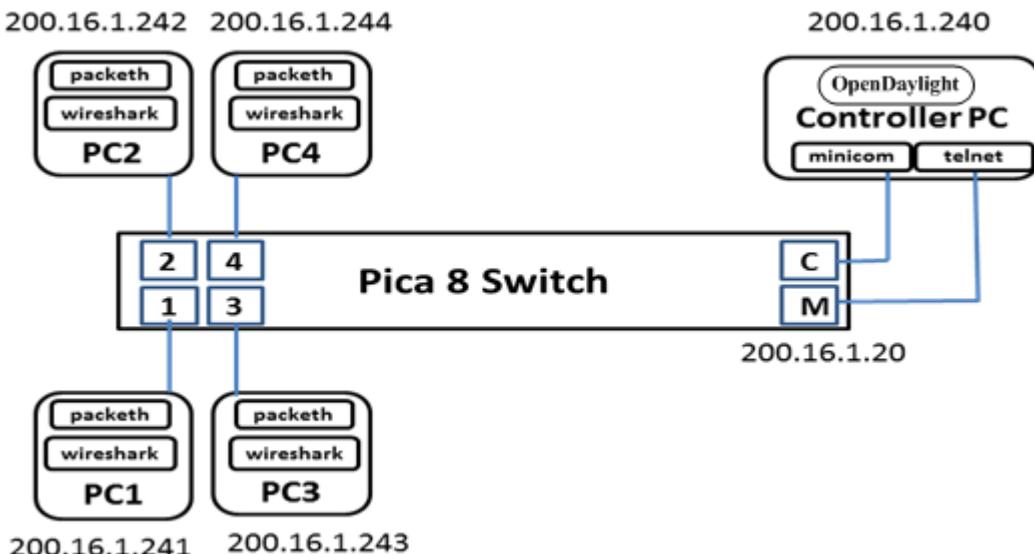
OpenDaylight Introduction

This document provides instructions on how to configure Pica8's open switches to work in various application scenarios. This guide assumes the reader has minimal to no knowledge of the Open Virtual Switch (OVS) implementation defined by <http://openvswitch.org/>; or the OpenFlow protocol, defined by <https://www.opennetworking.org/>. This guide will provide the tools required to configure Pica8's open switches as an OpenFlow switch. By reading this guide, user will gain insights on how to optimize the configuration to work in a specific application environment, while also learning about OVS and the OpenFlow protocol.

This starter kit provides screen shots and a list of off-the-shelf applications needed to complete the configuration, as well as tips on problems users may encounter during the setup. More documents or cookbooks on other subjects will be published periodically. This document provides a tutorial on how to:

- Configure Pica8 as an OVS OpenFlow switch
- Create bridges, add ports, show bridge and port statistics, status, and utilize the OVS database
- Configure flow tables for uni-directional, bi-directional, traffic switching, one-to-many multi-casting, mirroring, filtering, many-to-one aggregation, etc.
- Configure Pica8 OVS OpenFlow switches to interface with the OpenDaylight OpenFlow Controller

Figure 1 – Test bed configuration



In this document, the system configuration depicted in Figure 1 includes:

- A Pica8 P-3295 open switch with 48 x 1 GbE and 4 x 10 GbE uplinks
- 5 Linux PCs running Ubuntu 12.4.1 LTS, one is connected to the management LAN port (RJ45) and console port (RJ45F); this PC is referred to the controller PC. The OpenFlow controller will be running on this PC. Four PCs are connected to 1GbE port 1 to 4 and serve as a data terminal for generating and monitoring traffic
- Tools from installed on all the PCs are listed below. They can be installed through Linux installation utility *apt-get*
 - Terminal emulator minicom
 - Traffic monitoring tool Wireshark
 - Packet generator Packeth
 - ftp and ftppd
 - telnet and telnetd

Introduction to the OpenDaylight OpenFlow Controller

OpenDaylight is an OpenFlow controller that has been integrated with the Pica8 open switch with OVS 1.10 implementation and supports Open Flow v1.3. Additional OpenDaylight information can be found at the OpenDaylight website <http://www.opendaylight.org>. The purpose of Pica8 OpenDaylight integration is to provide an open source SDN platform that the SDN community can prototype, test, and develop for application in an open source environment, with an open flow switching platform for real traffic testing. With the configuration provided in this starter kit, user should be able to have real traffic running within a week to test out the application scenarios using OVS command. Both OVS and OpenDaylight are open source with Apache licenses that developers can access easily.

User can obtain a copy of OpenDaylight from <http://www.opendaylight.org/software/downloads>. To install OpenDaylight, follow the procedure in the OpenDaylight Getting Started Guide.

After installing OpenDaylight, user can first edit the configuration file in */opendaylight/configuration/config.ini* and then start the OpenDaylight controller by using the *./run.sh* command.

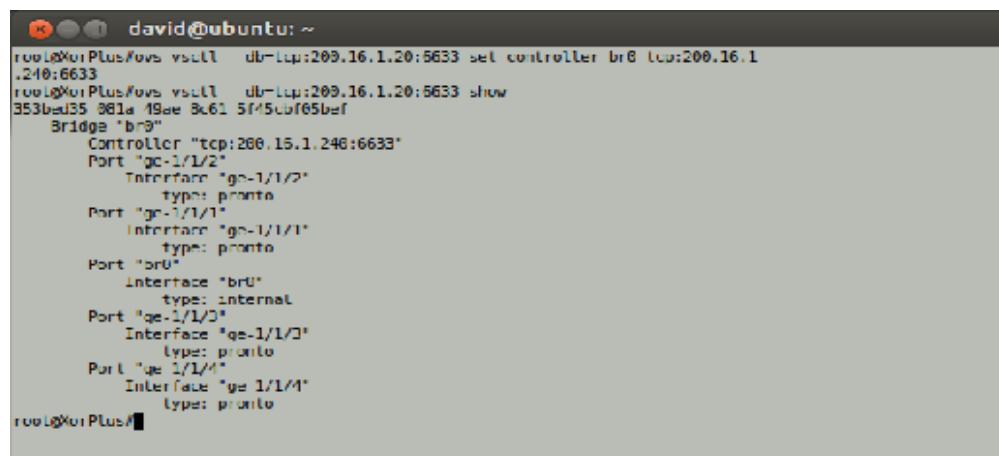
Figure 1

```
root@dev-42:/home/ychen/opendaylight# ./run.sh
osgi> 2014-05-27 10:45:50.871 CST [Start Level Event Dispatcher] INFO
o.o.c.c.s.internal.ClusterManager - I'm a GossipRouter will listen on port 12001
2014-05-27 10:45:51.016 CST [Start Level Event Dispatcher] INFO
o.o.c.c.s.internal.ClusterManager - Started GossipRouter
GossipRouter started at Tue May 27 10:45:51 CST 2014
Listening on port 12001 bound on address 0.0.0.0/0.0.0.0
Backlog is 1000, linger timeout is 2000, and read timeout is 0
2014-05-27 10:45:51.016 CST [Start Level Event Dispatcher] INFO
o.o.c.c.s.internal.ClusterManager - Starting the ClusterManager
2014-05-27 10:45:52.075 CST [fileinstall-./plugins] INFO
o.o.c.n.i.osgi.NetconfImplActivator - Starting TCP netconf server at /127.0.0.1:8383
2014-05-27 10:45:52.341 CST [fileinstall-./plugins] INFO
o.o.c.s.binding.impl.BrokerActivator - Binding Aware Broker initialized
2014-05-27 10:45:52.556 CST [ConfigPersister-registrator] INFO
o.o.c.n.p.i.ConfigPersisterNotificationHandler - Session id received from netconf server: 1
2014-05-27 10:45:52.556 CST [ConfigPersister-registrator] INFO
o.o.c.n.p.i.ConfigPersisterNotificationHandler - No last config provided by backend storage
PersisterImpl
[storage=org.opendaylight.controller.netconf.persist.impl.NoOpStorageAdapter@ade6f7]
2014-05-27 10:45:55.705 CST [Start Level Event Dispatcher] INFO
o.o.c.c.i.ConfigurationService - ConfigurationService Manager init
2014-05-27 10:45:56.239 CST [ControllerI/O Thread] INFO
o.o.c.p.o.core.internal.ControllerIO - Controller is now listening on any:6655
```

Configure OVS for OpenDaylight Open Flow Controller

In OVS, the controller property of the bridge created earlier needs to be added to include the controller IP address and port number. The command *ovs-vsctl set-controller br0 tcp:200.16.1.240:6633*, is to set a controller address for bridge *br0*. The command *ovs-vsctl show* can now show the bridge information. The connection status is not shown because the controller has not been started yet.

Figure 2 – Set OpenDaylight controller IP address



```
david@ubuntu: ~
root@dev-42:~# ovs-vsctl db=libvirt:200.16.1.20:6633 set controller br0 tcp:200.16.1.240:6633
root@dev-42:~# ovs-vsctl db=libvirt:200.16.1.20:6633 show
Bridge "br0"
  Controller "tcp:200.16.1.240:6633"
  Port "gc-1/1/2"
    Interface "gc-1/1/2"
      type: pphonto
  Port "gc-1/1/1"
    Interface "gc-1/1/1"
      type: pphonto
  Port "br0"
    Interface "br0"
      type: internal
  Port "qe-1/1/0"
    Interface "qe-1/1/0"
      type: pphonto
  Port "qe-1/1/1"
    Interface "qe-1/1/1"
      type: pphonto
Port "qe-1/1/2"
  Interface "qe-1/1/2"
    type: pphonto
Port "qe-1/1/3"
  Interface "qe-1/1/3"
    type: pphonto
Port "qe-1/1/4"
  Interface "qe-1/1/4"
    type: pphonto
```

The OpenDaylight controller will be running on the controller PC with IP address 200.16.1.240, using

default port 6633. The port number can be changed. For this exercise, the controller will be started with `./run.sh`

Figure 3 – Start OpenDaylight-manager

```
root@dev-42:~# cd /home/ychen/opendaylight/
root@dev-42:/home/ychen/opendaylight#
root@dev-42:/home/ychen/opendaylight# ./run.sh
osgi> 2014-05-28 09:24:55.270 CST [Start Level Event Dispatcher] INFO o.o.c.c.s.internal.ClusterManager - I'm a GossipRouter will listen on port 12001
2014-05-28 09:24:55.328 CST [Start Level Event Dispatcher] INFO o.o.c.c.s.internal.ClusterManager - Started GossipRouter
2014-05-28 09:24:55.328 CST [Start Level Event Dispatcher] INFO o.o.c.c.s.internal.ClusterManager - Starting the Cluster Manager
GossipRouter started at Wed May 28 09:24:55 CST 2014
Listening on port 12001 bound on address 0.0.0.0/0.0.0
BackLog is 1000, linger timeout is 2000, and read timeout is 0
2014-05-28 09:24:56.734 CST [fileinstall-/plugins] INFO o.o.c.n.i.osgi.NetconfImplActivator - Starting TCP netconf server at /127.0.0.1:8383
2014-05-28 09:24:57.002 CST [ConfigPersister-registrator] INFO o.o.c.n.p.i.ConfigPersisterNotificationHandler - Session id received from netconf server: 1
2014-05-28 09:24:57.002 CST [ConfigPersister-registrator] INFO o.o.c.n.p.i.ConfigPersisterNotificationHandler - No last config provided by backend storage PersisterImpl [storage=org.opendaylight.controller.netconf.persist.impl.NoopStorageAdapter@fle633]
2014-05-28 09:24:57.011 CST [fileinstall-/plugins] INFO o.o.c.s.binding.impl.BrokerActivator - Binding Aware Broker initialized
2014-05-28 09:24:59.337 CST [Start Level Event Dispatcher] INFO o.o.c.c.i.ConfigurationService - ConfigurationService Manager init
2014-05-28 09:25:00.264 CST [ControllerI/O Thread] INFO o.o.c.p.o.core.internal.ControllerIO - Controller is now listening on any:6655
2014-05-28 09:25:00.352 CST [Start Level Event Dispatcher] INFO o.o.c.s.t.provider.ToasterProvider - Provider Session initialized
2014-05-28 09:25:00.706 CST [Start Level Event Dispatcher] INFO o.o.c.s.l.i.LoadBalancerService - Running container name :default
2014-05-28 09:25:00.707 CST [Start Level Event Dispatcher] INFO o.o.c.s.l.i.LoadBalancerService[68%] y_sokkingManager [vips={}
```

Once the controller is started, the connection status will change to `is_connected: true`. The controller port information can also be shown using the command `ovs-ofctl show br0`.

Figure 4 – Show controller connection status

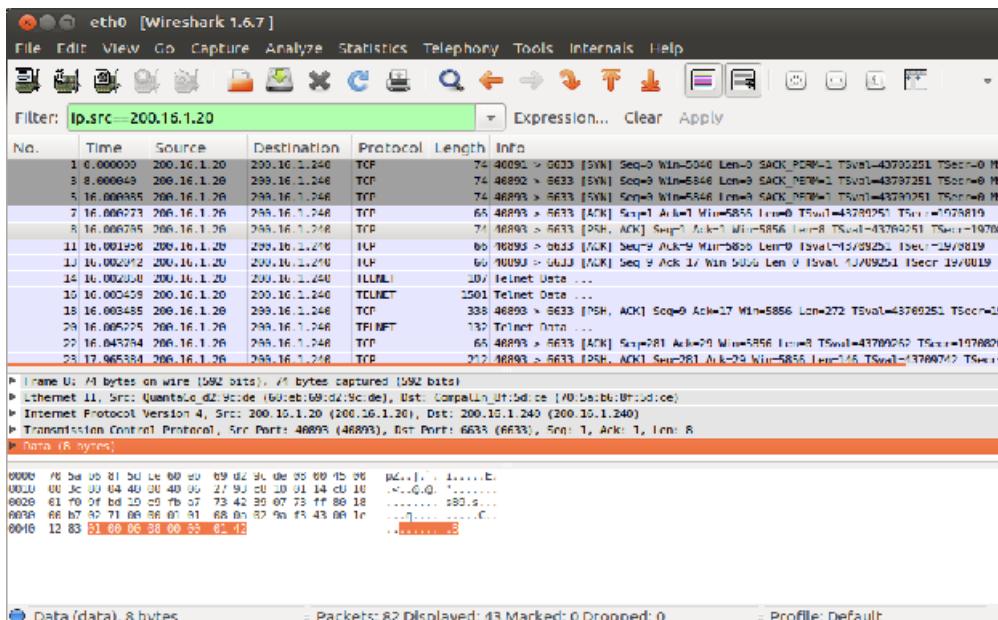
```
david@ubuntu: ~
root@XorPlus:~# ovs-ofctl show br0
Bridge: "br0"
  Controller "tcp:200.16.1.20:6633" is_connected: true
  Port "qc-1/1/2"
    Interface "ga-1/1/2" type: proto
  Port "qc-1/1/1"
    Interface "ga-1/1/1" type: proto
  Port "br0"
    Interface "br0" type: internal
  Port "ge 1/1/3"
    Interface "ge-1/1/3" type: proto
  Port "ge 1/1/4"
    Interface "ge-1/1/4" type: proto
root@XorPlus:~#
```

OpenDaylight Controller-OVS Interaction

Once the controller and OVS are connected, a set of messages will be exchanged. For example, the OVS sends an OFPT_HELLO message to the controller. The hello message is captured on the `wireshark` screen. The first byte of the message is the version number, and the second byte is the OFPT_TYPE. OFPT_HELLO message is type 0.

After the hello message from the switch, the controller sends OFPT_FEATURES_REQUEST (type=5) to retrieve the switch capabilities, including supported open flow version, switch configuration, and port hardware address. The switch sends OFPT_FEATURES_REPLY (type=6) to provide the feature information. The message is shown on both the controller console and the switch console.

Figure 6 – OFPT_HELLO message



The switch console information is provided by the snoop option of the *ovs-ofctl*/command. The command is ***ovs-ofctl snoop br0***. It shows the feature request from the controller and the feature reply with the bridge information. User can compare the switch console information with the controller console information to get a better understanding of the message exchange.

Figure 7 – ovs-ofctl snoop br0

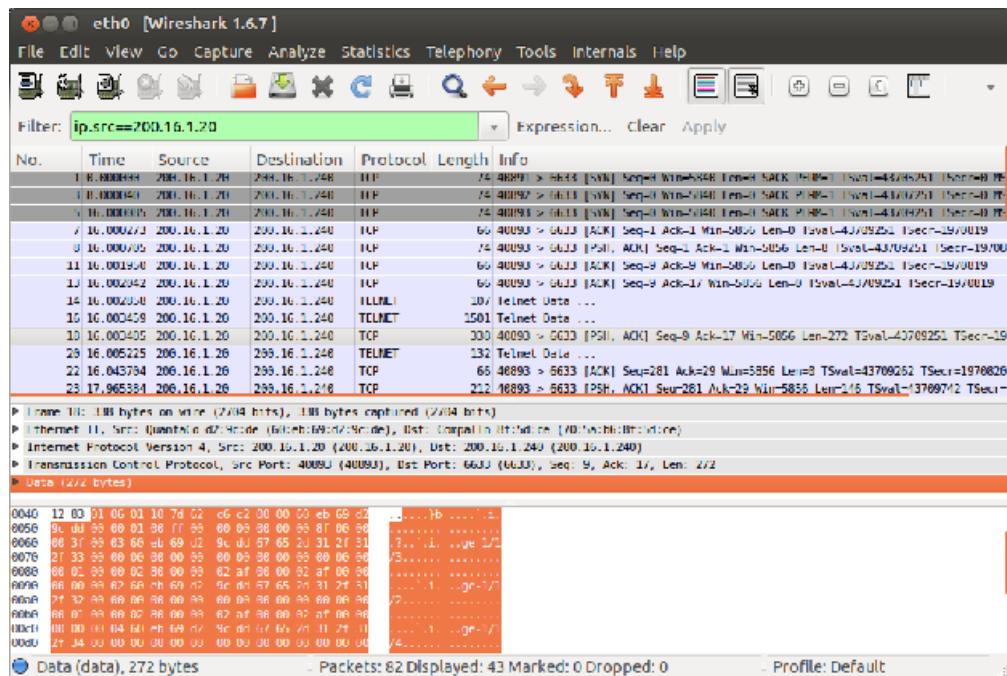
```
david@ubuntu:~
```

```
root@XorPlus:~# ovs-ofctl snoop br0
OFPT_FEATURES_REQUEST (xid=0x272de88e):
OFPT_FEATURES_REPLY (xid=0x272de00e): ver=0x1, dpid:00000eb69d29cd0
n_tables:255, n_buffers:256
features: capabilitiess=0x3f, advertised=613f
1(g-1/1/1): addr:69:cb:50:d2:9c:dd
    config: 0
    state: LINK_DOWN
    current: COPPER AUTO_NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
2(g-1/1/2): addr:69:eb:59:d2:9c:dd
    config: 0
    state: LINK_DOWN
    current: COPPER AUTO_NEG
    advertised: 10MD-ID 10MD-FD 100MD-ID 100MD-FD 1GB-FD COPPER AUTO_NEG
    supported: 10MB HD 10MB FD 100MB HD 100MB FD 1GB FD COPPER AUTO NEG
3(g-1/1/3): addr:69:eb:59:d2:9c:dd
    config: 0
    state: LINK_DOWN
    current: COPPER AUTO_NEG
    advertised: 10MB HD 10MB FD 100MB HD 100MB FD 1GB FD COPPER AUTO NEG
    supported: 10MB HD 10MB FD 100MB HD 100MB FD 1GB FD COPPER AUTO NEG
4(g-1/1/4): addr:69:eb:59:d2:9c:dd
    config: 0
    state: LINK_DOWN
    current: COPPER AUTO_NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO NEG
    supported: 10MD-ID 10MD-FD 100MD-ID 100MD-FD 1GB-FD COPPER AUTO NEG
LULN(b-9): addr:69:eb:69:d2:9c:dd
    config: PORT_DOWN
    state: LINK_DOWN
    current: 10MB-FD COPPER
OFPT_SET_CONFIG (xid=0x272de88f): flags=normal miss_send_len=128
ovs-ofctl: tcpdump exited with status 1
OFPT_PACKET_IN (xid=0x0): total_len=58 in_port=LOCAL data_len=58 buffer=0xffffffff00
priority:0; tunnel0;n_portiffe:txcl0(mac00:23:20:cf:91:63->ff:ff:ff:ff type05ff proto0 ttl0 ip0.0.0.0->0.0.0.0
ovs ofctl: tcpdump exited with status 1
```

The *wireshark* also captured the information. Notice the message type in the second byte is 6 representing the OFPT_FEATURES_REPLY. After the feature reply, the controller sends an OFPT_SET_CONFIG message to set the message parameters like the max length, etc. Once the controller is connected, the OVS changes its default behavior from a layer 2 switch to an OVS switch. It means the flooding is disabled and open flow packet processing starts. Each packet is processed based

on the flow table entry. Unmatched packet is forwarded to the controller for analysis unless a rule is defined to drop the packet. During initial start up with the controller, the flow table is empty. Therefore, packets received from any port are forwarded to the controller. The next message from the switch is type OFPT_PACKET_IN (type=10/0x0a).

Figure 8 – OFPT_FEATURE_REPLY message

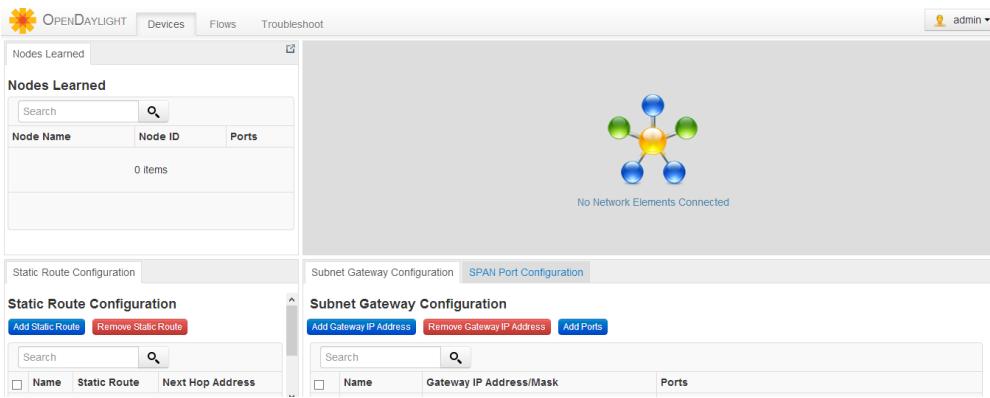


In this exercise, the *OpenDaylight controller* does not have any application to receive and process the OFPT_PACKET_IN message. Therefore, on the controller screen, a bunch of *unhandled_events* are printed on the console. At this point, the OpenDaylight-OVS open flow session is established and ready for Open Flow application to take over the event handling and flow configuration.

OpenDaylight Simple Switch Application

With only the controller connected, without any application, the ping between the PCs cannot work because the ARP requests are forwarded to the controller without any packet processing instructions in the flow tables. OpenDaylight code distribution comes with a set of applications to show how applications can be integrated. Next, an example will be shown of running the simple switch application. The application processes the *packet_in* messages (e.g., ICMP_REQUEST) and instructs the bridge to flood all other ports with the packets. Once the destination host receives the request and replies with its MAC address, this simple switch application sets up the flow table to forward traffic from source port to the correct destination port. This is the default switch behavior that has been tested. When starting the OpenDaylight controller with "/home/ychen/opendaylight# ./run.sh", user can configure the controller on the web, <http://10.10.50.42:8080>/

Figure 9 – Web configure



Message Type of Open Flow

```
# enum ofp_type
OFPT_HELLO = 0 # Symmetric message
OFPT_ERROR = 1 # Symmetric message
OFPT_ECHO_REQUEST = 2 # Symmetric message
OFPT_ECHO_REPLY = 3 # Symmetric message
OFPT_VENDOR = 4 # Symmetric message
OFPT_FEATURES_REQUEST = 5 # Controller/switch message
OFPT_FEATURES_REPLY = 6 # Controller/switch message
OFPT_GET_CONFIG_REQUEST = 7 # Controller/switch message
OFPT_GET_CONFIG_REPLY = 8 # Controller/switch message
OFPT_SET_CONFIG = 9 # Controller/switch message
OFPT_PACKET_IN = 10 # Async message
OFPT_FLOW_REMOVED = 11 # Async message
OFPT_PORT_STATUS = 12 # Async message
OFPT_PACKET_OUT = 13 # Controller/switch message
OFPT_FLOW_MOD = 14 # Controller/switch message
OFPT_PORT_MOD = 15 # Controller/switch message
OFPT_STATS_REQUEST = 16 # Controller/switch message
OFPT_STATS_REPLY = 17 # Controller/switch message
OFPT_BARRIER_REQUEST = 18 # Controller/switch message
OFPT_BARRIER_REPLY = 19 # Controller/switch message
OFPT_QUEUE_GET_CONFIG_REQUEST = 20 # Controller/switch message
OFPT_QUEUE_GET_CONFIG_REPLY = 21 # Controller/switch message
```

OVS Commands Reference 04

```
ovs-vsctl show
ovs-ofctl show br0
ovs-ofctl dump-ports br0
ovs-vsctl list-br
ovs-vsctl list-ports br0
ovs-vsctl list-ifaces br0
ovs-ofctl dump-flows br0
ovs-ofctl snoop br0
ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
ovs-vsctl del-br br0
ovs-vsctl set-controller br0 tcp:172.16.1.240:6633
ovs-vsctl del-controller br0
ovs-vsctl set Bridge br0 stp_enable=true
ovs-vsctl add-port br0 ge-1/1/1 - set interface ge-1/1/1 type=pica8
ovs-vsctl add-port br0 ge-1/1/2 - set interface ge-1/1/2 type=pica8
ovs-vsctl add-port br0 ge-1/1/3 - set interface ge-1/1/3 type=pica8
ovs-vsctl add-port br0 ge-1/1/4 - set interface ge-1/1/4 type=pica8
ovs-vsctl add-port br0 ge-1/1/1 type=pronto options:link_speed=1G
```

```

ovs-vsctl del-port br0 ge-1/1/1
ovs-ofctl add-flow br0 in_port=1,actions=output:2
ovs-ofctl mod-flows br0 in_port=1,dl_type=0x0800,nw_src=100.10.0.1,actions=output:2
ovs-ofctl add-flow br0 in_port=1,actions=output:2,3,4
ovs-ofctl add-flow br0 in_port=1,actions=output:4
ovs-ofctl del-flows br0
ovs-ofctl mod-port br0 1 no-flood
ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,nw_src=192.168.1.241,actions=output:3
ovs-ofctl add-flow br0
in_port=4,dl_type=0x0800,dl_src=60:eb:69:d2:9c:dd,nw_src=198.168.1.2,nw_dst=124.12.123.55,actions=output:1
ovs-ofctl mod-flows br0 in_port=4,dl_type=0x0800,nw_src=192.210.23.45,actions=output:3
ovs-ofctl del-flows br0 in_port=1

```

Connection to a Floodlight Controller

- Floodlight Controller Introduction
- Floodlight Open Flow Controller
- Test Topology
- Configure OVS
- Launch Floodlight
- Floodlight REST Interface

Floodlight Controller Introduction

This is the fourth document of the *Open SDN Starter Kit series*. This document provides instructions on how to configure Pica8's open switches in order to work with *Floodlight Controller*. This document assumes the reader has read the first two documents of the *Open SDN Starter Kit series*.

Floodlight Open Flow Controller

The *Floodlight* Open SDN Controller is an enterprise-class, Apache-licensed, Java-based OpenFlow Controller and it supports OpenFlow v1.0. In fact, *Floodlight* is not just an OpenFlow controller and it also includes a collection of applications built on top the *Floodlight* Controller. Additional *Floodlight* information can be found at *Floodlight* website <http://www.projectfloodlight.org/floodlight/>.

Users can either download the *Floodlight* source from <http://www.projectfloodlight.org/download/> or follow the *Installation Guide*,

<http://docs.projectfloodlight.org/display/floodlightcontroller/Installation+Guide> to install *Floodlight*. In this document, we follow the *Installation Guide* to clone the source to an Ubuntu 11.10 system:

Figure 1 – Clone Floodlight

```

frank@ubuntu:~/wd/pica8/bigswitch/test$ git clone git://github.com/floodlight/floodlight.git
Cloning into floodlight...
remote: Counting objects: 24067, done.
remote: Compressing objects: 100% (9042/9042), done.
remote: Total 24067 (delta 12504), reused 23511 (delta 11994)
Receiving objects: 100% (24067/24067), 26.64 MiB | 960 KiB/s, done.
Resolving deltas: 100% (12504/12504), done.
frank@ubuntu:~/wd/pica8/bigswitch/test$ ls
floodlight
frank@ubuntu:~/wd/pica8/bigswitch/test$ cd floodlight/
frank@ubuntu:~/wd/pica8/bigswitch/test/floodlight$ git checkout fl-last-passed-build
Note: checking out 'fl-last-passed-build'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at e890662... Merge pull request #405 from vy/patch-5
frank@ubuntu:~/wd/pica8/bigswitch/test/floodlight$
```

Why Make Changes?

By default, *Floodlight* forwards the unknown packets from the Pica8 switch to the destination. In this case, things may go well even if there are no flows set in the Pica8 switch. The purpose of this document is to show users how to use Floodlight REST interface to add flows to the Pica8 switch and verify the transmission of the traffic. In order to eliminate the confusion whether the packets are forwarded by Pica8 switch or *Floodlight* controller, we will disable *Floodlight's* default forwarding feature.

Changes to Floodlight

The *Floodlights* default setting is in \$floodlight/src/resources/floodlightdefault.properties. We need to **remove** the line "net.floodlightcontroller.forwarding.Forwarding," as shown below:

Figure 2 – Edit Floodlight Default Properties

```

floodlight.modules = net.floodlightcontroller.storage.memory.MemoryStorageSource, \
net.floodlightcontroller.core.FloodlightProvider, \
net.floodlightcontroller.threadpool.ThreadPool, \
net.floodlightcontroller.devicemanager.internal.DeviceManagerImpl, \
net.floodlightcontroller.devicemanager.internal.DefaultEntityClassifier, \
net.floodlightcontroller.staticflowentry.StaticFlowEntryPusher, \
net.floodlightcontroller.firewall.Firewall, \
net.floodlightcontroller.forwarding.Forwarding, \
net.floodlightcontroller.linkdiscovery.internal.LinkDiscoveryManager, \
net.floodlightcontroller.topology.TopologyManager, \
net.floodlightcontroller.flowcache.FlowCache, \
net.floodlightcontroller.flowcache.FlowReconcileManager, \
net.floodlightcontroller.jython.JythonDebugInterface, \
net.floodlightcontroller.counter.CounterStore, \
net.floodlightcontroller.debugcounter.DebugCounter, \
net.floodlightcontroller.perfmon.PktInProcessingTime, \
net.floodlightcontroller.ui.web.StaticWebRoutable, \
net.floodlightcontroller.loadbalancer.LoadBalancer
net.floodlightcontroller.restserver.RestApiServer.port = 8080
net.floodlightcontroller.core.FloodlightProvider.openflowport = 6633
net.floodlightcontroller.jython.JythonDebugInterface.port = 6655
net.floodlightcontroller.forwarding.Forwarding.idletimeout = 5
net.floodlightcontroller.forwarding.Forwarding.hardtimeout = 0
```

Build Floodlight

Before building *Floodlight*, we need to install *JDK* and *Ant*. Then, issue *ant* to build *Floodlight*.

Figure 3 – Build Floodlight

```
frank@ubuntu:~/wd/pica8/bigswitch/test/floodlight$ ant
Buildfile: /home/frank/wd/pica8/bigswitch/test/floodlight/build.xml

init:
    [mkdir] Created dir: /home/frank/wd/pica8/bigswitch/test/floodlight/target/bin
    [mkdir] Created dir: /home/frank/wd/pica8/bigswitch/test/floodlight/target/bin-test
    [mkdir] Created dir: /home/frank/wd/pica8/bigswitch/test/floodlight/target/lib
    [mkdir] Created dir: /home/frank/wd/pica8/bigswitch/test/floodlight/target/test

compile:
    [javac] Compiling 397 source files to /home/frank/wd/pica8/bigswitch/test/floodlight/target/bin
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.
    [copy] Copying 52 files to /home/frank/wd/pica8/bigswitch/test/floodlight/target/bin

compile-test:
    [javac] Compiling 79 source files to /home/frank/wd/pica8/bigswitch/test/floodlight/target/bin-test

dist:
    [jar] Building jar: /home/frank/wd/pica8/bigswitch/test/floodlight/target/floodlight.jar
    [jar] Building jar: /home/frank/wd/pica8/bigswitch/test/floodlight/target/floodlight-test.jar

BUILD SUCCESSFUL
Total time: 55 seconds
frank@ubuntu:~/wd/pica8/bigswitch/test/floodlight$
```

Then, the *Floodlight* Java Archive file, *floodlight.jar*, is generated under *target* directory and ready to be run.

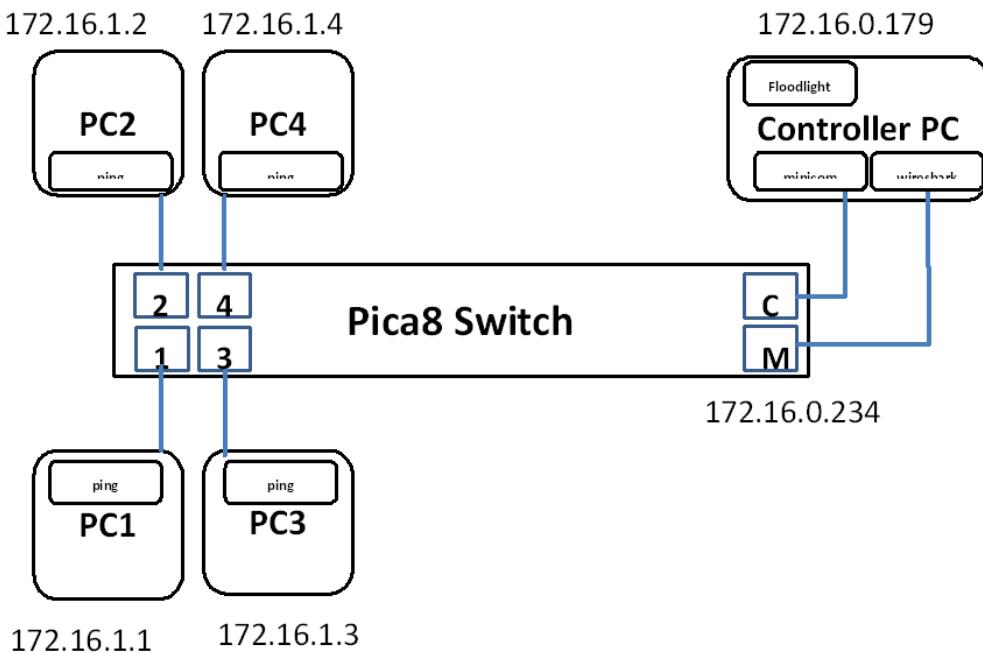
Figure 4 – floodlight.jar

```
frank@ubuntu:~/wd/pica8/bigswitch/test/floodlight$ ls
apps           floodlight_style_settings.xml  NOTICE.txt
build.xml      lib                         README.md
example        LICENSE.txt                  setup-eclipse.sh
findbugs-exclude.xml logback.xml            src
floodlight.sh   Makefile                   target
frank@ubuntu:~/wd/pica8/bigswitch/test/floodlight$ ls target
bin  bin-test  floodlight.jar  floodlight-test.jar  lib  test
frank@ubuntu:~/wd/pica8/bigswitch/test/floodlight$
```

Test Topology

The following picture shows the test topology which is similar to the topology in the first two documents of the *Open SDN Starter Kit series*, even though the IP addresses are different.

Figure 5 – Test Topology



In this document, the systems depicted in the above diagram include:

- A Pica8 switch which is P-3295
- 5 Linux PCs running Ubuntu 11.10
 - The one, connected to the P-3295 management port (RJ45) and console port (RJ45F), is referred as the controller PC. The *Floodlight* controller will be running on this PC.
 - The other four PCs are connected to physical port 1 to 4 and serve as a data terminal to verify the flow.

Configure OVS

In this document, we start OVS manually. After powering on the Pica8 switch, users will see the following messages on the console display. Select choice " 2 " for OVS and enter "yes " to start OVS by manual.

Figure 6 – Start OVS Manually

```

Flash Information :
Device Id      : 0x00000022
Device Flags   : 0x00000000
Device Blocks  : 0x00000100
Device Block Info : 0x00020000

I2C Device Drivers (c) 2007 Quanta Computer Inc.
Registered with major 242

PCI Driver v0.10 (c) 2007 Quanta Computer Inc.
Registered with major 244

WDI Driver v0.1 (c) 2008 Quanta Computer Inc.
Registered with major 245
linux_kernel_bde: module license 'Proprietary' taints kernel.
Checking file system integrity on '/dev/hdal' ...

File system OK
net.netfilter.nf_conntrack_acct = 1
net.ipv6.conf.all.forwarding = 1
System initiating...Please wait...
    Please choose which to start: PicOS L2/L3, PicOS OVS/OpenFlow, or System She
ll:
    (Will choose default entry if no input in 10 seconds.)
    [1] PicOS L2/L3 * default
    [2] PicOS Open vSwitch/OpenFlow
    [3] System Shell
    [4] Boot Menu Editor
Enter your choice (1,2,3,4):2

PicOS Open vSwitch/OpenFlow is selected.

Note: Defaultly, the OVS server is runned with static local management IP and
port 6633.
The default way of vswitch connecting to server is PTCP.
If you do not want default configuration, choose manual start!

sh: you need to specify whom to kill
Do you want start the OVS by manual? (yes/no) yes
You need start root@PicOS-OVS#
root@PicOS-OVS#

```

Please refer to PicOS 1.6 OVS Configuration Guide,
<http://www.pica8.org/document/picos-1.6-ovs-configuration-guide.pdf>, for the details how to configure OVS. First, we need to provide a fixed IP address to the Pica8 switch, create the OVS database, and launch the OVS database server and the switch daemon. Here are the commands:

- ifconfig eth0 172.16.0.234/24
- ovsdb-tool create /ovs/ovs-vswitchd.conf.db /ovs/bin/vswitch.ovsschema
- ovsdb-server /ovs/ovs-vswitchd.conf.db --remote=ptcp: 6632 :172.16.0.234 &
- ovs-vswitchd tcp:172.16.0.234:6632 --pidfile=pica8 --overwrite-pidfile > /var/log/ovs.log 2>
 /dev/null &

Figure 7 – Configure OVS - 1

```

root@PicOS-OVS# ifconfig eth0 172.16.0.234/24
root@PicOS-OVS# PHY: 24520:00 - Link is Up - 1000/Full
ovsdb-tool create /ovs/ovs-vswitchd.conf.db /ovs/bin/vswitch.ovss
chema
root@PicOS-OVS#
root@PicOS-OVS# ovsdb-server /ovs/ovs-vswitchd.conf.db --remote=ptcp:6632:172.16.
0.234 &
root@PicOS-OVS# 2013-06-11T19:17:50Z|00001|ovsdb_server|INFO|ovsdb-server (Open v
Switch) 1.9.90

root@PicOS-OVS# ovs-vswitchd tcp:172.16.0.234:6632 --pidfile=pica8 --overwrit
e-pidfile > /var/log/ovs.log 2> /dev/null &
root@PicOS-OVS#

```

In the following steps, we create the bridge, br0, and add 4 physical ports to it and set up its connection to a specific OpenFlow Controller (*Floodlight* in this case):

- ovs-vsctl add-br br0 – set bridge br0 datapath_type=pica8

- ovs-vsctl add-port br0 ge-1/1/1 – set interface ge-1/1/1 type=pica8
- ovs-vsctl add-port br0 ge-1/1/2 – set interface ge-1/1/2 type=pica8
- ovs-vsctl add-port br0 ge-1/1/3 – set interface ge-1/1/3 type=pica8
- ovs-vsctl add-port br0 ge-1/1/4 – set interface ge-1/1/4 type=pica8
- ovs-vsctl set-controller br0 tcp:172.16.0.179:6633

Figure 8 – Configure OVS - 2

```
root@PicOS-OVS#
root@PicOS-OVS#ovs-vsctl --db=tcp:172.16.0.234:6632 add-br br0 -- set bridge br0 datapath_type=pica8
device ovs-pica8 entered promiscuous mode
device br0 entered promiscuous mode
root@PicOS-OVS#ovs-vsctl --db=tcp:172.16.0.234:6632 add-port br0 ge-1/1/1 -- set interface ge-1/1/1 type=pica8
root@PicOS-OVS#ovs-vsctl --db=tcp:172.16.0.234:6632 add-port br0 ge-1/1/2 -- set interface ge-1/1/2 type=pica8
root@PicOS-OVS#ovs-vsctl --db=tcp:172.16.0.234:6632 add-port br0 ge-1/1/3 -- set interface ge-1/1/3 type=pica8
root@PicOS-OVS#ovs-vsctl --db=tcp:172.16.0.234:6632 add-port br0 ge-1/1/4 -- set interface ge-1/1/4 type=pica8
root@PicOS-OVS#ovs-vsctl --db=tcp:172.16.0.234:6632 set-controller br0 tcp:172.16.0.179:6633
root@PicOS-OVS#
```

We can verify the configuration by issuing:

- ovs-vsctl show

Figure 9 – Show OVS Configuration

```
root@PicOS-OVS#ovs-vsctl --db=tcp:172.16.0.234:6632 show
5388cf70-f685-4de0-b180-06ce9d36e013
Bridge "br0"
  Controller "tcp:172.16.0.179:6633"
  Port "br0"
    Interface "br0"
      type: internal
  Port "ge-1/1/3"
    Interface "ge-1/1/3"
      type: "pica8"
  Port "ge-1/1/4"
    Interface "ge-1/1/4"
      type: "pica8"
  Port "ge-1/1/1"
    Interface "ge-1/1/1"
      type: "pica8"
  Port "ge-1/1/2"
    Interface "ge-1/1/2"
      type: "pica8"
root@PicOS-OVS#
```

In the above pictures, it shows that the OpenFlow Controller has been defined. After the connection between Pica8 switch and OpenFlow Controller, the same command will show the connection status.

Launch Floodlight

It makes no difference whether to launch *Floodlight* before or after bringing up Pica8 switch. In this document, we start *Floodlight* after bringing up Pica8 switch.

Figure 10 – Start Floodlight

```
frank@ubuntu:~/wd/pica8/bigswitch/floodlight$ java -jar target/floodlight.jar
15:47:25.601 [main] INFO n.f.c.module.FloodlightModuleLoader - Loading default modules
15:47:25.609 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Starting module loader
15:47:25.611 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.core.FloodlightProvider
15:47:25.618 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.storage.memory.MemoryStorage
15:47:25.624 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.devicemanager.internal.DeviceManager
15:47:25.632 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.linkdiscovery.internal.LinkDiscovery
15:47:25.635 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.topology.TopologyManager
15:47:25.648 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.forwarding.Forwarding
15:47:25.649 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.flowcache.FlowCache
15:47:25.650 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.flowcache.FlowReconcileManager
15:47:25.655 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.core.OFFMessageFilterManager
15:47:25.656 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.staticflowentry.StaticFlowEntry
15:47:25.657 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.perfmon.PktInProcessingTime
15:47:25.658 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.perfmon.NullPktInProcessingTime
15:47:25.659 [main] DEBUG n.f.c.module.FloodlightModuleLoader - Found module net.floodlightcontroller.restserver.RestApiServer
```

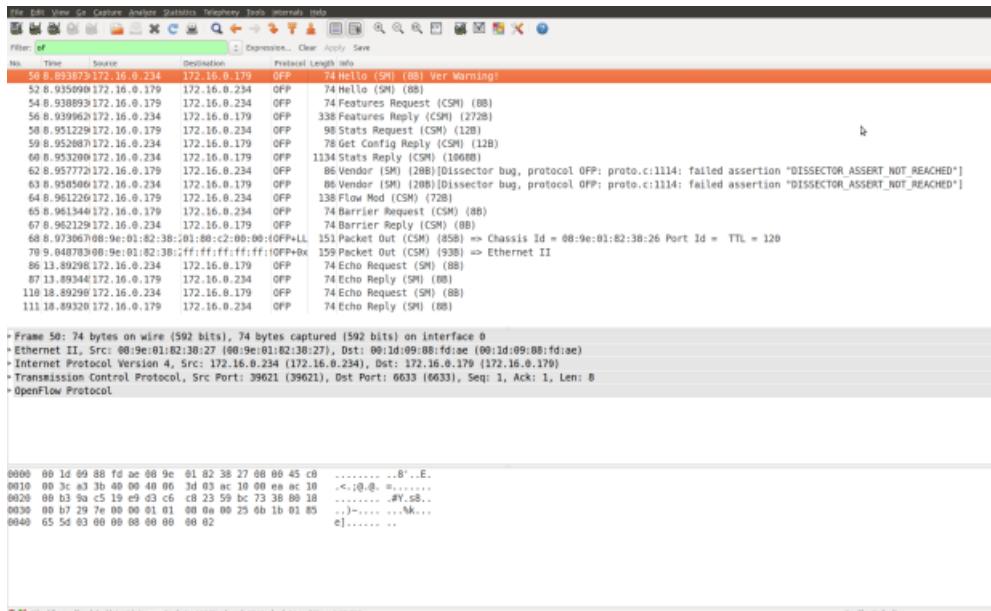
The following figure shows the connection between Pica8 switch and *Floodlight*.

Figure 11 – Start Floodlight

```
.....
17:28:38.731 [New I/O server worker #1-1] INFO n.f.core.internal.Controller - New switch connection from /172.16.0.234:50296
17:28:38.788 [New I/O server worker #1-1] ERROR n.f.core.internal.Controller - Disconnecting switch null due to IO Error: Broken
17:28:38.789 [New I/O server worker #1-1] INFO n.f.core.internal.Controller - Disconnected switch null
17:28:39.326 [New I/O server worker #1-2] INFO n.f.core.internal.Controller - New switch connection from /172.16.0.234:50297
17:28:39.383 [New I/O server worker #1-2] DEBUG n.f.core.internal.Controller - Waiting for switch description from switch /172.16.0.234:50297
17:28:39.384 [New I/O server worker #1-2] INFO n.f.core.internal.Controller - Switch 67:8c:08:01:82:38:26 bound to class net
17:28:39.387 [New I/O server worker #1-2] INFO n.f.core.internal.Controller - Switch Desc - Vendor: Pica8, Inc. Model: Open vSwitch
17:28:39.387 [New I/O server worker #1-2] DEBUG n.f.core.internal.Controller - This controller's role is MASTER, sending initial
17:28:39.390 [New I/O server worker #1-2] DEBUG n.f.core.internal.RoleChanger - Handling role reply for role MASTER from OFSSwitch
17:28:39.390 [New I/O server worker #1-2] DEBUG n.f.core.internal.RoleChanger - Received role reply message from net.floodlightcontroller.ofswitch
17:28:39.400 [main] DEBUG n.f.s.StaticFlowEntryPusher - Switch 67:8c:08:01:82:38:26 connected; processing its static entries
17:28:39.407 [New I/O server worker #1-2] DEBUG n.f.core.internal.Controller - Received a Barrier Reply, no listeners for it
17:28:40.482 [New I/O server worker #1-2] DEBUG n.f.d.internal.DeviceManagerImpl - New device created: Device [deviceKey=0, entityID=0]
17:28:41.645 [New I/O server worker #1-2] DEBUG n.f.d.internal.DeviceManagerImpl - New device created: Device [deviceKey=1, entityID=1]
```

We can also use Wireshark to capture the traffic between Pica8 switch and *Floodlight* as shown below:

Figure 12 – Wireshark Captures



We can also use the following command to show the connection status:

- ovs-vsctl show

Figure 13 – Show OVS Connection Status

```

root@PicOS-OVS# ovs-vsctl --db=tcp:172.16.0.234:6632 show
5388cf70-f685-4de0-b180-06ce9d36e013
  Bridge "br0"
    Controller "tcp:172.16.0.179:6633"
      is connected: true
    Port "br0"
      Interface "br0"
        type: internal
    Port "ge-1/1/3"
      Interface "ge-1/1/3"
        type: "pica8"
    Port "ge-1/1/4"
      Interface "ge-1/1/4"
        type: "pica8"
    Port "ge-1/1/1"
      Interface "ge-1/1/1"
        type: "pica8"
    Port "ge-1/1/2"
      Interface "ge-1/1/2"
        type: "pica8"
root@PicOS-OVS#
root@PicOS-OVS#

```

We can now use the following command to verify that all of the physical ports are connected:

- ovs-ofctl show br0

Figure 14 – Show Physical Port Status

```

root@PicOS-OVS# ovs-ofctl show br0
OFPT_FEATURES_REPLY (OF1.2) (xid=0x2): dpid:678c089e01823826
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS
 1(ge-1/1/1): addr:08:9e:01:82:38:26
    config: 0
    state: LINK_UP
    current: 1GB-FD COPPER AUTO_NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    peer: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER
    speed: 1000 Mbps now, 1000 Mbps max
 2(ge-1/1/2): addr:08:9e:01:82:38:26
    config: 0
    state: LINK_UP
    current: 1GB-FD COPPER AUTO_NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    peer: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER
    speed: 1000 Mbps now, 1000 Mbps max
 3(ge-1/1/3): addr:08:9e:01:82:38:26
    config: 0
    state: LINK_UP
    current: 1GB-FD COPPER AUTO_NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    peer: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER
    speed: 1000 Mbps now, 1000 Mbps max
 4(ge-1/1/4): addr:08:9e:01:82:38:26
    config: 0
    state: LINK_UP
    current: 1GB-FD COPPER AUTO_NEG
    advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
    peer: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER
    speed: 1000 Mbps now, 1000 Mbps max
LOCAL(br0): addr:08:9e:01:82:38:26
  config: 0
  state: LINK_UP
  current: 10MB-FD COPPER
  supported: 10MB-FD COPPER
  speed: 10 Mbps now, 10 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.2) (xid=0x4): frags=normal miss_send_len=0
root@PicOS-OVS#

```

At this moment, there are no flows defined:

- ovs-ofctl dump-flows br0

Figure 15 – No Flows defined in Pica8 Switch

```
root@PicOS-OVS#
root@PicOS-OVS#ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.2) (xid=0x2):
root@PicOS-OVS#
```

If we try to ping from PC1 to PC2, it fails:

Figure 16 – Ping Fails

```
PC1> ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
From 172.16.1.1 icmp_seq=1 Destination Host Unreachable
From 172.16.1.1 icmp_seq=2 Destination Host Unreachable
From 172.16.1.1 icmp_seq=3 Destination Host Unreachable
From 172.16.1.1 icmp_seq=4 Destination Host Unreachable
From 172.16.1.1 icmp_seq=5 Destination Host Unreachable

--- 172.16.1.2 ping statistics ---
5 packets transmitted, 0 received, +5 errors, 100% packet loss, time 4016ms pipe 3
```

Floodlight REST Interface

We can use the browser to view *Floodlights* real time information. The URL can be

<http://172.16.0.179:8080/ui/index.html> or <http://127.0.0.1:8080/ui/index.html> if users access it from Controller PC.

Figure 17 – Access Floodlight Info

The screenshot shows the Floodlight REST Interface dashboard. At the top, there's a navigation bar with links for Dashboard, Topology, Switches, and Hosts, and a checkbox for 'Live updates'. Below the navigation is a 'Controller Status' section with the following details:

Hostname:	localhost:6633
Healthy:	true
Uptime:	3543 s
JVM memory bloat:	39499960 free out of 131530752
Modules loaded:	n.f.topology.TopologyManager, n.f.flowcache.FlowReconcileManager, n.f.deviceManager.internal.DefaultEntityClassifier, n.f.storage.memory.MemoryStorageSource, n.f.debugCounter.DebugCounter, n.f.counter.CounterStore, n.f.restServer.RestApiServer, n.f.firewall.Firewall, n.f.pedmon.PingProcessingTime, n.f.core.FloodlightProvider, n.f.deviceManager.internal.DeviceManagerImpl, n.f.linkDiscovery.internal.LinkDiscoveryManager, n.f.threadpool.ThreadPool, n.f.flowcache.FlowCache, n.f.staticFlowEntry.StaticFlowEntryPusher, n.f.loadbalancer.LoadBalancer,

Below this is a 'Switches (1)' section with a table:

DPID	IP Address	Vendor	Packets	Bytes	Flows	Connected Since
67:8c:08:9e:01:82:38:26	172.16.0.234:54145	Pica8, Inc.	0	0	0	6/12/2013 2:13:05 PM

Finally, there's a 'Hosts (1)' section with a table:

MAC Address	IP Address	Switch Port	Last Seen
67:8c:08:9e:01:82:38:26	172.16.1.1	67:8c:08:9e:01:82:38:26	6/12/2013 2:14:12 PM

User should pay attention to the DPID in the picture above. User needs to replace it with user's own DPID in the following tests.

curl

Use the command line tool curl for transferring data with URL syntax. It is also the tool to send Floodlight REST APIs to configure Pica8 switch flows. We can use apt-get to install it on Controller PC:

- sudo apt-get install curl

Add Flows

Here is the command to add a flow from port 1 to port 2:

- curl -d '{"switch": "67:8c:08:9e:01:82:38:26", "name":"pc1-
pc2", "cookie": "0", "priority": "0",
"ingress-port": "1", "active": "true", "actions": "output=2"}'
<http://127.0.0.1:8080/wm/staticflowentrypusher/json>

We need another flow from port 2 to port 1 in order to make the ping between PC1 and PC2 work.

- curl -d '{"switch": "67:8c:08:9e:01:82:38:26", "name": "pc2-
pc1", "cookie": "0", "priority": "0",
"ingress-port": "2", "active": "true", "actions": "output=1"}'
<http://127.0.0.1:8080/wm/staticflowentrypusher/json>

Now, the ping from PC1 to PC2 works.

Figure 18 – Ping Successes

```
PC1> ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
64 bytes from 172.16.1.2: icmp_req=1 ttl=64 time=1.83 ms
64 bytes from 172.16.1.2: icmp_req=2 ttl=64 time=0.176 ms
64 bytes from 172.16.1.2: icmp_req=3 ttl=64 time=0.159 ms
64 bytes from 172.16.1.2: icmp_req=4 ttl=64 time=0.133 ms
64 bytes from 172.16.1.2: icmp_req=5 ttl=64 time=0.154 ms

--- 172.16.1.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 0.133/0.491/1.833/0.671 ms
PC1>
```

We can add the same flows between port 3 and port 4 to make ping working between them.

Figure 19 – Flows Added

```
root@PicOS-OVS# ovs-ofctl dump-flows br0
OVS1_FLOW reply (OF1.2) (xid=0x2):
cookie=0xa000000000000000, duration=2169.681s, table=0, n_packets=22, n_bytes=2234, priority=0,in_port=1 actions=output:2
cookie=0xa000000000000000, duration=2065.64s, table=0, n_packets=22, n_bytes=2234, priority=0,in_port=2 actions=output:1
cookie=0xa000000000000000, duration=16.046s, table=0, n_packets=0, n_bytes=0, priority=0,in_port=4 actions=output:3
cookie=0xa000000000000000, duration=61.684s, table=0, n_packets=0, n_bytes=0, priority=0,in_port=3 actions=output:4
root@PicOS-OVS#
root@PicOS-OVS#
```

Delete Flows

We can remove all of the flows by issuing:

- curl <http://127.0.0.1:8080/wm/staticflowentrypusher/clear/67:8c:08:9e:01:82:38:26/json>

Configuration Guide for Atrium Stack on ONOS Controller

ONOS is a production ready controller software, capable of enabling applications at scale with its architecture. To get more information on ONOS, please visit their home page.

ONF developed a BGP router application on the ONOS controller called the Atrium. This is a fully functional production ready peering application that uses OpenFlow enabled switches. To get relevant roadmap information and additional information on Atrium, please refer to the Atrium page.

- ONOS Introduction
- Installation Guide
- ONOS Configuration Guide
- Quagga Configuration Guide
- PicOS Configuration Guide

ONOS

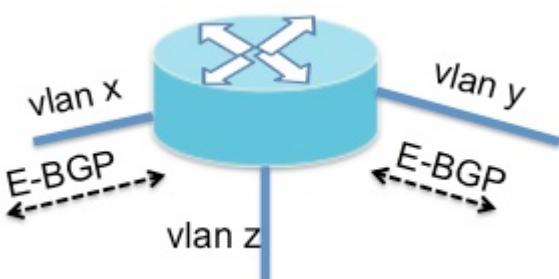
ONOS (Open Network Operating System) is the first open source SDN network operating system targeted specifically at the service provider and mission critical networks. ONOS is purpose built to provide high availability (HA), scale out, and performance for large scale networks. In addition, ONOS has useful Northbound abstractions and APIs to enable easier application development, as well as Southbound abstractions and interfaces to control OpenFlow ready and legacy devices. Thus, ONOS will:

- bring carrier grade features (scale, availability, and performance) to the SDN control plane
- enable web style agility
- help service providers migrate their existing networks to white boxes
- lower service provider CapEx and OpEx

ONOS has been developed in concert with leading service providers (AT&T, NTT Communications), R&E network operators (Internet2, CNIT, CREATE-NET), collaborators (SRI, Infoblox), and with ONF to validate its architecture through real world use cases.

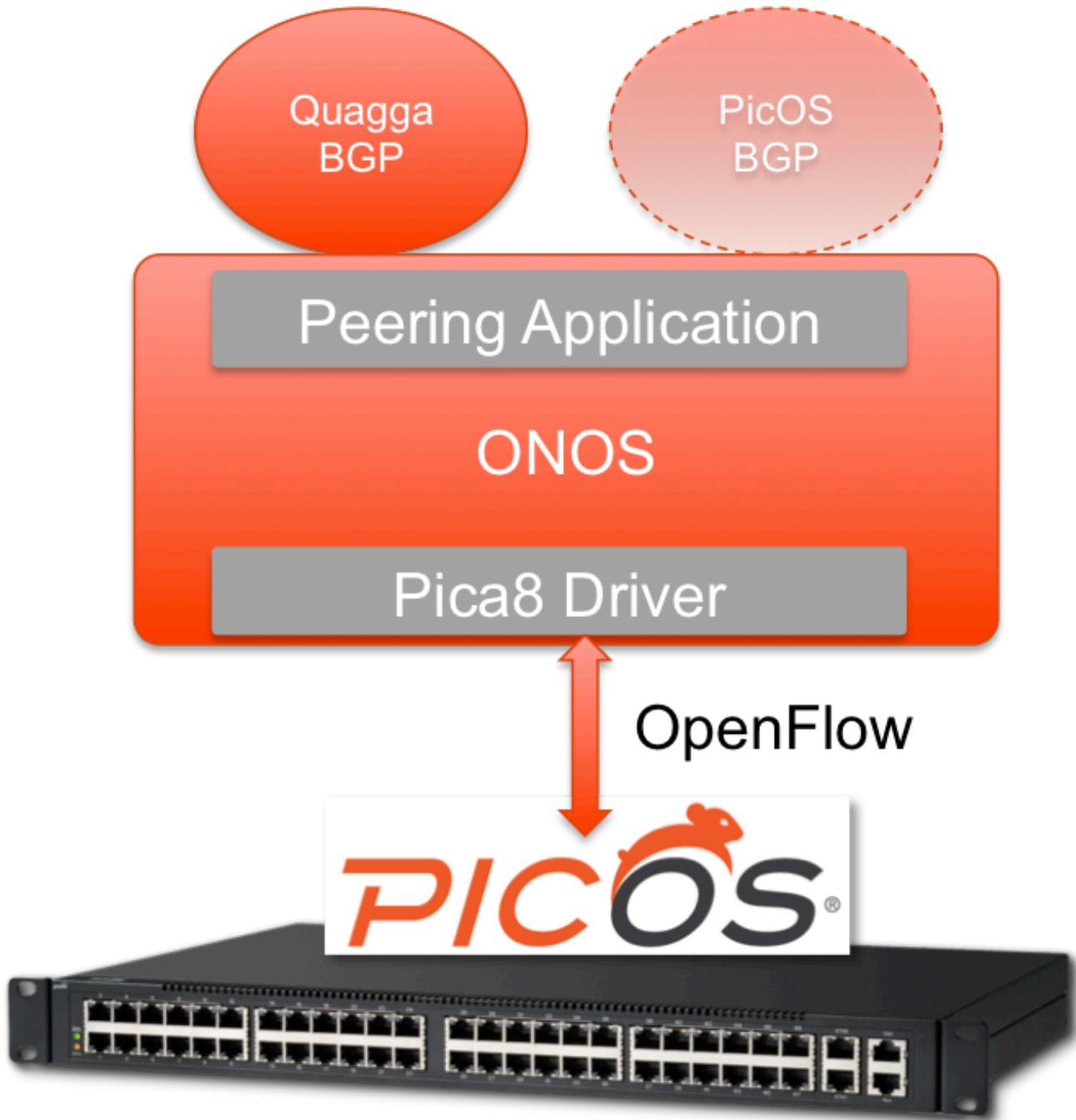
Atrium

In the first release (2015/A), Atrium is an open-source router that speaks BGP with other routers and programs flows on the open flow switch packets between ports or VLANs based on the next-hop learned via BGP peering.



Atrium creates a vertically-integrated stack to produce an SDN based router. This stack can have the forms shown below.

The stack includes a controller (ONOS) with a peering application (called BGP Router) integrated with an instance of Quagga BGP. The controller uses OpenFlow v1.3.4 to communicate with the hardware switch. The peering application uses multi-tables to install flows on the switch.



Installation Guide

Distribution VM

To get started with Atrium Release 2015/A, download the distribution VM (Atrium_2015_A.ova) from here: size ~ 2GB

https://www.dropbox.com/s/vw7k5y2vkhfytgx/Atrium_2015_A.ova?dl=0

login: admin

password: bgprouter

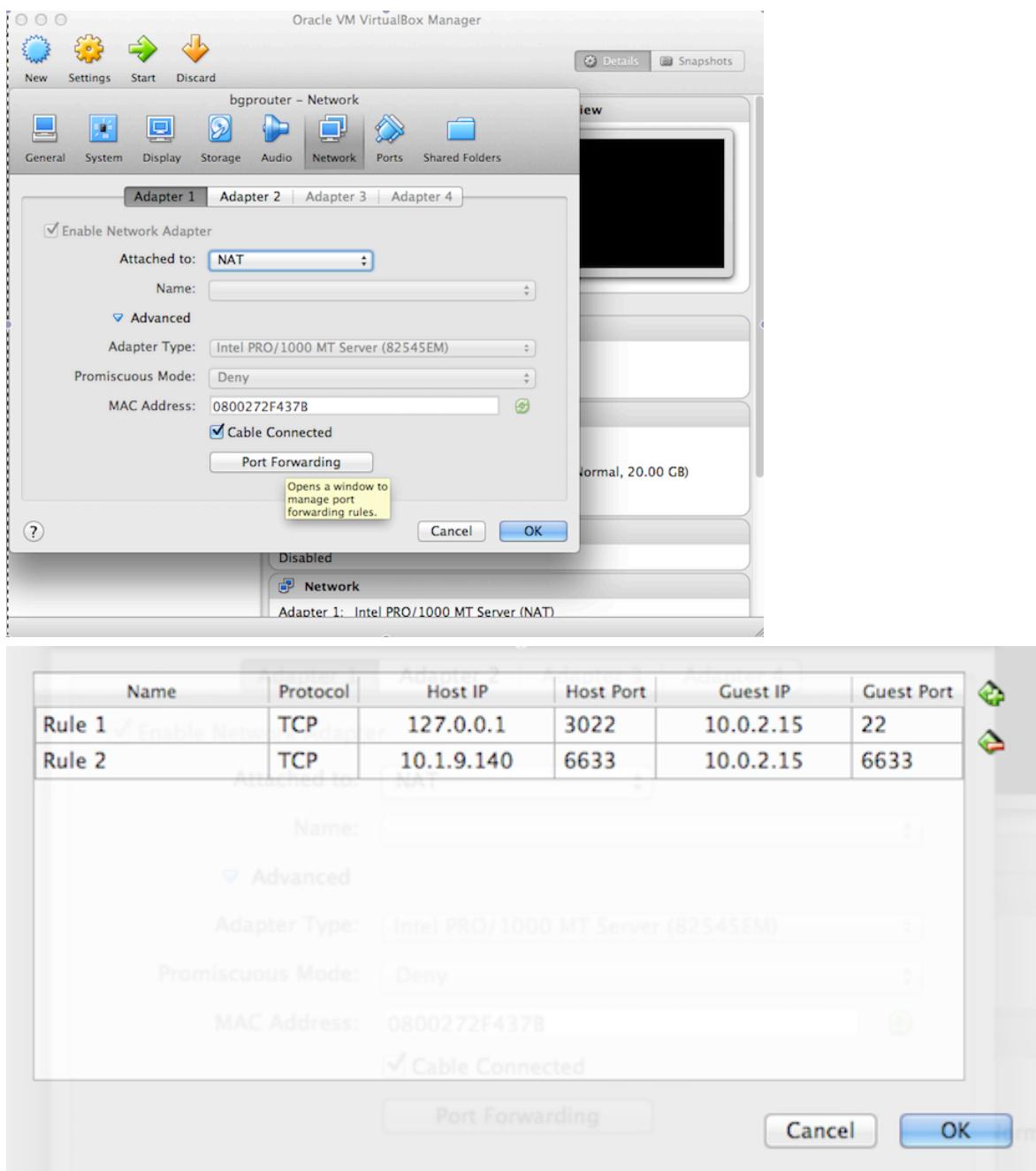
NOTE: This distribution VM is NOT meant for development. The only purpose is to help get a working system up and running for testing and deployment as painlessly as possible. A developer guide using mechanisms other than this VM will be available shortly after the release.

The VM can be run on any desktop, laptop, or server with virtualization software (VirtualBox, Parallels, VMWare Fusion, VMWare Player, etc.). We recommend using VirtualBox for non-server uses. For running on a server, see the subsection below.

Get a recent version of VirtualBox to import and run the VM. We recommend the following:

- 1) using 2 cores and at least 4GB of RAM.
- 2) For networking, user can "Disable" the **2nd** Network Adapter. We only need the 1st network adapter for this release.
- 3) User could choose the primary networking interface (Adapter 1) for the VM to be NATed or "bridged". If choosing to NAT, user would need to create the following port-forwarding rules. The first rule allows user to ssh into user's VM with a command from a Linux or MAC terminal like the one below:

```
$ ssh -X -p 3022 admin@localhost
```



The second rule allows user to connect an external switch to the controller running within the VM (the guest-machine) using the IP address of the host-machine (in the example its 10.1.9.140) on the host-port 6633.

If choosing to bridge (with DHCP) instead of NAT, user should login to the VM to see what IP address was assigned by user's DHCP server (on the eth0 interface). Then, user should use ssh to get in to the VM from a terminal:

```
$ ssh -X admin@<assigned-ip-addr>
```

User can login to the VM with the following credentials --> **login:** admin, **password:** bgprouter

Once in, user should try to ping the outside world as a sanity check (ping www.cnn.com).

Running the Distribution VM on a Server

The Atrium_2015_A.ova file is simply a tar file containing the disk image (vmdk file) and some configuration (ovf file). Most server virtualization softwares can directly run the vmdk file. However, most people prefer to run qcow2 format in servers. First, untar the ova file

```
$ tar xvf Atrium_2015_A.ova
```

Use the following command to convert the vmdk file to qcow2. Then, the virtualization software can be used to create a VM using qcow2 image.

```
$ qemu-img convert -f vmdk Atrium_2015_A-disk1.vmdk -O qcow2 Atrium_2015_A-disk1.qcow2
```

Running the Distribution VM on the Switch

While it should be possible to run the controller and other software that is part of the distribution VM directly on the switch CPU in a linux based switch OS, it is not recommended. This VM has not been optimized for such an installation, and it has not been tested in such a configuration.

Installation Steps

Once user has the VM up and running, the following steps will help bring up the system.

User has two choices:

A) Bring up the Atrium Router completely in software and completely self-contained in this VM. In addition, a complete test infrastructure (other routers to peer with, hosts to ping from, etc.) that can be played with (via the router-test.py script) will be provided. Note that when using this setup, software switches emulate hardware pipelines. Head over to the "Running Manual Tests" section on the Test Infrastructure page.

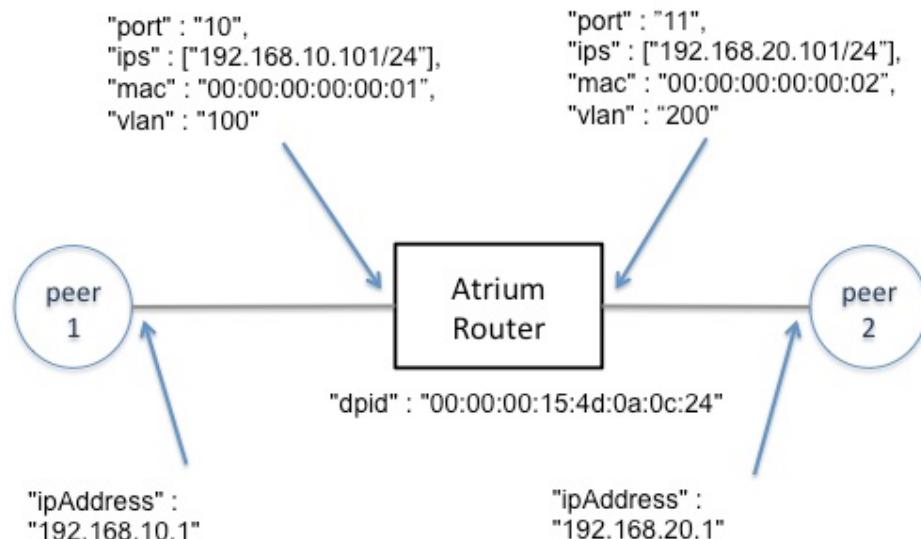
B) Bring up the Atrium Router in hardware, working with one of the seven OpenFlow switches Pica8 has certified to work for Project Atrium. Follow the directions below:

Basically, user needs to configure the controller/app, bring up Quagga, connect it to ONOS (via the router-deploy.py script), and configure the switch that is being worked on to connect it to the controller. The following pages will help with that:

1. Configure and run ONOS
2. Configure and run Quagga
3. Configure Pica8 P-3295

Configuring ONOS

The distribution VM has taken care of most things necessary to configure and run ONOS. What still needs to be done is the routing application within ONOS. At this time, Atrium does not support the ability to change configuration at runtime, an issue which is known. Therefore, it is necessary to pre-configure the controller/app with router-interface addresses and expected BGP peers before launching ONOS. For an example configuration, please see the network shown below.

Figure 1**Figure 2**

```

admin@atrium:~$ cat Applications/config/addresses.json
{
  "addresses" : [
    {
      "dpid" : "00:00:00:15:4d:0a:0c:24",
      "port" : "10",
      "ips" : ["192.168.10.101/24"],
      "mac" : "00:00:00:00:00:01",
      "vlan" : "100"
    },
    {
      "dpid" : "00:00:00:15:4d:0a:0c:24",
      "port" : "11",
      "ips" : ["192.168.20.101/24"],
      "mac" : "00:00:00:00:00:02",
      "vlan" : "200"
    }
  ]
}
  
```

With reference to the topology diagram, the addresses of our router-interfaces are being specified. The "dpid" refers to the OpenFlow datapath-id of the data-plane switch, "port" refers to the data-plane switch port (the OpenFlow port number), "vlan" is one vlan-id configured on that port, "ips" are the set of ip-addresses assigned to that port/vlan, and "mac" is the mac-address assigned to that port.

A few things to note:

- To assign another vlan to the same port, user needs to create another block within the "addresses" array similar to the two shown above.
- The "mac" does not need to be the actual mac address of the physical switch-port. The "mac" does, however, have to be unique because this is what the router uses to reply to ARP-requests for its interface IPs (on that vlan).

- The "mac" can be the same for all port-vlans. In this case, user is essentially using a single "Router-MAC", which is used to reply to ARP-requests for any interface IP and the source-mac address on all Ethernet frames sent out of this router.
- Most switches do not have any dataplane restrictions on the use of "mac" addresses. However, Pica8 switches **require** the use of a single Router MAC (08:9e:01:82:38:68). See the example config files: Applications/config/pica_addresses.json and Applications/config/pica_sdnip.json

Applications/config/sdnip.json is addressed below.

Figure 3

```
admin@atrium:~$ cat Applications/config/sdnip.json
{
    "bgpPeers" : [
        {
            "attachmentDpid" : "00:00:00:15:4d:0a:0c:24",
            "attachmentPort" : "10",
            "ipAddress" : "192.168.10.1"
        },
        {
            "attachmentDpid" : "00:00:00:15:4d:0a:0c:24",
            "attachmentPort" : "11",
            "ipAddress" : "192.168.20.1"
        }
    ],
    "bgpSpeakers" : [
        {
            "name" : "bgpSpeaker1",
            "attachmentDpid" : "00:00:00:00:00:00:aa",
            "attachmentPort" : "1",
            "macAddress" : "aa:bb:cc:dd:ee:ff",
            "interfaceAddresses" : [
                {
                    "interfaceDpid" : "00:00:00:15:4d:0a:0c:24",
                    "interfacePort" : "10",
                    "ipAddress" : "192.168.10.101"
                },
                {
                    "interfaceDpid" : "00:00:00:15:4d:0a:0c:24",
                    "interfacePort" : "11",
                    "ipAddress" : "192.168.20.101"
                }
            ]
        }
    ]
}
```

"bgpPeers" refers to other routers connected to our Atrium router. These peers can be traditional routers or other Atrium routers. "bgpSpeakers" refers to our Atrium router (i.e there is only one bgpSpeaker).

For the "bgpPeers", the app needs to know two things: the IP address of the peer's interface and where it is attached on the Atrium router. In the example above, for peer1, the IP address of the peer's interface is 192.168.10.1, and it is attached to the Atrium router (attachmentDpid: "00:00:00:15:4d:0a:0c:24") on port 10.

For the "bgpSpeakers", the app needs to know the Atrium router's interface IP addresses in the "interfaceAddresses" section. (Do not change information under "bgpSpeakers" like "attachmentDpid" and "attachmentPort".)

Then, ONOS configuration is complete.

Launching ONOS for a Test

If launching ONOS for testing, user can run the following command from the shell (over ssh). If deploying the router, however, user should follow the instructions below.

To ensure that there are no remaining processes left over from previous runs, please run "../router-cleanup.sh".

Figure 4

```
admin@atrium:~$ ./router-cleanup.sh
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller udpbwtest mnexec ivs 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller udpbwtest mnexec ivs 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | grep -o 'dp[0-9]+*' | sed 's/dp/nl:/'
*** Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)' | ip link del sw1-eth4;ip link del as6sw-eth2;ip link del rootTest0n-eth0;ip link del swTest0n-eth1 ) 2> /dev/null
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
admin@atrium:~$
```

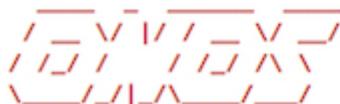
ONOS may already be running. Use the following command to ensure that ONOS has not started automatically.

```
admin@atrium:~$ onos-service localhost stop
```

Then, to launch, run "ok clean" (onos-service start/stop can also be used to launch ONOS.)

Figure 5

```
admin@atrium:~$ ok clean
Creating local cluster configs for IP 127.0.0.1...
Staging builtin apps...
Customizing apps to be auto-activated: drivers,openflow,proxyarp,bgprouter...
Welcome to Open Network Operating System (ONOS)!
```



```
Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown ONOS.
```

```
onos>
```

To check, enter the command "log:tail" on the ONOS (karaf) CLI.

```
onos> log:tail
```

Scroll through the log and verify that the lines, like those in the figure below, can be seen.

Figure 6

```
2015-06-25 17:18:52,483 | INFO | FelixStartLevel | RoutingConfigurationImpl | 166 - org.onosproject.onos-app-routing - 1.2.0.SNAPSHOT | Routing configuration service started
2015-06-25 17:18:52,495 | INFO | FelixStartLevel | BgpRouter | 165 - org.onosproject.onos-app-bgprouter - 1.2.0.SNAPSHOT | Router dpid: of:000000154d0a0c24
2015-06-25 17:18:52,495 | INFO | FelixStartLevel | BgpRouter | 165 - org.onosproject.onos-app-bgprouter - 1.2.0.SNAPSHOT | Control Plane OVS dpid: of:0000000000000aa
2015-06-25 17:18:52,502 | DEBUG | FelixStartLevel | BgpSessionManager | 166 - org.onosproject.onos-app-routing - 1.2.0.SNAPSHOT | BGP Session Manager start.
2015-06-25 17:18:52,537 | INFO | FelixStartLevel | BgpRouter | 165 - org.onosproject.onos-app-bgprouter - 1.2.0.SNAPSHOT | BgpRouter started
```

The "Router dpid" should be the one the user just configured in the config files. The "Control Plane OVS dpid" will be explained in the next section.

User is now ready to configure Quagga.

Launching ONOS for Deployment

For deployment, it is best to launch ONOS within a terminal multiplexer like tmux because, among other things, tmux gives user the ability to maintain persistent working state on remote servers (in our case, on the distribution VM) while detaching and re-attaching at will.

For more on tmux, please see this tutorial.

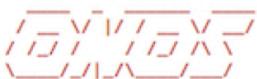
While still connected via ssh into the distribution VM and having already configured the controller but before ONOS is launched, user should enter

```
admin@atrium:~$ tmux new
```

This command should bring up a window like the one shown below. Notice the "bar" at the bottom, identifying the tmux session.

Figure 7

```
RTNETLINK answers: File exists
admin@atrium:~$ ok clean
Creating local cluster configs for IP 127.0.0.1...
Staging builtin apps...
Customizing apps to be auto-activated: drivers,openflow,proxyarp,bgprouter...
Welcome to Open Network Operating System (ONOS)!
```



```
Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown ONOS.
```

```
onos>
```

Now, user can enter "ok clean" to launch ONOS (after running "./router-cleanup.sh" and possibly "onos-service localhost stop").

To detach from the window, enter 'Ctrl-b' followed by 'd'.

User can view the active tmux windows by typing:

```
admin@atrium:~$ tmux ls
0: 1 windows (created Thu Jun 25 17:39:27 2015) [109x36]
```

To reattach to the session:

```
admin@atrium:~$ tmux at -t 0
```

User is now ready to configure Quagga.

Quagga Configuration Guide

Configuring Quagga BGP & Connecting to ONOS

Quagga comes with a rich (Cisco-like) CLI. However, it is not enough to simply configure Quagga BGP. We also need to configure Zebra, as well as the Linux-host on which Quagga runs. In addition, we need to hook up Quagga to ONOS in the control plane. This hookup happens on two different fronts: 1) where BGP communication from the data-plane hardware switch is delivered to ONOS, which in-turn delivers those packets to Quagga (and the reverse communication); and 2) where the ONOS lightweight BGP implementation peers with Quagga BGP using iBGP.

Sounds complicated! But part of Project Atrium's goal is to make life easier for the end user. And so, we have created a script that users can modify in a couple of places and launch with one command that would take care of all the pieces above. If interested in understanding the underlying plumbing, user should refer to the System Architecture.

The script resides at the top level of the directory structure in the distribution VM: **router-deploy.py**

Here is what it looks like:

Figure 1

```

File Edit Options Buffers Tools Python Help
class SDNTopo( Topo ):
    "Sets up control plane components for Router deployment"

    def __init__( self, *args, **kwargs ):
        Topo.__init__( self, *args, **kwargs )

        # Set up Control Plane OVS
        s1 = self.addSwitch('s1', dpid='000000000000aa')

        # Set up BGP speaker
        sdnAs = 65000

        bgpleth0 = { 'ipAddrs' : ['1.1.1.11/24'] }

        bgpleth1 = [
            { 'vlan': 100,
              'mac':'00:00:00:00:00:01',
              'ipAddrs' : ['192.168.10.101/24'] },
            { 'vlan': 200,
              'mac':'00:00:00:00:00:02',
              'ipAddrs' : ['192.168.20.101/24'] }
        ]

        bgp1Intfs = { 'bgp1-eth0' : bgpleth0,
                      'bgp1-eth1' : bgpleth1 }

        neighbors = [ { 'address': '192.168.10.1', 'as': 65001 },
                      { 'address': '192.168.20.1', 'as': 65002 },
                      { 'address': '1.1.1.1', 'as': sdnAs, 'port': 2000 } ]

        bgp1 = self.addHost( "bgp1", intfDict=bgp1Intfs, asNum=sdnAs,
                             neighbors=neighbors, routes=[], cls=BgpRouter)

        # Set up control plane connectivity
        root1 = self.addHost('root1', ip='1.1.1.1/24', inNamespace=False)

        self.addLink( bgp1, root1 )
        self.addLink( bgp1, s1 )

if __name__ == "__main__":
    setLogLevel('debug')
    topo = SDNTopo()

    net = Mininet(topo=topo, controller=RemoteController, switch=OVSSwitch)
    net.start()

    CLI(net)
--:-- router-deploy.py 18% L48 (Python)-----

```

The highlighted boxes in the script above show what user needs to change for the network:

1. The Atrium Router speaks BGP, so it must have an AS number. Set the AS number in the first box.
2. The second box is where user configures the same interface addresses (mac/vlan/ip) for the Atrium Router as was done when configuring ONOS. Note that there is no need to configure "port" information here. Also note that if user had configured the same mac address for all ports (eg. the RouterMAC case for Pica8 switches), the same would need to be done here.
3. The third box is where user configures the peer's (also called neighbor) IP address and AS number. Do not change the last line in this box: it refers to the ONOS iBGP peer.

User may have noticed that step#2 could be done using Linux commands, and steps 1 and 3 can be done using the Quagga CLI. User may also have noticed that we did not give the ability to advertise routes (also known as 'networks' in BGP terminology). This is a known issue with the Atrium router

implementation, an issue which will be addressed in the next release. For this release, the Atrium router can only be used in transit, ie. it can receive and readvertise routes, but it cannot originate route advertisements itself.

Launching Quagga

Similar to what was shown in the "Launching ONOS" section, quagga can be launched simply by running the command in bash (over ssh) if it is just to be run for testing. For deployment, it is better to use tmux and launch quagga in a separate tmux window/pane.

In this example, a new "pane" to launch Quagga was created in the same tmux "window" where ONOS was launched.

In bash, listing the current tmux session should show what was created for launching ONOS.

```
$ tmux ls
0: 1 windows (created Thu Jun 25 17:39:27 2015) [109x36]
```

Attach to the session:

```
$ tmux att -t 0
```

That should bring up the window where ONOS was launched.

Enter the keys Ctrl-b followed by " (double quotation mark). This command will split the window vertically into two panes.

In the lower pane, start the script user just edited in the background.

```
$ sudo ./router-deploy.py &
```

As a checkpoint, user should see the following responses when checking the following commands.

Figure 2

```
admin@atrium:~$ ps aux | grep mininet
root      7474  0.0  0.0  21168  2156 pts/4    Ss+  17:09   0:00 bash --norc -is mininet:c0
root      7480  0.1  0.0  21168  2156 pts/5    Ss+  17:09   0:00 bash --norc -is mininet:bgp1
root      7486  0.0  0.0  21168  2160 pts/6    Ss+  17:09   0:00 bash --norc -is mininet:root1
root      7492  0.0  0.0  21168  2156 pts/7    Ss+  17:09   0:00 bash --norc -is mininet:s1
admin     7627  0.0  0.0  11736   932 pts/3    S+   17:09   0:00 grep --color=auto mininet
```

Here, "bgp1" represents the linux-host (container) that runs Quagga. User can enter this host with:

```
$ ./mininet/util/m bgp1
```

Then, user can telnet into the Quagga BGP process with:

```
$ telnet localhost 2605
```

password: **sdnip**

Now user is in the Quagga CLI. To see the status of BGP peering, enter:

```
> show ip bgp summary
```

Figure 3

```

admin@atrium:~-
2015-06-26 17:09:26,923 | INFO | event-dispatch-0 | FlowObjectiveManager
initializing driver
2015-06-26 17:09:26,968 | INFO | nos-topo-build-1 | TopologyManager
onTime=1435363766942, computeCost=941970, clusters=1, devices=1, links=0] changed
2015-06-26 17:09:26,995 | INFO | ew I/O worker #2 | PortStatsCollector
@ 00:00:00:00:00:00:aa
2015-06-26 17:09:26,997 | INFO | event-dispatch-0 | BgpRouter
2015-06-26 17:09:27,000 | INFO | ew I/O worker #2 | GroupStatsCollector
@ 00:00:00:00:00:00:aa
2015-06-26 17:09:27,017 | INFO | ew I/O worker #2 | OFChannelHandler
2015-06-26 17:09:27,026 | INFO | event-dispatch-0 | TunnellingConnectivityManager
2015-06-26 17:09:27,039 | INFO | ew I/O worker #2 | ntrollerImpl@OpenFlowSwitchAgent
@ 00:00:00:00:00:aa
2015-06-26 17:09:29,685 | DEBUG | event-dispatch-0 | Router
@ 00:29,651, type=HOST_ADDED, subject=StoredHost[id=00:00:00:00:00:01/100, mac=00:00:00:00:00:01/100, location=Hostlocation[elementId=of:00000000000000aa, portNumber=1], ipAddresses=[192.168.10.101]]
2015-06-26 17:09:29,696 | DEBUG | event-dispatch-0 | Router
@ 00:00:00:00:01
2015-06-26 17:09:29,697 | DEBUG | event-dispatch-0 | Router
@ 00:29,672, type=HOST_ADDED, subject=StoredHost[id=00:00:00:00:00:02/200, mac=00:00:00:00:00:02/200, location=Hostlocation[elementId=of:00000000000000aa, portNumber=1], ipAddresses=[192.168.20.101]]
2015-06-26 17:09:29,697 | DEBUG | event-dispatch-0 | Router
@ 00:00:00:00:02
2015-06-26 17:09:32,634 | DEBUG | w I/O worker #18 | BgpSession
@ 1.1.1.1:2000
2015-06-26 17:09:32,637 | DEBUG | w I/O worker #18 | BgpSessionManager
2015-06-26 17:09:32,643 | DEBUG | w I/O worker #18 | BgpOpen
2015-06-26 17:09:32,644 | DEBUG | w I/O worker #18 | BgpOpen
2015-06-26 17:09:32,645 | DEBUG | w I/O worker #18 | BgpOpen
2015-06-26 17:09:32,645 | DEBUG | w I/O worker #18 | BgpOpen
2015-06-26 17:09:32,647 | DEBUG | w I/O worker #18 | BgpOpen
AS 65000 BGP-ID 192.168.10.101 Holdtime 9

admin@atrium:~$ ./mininet/util/m bgp1
ONOS_CELL=sdnip_single_instance
OCI=127.0.0.1
OCL=127.0.0.1
OCN=127.0.0.1
ONOS_APPS=drivers,openflow,proxyarp,bgprouter
ONOS_USER=admin
ONOS_NIC=127.0.0.1*
root@atrium:~# telnet localhost 2605
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^}'.

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
bgp1> sh ip bgp su
bgp1> sh ip bgp summary
BGP router identifier 192.168.10.101, local AS number 65000
RIB entries 0, using 0 bytes of memory
Peers 3, using 13 KiB of memory

Neighbor      V   AS MsgRcvd MsgSent  TblVer  InQ OutQ Up/Down State/PfxRcd
1.1.1.1        V   450000    43    46      0     0  00:02:18      0
192.168.10.1  V   450001    0     0      0     0 never   Active
192.168.20.1  V   450002    0     0      0     0 never   Active

Total number of neighbors 3
bgp1> █
```

The screenshot above shows that the iBGP peering session between ONOS (1.1.1.1) and Quagga is up. The screenshot also shows that the BGP peering session with the neighbors (peers) is not up. This display is due to the fact that the dataplane switch and controller have not been connected.

User is now ready to connect a switch of choice to the controller.

Configuration & Launch

Atrium Specific Configuration

1. Enable multi-table on PicOS. Table 252 is the FDB for IP entries. The preference setting instructs the switch to prioritize match on FDB entries over TCAM.

```
admin@PicOS-OVSS$ ovs-vsctl set-l3-mode TRUE 252
```

```
admin@PicOS-OVSS$ ovs-vsctl set-l2-l3-preference TRUE
```

Verify multi-tables are enabled.

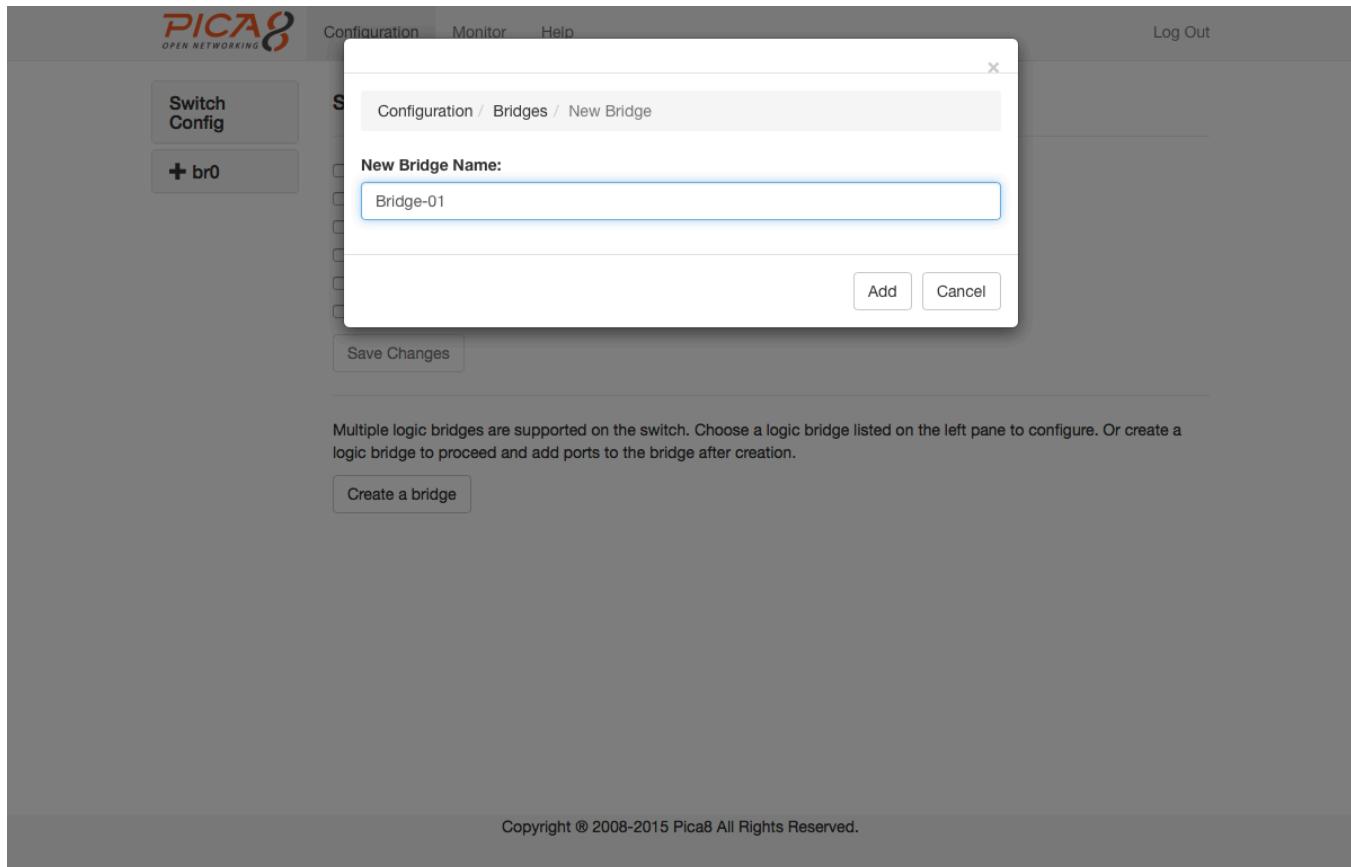
```
admin@PicOS-OVS$ovs-vsctl show-12-13-preference
```

12/13 flow preference is enabled

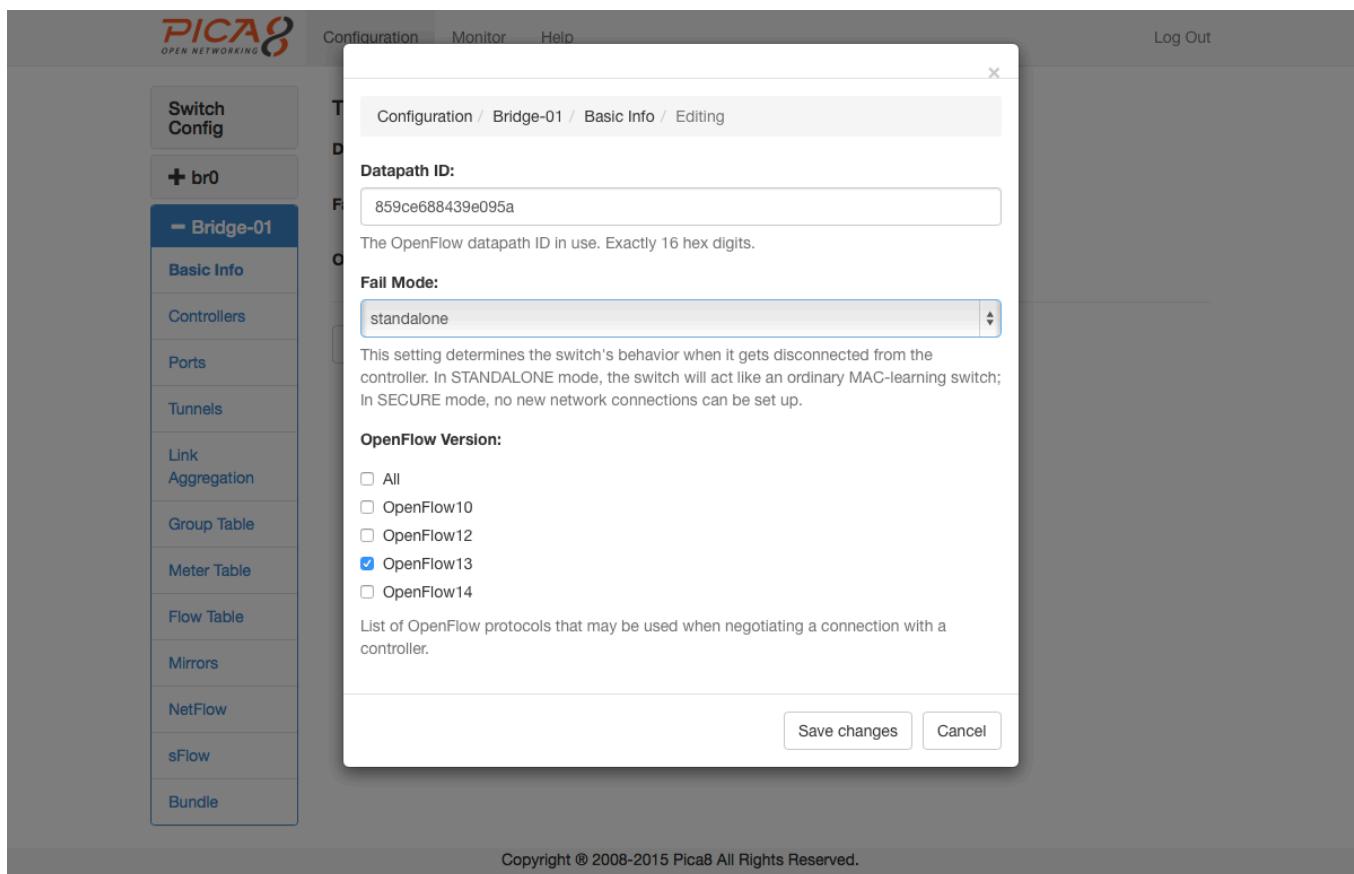
```
admin@Picos-OVS$ovs-vsctl show-13-mode
```

13 mode is enabled, table id is 252

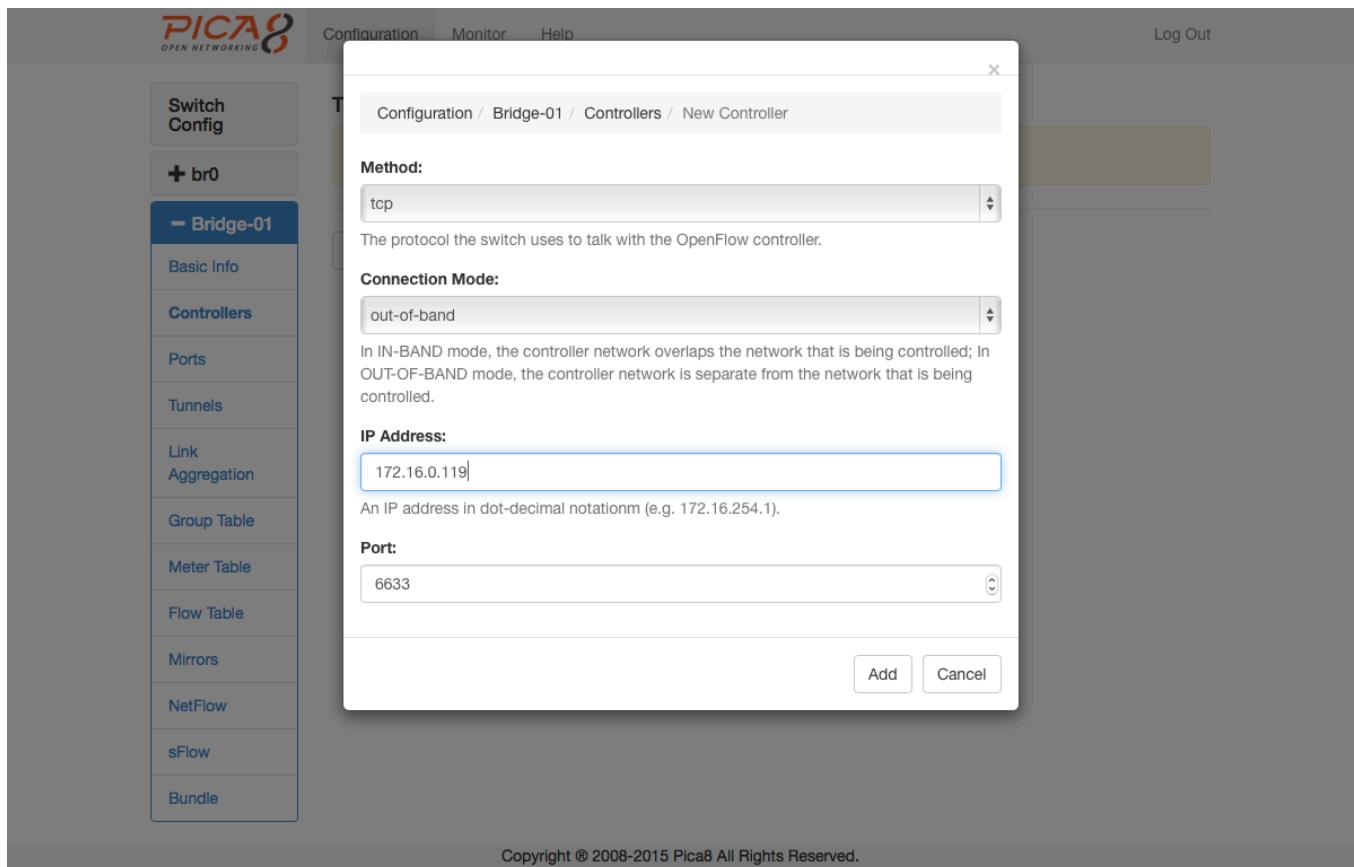
2. Next, create a bridge. Using the GUI is the easiest way: point a browser to the switch management port IP address, use the configuration tab, click on the "Create a bridge" button, and give the bridge a name.



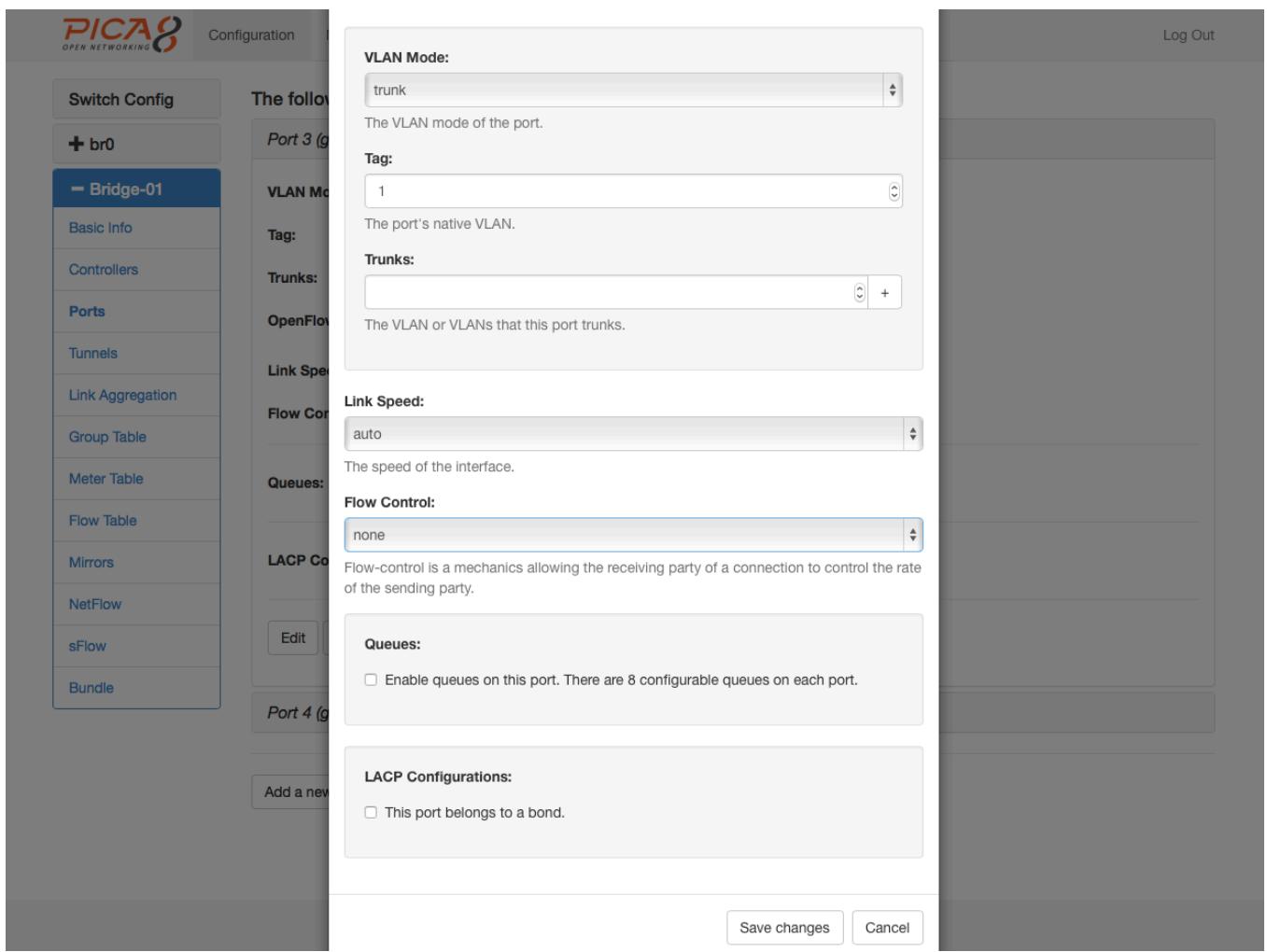
3. Click on the newly created bridge and then "Basic Info". Hit the Edit button. Give a Datapath ID (same one configured in ONOS). Select OpenFlow version 13. The fail mode does not matter.



4. Add a new controller: tcp, out-of-band, <controller-IP> and port 6633



5. Add as many ports as wanted to the bridge. Select the port number. Select TRUNK mode (important). Don't worry about the Tag/Trunk and there is no Flow Control.



Step 6: Finally, remember to use the routerMac: 08:9e:01:82:38:68 for all ports when doing configuration in the controller (in addresses.json and router-deploy.py)

Step 7: Verify the appropriate flows have been installed

```
admin@PicOS-OVS$ovs-ofctl dump-flows atrium-br0
OFPST_FLOW reply (OF1.3) (xid=0x2):
cookie=0x20000001fc94a5, duration=524.485s, table=0, n_packets=n/a, n_bytes=3648, send_flow_rem priority=65535,arp,in_port=20,d1_vlan=200,d1_dst=ff:ff:ff:ff:ff:ff
actions=CONTROLLER:65535
cookie=0x20000001faf7b7, duration=524.485s, table=0, n_packets=n/a, n_bytes=4480, send_flow_rem priority=65535,arp,in_port=10,d1_vlan=100,d1_dst=ff:ff:ff:ff:ff:ff
actions=CONTROLLER:65535
cookie=0x200000af776b1b, duration=524.485s, table=0, n_packets=n/a, n_bytes=1664, send_flow_rem priority=65535,arp,in_port=20,d1_vlan=200,d1_dst=08:9e:01:82:38:68
actions=CONTROLLER:65535
cookie=0x200000af75c32d, duration=524.485s, table=0, n_packets=n/a, n_bytes=2688, send_flow_rem priority=65535,arp,in_port=10,d1_vlan=100,d1_dst=08:9e:01:82:38:68
actions=CONTROLLER:65535
cookie=0x200000cb4300f, duration=524.527s, table=0, n_packets=n/a, n_bytes=0, send_flow_rem priority=0 actions=drop
cookie=0x150000c20cbda8, duration=524.485s, table=0, n_packets=n/a, n_bytes=280738, send_flow_rem priority=65535,ip,in_port=20,d1_vlan=200,d1_dst=08:9e:01:82:38:68,nw_dst=192.168.20.101 actions=CONTROLLER:65535
cookie=0x150000c17e35b0, duration=524.485s, table=0, n_packets=n/a, n_bytes=274262, send_flow_rem priority=65535,ip,in_port=10,d1_vlan=100,d1_dst=08:9e:01:82:38:68,nw_dst=192.168.10.101 actions=CONTROLLER:65535
cookie=0x150000000e1781, duration=9.026s, table=252, n_packets=n/a, n_bytes=n/a, send_flow_rem priority=220,ip,d1_vlan=100,d1_dst=08:9e:01:82:38:68,nw_dst=1.0.0.0/24 actions=set_field:08:9e:01:82:38:68->eth_src, set_field:00:90:0b:1f:04:8b->eth_dst, set_field:4196->vlan_vid,output:10
cookie=0x150000000e1781, duration=9.064s, table=252, n_packets=n/a, n_bytes=n/a, send_flow_rem priority=220,ip,d1_vlan=200,d1_dst=08:9e:01:82:38:68,nw_dst=1.0.0.0/24 actions=set_field:08:9e:01:82:38:68->eth_src, set_field:00:90:0b:1f:04:8b->eth_dst, set_field:4196->vlan_vid,output:10
cookie=0x150000000e1781, duration=427.557s, table=252, n_packets=n/a, n_bytes=n/a, send_flow_rem priority=220,ip,d1_vlan=200,d1_dst=08:9e:01:82:38:68,nw_dst=2.0.0.0/24 actions=set_field:08:9e:01:82:38:68->eth_src, set_field:00:90:0b:23:7d:17->eth_dst, set_field:4296->vlan_vid,output:20
cookie=0x150000000e1781, duration=427.558s, table=252, n_packets=n/a, n_bytes=n/a, send_flow_rem priority=220,ip,d1_vlan=100,d1_dst=08:9e:01:82:38:68,nw_dst=2.0.0.0/24 actions=set_field:08:9e:01:82:38:68->eth_src, set_field:00:90:0b:23:7d:17->eth_dst, set_field:4296->vlan_vid,output:20
```

Managing PicOS Switches with OpenDaylight

The OpenDaylight controller is a Java application that runs in a JVM (Java Virtual Machine). You can run OpenDaylight on any operating system (including Windows) with a JVM installed, though Linux is the recommended platform for running OpenDaylight.

We are using a *Ubuntu 14.04.3 LTS Desktop* virtual machine in VirtualBox, to install and run OpenDaylight.

We are using Lithium-SR1, which is the current release of OpenDaylight at the time of this writing. You can download Lithium-SR1 from <http://www.opendaylight.org/downloads/>.

```
odl@pica8:~$ wget
https://nexus.opendaylight.org/content/groups/public/org/opendaylight/integration/distribution-karaf/0.3.1-Lithium-SR1/distribution-karaf-0.3.1-Lithium-SR1.zip
--2015-09-13 15:35:02--
https://nexus.opendaylight.org/content/groups/public/org/opendaylight/integration/distribution-karaf/0.3.1-Lithium-SR1/distribution-karaf-0.3.1-Lithium-SR1.zip
Resolving nexus.opendaylight.org (nexus.opendaylight.org)... 23.253.119.7
Connecting to nexus.opendaylight.org (nexus.opendaylight.org)|23.253.119.7|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 280858616 (268M) [application/zip]
Saving to: "distribution-karaf-0.3.1-Lithium-SR1.zip"

100%[=====] 280,858,616 788KB/s
in 5m 54s

2015-09-13 15:40:57 (775 KB/s) - "distribution-karaf-0.3.1-Lithium-SR1.zip" saved
[280858616/280858616]
```

Before you can run OpenDaylight, you need to install Java.

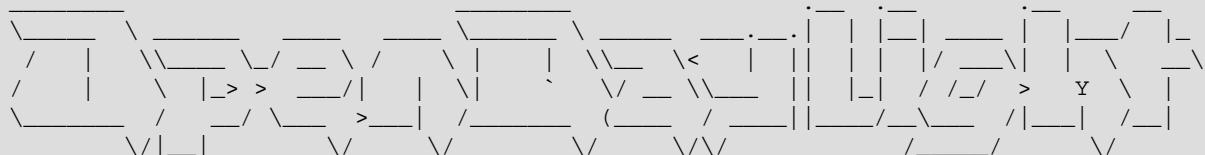
```
odl@pica8:~$ sudo apt-get install default-jre
```

Unzip the downloaded zip file *distribution-karaf-0.3.1-Lithium-SR1.zip*.

```
odl@pica8:~$ sudo unzip distribution-karaf-0.3.1-Lithium-SR1.zip
```

Run the **karaf** file in the **bin** folder, to launch OpenDaylight.

```
odl@pic8:~$ cd distribution-karaf-0.3.1-Lithium-SR1
odl@pic8:~/distribution-karaf-0.3.1-Lithium-SR1$ sudo bin/karaf
karaf: JAVA_HOME not set; results may vary
```



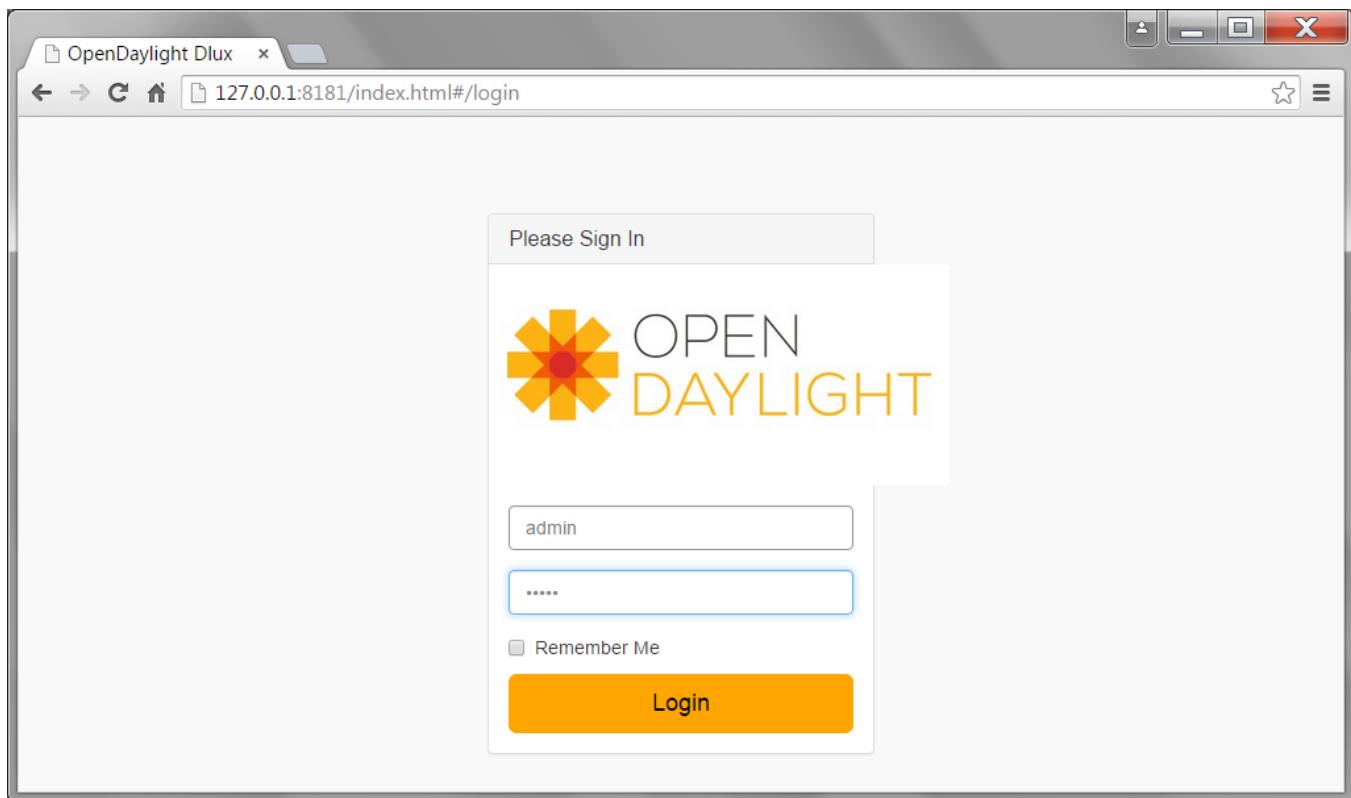
```
Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.
```

```
opendaylight-user@root>
```

OpenDaylight doesn't come with any features installed by default. You have to use the console to install the required features. We are going to install a few additional packages including DLUX, the web interface of OpenDaylight.

```
opendaylight-user@root>feature:install odl-restconf-all odl-l2switch-switch
odl-mdsal-apidocs odl-dlux-all
```

That's all you need to do to get OpenDaylight up and running. Open a browser and go to <http://127.0.0.1:8181/index.html> to reach DLUX - the web interface of OpenDaylight. Replace `127.0.0.1` with the IP address of the machine which you installed OpenDaylight on.



Login to DLUX using `admin` as both username and password.

i You may need to wait for a few minutes before you can successfully login to the web interface of OpenDaylight.

Installing OpenDaylight on Linux

The OpenDaylight controller is a Java application that runs in a JVM (Java Virtual Machine). You can run OpenDaylight on any operating system (including Windows) with a JVM installed, though Linux is the recommended platform for running OpenDaylight.

We are using a *Ubuntu 14.04.3 LTS Desktop* virtual machine in VirtualBox, to install and run OpenDaylight.

We are using Lithium-SR1, which is the current release of OpenDaylight at the time of this writing. You can download Lithium-SR1 from <http://www.opendaylight.org/downloads/>.

```
odl@pica8:~$ wget
https://nexus.opendaylight.org/content/groups/public/org/opendaylight/integration/distribution-karaf/0.3.1-Lithium-SR1/distribution-karaf-0.3.1-Lithium-SR1.zip
--2015-09-13 15:35:02--
https://nexus.opendaylight.org/content/groups/public/org/opendaylight/integration/distribution-karaf/0.3.1-Lithium-SR1/distribution-karaf-0.3.1-Lithium-SR1.zip
Resolving nexus.opendaylight.org (nexus.opendaylight.org)... 23.253.119.7
Connecting to nexus.opendaylight.org (nexus.opendaylight.org)|23.253.119.7|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 280858616 (268M) [application/zip]
Saving to: "distribution-karaf-0.3.1-Lithium-SR1.zip"

0%[=====] 0.00M/280858616 [00:00<00:00]
```

```
100%[=====] 280,858,616 788KB/s
in 5m 54s

2015-09-13 15:40:57 (775 KB/s) - distribution-karaf-0.3.1-Lithium-SR1.zip saved
[280858616/280858616]
```

Before you can run OpenDaylight, you need to install Java.

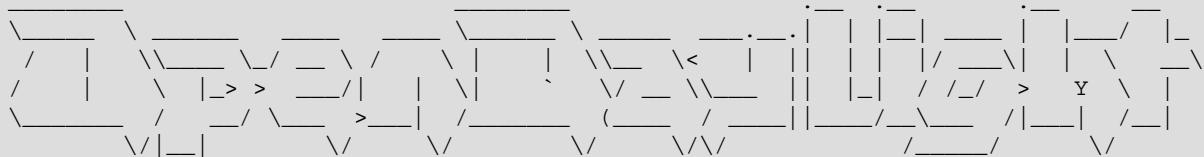
```
odl@pica8:~$ sudo apt-get install default-jre
```

Unzip the downloaded zip file *distribution-karaf-0.3.1-Lithium-SR1.zip*.

```
odl@pica8:~$ sudo unzip distribution-karaf-0.3.1-Lithium-SR1.zip
```

Run the **karaf** file in the **bin** folder, to launch OpenDaylight.

```
odl@pica8:~$ cd distribution-karaf-0.3.1-Lithium-SR1
odl@pica8:~/distribution-karaf-0.3.1-Lithium-SR1$ sudo bin/karaf
karaf: JAVA_HOME not set; results may vary
```



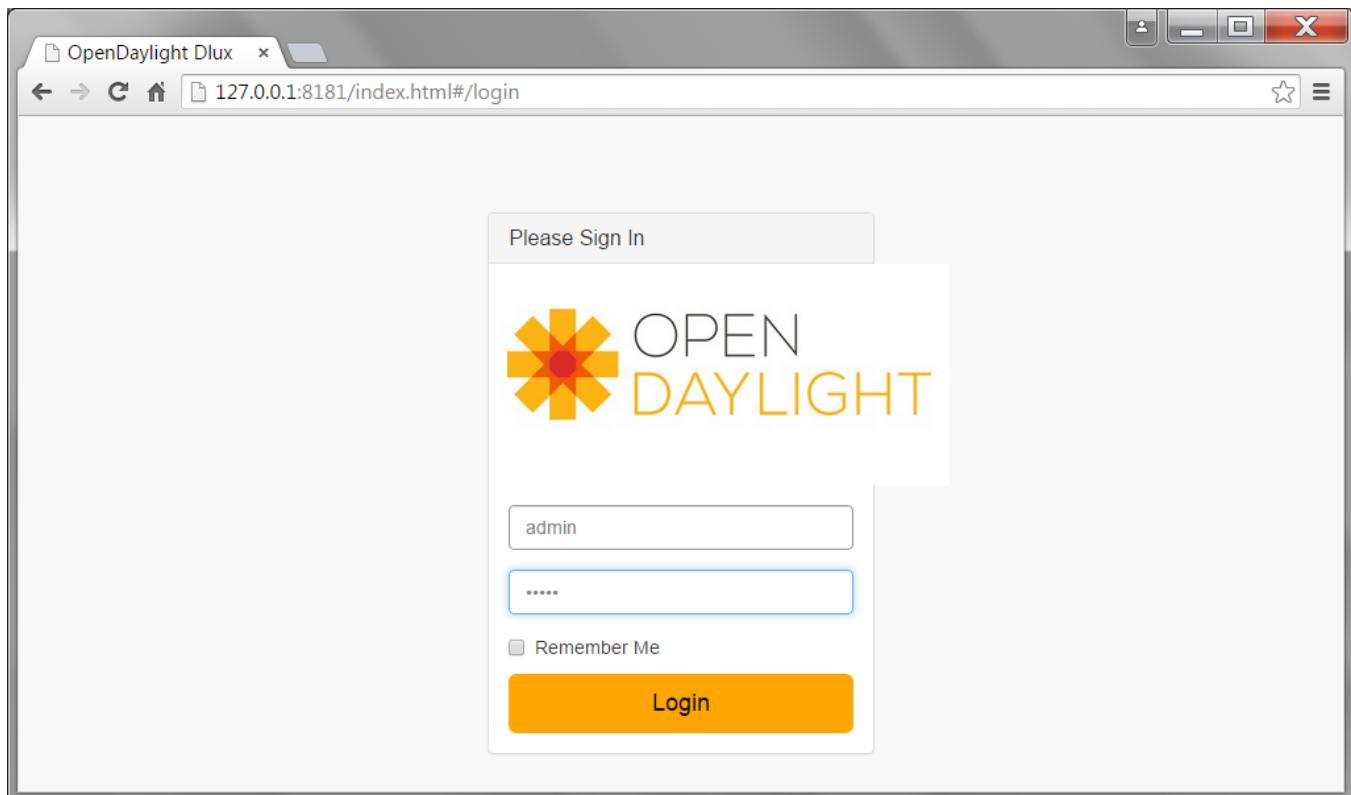
```
Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
```

OpenDaylight doesn't come with any features installed by default. You have to use the karaf console to install the required features. We are going to install a few additional packages including DLUX (OpenDaylight User Experience), the web interface of OpenDaylight.

```
opendaylight-user@root>feature:install odl-restconf-all odl-l2switch-switch
odl-mdsal-apidocs odl-dlux-all
```

That's all you need to do to get OpenDaylight up and running. Open a browser and go to <http://127.0.0.1:8181/index.html> to reach DLUX - the web interface of OpenDaylight. Replace *127.0.0.1* with the IP address of the machine which you installed OpenDaylight on.



Login to DLUX using *admin* as both username and password.

i You may need to wait for a few minutes before you can successfully login to the web interface of OpenDaylight.

Installing OpenDaylight on Windows

The OpenDaylight controller is a Java application that runs in a JVM (Java Virtual Machine). You can run OpenDaylight on any operating system with a JVM installed. Linux is the recommended platform for running OpenDaylight. However, you can install and run OpenDaylight on Microsoft Windows without issues.

You have to download OpenDaylight from <http://www.opendaylight.org/downloads>. At the time of this writing, Lithium-SR1 is current release of OpenDaylight that we used to develop this document.

OpenDaylight can be downloaded as pre-built zip and tar files. You should download the zip file **distribution-karaf-0.3.1-Lithium-SR1.zip** and save it to **C:** on your computer. Next, unzip the zip file to **C:\distribution-karaf-0.3.1-Lithium-SR1** using WinRAR or another utility.

! You must have JDK (Java Development Kit) installed before running OpenDaylight. You also need to set the JAVA_HOME system environment variable as described here.

Go to **C:\distribution-karaf-0.3.1-Lithium-SR1\bin** and run the **karaf.bat** batch file from the CLI (command-line interface).

```
C:\distribution-karaf-0.3.1-Lithium-SR1\bin>karaf.bat
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=512m; support was
removed in 8.0
```



```
Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.
opendaylight-user@root>
```

Use **karaf** to install the following features:

```
opendaylight-user@root>feature:install odl-restconf
opendaylight-user@root>feature:install odl-l2switch-switch
opendaylight-user@root>feature:install odl-mdsal-apidocs
opendaylight-user@root>feature:install odl-dlux-core
opendaylight-user@root>feature:install odl-adsal-all
opendaylight-user@root>feature:install odl-adsal-northbound
```

Configuring PicOS Switch for OpenDaylight

We are using the Pica8 P-3297 switch running PicOS 2.6.

```
admin@Spine1$version
Copyright (C) 2009-2014 Pica8, Inc.
=====
Hardware Model          : P-3297
Linux System Version/Revision : 2.6/21490
Linux System Released Date   : 05/24/2015
L2/L3 Version/Revision      : 2.6/21490
L2/L3 Released Date        : 05/24/2015
OVS/OF Version/Revision     : 2.6/21490
OVS/OF Released Date       : 05/24/2015
```

Verify that the switch is running in PicOS OVS mode.

```
admin@Spine1$sudo service picos status
xorp_rtrmgr is not running ... failed!
ovsdb-server is running.
ovs-vswitchd is running.
```

You can also change the PicOS mode using the procedure described here.

Display an overview of the database contents, using the **ovs-vsctl show** command.

```
admin@Spine1$ovs-vsctl show
1bd78e1e-6b6e-472b-b137-357c9190210a
```

Create a bridge named **br0** using the **ovs-vsctl add-bridge bridge-name** command.