

dpctl

- dpctl: command-line utility that sends basic OpenFlow messages, useful for viewing switch port and flow stats, plus manually inserting flow entries.
- little tool for first steps and debugging
- talks directly to the switch (This does not need any controller or respect any Flowvisor restrictions!)
- so no infrastructure needed
- part of the reference controller
- switch must support it, listening port either configurable or 6633.

dpctl Example Usage

- Create a SSH window if you don't already have one.
- Connect all team members via screen.
- Create wired connections from your host to your VLAN.
- Run:
 - `$ dpctl show tcp:10.101.1.1?:6633`
 - The 'show' command connects to the switch and dumps out its port state and capabilities.

dpctl Example Usage

- `dpctl show tcp:10.101.1.1?:6633`
 - `features_reply (xid=0x21355842): ver:0x1, dpid:a2c27d7772d80`
 - `n_tables:2, n_buffers:256`
 - `features: capabilities:0x87, actions:0x7ff`
 - `23(23): addr:2c:27:d7:77:2d:a9, config: 0, state:0`
 - `current: 1GB-FD AUTO_NEG`
 - `supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD AUTO_NEG`
 - `24(24): addr:2c:27:d7:77:2d:a8, config: 0, state:0`
 - `current: 1GB-FD AUTO_NEG`
 - `supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD AUTO_NEG`
 - `LOCAL(local): addr:2c:27:d7:77:2d:80, config: 0, state:0`
 - `get_config_reply (xid=0xec6a5f73): miss_send_len=0`

dpctl Example Usage

- Dpctl status shows us information about the switch
 - Mostly controller related
 - Whether on not connected
 - Connected since
 - Address of controller
 - Messages sent and received

dpctl Example Usage

- \$ dpctl status tcp:127.0.0.1:6634 (not on HP)
 - config.datapath-id=000000000000000001
 - switch.now=16068
 - switch.uptime=6825
 - switch.pid=6811
 - remote.name=tcp:127.0.0.1:6633
 - remote.state=BACKOFF
 - remote.backoff=8
 - remote.probe-interval=5

dpctl Example Usage

- `$ dpctl status tcp:127.0.0.1:6634`
 - `remote.is-connected=false`
 - `remote.sent-msgs=0`
 - `remote.received-msgs=0`
 - `remote.attempted-connections=856`
 - `remote.successful-connections=0`
 - `remote.last-connection=6825`
 - `remote.last-received=6825`
 - `remote.time-connected=0`
 - `remote.state-elapsed=2`

dpctl Example Usage

- Let us add some first flow entries so we can ping from h1 to h2
 - Test to ping Host 1 from Host 2 (should fail as we do not have any flow entries yet)
 - Add the flow entries (change port numbers):
 - `$ dpctl add-flow tcp:10.101.1.1?:6633 in_port=1,actions=output:2`
 - `$ dpctl add-flow tcp:10.101.1.1?:6633 in_port=2,actions=output:1`
- Ping should work now!

dpctl Example Usage

- \$ dpctl show-protostat tcp:10.101.1.1?:6633
(not on HP)
 - error_msg (xid=0xe02519b2): type1(OFPET_BAD_REQUEST)
code3(OFPBRC_BAD_VENDOR) payload:
 - vendor (xid=0xe02519b2):
 - dpctl: bad reply

Dpctl protostat example output

OpenFlow protocol version 0x97 statistical information

Protocol message:

Rcvd: 12642 total msgs, 0 unknown msgs

1 hello, 0 errors, 0 echo, 0 echo reply, 0 vendor

1 feats, 0 feats reply

0 get config, 0 get config reply, 1 set config

0 packet in, 0 flow expried, 0 port status

10799 packet out, 1829 flow mod, 0 port mod

11 stats, 0 stats reply

Sent: 13151 total msgs, 0 unknown msgs

1 hello, 3 errors, 0 echo, 0 echo reply, 0 vendor

0 feats, 1 feats reply

0 get config, 0 get config reply, 0 set config

11461 packet in, 1674 flow expried, 3 port status

0 packet out, 0 flow mod, 0 port mod

0 stats, 11 stats reply

Flow manipulation:

Rcvd: 1764 add, 0 modify, 0 modify strict

65 delete, 0 delete strict, 0 unknown ops

Sent: 0 add, 0 modify, 0 modify strict

0 delete, 0 delete strict, 0 unknown ops

Error notification:

Rcvd: 0 hello fail: 0 incompat

0 bad request: 0 version, 0 type, 0 stat

0 vendor

0 bad action: 0 type, 0 len, 0 vendor, 0 vendor type

0 out port, 0 argument

0 flow mod fail: 0 all tables full

0 unknown type, 0 unknown code

Sent: 0 hello fail: 0 incompat

0 bad request: 0 version, 0 type, 0 stat

0 vendor

3 bad action: 0 type, 0 len, 0 vendor, 0 vendor type

3 out port, 0 argument

0 flow mod fail: 0 all tables full

0 unknown type, 0 unknown code



dpctl Example Usage

- `$ dpctl dump-ports tcp:10.101.1.1?:6633`
 - Gives physical port information
 - Rx, Tx counters
 - Error counters

dpctl Example Usage

- `$ dpctl dump-ports tcp:10.101.1.1?:6633`
 - `stats_reply (xid=0xb2eeb981): flags=none type=4(port)`
 - 3 ports
 - port 23: rx pkts=2756, bytes=527428, drop=0, errs=0, frame=?, over=?, crc=?
 - tx pkts=2721, bytes=523911, drop=0, errs=0, coll=?
 - port 24: rx pkts=2733, bytes=525187, drop=0, errs=0, frame=?, over=?, crc=?
 - tx pkts=2727, bytes=525296, drop=0, errs=0, coll=?
 - port 65534: rx pkts=?, bytes=?, drop=?, errs=?, frame=?, over=?, crc=?
 - tx pkts=?, bytes=?, drop=?, errs=?, coll=?

dpctl Example Usage

- `dpctl mod-ports tcp:10.101.1.1?:6633`
- Allows manipulation of the switch ports
 - Possible commands are:
 - Up
 - Down
 - Flood
 - Noflood

dpctl Example Usage

- `$ dpctl mod-ports tcp:10.101.1.1?:6633 2 down`
 - Ping should fail now
- `$ dpctl mod-ports tcp:10.101.1.1?:6633 2 up`
 - Ping works again
- Possible commands are up/down/flood/noflood

dpctl Example Usage

- `$ dpctl dump-flows tcp:10.101.1.1?:6633`
 - Gives us information about the flows installed
 - Rule itself
 - Timeouts
 - Actions
 - Packets and bytes processed by flow

dpctl Example Usage

- `$ dpctl dump-flows tcp:10.101.1.1?:6633`
 - `stats_reply (xid=0x63b24b68): flags=none type=1(flow)`
 - `cookie=0, duration_sec=460s, duration_nsec=129000000s, table_id=1, priority=32768, n_packets=456, n_bytes=43680, idle_timeout=60,hard_timeout=0,in_port=3,actions=output:2`
 - `cookie=0, duration_sec=464s, duration_nsec=401000000s, table_id=1, priority=32768, n_packets=456, n_bytes=43680, idle_timeout=60,hard_timeout=0,in_port=2,actions=output:3`

dpctl Example Usage

- Lets change the flow to work on IP's
- `dpctl add-flow tcp:10.101.1.1?:6633 ip,nw_dst=10.0.0.2,priority=33000,actions=output:2`
- `dpctl add-flow tcp:10.101.1.1?:6633 ip,nw_dst=10.0.0.3,priority=33000,actions=output:3`
- **Priority changed!**

dpctl Example Usage

Flow fields and syntax:

- in_port=port_no
- dl_vlan=vlan
- dl_src=mac
- dl_dst=mac
- dl_type=ethertype
- nw_src=ip[/netmask]
- nw_dst=ip[/netmask]
- nw_proto=proto

dpctl Example Usage

Flow fields and syntax:

nw_tos=tos/dscp

tp_dst=port

icmp_type=type

icmp_code=code

The following shorthand notations are also available:

- ip Same as dl_type=0x0800
- icmp Same as dl_type=0x0800,nw_proto=1
- tcp Same as dl_type=0x0800,nw_proto=6
- udp Same as dl_type=0x0800,nw_proto=17
- arp Same as dl_type=0x0806

dpctl Example Usage

- What could be done with static flow rules?
 - Using `nw_src` – Load balancing flows from different sources to web servers
 - Using `dl_src` – sort wireless from wired clients by looking for mac-address prefixes coming from wireless vendors
 - Using `In_port + Out_port` – create basic virtual circuits between end points

dpctl Example Usage

- `dpctl benchmark tcp:10.101.1.1?:6633 100 100`
 - Sending 100 packets * 108 bytes (with header) = 10800 bytes total
 - Finished in 9.7 ms (10349 packets/s) (1117665 bytes/s)
- `dpctl benchmark tcp:10.101.1.1?:6633 1000 1000`
 - Sending 1000 packets * 1008 bytes (with header) = 1008000 bytes total
 - Finished in 94.4 ms (10594 packets/s) (10678984 bytes/s)

Dpctl command list

For local datapaths and remote switches:

- show SWITCH show basic information
- status SWITCH [KEY] report statistics (about KEY)
- show-protostat SWITCH report protocol statistics
- dump-desc SWITCH print switch description
- dump-tables SWITCH print table stats
- mod-port SWITCH IFACE ACT modify port behavior
- dump-ports SWITCH [PORT] print port statistics
- desc SWITCH STRING set switch description
- dump-flows SWITCH print all flow entries
- dump-flows SWITCH FLOW print matching FLOWS
- dump-aggregate SWITCH print aggregate flow statistics
- dump-aggregate SWITCH FLOW print aggregate stats for FLOWS
- add-flow SWITCH FLOW add flow described by FLOW
- add-flows SWITCH FILE add flows from FILE
- mod-flows SWITCH FLOW modify actions of matching FLOWS
- del-flows SWITCH [FLOW] delete matching FLOWS
- monitor SWITCH print packets received from SWITCH
- execute SWITCH CMD [ARG...] execute CMD with ARGS on SWITCH

