1 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, you will have the tools you need to configure Pica8's open switches as an OpenFlow switch. You will also gain insights on how to optimize the configuration to work in your 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 highlighting the problems you 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 Pica 8 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 Pica 8 OVS OpenFlow switches to interface with the RYU OpenFlow Controller

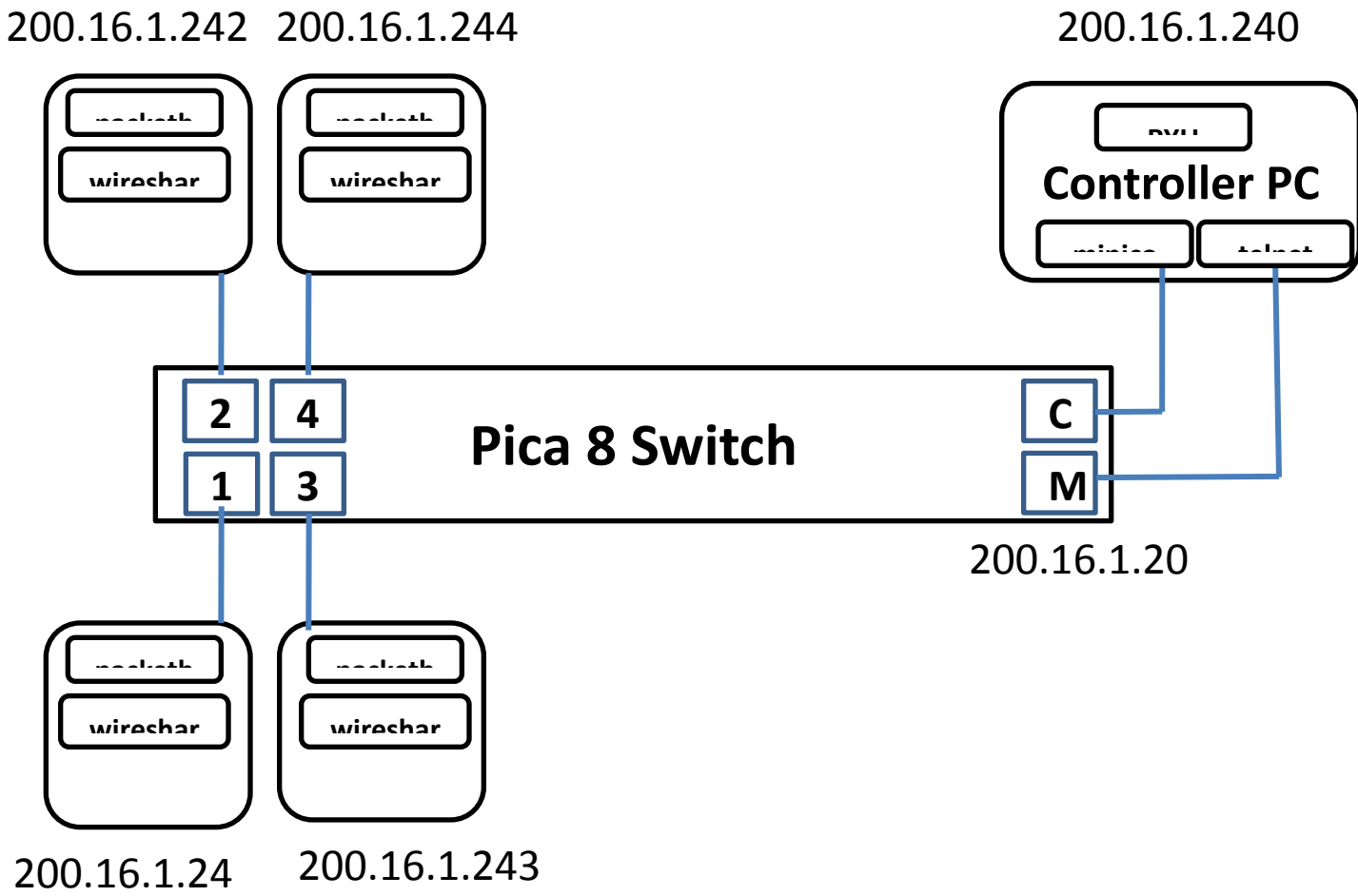

Figure 1 – Test bed configuration

In this document, the system configuration depicted in Figure 1 includes:

- A Pica8 P-3295 open switch with 48 x 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 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 ftpd
- telnet and telnetd

2 Power on Configuration

To start, configure your terminal emulator to the following configuration:

- 115200 8N1
- No hardware flow control
- No software flow control

To start the switch, a console cable is required to connect the switch console port to the serial port on the controller PC. Run the terminal emulator on the console port from the controller PC, then power on the switch.

Figure 2 shows the console output; do not hit any keys until you see the “XorPlus login: “ string. Log in as “root” with password “pica8” to boot into Linux prompt.

```
[ ok ] Cleaning up temporary files....
INIT: Entering runlevel: 2
ip_tables: (C) 2000-2006 Netfilter Core Team

CPU MMAP Driver (c) 2007 Quanta Computer Inc.
Registered with major 243

Flash Driver v0.10 (c) 2007 Quanta Computer Inc.
Registered with major 240

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

WDT Driver v0.1 (c) 2008 Quanta Computer Inc.
Registered with major 245
linux_kernel_bde: module license 'Proprietary' taints kernel.
checking partition 2 ...
partition 2 file system is ok
net.netfilter.nf_conntrack_acct = 1
net.ipv6.conf.all.forwarding = 1
[ ok ] Starting enhanced syslogd: rsyslogd.
[ ok ] Starting periodic command scheduler: cron.
[ ok ] Starting internet superserver: xinetd.
Pica8 Auto Provisioning Tool - checking updates ....
No tftp server address found, exit now
[....] Starting: PicOS L2/L3..PowerPC Book-E Watchdog Timer Enabled (wdt_period=29)
.........................PHY: 24520:01 - Link is Up - 1000/Full
[ ok .....
[ ok ] Starting OpenBSD Secure Shell server: sshd.

XorPlus login:
```

Figure 2 – Power on console output

3 Configure Switch

Launch ***picos_boot*** shell script to set the system initialization default to OVS mode and the switch will display the menu in Figure 3.

```
XorPlus login: root
Password:

root@XorPlus$picos_boot
Please configure the default system start-up options:
 (Press other key if no change)
    [1]  PicOS L2/L3    * default
    [2]  PicOS Open vSwitch/OpenFlow
    [3]  No start-up options
Enter your choice (1,2,3):2

PicOS Open vSwitch/OpenFlow is selected.
```

Figure 3 – System Initialization Menu

Enter your choice "***2***" for OVS mode and the script will keep asking for the switch static IP address and the gateway IP address as shown in Figure 4.

```
  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.

Please set a static IP and netmask for the switch (e.g. 128.0.0.10/24) : 200.16.1.20/24

Please set the gateway IP (e.g 172.168.1.2):200.16.1.1
Please restart the PicOS service
```

Figure 4 – IP address and gateway IP address

Next, stop the L2/L3 processes and start OVS processes by issuing ***service picos restart*** as shown in Figure 5.

```
root@XorPlus$service picos restart
[....] Stopping: PicOS L2/L3.....................................
[....] Stopping enhanced syslogd: rsyslogd.
[....] Starting enhanced syslogd: rsyslogd.
[....] Stopping internet superserver: xinetd.
[....] Starting internet superserver: xinetd.
[....] Restarting OpenBSD Secure Shell server: sshd.
[....] Create OVS database file.
[....] Starting: PicOS Open vSwitch/OpenFlow.
[....] Starting web server: lighttpd.
root@XorPlus$
```

Figure 5 –Start picos service

Next, 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.

```
root@XorPlus$ps -A
  PID TTY          TIME CMD
    1 ?        00:00:00 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 ksoftirqd/0
    4 ?        00:00:00 watchdog/0
    5 ?        00:00:00 events/0
    6 ?        00:00:00 khelper
   48 ?        00:00:00 kblockd/0
   55 ?        00:00:00 ata/0
   56 ?        00:00:00 ata_aux
   58 ?        00:00:00 kseriod
   99 ?        00:00:00 pdflush
  100 ?        00:00:00 pdflush
  101 ?        00:00:00 kswapd0
  147 ?        00:00:00 aio/0
  156 ?        00:00:00 nfsiod
  831 ?        00:00:00 ftld
  853 ?        00:00:00 rpciod/0
  857 ?        00:00:00 kjournald
 2343 ?        00:00:00 cron
 3542 tty1     00:00:00 getty
 3543 tty2     00:00:00 getty
 3550 tty3     00:00:00 getty
 3552 ttyS0    00:00:00 login
 4360 ttyS0    00:00:00 bash
 5430 ?        00:00:00 rsyslogd
 5487 ?        00:00:00 xinetd
 5514 ?        00:00:00 sshd
 5559 ttyS0    00:00:00 ovsdb-server
 5561 ttyS0    00:02:30 ovs-vswitchd
 5599 ?        00:00:00 lighttpd
 5600 ?        00:00:00 python
 5647 ttyS0    00:00:00 ps
root@XorPlus$
```

Figure 6 – OVS Processes

4 Configure Bridge

In PicOS 2.1 and later versions, there are no needs to provide DB IP address and port number in *ovs-vsctl* command.

In this section, we will show to lissue ***ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8*** command to create a new bridge. In our configuration, the bridge needs four 1GbE ports for our exercise. To add each

1GbE port to the bridge, we will 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 as shown below.

```
root@XorPlus$ovs-vsctl add-br br0 -- set bridge br0 datapath_type=pica8
device ovs-pica8 entered promiscuous mode
device br0 entered promiscuous mode
root@XorPlus$ovs-vsctl  add-port br0 ge-1/1/1 -- set interface ge-1/1/1 type=pica8
root@XorPlus$ovs-vsctl  add-port br0 ge-1/1/2 -- set interface ge-1/1/2 type=pica8
root@XorPlus$ovs-vsctl  add-port br0 ge-1/1/3 -- set interface ge-1/1/3 type=pica8
root@XorPlus$ovs-vsctl  add-port br0 ge-1/1/4 -- set interface ge-1/1/4 type=pica8
```

Figure 7 – Add bridge and ports

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.

```
root@XorPlus$ovs-vsctl show
abf7100c-2526-4e81-baec-f22bc3fe4344
    Bridge "br0"
        Port "ge-1/1/1"
            Interface "ge-1/1/1"
                type: "pica8"
        Port "ge-1/1/3"
            Interface "ge-1/1/3"
                type: "pica8"
        Port "br0"
            Interface "br0"
                type: internal
        Port "ge-1/1/4"
            Interface "ge-1/1/4"
                type: "pica8"
        Port "ge-1/1/2"
            Interface "ge-1/1/2"
                type: "pica8"
```

Figure 8 – Show bridge and ports in database

Next, let us monitor the port status and examine the port configuration with ***ovs-ofctl show br0*** command.

```
root@XorPlus$ovs-ofctl show br0
OFPT_FEATURES_REPLY (OF1.3) (xid=0x2): dpid:5e3ee89a8ffbc405
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS
OFPST_PORT_DESC reply (OF1.3) (xid=0x4):
 1(ge-1/1/1): addr:e8:9a:8f:fb:c4:05
     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
 2(ge-1/1/2): addr:e8:9a:8f:fb:c4:05
     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:       100MB-HD 100MB-FD 1GB-FD COPPER
     speed: 1000 Mbps now, 1000 Mbps max
 3(ge-1/1/3): addr:e8:9a:8f:fb:c4:05
     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
 4(ge-1/1/4): addr:e8:9a:8f:fb:c4:05
     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:       100MB-HD 100MB-FD 1GB-FD COPPER
     speed: 1000 Mbps now, 1000 Mbps max
 LOCAL(br0): addr:e8:9a:8f:fb:c4:05
     config:     0
     state:      LINK_UP
     current:    10MB-FD COPPER
     supported:  10MB-FD COPPER
     speed: 10 Mbps now, 10 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.3) (xid=0x6): frags=normal miss_send_len=0
```

Figure 8 – Port status

In the example provided, port state is *LINK*_DOWN 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, let us 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 zeroes.

```
root@XorPlus$ovs-ofctl dump-ports br0
OFPST_PORT reply (OF1.3) (xid=0x2): 5 ports
  port  1: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=61.928s
  port  2: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=44.213s
  port  3: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=30.939s
  port  4: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=19.967s
  port LOCAL: rx pkts=8, bytes=816, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=84.558s
root@XorPlus$
```

Figure 9 – Port statistics

5 Configure port

A port can be added, deleted, turned up, or turned down dynamically. We have tested the *add-port* command, to delete a port, use ***ovs-vsctl del-port br0 ge-1/1/1***. Port state can also be modified with the *mod−port* 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 no−stp
  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 no−receive / receive−stp or no−receive−stp
  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 no−receive setting applies to all packets except 802.1D spanning tree packets, which are separately controlled by receive−stp or no−receive−stp.
- forward or no−forward
  Allow or disallow forwarding of traffic to this interface. By default, forwarding is enabled.
- flood or no−flood

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.

- packet−in or no−packet−in
  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 *mod−port* changes.

6 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 the 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.

```
root@XorPlus$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=855.426s, table=0, n_packets=8, n_bytes=816, priority=0 actions=NORMAL
```

Figure 10 – Default Flow

Now, let us 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.

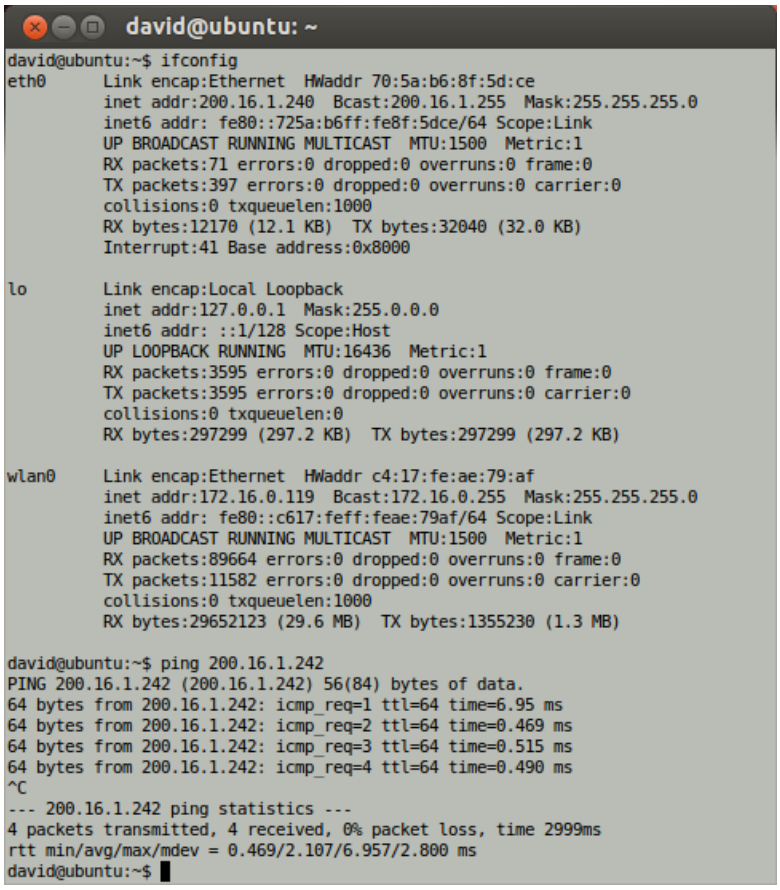

Figure 11 – Ping test

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; reader can try the telnet and ftp sessions to test the connectivity and bridge functionalities.

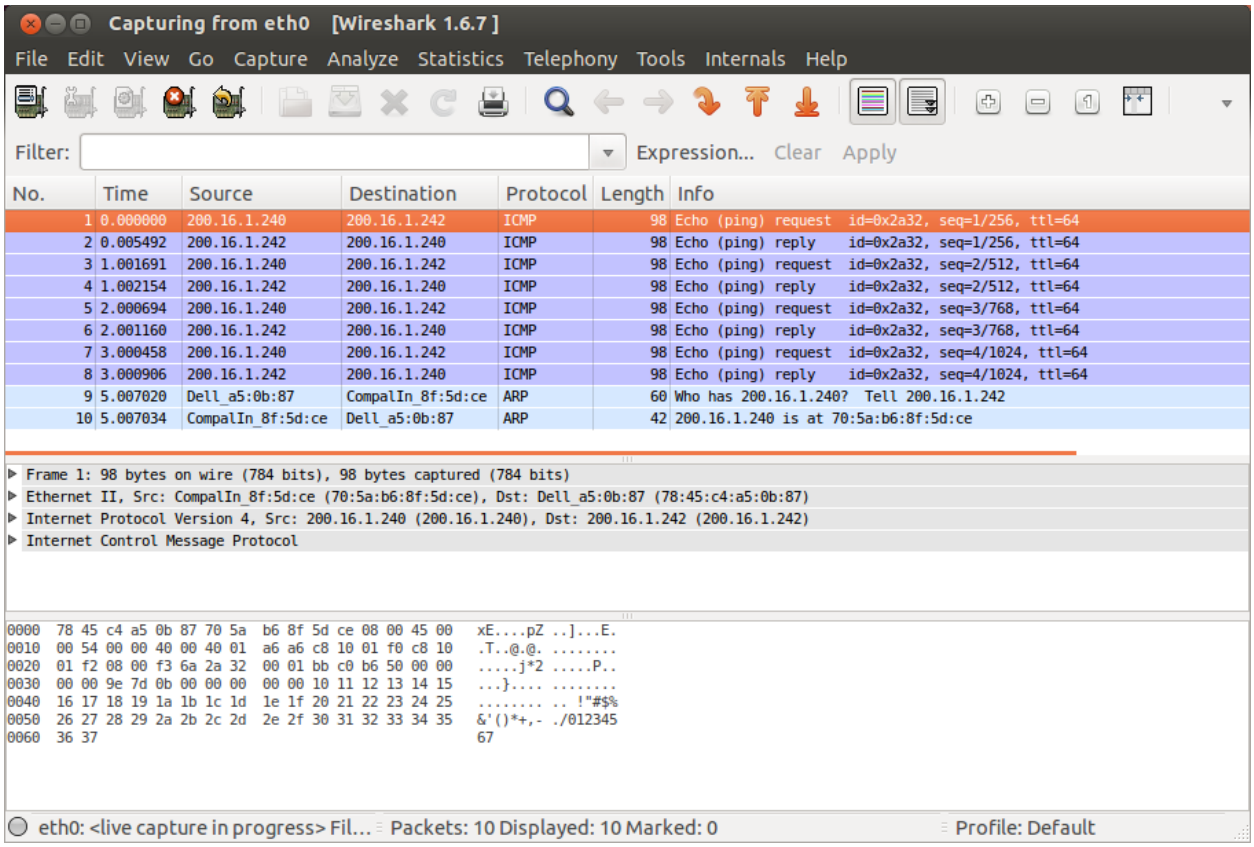

Figure 12 – ICMP request/reply

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.

7 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=pronto
ovs-vsctl add-port br0 ge-1/1/2 -- set interface ge-1/1/2 type=pronto
ovs-vsctl add-port br0 ge-1/1/3 -- set interface ge-1/1/3 type=pronto
ovs-vsctl add-port br0 ge-1/1/4 -- set interface ge-1/1/4 type=pronto
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