Label Switch Router (LSR) for NetFPGA

User Manual, Version 1.4
June 31, 2011

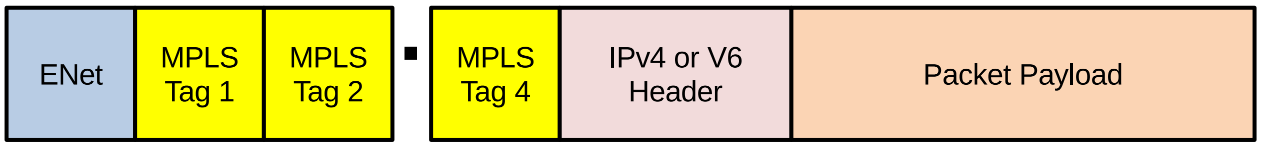
Developed by Algo-Logic Systems for Google

http://Algo-Logic.com

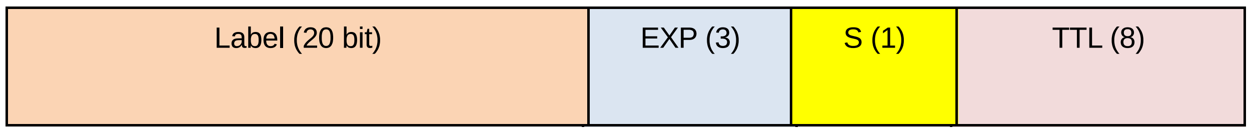
Multiprotocol Label Switching (MPLS) is a mechanism that allows networks to carry data from one network node to the next with the help of short labels. MPLS enables creation of virtual links between distant nodes and can readily forward packets between data centers by encapsulating packets of multiple network protocols.

The MPLS Label Switch Router is a hardware-accelerated LSR implemented built by Algo-Logic Systems for Google that adds MPLS label switching functionality over and beyond the default NetFPGA reference switch.

The Structure of an MPLS packet with four MPLS tags is shown below.



MPLS tags can be seen immediately after ether type in Ethernet Header. Each MPLS Tag consists of 32 bits in the following format.

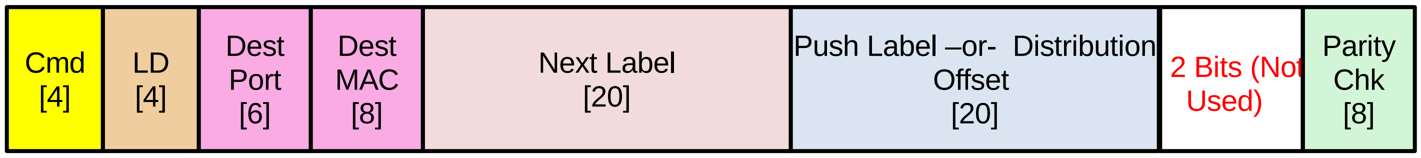


“EXP” is the experimental bits and we are not using it.

“S” indicates the end of Tag. If S is “1” it indicates that corresponding Tag is the last one in the MPLS Tag stack. 8 bits in the LSB side is allotted for TTL. 20 bits in the MSB side is the MPLS label and is used for Label switching operations. The MPLS operations that we have implemented are

- **Swap**
- **Push**
- **Swap+Push**
- **Pop**
- **Pop+Swap**

The operations to be performed in the input MPLS packets are decided by the 72 bit MPLS words which are stored in the SRAM of 1G NetFPGA card. The address of the 72 bit MPLS word is extracted from the 20 bit label in the first MPLS Tag. So whenever a packet enters in the NetFPGA hardware pipeline we extract the 20 bit label from the first MPLS Tag and do a lookup operation for obtaining the corresponding 72 bit MPLS word from SRAM. The format of the 72 MPLS word stored in SRAM is shown below.



Each field of the 72 bit MPLS word is given below

Cmd – MPLS Commands: 4 bits, which are represented as

No Op=0, Swap=1, Push=2, Swap+Push=3, Pop=4 and Pop+Swap=5

LD - Load Distribution Count: 4 bits
The number of load distribution entries starting from the Load distribution offset. Default value is 0

Dest Port
Address of the output Physical port (nf2c0, nf2c1 …)

Dest MAC
Index to 256-entry table of MAC addresses stored in on-chip BRAM

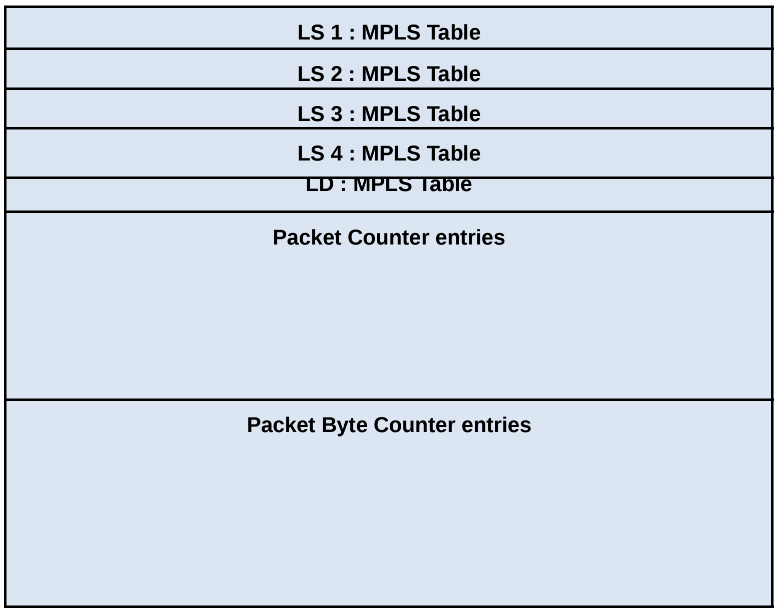
Next Label
20 bit MPLS label used to swap with the 20 bit label of the first MPLS Tag in the incoming packet

Push Label –or- Distribution offset
For commands which do push uses these 20 bits as the pushing MPLS label.
If LD > 0 these 20 bits will be the starting address of the load distribution entries.

Parity/Checksum
N-bit check of bits

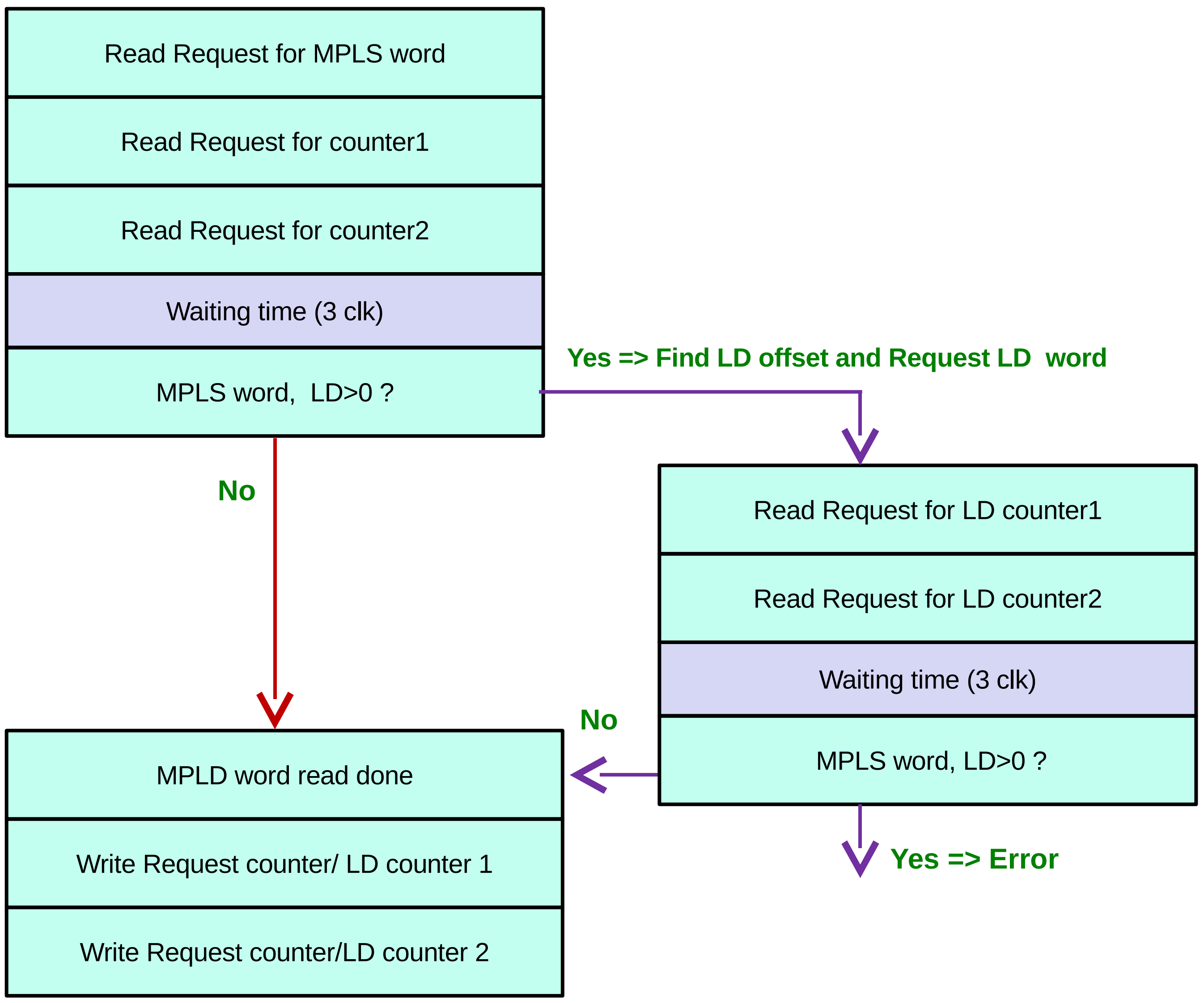
Label spaces and partitions in SRAM:

SRAM in a 1G NetFPGA board can store 256k words with a width of 72bits. The total label space is divided in to four (LS1, LS2, LS3 and LS4) and each space is assigned to each input port. There is a separate space for the load distribution entries which is named as LD space. For each entry in the LS space and LD space there will two counters. First counter gives the total number of packets which uses that particular entry and the second counter indicates the total byte count of those packets. Each of these counters take 1/3 of the total SRAM space. The whole SRAM space allotted for these spaces are shown below.



SRAM Lookup:

If the Ethernet parser find a MPLS Unicast packet the first 20 bits from the first MPLS Tag is extracted and added with a constant value (software offset) to find out the lookup address. After the SRAM lookup the Ethernet parser module will check the 4 bit LD value in the 72 bit MPLS word. If the LD value is zero the received MPLS word is latched and the corresponding Packet counter and Packet byte counter is incremented. If LD value is not equal to zero then the 20 bit LD offset value is latched. Consider "n" is the LD value. This indicates that there will


Read Request for MPLS word
Read Request for counter1
Read Request for counter2
Waiting time (3 clk)
MPLS word, LD>0 ?
Yes => Find LD offset and Request LD word
No
Read Request for LD counter1
Read Request for LD counter2
Waiting time (3 clk)
MPLS word, LD>0 ?
No
Yes => Error
MPLD word read done
Write Request counter/ LD counter 1
Write Request counter/LD counter 2

be n valid LD entries starting from the address 'LD offset' and from these n entries   one is to be selected for the later MPLS operations. A hash value is generated by making use of the following parameters.

- Source IP address (32 bit) of the packet.
- Destination IP address (32 bit) of the packet.
- TCP source port (16 bit) if the packet is a TCP packet (or UDP source port if the packet is a UDP packet)
- TCP destination port (16 bit) if the packet is a TCP packet (or UDP destination port if the packet is a UDP packet)

  Let k is the hash value, then (k modulo n) will give a number p whose value will be in between 0 and n-1.

The address of the LD entry selected = (LD offset + p)

A second SRAM look up operation is performed for this selected address. After latching the MPLS word corresponding packet counter and packet byte counter are incremented.

BRAM Lookup:

Dest Mac field (8 bits) of latched MPLS word is used for BRAM lookup to get the destination Mac address of the outgoing packet. A maximum of 256 Mac addresses can be stored in the BRAM.

Packet flow through NetFPGA hardware pipeline:

Mac receiver unit receives packet and direct it to output look-up module through the input arbiter.

In Output look-up, module header of the packet is parsed by the help of header parser module.

It latches all the header information in a status FIFO. The packets are stored in a FIFO in the look-up module.

The Actions in the output look-up module are taking place in the following order.

If the packet is MPLS multicast it is forwarded to CPU.

If the Multicast bit is not set and the Destination Mac Address is not matching, then the packet is dropped and the corresponding counter (Not for us counter) is incremented.

If the packet is MPLS unicast and SRAM lookup address derived from 20 bit MPLS Tag is out of the expected label space, then the packet is dropped and the corresponding counter (LS error counter) is incremented.

If the packet is not MPLS unicast it is forwarded to CPU.

If the packet is MPLS unicast, it waits for the MPLS look-up action to complete.

After getting the MPLS word from the SRAM Corresponding packet counter and packer byte counter content in the SRAM is incremented.

Index for the Destination Mac address is latched from the 72 bit MPLS word and BRAM lookup is performed.

If there a parity error in the look-up MPLS word it is dropped and corresponding counter is incremented

If there is no error, then the MPLS unicast packet with the modified Destination mac and output port are passed to the MPLS processing unit along with the MPLS look-up word from SRAM.

MPLS processing unit check the TTL and if it is expired forward it to CPU and corresponding counter is incremented.

Else it Modify the Packet header according to the “cmd” bit in the MPLS word and it is passed to the output queue module.

Output queue module check the output port address and forward the packet to corresponding DRAM queue and then it is forwarded to corresponding Mac transmitting unit.

Register Interface:

MPLS word entry in SRAM and output Mac address entry in BRAM, are populated using register interface. Address of all registers, SRAM base and Block ram base are defined in ~/netfpga/projects/mpls_switch/lib/C/reg_defines_mpls_switch.h

NetFPGA register interface support the read and write of 32 bit wide word. In order to write a single SRAM entry (72 bits) we make use of SRAM address locations between “0Xxxxxxx0” and “0Xxxxxxxf”. 32 bits of SRAM entry in the LSB side is buffered if we use the address location 0XxxxxxxC, Next 32 bits (middle) is buffered for the address location 0Xxxxxxx8 and the next 8 bits (MSB side) are buffered for the address location 0Xxxxxxx4. These buffered 72 bits is written to the SRAM when we write a trigger value of “0xffffffff” to the address location “0Xxxxxxx0”. The following example illustrate the procedure to write a 72 bit SRAM entry “0X12345678ABCDFFFF98” to the SRAM locations from address 0X1000000 to 0X100000f.

```
writeReg(&nf2, 0x100000C, 0xCDFFFF98);
writeReg(&nf2, 0x1000008, 0x345678AB);
writeReg(&nf2, 0x1000004, 0x00000012);
writeReg(&nf2, 0x1000000, 0xffffffff);
```

Writing a Mac address in BRAM also follow the same procedure. The following example illustrate how we write a Mac address “12345678ABCD” to the BRAM region spanned between “0x2002030” and “0x200203f”.

```
  writeReg(&nf2, 0x100000C, 0x5678ABCD);
writeReg(&nf2, 0x1000008, 0x00001234);
writeReg(&nf2, 0x1000000, 0xffffffff);
```

Other important registers needed for proper MPLS operations are added below.

```
#define SWITCH_OP_LUT_MAC0_HI_REG         0x2000100
#define SWITCH_OP_LUT_MAC0_LO_REG         0x2000104
#define SWITCH_OP_LUT_MAC1_HI_REG         0x2000108
#define SWITCH_OP_LUT_MAC1_LO_REG         0x200010c
#define SWITCH_OP_LUT_MAC2_HI_REG         0x2000110
```

```
#define SWITCH_OP_LUT_MAC2_LO_REG        0x2000114
#define SWITCH_OP_LUT_MAC3_HI_REG        0x2000118
#define SWITCH_OP_LUT_MAC3_LO_REG        0x200011c
```

Above registers are used for storing Mac Address of the NetFPGA Mac ports. This Mac address is used for matching with the destination Mac address of the incoming packets.

```
#define SWITCH_OP_LUT_LS1_BASE_REG       0x2000120
#define SWITCH_OP_LUT_LS1_BOUND_REG      0x2000124
#define SWITCH_OP_LUT_LS2_BASE_REG       0x2000128
#define SWITCH_OP_LUT_LS2_BOUND_REG      0x200012c
#define SWITCH_OP_LUT_LS3_BASE_REG       0x2000130
#define SWITCH_OP_LUT_LS3_BOUND_REG      0x2000134
#define SWITCH_OP_LUT_LS4_BASE_REG       0x2000138
#define SWITCH_OP_LUT_LS4_BOUND_REG      0x200013c
#define SWITCH_OP_LUT_LD_BASE_REG        0x2000140
#define SWITCH_OP_LUT_LD_BOUND_REG       0x2000144
```

Above registers are used for storing Base addresses and Bound for four label spaces and one LD space.

```
#define SWITCH_OP_LUT_SOFT_OFFSET_REG    0x2000150
```

Used to load software offset. The content of this register is added with the 20 bit MPLS label of the first MPLS Tag in the incoming packet to find out the SRAM lookup address.

```
#define SWITCH_OP_LUT_NOT_FOR_US_REG     0x2000154
```

Content of this register gives the number packets dropped due to the mismatching destination Mac address

```
#define SWITCH_OP_LUT_PARITY_ERROR_REG   0x2000158
```

Content of this register gives the number of times parity error found while SRAM MPLS lookup.

```
#define SWITCH_OP_LUT_TTL_ERROR_REG      0x200015c
```

Content of this register gives TTL error count

```
#define SWITCH_OP_LUT_LS_ERROR_REG       0x2000160
```

This counter is incremented if the index for the SRAM lookup is not matching with the assigned label space for the incoming packet.

```
#define SWITCH_OP_LUT_LD_ERROR_REG       0x2000164
```

Gives the LD error count.

Software :

Software programs to help populating SRAM and BRAM and other register entries are provided in ~/netfpga/projects/mpls_switch/sw

swap is the command for entering a swap entry with ld value zero in sram and it should be followed
1: Destination port number
2: SRAM entry number
3: BRAM Mac entry number and
4: Next mpls label

push is the command for entering a push entry with ld value zero in sram and it should be followed
1: Destination port number
2: SRAM entry number
3: BRAM Mac entry number and
4: push label

pop is the command for entering a pop entry with ld value zero in sram and it should be followed
1: Destination port number
2: SRAM entry number
3: BRAM Mac entry number

spush is the command for entering a swap+push entry with ld value zero in sram and it should be followed by
1: Destination port number
2: SRAM entry number
3: BRAM Mac entry number
4: Next mpls label
5: push label

pswap is the command for entering a pop+swap entry with ld value zero in sram and it should be followed
1: SRAM entry number

ld is the command for entering an entry with a nonzero ld value in sram and it should be followed by
1: ld value
2: SRAM entry number
3: Distribution offset

mac_out is the command for entering a mac entry in bram and it should be followed by
1: mac address without colon or space in between
2: BRAM entry number

lsld_init is the command for entering label space, ld space and pkt counter base and bound and it should be followed by
1: Label space 1 base address
2: Label space 1 bound (number of entries)
3: Label space 2 base address
4: Label space 2 bound (number of entries)
5: Label space 3 base address
6: Label space 3 bound (number of entries)
7: Label space 4 base address
8: Label space 4 bound (number of entries)
9: Ld space base address
10: Ld space bound (number of entries)

11: packet counter space base address
12: Packet byte counter space base address

mac0_add, mac1_add, mac2_add, mac3_add are the commands for entering mac addresses of corresponding mac ports and these should be followed by
1: mac address without colon or space in between

lsr_init is the command for initializing label space registers, ld space registers, software offset register, Mac address registers and error count registers with default values.
Following are some underlying C functions which can be used to perform MPLS operations.

`lsr_fn` is a function which can be used to enter a 72 bit SRAM entry.

```
int lsr_fn(int dest_port,int ld_entry, int mac_bram_entry, int
next_label, int next_label2,int op, int entry_num)
```

- **`dest_port`** – Destination port
- **`ld_entry`** - Load distribution entry
- **`int mac_bram_entry`** – Index of BRAM to read the output Mac Address
- **`next_label`** – Label used for the swap operation (input any value if it is not used)
- **`next_label2`** - load distribution offset or push value (input any value if it is not used)
- **`op`** – MPLS cmd (value between 0 to 5)
- **`entry_num`** – Index of SRAM to write the MPLS word

`mac_out` is the function to write a Mac address entry to the BRAM lookup Table.

```
int mac_out(int mac_oh, int mac_ol, int entry_num)
```

- **`mac_oh`** - value for 16 bits in the MSB side of the Mac address
- **`mac_ol`** - value for 32 bits in the LSB side of the Mac address
- **`entry_num -`** – Index of BRAM to write the Mac address

`lsr1_init` is the function to write Label space 1 starting index and bound (number of entries in SRAM) to corresponding registers.

```
int lsr1_init(int base_address, int bound)
```

- `base_address` – Starting Address of the Label space 1 in SRAM
- `bound` - Number of entries for the Label space 1 in SRAM

Similarly `lsr1_init`, `lsr1_init`, `lsr1_init` can be used to enter base and bound values for other label spaces.

`lsr_ld_init` is the function to write LD space starting index and bound to corresponding registers.

```
int lsr_ld_init(int base_addr, int bound)
```

- `base_addr` – Starting Address of the LD space in SRAM
- `bound` - Number of entries for the LD space in SRAM

`cnt_base_init` is the function to write packet counter and packet byte counter starting indexes to corresponding registers.

```
int cnt_base_init(int packet_counter_base_addr, int
byte_counter_base_addr)
```

- `packet_counter_base_addr` – Starting Address of the packet counter in SRAM
- `byte_counter_base_addr` - Starting Address of the packet byte counter in SRAM

`soft_offset_init` is the function to write software offset value to corresponding register.

```
int soft_offset_init(int soft_offset)
```

- `soft_offset` – software offset value .

`mac0_init` is the function to enter the Mac address of the Port 0 of the NetFPGA board

```
int mac0_init(int mac0_hi, int mac0_lo)
```

- `mac0_hi` – value for 16 bits in the MSB side of the Mac address
- `mac0_lo` - value for 32 bits in the LSB side of the Mac address

similarly `mac1_init, mac2_init, mac3_init` can be used to enter Mac addresses for other three ports.

`lsr_init` is the function which set up default values in different registers

```
int lsr_init()
```

This function initializes the following parameters

- Label space and LD space base and bound values. Equally distributed entries for all the four label spaces (corresponds to each port) and LD space. default is to assign each port N=8888 h=34,952d entries of the total of 512k entries 3 table regions
- packet counter and Packet byte counter base address
- offset value added to input label value (-1,000,000 in 2's complement; as per Juniper default)
- MAC address of nf2c0: MSB,LSB = 00:90:69:B1:D0:7E
- Reset count of packets arriving with MAC != self
- Reset counter corresponds to Parity errors in SRAM lookups
- Reset counter corresponds to TTL expirations
- Reset counter corresponds to Label Space out-of-bound error
- Reset counter corresponds to Load Distribution error

User can also read and write any registers using regread and regwrite commands which can be find in ~/netfpga/lib/C/reg_access. For example user can use regread 0x0600028 to read the number of packets transmitted through mac-0 port.

Implementation and testing:

(1) Starting at the top-level of a NetFPGA base distribution

(typically: /home/user/netfpga), extract tarball file to populate lib and project/mpls-switch sub-directories.

eg:

```
cd ~/netfpga
tar xvf lsr-netFPGA.tar
```

(2) Download the LSR bitfile to the netfpga

```
nf_download ./bitfiles/mpls_switch.bit
```

(3) Compile the mpls control and configuration software

```
cd ./projects/mpls_switch/sw
./make
```

(4) Run the command-line program to initialize LSR switch

```
./lsr lsr_init
```

Will initialize tables with default values.

(5) For testing,

* connect eth1 (NIC) to nf2c0 (Port 0 of NetFPGA) with short cat5e cable,

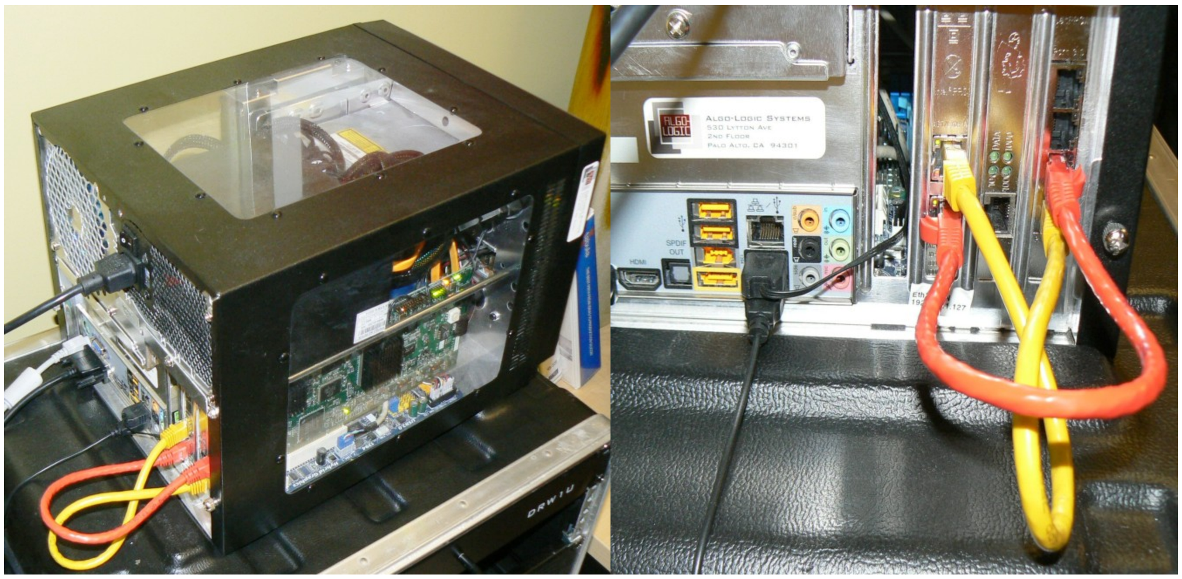
* connect eth2 (NIC) to nf2c1 (Port 1 of NetFPGA) with short cat5e cable,

* run wireshark on eth2 to monitor output packets
* and run command like:

```
cd /../pcap_files

tcpreplay intf1=eth1 preswap.cap
```

to verify operation of lsr switch with sample packets

The cable connections between the Host machine and the NetFPGA board is shown below



(6) To add entries to SRAM to perform specfic

swap, pop, push, swap+push, and pop+swap operations,

use the command syntax documented by running

```
./lsr
```

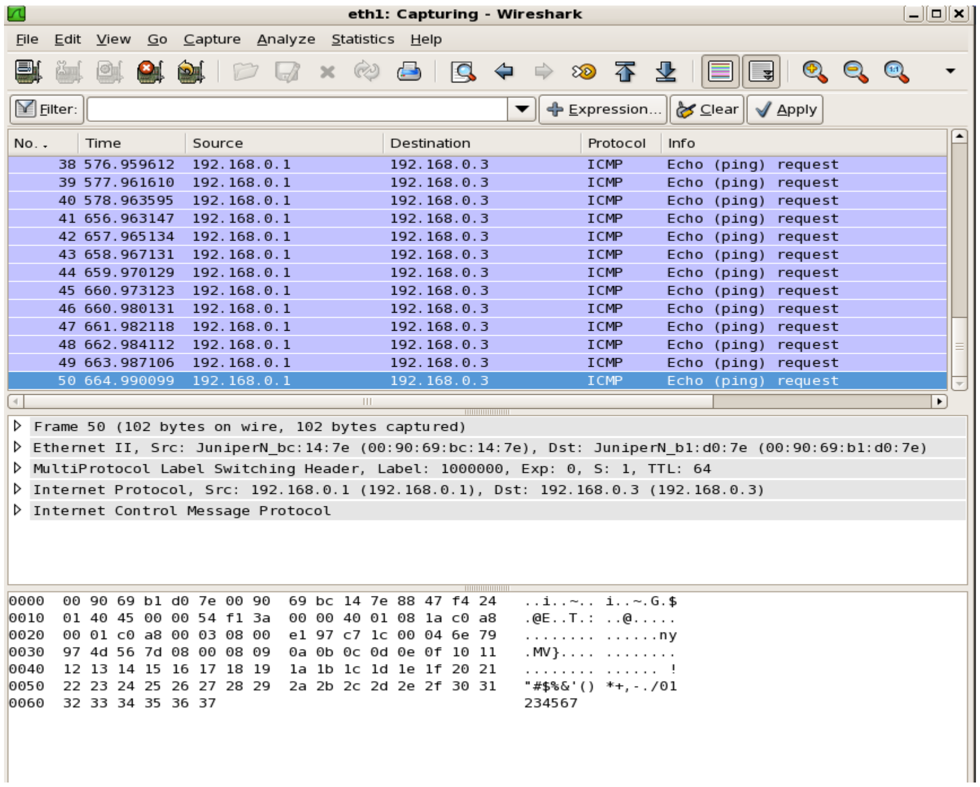
Results :

Wireshark capture of of the packet send to the NetFPGA port 0 and the output packet from NetFPGA port 1 is shown below.

Swap operation

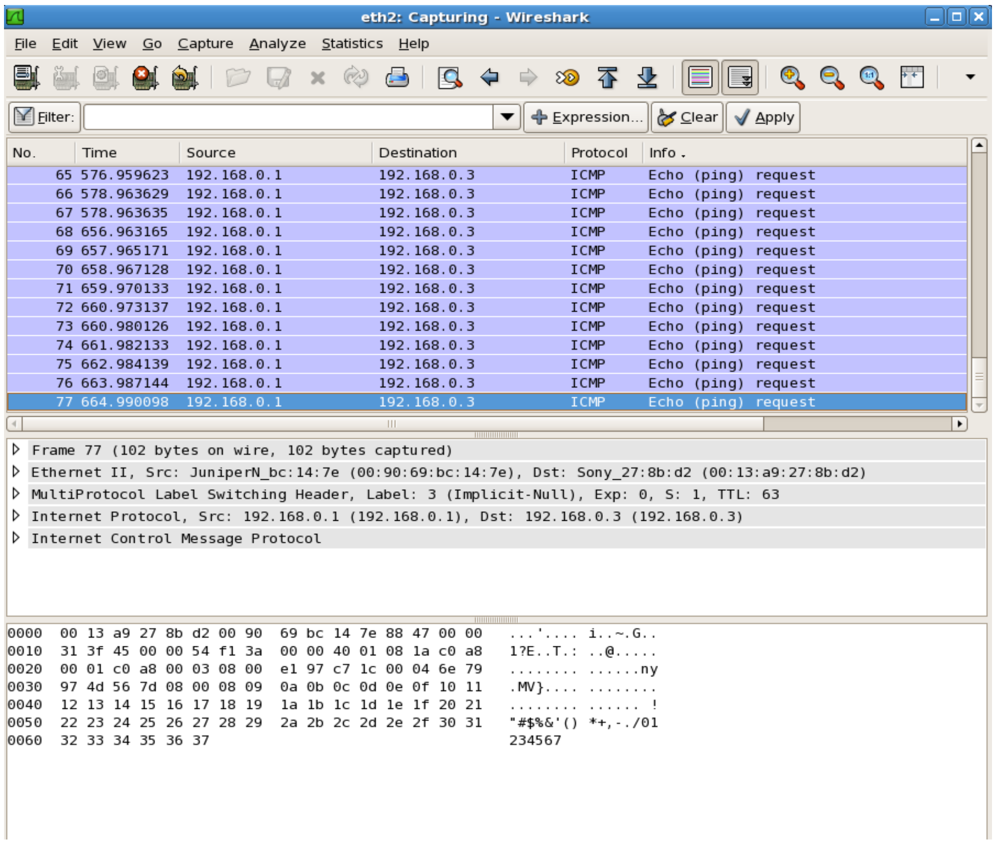
The following commands are used for setup

```
./lsr lsr_init

./lsr swap  2 0 3 3

./lsr mac_out  13a9278bd2 3

tcpreplay intf1=eth1 preswap.cap
```



eth1: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No. .  Time        Source        Destination   Protocol  Info
    38 576.959612  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    39 577.961610  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    40 578.963595  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    41 656.963147  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    42 657.965134  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    43 658.967131  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    44 659.970129  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    45 660.973123  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    46 660.980131  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    47 661.982118  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    48 662.984112  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    49 663.987106  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    50 664.990099  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request

Frame 50 (102 bytes on wire, 102 bytes captured)
Ethernet II, Src: JuniperN_bc:14:7e (00:90:69:bc:14:7e), Dst: JuniperN_b1:d0:7e (00:90:69:b1:d0:7e)
MultiProtocol Label Switching Header, Label: 1000000, Exp: 0, S: 1, TTL: 64
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.3 (192.168.0.3)
Internet Control Message Protocol

0000  00 90 69 b1 d0 7e 00 90  69 bc 14 7e 88 47 f4 24   ..i..~.. i..~.G.$
0010  01 40 45 00 00 54 f1 3a  00 00 40 01 08 1a c0 a8   .@E..T.: ..@.....
0020  00 01 c0 a8 00 03 08 00  e1 97 c7 1c 00 04 6e 79   ........ ......ny
0030  97 4d 56 7d 08 00 08 09  0a 0b 0c 0d 0e 0f 10 11   .MV}.... ........
0040  12 13 14 15 16 17 18 19  1a 1b 1c 1d 1e 1f 20 21   ........ ...... !
0050  22 23 24 25 26 27 28 29  2a 2b 2c 2d 2e 2f 30 31   "#$%&'() *+,-./01
0060  32 33 34 35 36 37                                  234567
```


eth2: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No.   Time        Source        Destination   Protocol  Info
   65 576.959623  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   66 578.963629  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   67 578.963635  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   68 656.963165  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   69 657.965171  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   70 658.967128  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   71 659.970133  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   72 660.973137  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   73 660.980126  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   74 661.982133  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   75 662.984139  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   76 663.987144  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
   77 664.990098  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
```

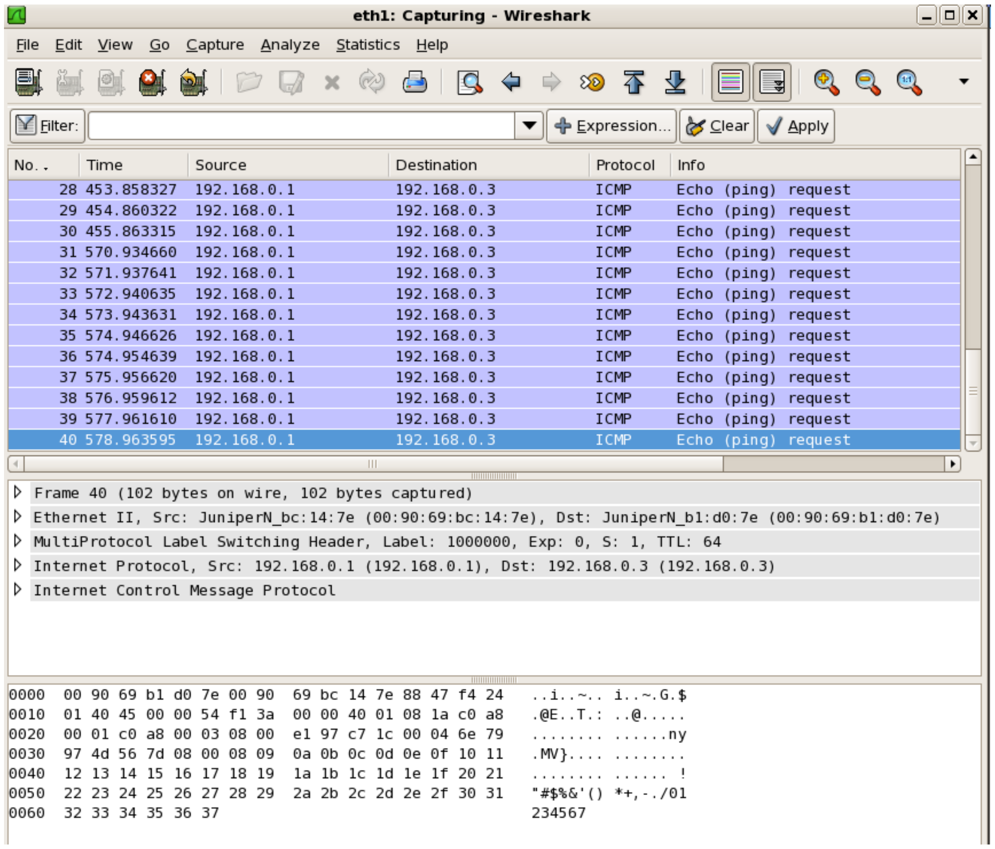
```
Frame 77 (102 bytes on wire, 102 bytes captured)
Ethernet II, Src: JuniperN_bc:14:7e (00:90:69:bc:14:7e), Dst: Sony_27:8b:d2 (00:13:a9:27:8b:d2)
MultiProtocol Label Switching Header, Label: 3 (Implicit-Null), Exp: 0, S: 1, TTL: 63
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.3 (192.168.0.3)
Internet Control Message Protocol
```

```
0000  00 13 a9 27 8b d2 00 90  69 bc 14 7e 88 47 00 00   ...'.... i..~.G..
0010  31 3f 45 00 00 54 f1 3a  00 00 40 01 08 1a c0 a8   1?E..T.: ..@.....
0020  00 01 c0 a8 00 03 08 00  e1 97 c7 1c 00 04 6e 79   ........ ......ny
0030  97 4d 56 7d 08 00 08 09  0a 0b 0c 0d 0e 0f 10 11   .MV}.... ........
0040  12 13 14 15 16 17 18 19  1a 1b 1c 1d 1e 1f 20 21   ........ ...... !
0050  22 23 24 25 26 27 28 29  2a 2b 2c 2d 2e 2f 30 31   "#$%&'() *+,-./01
0060  32 33 34 35 36 37                                 234567
```

Push operation

The following commands are used for setup

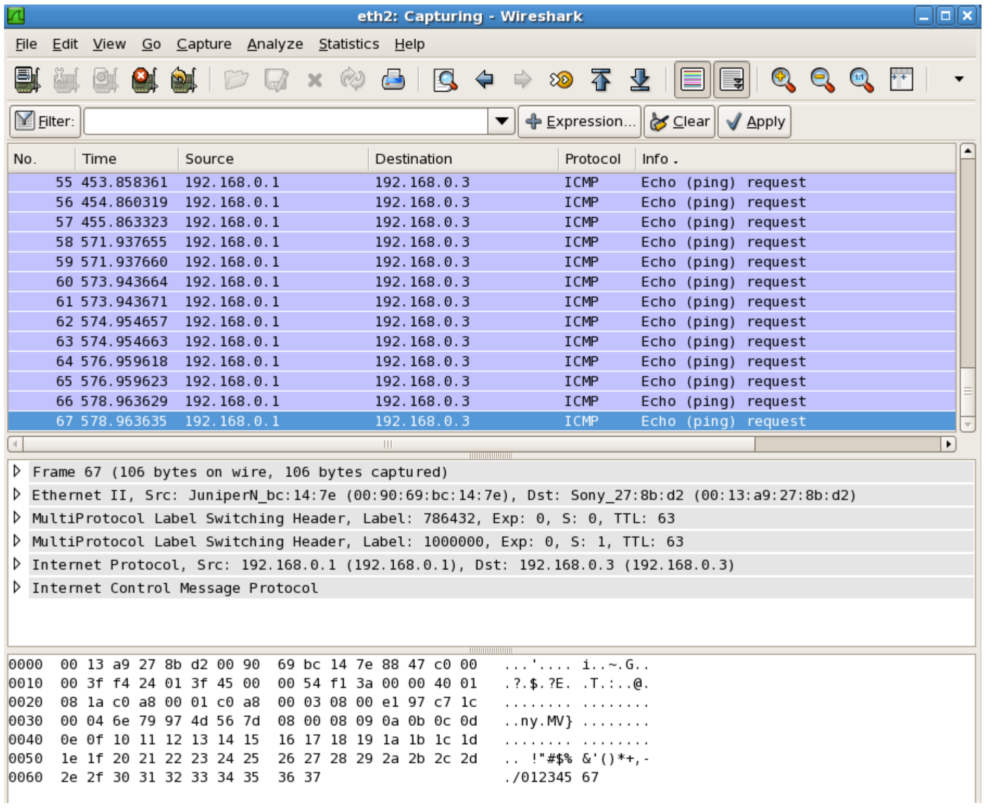
```
./lsr lsr_init

./lsr push  2 0 3 786432

./lsr mac_out  13a9278bd2 3

tcpreplay intf1=eth1 preswap.cap
```


eth1: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No. .  Time        Source       Destination  Protocol Info
    28 453.858327  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    29 454.860322  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    30 455.863315  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    31 570.934660  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    32 571.937641  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    33 572.940635  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    34 573.943631  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    35 574.946626  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    36 574.954639  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    37 575.956620  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    38 576.959612  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    39 577.961610  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    40 578.963595  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
```

```
▷ Frame 40 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: JuniperN_bc:14:7e (00:90:69:bc:14:7e), Dst: JuniperN_b1:d0:7e (00:90:69:b1:d0:7e)
▷ MultiProtocol Label Switching Header, Label: 1000000, Exp: 0, S: 1, TTL: 64
▷ Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.3 (192.168.0.3)
▷ Internet Control Message Protocol
```

```
0000  00 90 69 b1 d0 7e 00 90  69 bc 14 7e 88 47 f4 24   ..i..~.. i..~.G.$
0010  01 40 45 00 00 54 f1 3a  00 00 40 01 08 1a c0 a8   .@E..T.: ..@.....
0020  00 01 c0 a8 00 03 08 00  e1 97 c7 1c 00 04 6e 79   ........ ......ny
0030  97 4d 56 7d 08 00 08 09  0a 0b 0c 0d 0e 0f 10 11   .MV}.... ........
0040  12 13 14 15 16 17 18 19  1a 1b 1c 1d 1e 1f 20 21   ........ ...... !
0050  22 23 24 25 26 27 28 29  2a 2b 2c 2d 2e 2f 30 31   "#$%&'() *+,-./01
0060  32 33 34 35 36 37                                  234567
```


eth2: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No.    Time        Source       Destination  Protocol Info .
    55 453.858361  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    56 454.860319  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    57 455.863323  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    58 571.937655  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    59 571.937660  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    60 573.943664  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    61 573.943671  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    62 574.954657  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    63 574.954663  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    64 576.959618  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    65 576.959623  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    66 578.963629  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
    67 578.963635  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
```

```
▷ Frame 67 (106 bytes on wire, 106 bytes captured)
▷ Ethernet II, Src: JuniperN_bc:14:7e (00:90:69:bc:14:7e), Dst: Sony_27:8b:d2 (00:13:a9:27:8b:d2)
▷ MultiProtocol Label Switching Header, Label: 786432, Exp: 0, S: 0, TTL: 63
▷ MultiProtocol Label Switching Header, Label: 1000000, Exp: 0, S: 1, TTL: 63
▷ Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.3 (192.168.0.3)
▷ Internet Control Message Protocol
```

```
0000  00 13 a9 27 8b d2 00 90  69 bc 14 7e 88 47 c0 00   ...'.... i..~.G..
0010  00 3f f4 24 01 3f 45 00  00 54 f1 3a 00 00 40 01   .?.$.?E. .T.:..@.
0020  08 1a c0 a8 00 01 c0 a8  00 03 08 00 e1 97 c7 1c   ........ ........
0030  00 04 6e 79 97 4d 56 7d  08 00 08 09 0a 0b 0c 0d   ..ny.MV} ........
0040  0e 0f 10 11 12 13 14 15  16 17 18 19 1a 1b 1c 1d   ........ ........
0050  1e 1f 20 21 22 23 24 25  26 27 28 29 2a 2b 2c 2d   .. !"#$% &'()*+,-
0060  2e 2f 30 31 32 33 34 35  36 37                     ./012345 67
```

Pop operation

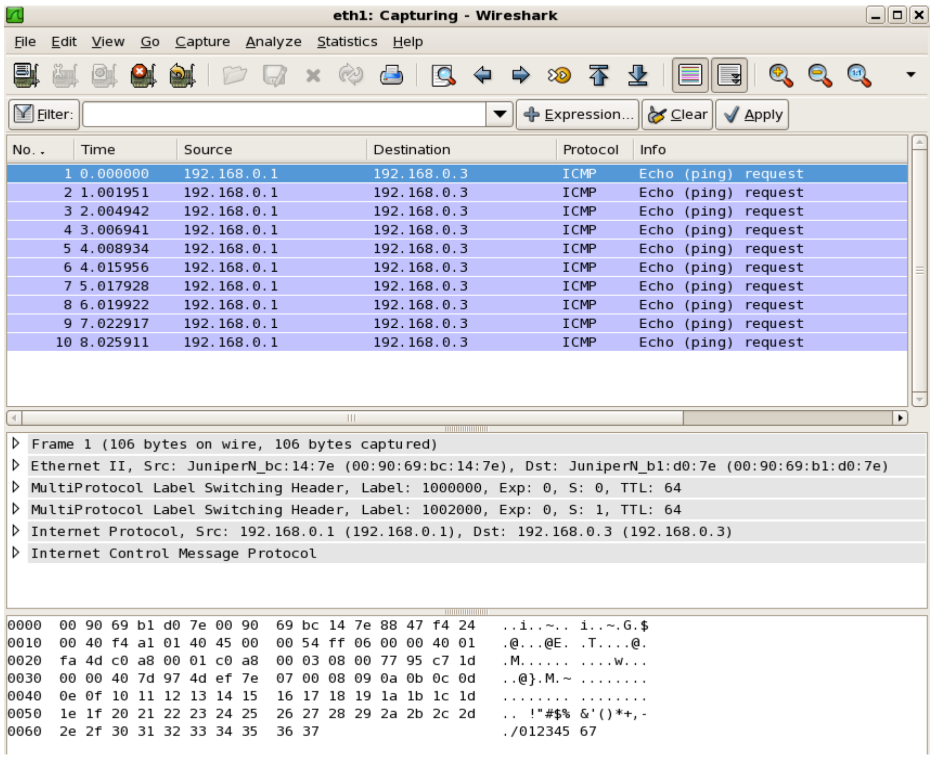
The following commands are used for setup

./lsr lsr_init

./lsr pop 2 0 3

./lsr mac_out 13a9278bd2 3

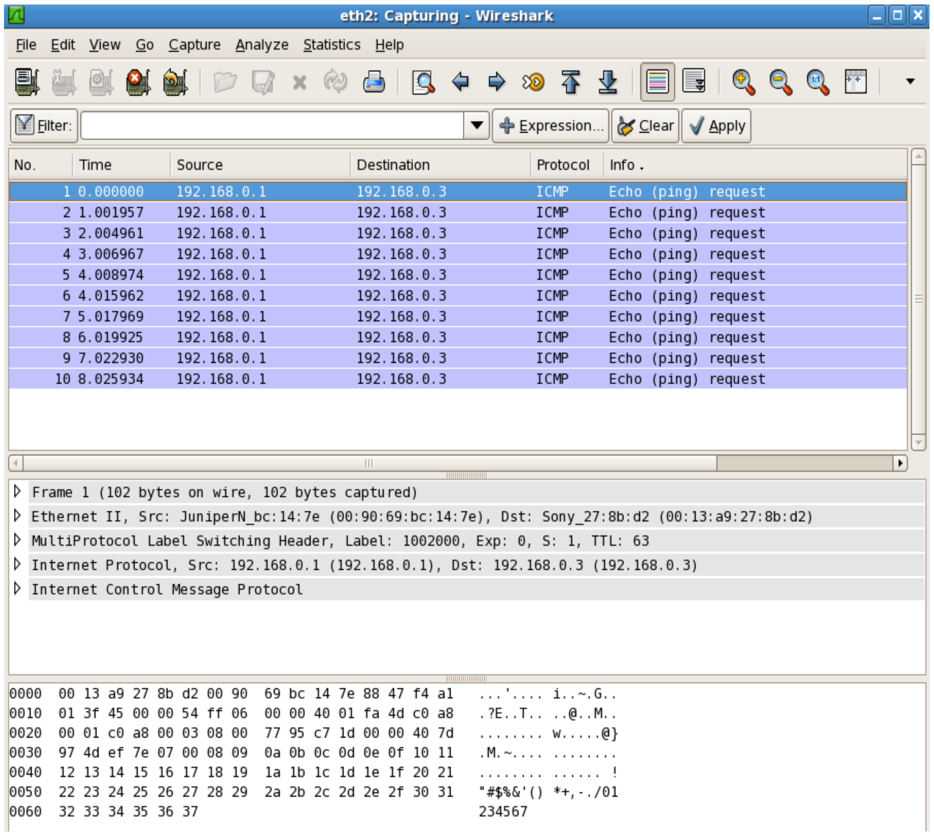
tcpreplay intf1=eth1 predoublepop.cap


eth1: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
2	1.001951	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
3	2.004942	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
4	3.006941	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
5	4.008934	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
6	4.015956	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
7	5.017928	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
8	6.019922	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
9	7.022917	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
10	8.025911	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request

```
Frame 1 (106 bytes on wire, 106 bytes captured)
Ethernet II, Src: JuniperN_bc:14:7e (00:90:69:bc:14:7e), Dst: JuniperN_b1:d0:7e (00:90:69:b1:d0:7e)
MultiProtocol Label Switching Header, Label: 1000000, Exp: 0, S: 0, TTL: 64
MultiProtocol Label Switching Header, Label: 1002000, Exp: 0, S: 1, TTL: 64
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.3 (192.168.0.3)
Internet Control Message Protocol
```

```
0000  00 90 69 b1 d0 7e 00 90  69 bc 14 7e 88 47 f4 24   ..i..~.. i..~.G.$
0010  00 40 f4 a1 01 40 45 00  00 54 ff 06 00 00 40 01   .@...@E. .T....@.
0020  fa 4d c0 a8 00 01 c0 a8  00 03 08 00 77 95 c7 1d   .M...... ....w...
0030  00 00 40 7d 97 4d ef 7e  07 00 08 09 0a 0b 0c 0d   ..@}.M.~ ........
0040  0e 0f 10 11 12 13 14 15  16 17 18 19 1a 1b 1c 1d   ........ ........
0050  1e 1f 20 21 22 23 24 25  26 27 28 29 2a 2b 2c 2d   .. !"#$% &'()*+,-
0060  2e 2f 30 31 32 33 34 35  36 37                     ./012345 67
```

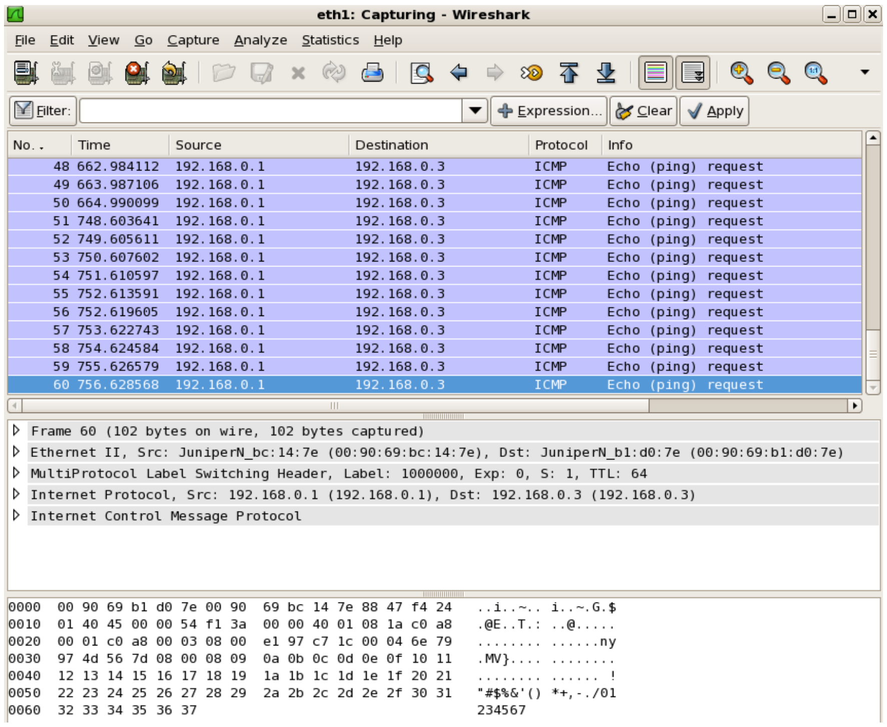

eth2: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No.  Time        Source         Destination    Protocol  Info
   1 0.000000    192.168.0.1    192.168.0.3    ICMP      Echo (ping) request
   2 1.001957    192.168.0.1    192.168.0.3    ICMP      Echo (ping) request
   3 2.004961    192.168.0.1    192.168.0.3    ICMP      Echo (ping) request
   4 3.006967    192.168.0.1    192.168.0.3    ICMP      Echo (ping) request
   5 4.008974    192.168.0.1    192.168.0.3    ICMP      Echo (ping) request
   6 4.015962    192.168.0.1    192.168.0.3    ICMP      Echo (ping) request
   7 5.017969    192.168.0.1    192.168.0.3    ICMP      Echo (ping) request
   8 6.019925    192.168.0.1    192.168.0.3    ICMP      Echo (ping) request
   9 7.022930    192.168.0.1    192.168.0.3    ICMP      Echo (ping) request
  10 8.025934    192.168.0.1    192.168.0.3    ICMP      Echo (ping) request

Frame 1 (102 bytes on wire, 102 bytes captured)
Ethernet II, Src: JuniperN_bc:14:7e (00:90:69:bc:14:7e), Dst: Sony_27:8b:d2 (00:13:a9:27:8b:d2)
MultiProtocol Label Switching Header, Label: 1002000, Exp: 0, S: 1, TTL: 63
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.3 (192.168.0.3)
Internet Control Message Protocol

0000  00 13 a9 27 8b d2 00 90  69 bc 14 7e 88 47 f4 a1   ...'.... i..~.G..
0010  01 3f 45 00 00 54 ff 06  00 00 40 01 fa 4d c0 a8   .?E..T.. ..@..M..
0020  00 01 c0 a8 00 03 08 00  77 95 c7 1d 00 00 40 7d   ........ w.....@}
0030  97 4d ef 7e 07 00 08 09  0a 0b 0c 0d 0e 0f 10 11   .M.~.... ........
0040  12 13 14 15 16 17 18 19  1a 1b 1c 1d 1e 1f 20 21   ........ ...... !
0050  22 23 24 25 26 27 28 29  2a 2b 2c 2d 2e 2f 30 31   "#$%&'() *+,-./01
0060  32 33 34 35 36 37                                  234567
```

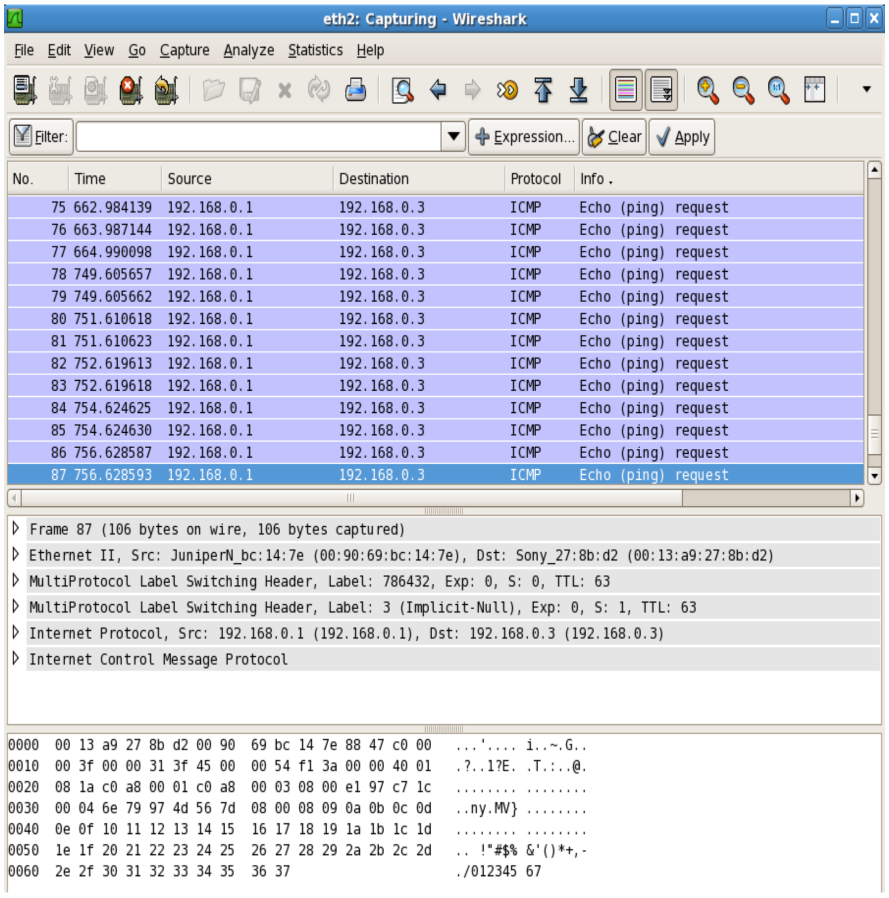
Swap+push operation

The following commands are used for setup

```
./lsr lsr_init

./lsr spush  2 0 3 3 786432

./lsr mac_out  13a9278bd2 3

tcpreplay intf1=eth1 preswap.cap
```


eth1: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No. .  Time        Source        Destination   Protocol  Info
    48 662.984112  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    49 663.987106  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    50 664.990099  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    51 748.603641  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    52 749.605611  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    53 750.607602  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    54 751.610597  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    55 752.613591  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    56 752.619605  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    57 753.622743  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    58 754.624584  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    59 755.626579  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    60 756.628568  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request

Frame 60 (102 bytes on wire, 102 bytes captured)
Ethernet II, Src: JuniperN_bc:14:7e (00:90:69:bc:14:7e), Dst: JuniperN_b1:d0:7e (00:90:69:b1:d0:7e)
MultiProtocol Label Switching Header, Label: 1000000, Exp: 0, S: 1, TTL: 64
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.3 (192.168.0.3)
Internet Control Message Protocol

0000  00 90 69 b1 d0 7e 00 90  69 bc 14 7e 88 47 f4 24   ..i..~.. i..~.G.$
0010  01 40 45 00 00 54 f1 3a  00 00 40 01 08 1a c0 a8   .@E..T.: ..@.....
0020  00 01 c0 a8 00 03 08 00  e1 97 c7 1c 00 04 6e 79   ........ ......ny
0030  97 4d 56 7d 08 00 08 09  0a 0b 0c 0d 0e 0f 10 11   .MV}.... ........
0040  12 13 14 15 16 17 18 19  1a 1b 1c 1d 1e 1f 20 21   ........ ...... !
0050  22 23 24 25 26 27 28 29  2a 2b 2c 2d 2e 2f 30 31   "#$%&'() *+,-./01
0060  32 33 34 35 36 37                                  234567
```

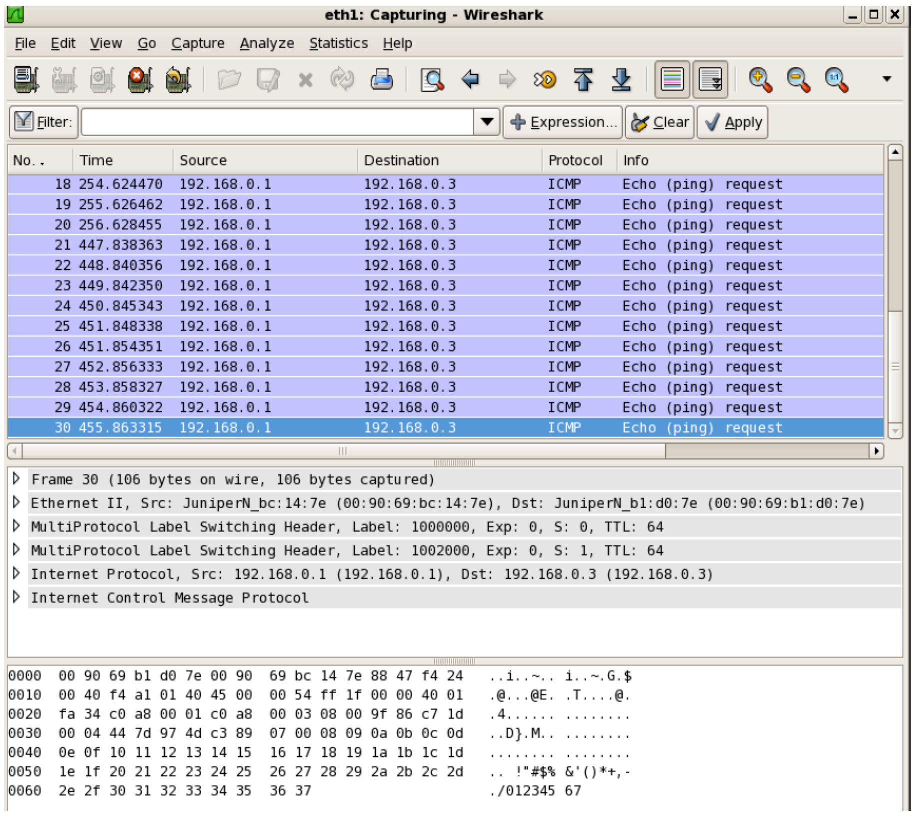

eth2: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No.    Time        Source        Destination   Protocol  Info .
    75 662.984139  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    76 663.987144  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    77 664.990098  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    78 749.605657  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    79 749.605662  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    80 751.610618  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    81 751.610623  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    82 752.619613  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    83 752.619618  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    84 754.624625  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    85 754.624630  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    86 756.628587  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request
    87 756.628593  192.168.0.1   192.168.0.3   ICMP      Echo (ping) request

Frame 87 (106 bytes on wire, 106 bytes captured)
Ethernet II, Src: JuniperN_bc:14:7e (00:90:69:bc:14:7e), Dst: Sony_27:8b:d2 (00:13:a9:27:8b:d2)
MultiProtocol Label Switching Header, Label: 786432, Exp: 0, S: 0, TTL: 63
MultiProtocol Label Switching Header, Label: 3 (Implicit-Null), Exp: 0, S: 1, TTL: 63
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.3 (192.168.0.3)
Internet Control Message Protocol

0000  00 13 a9 27 8b d2 00 90  69 bc 14 7e 88 47 c0 00   ...'.... i..~.G..
0010  00 3f 00 00 31 3f 45 00  00 54 f1 3a 00 00 40 01   .?..1?E. .T.:..@.
0020  08 1a c0 a8 00 01 c0 a8  00 03 08 00 e1 97 c7 1c   ........ ........
0030  00 04 6e 79 97 4d 56 7d  08 00 08 09 0a 0b 0c 0d   ..ny.MV} ........
0040  0e 0f 10 11 12 13 14 15  16 17 18 19 1a 1b 1c 1d   ........ ........
0050  1e 1f 20 21 22 23 24 25  26 27 28 29 2a 2b 2c 2d   .. !"#$% &'()*+,-
0060  2e 2f 30 31 32 33 34 35  36 37                     ./012345 67
```

Pop+swap operation

The following commands are used for setup

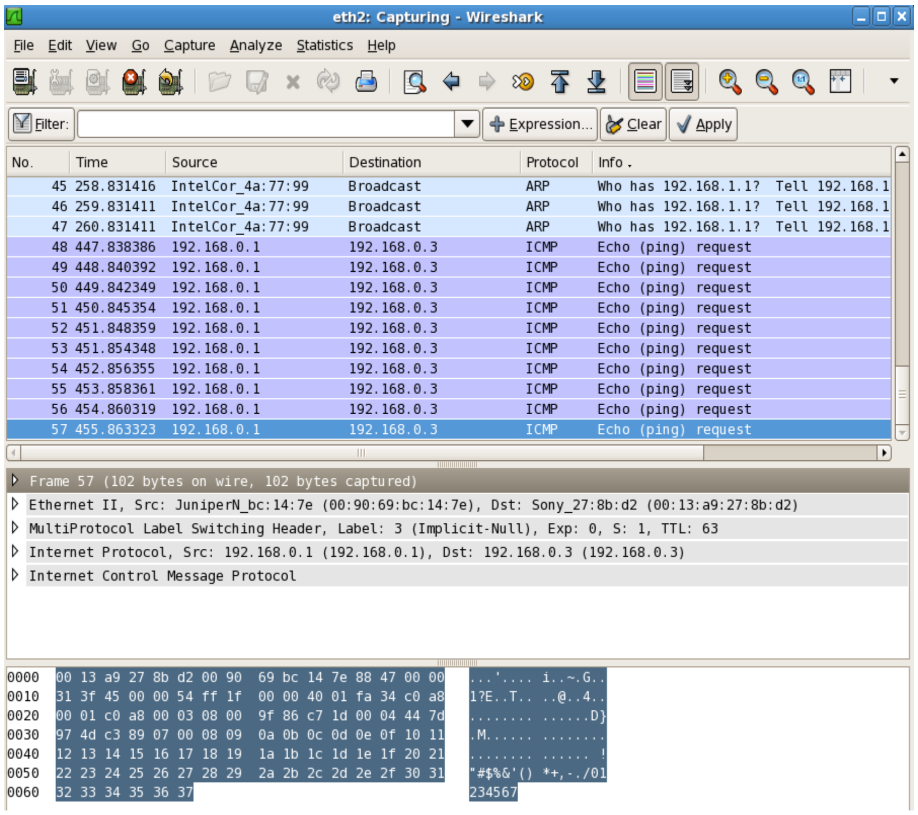
```
./lsr lsr_init

./lsr pswap  0

 ./lsr swap  2 2000 3 9

./lsr mac_out  13a9278bd2 3

tcpreplay intf1=eth1 predoublepop.cap
```


eth1: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No. . Time        Source       Destination  Protocol Info
   18 254.624470  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   19 255.626462  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   20 256.628455  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   21 447.838363  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   22 448.840356  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   23 449.842350  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   24 450.845343  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   25 451.848338  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   26 451.854351  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   27 452.856333  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   28 453.858327  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   29 454.860322  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
   30 455.863315  192.168.0.1  192.168.0.3  ICMP     Echo (ping) request
```

```
Frame 30 (106 bytes on wire, 106 bytes captured)
Ethernet II, Src: JuniperN_bc:14:7e (00:90:69:bc:14:7e), Dst: JuniperN_b1:d0:7e (00:90:69:b1:d0:7e)
MultiProtocol Label Switching Header, Label: 1000000, Exp: 0, S: 0, TTL: 64
MultiProtocol Label Switching Header, Label: 1002000, Exp: 0, S: 1, TTL: 64
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.3 (192.168.0.3)
Internet Control Message Protocol
```

```
0000  00 90 69 b1 d0 7e 00 90  69 bc 14 7e 88 47 f4 24   ..i..~.. i..~.G.$
0010  00 40 f4 a1 01 40 45 00  00 54 ff 1f 00 00 40 01   .@...@E. .T....@.
0020  fa 34 c0 a8 00 01 c0 a8  00 03 08 00 9f 86 c7 1d   .4...... ........
0030  00 04 44 7d 97 4d c3 89  07 00 08 09 0a 0b 0c 0d   ..D}.M.. ........
0040  0e 0f 10 11 12 13 14 15  16 17 18 19 1a 1b 1c 1d   ........ ........
0050  1e 1f 20 21 22 23 24 25  26 27 28 29 2a 2b 2c 2d   .. !"#$% &'()*+,-
0060  2e 2f 30 31 32 33 34 35  36 37                     ./012345 67
```


eth2: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No.   Time        Source            Destination    Protocol  Info
   45 258.831416  IntelCor_4a:77:99 Broadcast      ARP       Who has 192.168.1.1?  Tell 192.168.1
   46 259.831411  IntelCor_4a:77:99 Broadcast      ARP       Who has 192.168.1.1?  Tell 192.168.1
   47 260.831411  IntelCor_4a:77:99 Broadcast      ARP       Who has 192.168.1.1?  Tell 192.168.1
   48 447.838386  192.168.0.1       192.168.0.3    ICMP      Echo (ping) request
   49 448.840392  192.168.0.1       192.168.0.3    ICMP      Echo (ping) request
   50 449.842349  192.168.0.1       192.168.0.3    ICMP      Echo (ping) request
   51 450.845354  192.168.0.1       192.168.0.3    ICMP      Echo (ping) request
   52 451.848359  192.168.0.1       192.168.0.3    ICMP      Echo (ping) request
   53 451.854348  192.168.0.1       192.168.0.3    ICMP      Echo (ping) request
   54 452.856355  192.168.0.1       192.168.0.3    ICMP      Echo (ping) request
   55 453.858361  192.168.0.1       192.168.0.3    ICMP      Echo (ping) request
   56 454.860319  192.168.0.1       192.168.0.3    ICMP      Echo (ping) request
   57 455.863323  192.168.0.1       192.168.0.3    ICMP      Echo (ping) request
```

```
Frame 57 (102 bytes on wire, 102 bytes captured)
Ethernet II, Src: JuniperN_bc:14:7e (00:90:69:bc:14:7e), Dst: Sony_27:8b:d2 (00:13:a9:27:8b:d2)
MultiProtocol Label Switching Header, Label: 3 (Implicit-Null), Exp: 0, S: 1, TTL: 63
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.3 (192.168.0.3)
Internet Control Message Protocol
```

```
0000  00 13 a9 27 8b d2 00 90  69 bc 14 7e 88 47 00 00   ...'.... i..~.G..
0010  31 3f 45 00 00 54 ff 1f  00 00 40 01 fa 34 c0 a8   1?E..T.. ..@..4..
0020  00 01 c0 a8 00 03 08 00  9f 86 c7 1d 00 04 44 7d   ........ ......D}
0030  97 4d c3 89 07 00 08 09  0a 0b 0c 0d 0e 0f 10 11   .M...... ........
0040  12 13 14 15 16 17 18 19  1a 1b 1c 1d 1e 1f 20 21   ........ ...... !
0050  22 23 24 25 26 27 28 29  2a 2b 2c 2d 2e 2f 30 31   "#$%&'() *+,-./01
0060  32 33 34 35 36 37                                  234567
```

For Additional help and Assistance:

Contact: lsr_support@algo-logic.com