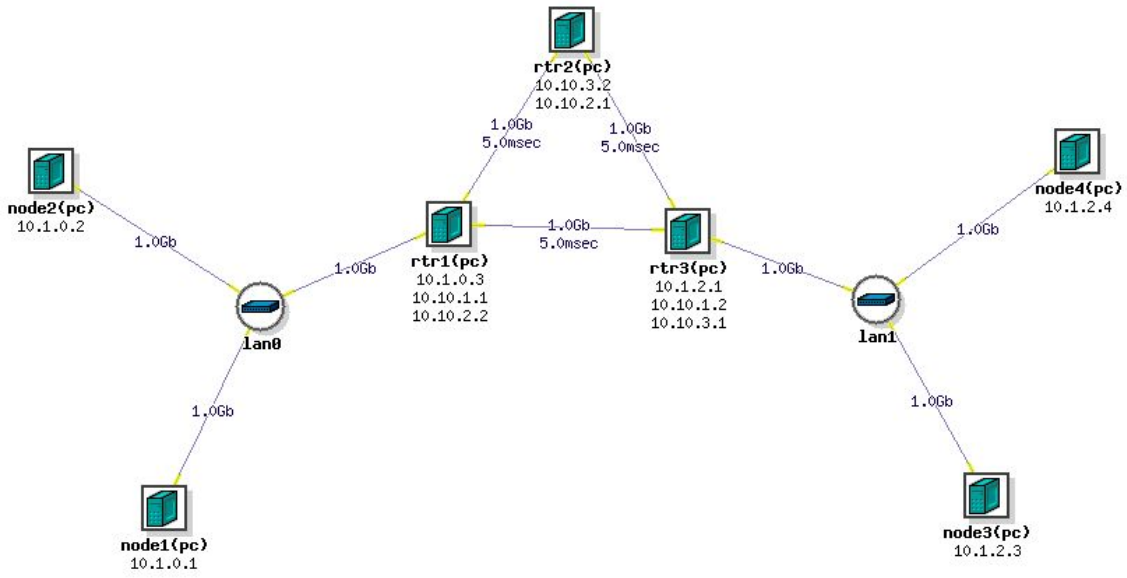
Lab 8 Custom Router
Group: Route-On

Group Member:
Hao Zhang, Vidhi Goel, Venkatasrinivaschowdary Reddy
Oct 3rd, 2015

A simple IP router is built in user space which broadly has 3 functionalities as mentioned in the problem statement. Our Custom Software Router captures the packet(packet sniffer and filter), determines the next hop (packet processor) and then sends the packet to the next hop (packet injector). Following report briefly describes the process and workflow our team adopted for accomplishing this lab.

1. Network Configuration

We used the same NS file as in lab7. Below is the created network:



The route to reach node3 and node4 is a CIDR route. in the notation of 10.1.2.0/24. We use it on rtr1 and rtr2, as these routers does not need to know about the details of that subnet. They just need to know which next hop to send if some packet has a destination of that subnet. In this case, the rtr3 is the gateway of 10.1.2.0/24 subnet and it should resolve all packets coming to this subnet.

2. ARP Lookup

Before deleting all the routes, we perform ping to Node3, Node4, rtr1 and rtr2 to make sure that ARP cache is updated. We then parse the arp table from /proc/net/arp and store it into our local data structure of arp table. Below is the ARP table:

ARP table

```
sc558bq@rtr3:~$ cat /proc/net/arp
IP address       HW type     Flags       HW address            Mask     Device
10.1.2.4         0x1         0x2         00:15:17:1c:d7:0e     *        eth4
192.168.1.254    0x1         0x2         00:1b:21:cd:de:b1     *        eth1
10.10.1.1        0x1         0x2         00:0e:0c:09:5d:ed     *        eth0
10.1.2.3         0x1         0x2         00:15:17:1c:d9:60     *        eth4
10.10.3.2        0x1         0x2         00:04:23:a6:57:79     *        eth3
```

What is this route in CIDR notation? Which routers did you use it on? When you include this route in your router it must be stored in collapsed form.

3. Routing Table

Here, we explain collapsed CIDR route. To reach node3 and node4, the collapsed route in CIDR notation is 10.1.2.0/24. This route was used in rtr1 and rtr2 to reach node3 and node4.
In our custom routing table also, we use the collapsed/aggregated CIDR routes as explained later.
After deleting all 10.x.x.x entries from rtr3, our kernel routing table looks like below:
Kernel routing table
We deleted all the 10.x.x.x entries from the kernel routing table and it looks like below:

```
sc558bq@rtr3:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0         255.255.252.0   U     0      0        0 eth1
0.0.0.0         192.168.1.254   0.0.0.0         UG    0      0        0 eth1
```

Custom Router routing table
We created our custom routing table that will be used by rtr3 and stored it in a local data structure. We can modify the metric in the custom routing table to see the effect of low metric on the path that rtr3 decides while forwarding the packet. **The custom routing table uses aggregated CIDR routes like 10.1.0.0** and looks like below:

```
Destination Gateway Genmask Metric  Iface
10.10.3.0   0.0.0.0 255.255.255.0   0   eth3
10.10.1.0   0.0.0.0 255.255.255.0   0   eth0
10.1.0.0    10.10.1.1   255.255.255.0   2   eth0
10.1.0.0    10.10.3.2   255.255.255.0   5   eth3
10.10.2.0   10.10.1.1   255.255.255.0   2   eth0
10.1.2.0    0.0.0.0 255.255.255.0   0   eth4
```

4. Blocking Kernel from sending Destination Unreachable

We are stopping kernel from sending Destination Unreachable messages by updating the firewall (IP-TABLES) with the following command.
“sudo iptables -A OUTPUT -p icmp --icmp-type destination-unreachable -j DROP”

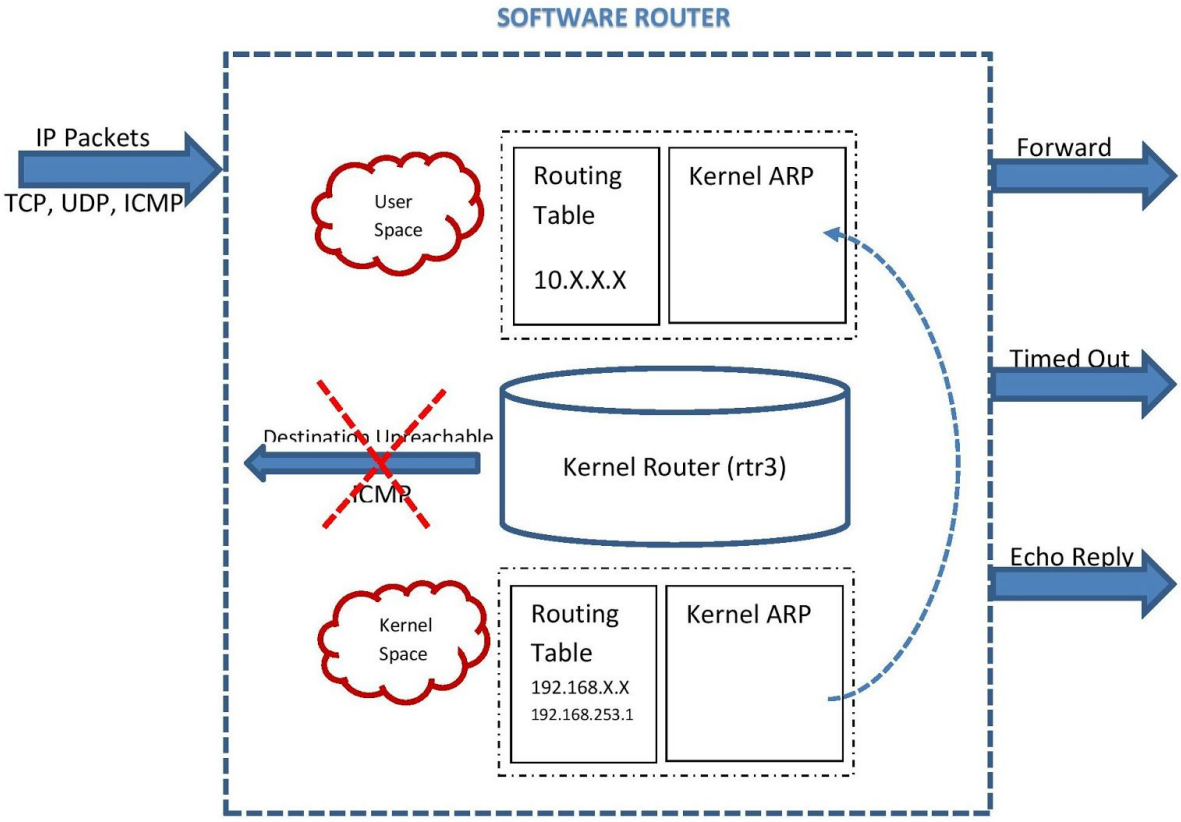
“sudo /sbin/iptables-save”

First command is used to stop the Kernel from sending output icmp messages of type “destination unreachable”.
Later on we need to save this rule by executing the ip-tables save command.
Why are these icmp destination unreachable messages generated ?
The Kernel Routing Table is not up to date. So after receiving packets the Kernel could not forward the packets as it cannot find the destination in the routing table. Hence forcing the Kernel to send Destination Unreachable message to the host.
Why does this cause a problem ?
As the kernel keeps on sending these destination unreachable messages the Network performance and thereby the throughput drops.
How can you block these messages ?
These messages can be blocked by creating filters/rules in the IP-TABLES which act as firewall to the Network Interface. The rule to be created is sudo iptables -A OUTPUT -p icmp --icmp-type destination-unreachable -j DROP. This is created under OUTPUT.

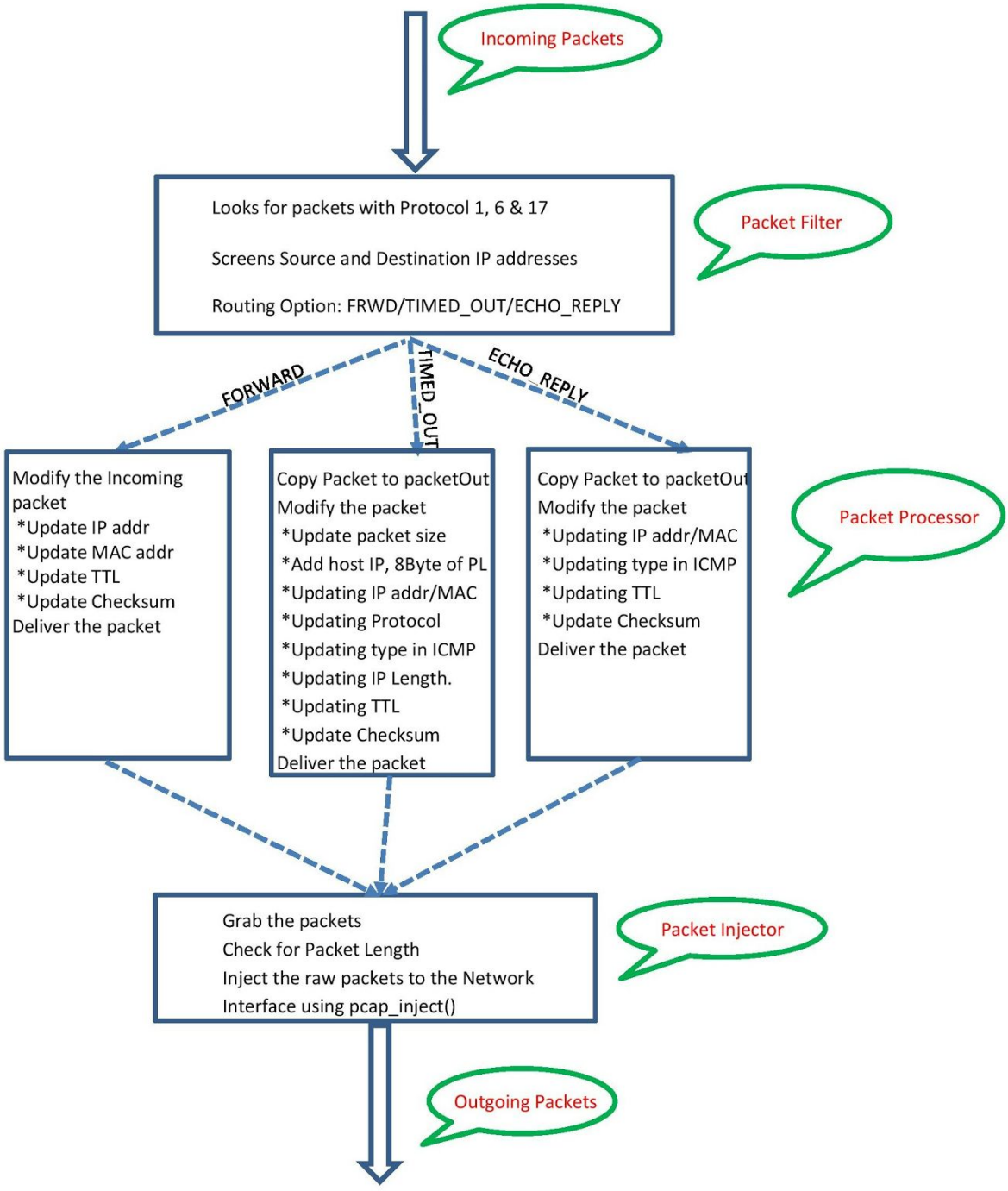
Also we implemented ICMP time exceeded reply and ICMP Echo reply functions will allow traceroute to function smoothly. This code is present in router_util.c script.

5. Design Specifications

The Block Diagram shows our model of our Custom Software Router



Flow chart showing the algorithm for our Router



We are capturing all incoming packets and processing only IP packets with ICMP or TCP or UDP protocols. Apart from this we are also processing packets originating from 10.X.X.X IP addresses.

After filtering, based on protocol and TTL (time-to-live) we are processing the incoming packets into 3 routing options namely: Forward, Timed out and Echo Reply.

In Forward packet routine, we basically check for TTL and update Destination IP/MAC addresses. Lastly we calculate checksum for the updated packet and push it to packet injector.
In ICMP Timed out module we basically copy the incoming packet into a packet of size MTU. Now update the size of the packet by adding IP header and 8 Bytes data from Payload from incoming packet. Now we update the IP/MAC addresses. Also we will update IP length, IP Protocol, TTL(=64), ICMP type, ICMP code and checksum. After updating we will send it to packet injector.
In ICMP Echo reply also we copy the incoming packet into packetOut and update IP/MAC Addresses, TTL, ICMP type, ICMP code and checksum. Once done we will push this packet to packet injector.

6. Code Review

Router.c

The execution begins at router.c. The module mainly calls the helper modules in router_util.c to decide the routing option and call the corresponding routine to fulfill the action. Below is the method description:

1. *main*: calls routine to create routing table and arp table. Scans all the network devices, and record all “eth” devices. It creates multiple threads to handle all the ethernet devices at the same time. Once the threads are started, packet sniffer program starts sniffing packets entering through that Network Interface
2. *sniffer*: It opens the libpcap interface to start sniffing the network interface. It loops through the sniffing process and calls the process packet routine when a packet is received/
3.*process_packet*: It filters the packets based on Protocol and Source IP address. After filtering we start routing the packets based on 3 categories namely, FORWARD, TIMED OUT & ECHO REPLY. It also has function to inject packets to the Network Interface.

router_util.c

This module contains all the util methods to perform the below major operations:

1. *read_arp_cache* : Read the /proc/net/arp file and update into local data structure.
2. *rt_lookup* : finds the routing table entry we are going to use to modify the packet.
3. *routing_opt* : With a raw packet and ip addr of this interface, it decides what we want to do with it
4. *modify_packet_new*: It is a response to P_FORWARD, returned by routing_opt() It takes in a raw packet, find next hop and gateway and modifies the packet.
5. *generate_icmp_time_exceed_packet*: It generates ICMP “time exceeded” packet when P_TIMEOUT is the routing option.
6. *generate_icmp_echo_reply_packet*: It generate ICMP Echo Reply packet when P_ICMPREPLY is the routing option
7. *updateIPHeader*: Takes the entire packet and updates the IP Header fields
8. *updateEtherHeader*: Takes the Ethernet header and updates the corresponding source and destination MAC address
9. *getLocalMac* : Returns the MAC address of a particular ethernet device.

10. *getMACfromIP_new*: Gives the MAC address of the remote host from ARP table based on IP address and device name has code for keep functions like modifying packet to forward it to next hop, ICMP time exceeded reply and ICMP echo reply.

packet_util.c

This has the following helper modules:

1. *calc_ip_checksum*: It calculates the IP checksum for the modified packet
2. *calc_icmp_checksum*: It calculates ICMP checksum for the created ICMP packet
3. *print_packet_handler*: prints the entire packet to logfile.
4. *print_ethernet_header*: prints Ethernet Header to logfile.
5. *print_ip_packet*: prints IP packet to logfile
6. *print_icmp_packet*: prints ICMP packet to logfile
7. *print_tcp_packet*: prints TCP packet to logfile
8. *print_udp_packet*: print UDP packet to logfile

route.c

In this module, we read routing table.txt file and create a Routing Table and Display it by printing it onto the screen.

We will be attaching the C code along with this report.

Code Optimization:

Our first version of router gave a RTT of node1 pinging node3/node4 of 150ms. We did the following optimizations

a. Reduce the number of system calls by caching ARP table and local interface information
b. Reduce the number of pcap handler by having one thread sniffing on each interface, and threads share interface handlers among each other
c. Copying data using memcpy() rather than looping or "=" assigning

Code Run:

```
$ make clean
$ make router
$ sudo ./router
```

7. Scripts

We created a startup.sh script that should be executed before starting our custom router. This script contains commands to ping the neighbouring nodes in order to get their MAC address updated in ARP cache. Then, we delete the routes to all the 10.xx.xx.xx IP addresses present in kernel routing table of rtr3. We also block the kernel "Destination Unreachable" message using the IP rule in our script.

8. PING result

We tried to ping from node4, to rtr1, rtr2, rtr3, node1, node2. They all work, and RTT are as expected - there is little delay on our router.

```
sc558bq@node4:~$ ping -c5 rtr3
PING rtr3-lan1 (10.1.2.1) 56(84) bytes of data.
64 bytes from rtr3-lan1 (10.1.2.1): icmp_seq=1 ttl=64 time=0.168 ms
64 bytes from rtr3-lan1 (10.1.2.1): icmp_seq=2 ttl=64 time=0.100 ms
64 bytes from rtr3-lan1 (10.1.2.1): icmp_seq=3 ttl=64 time=0.131 ms
64 bytes from rtr3-lan1 (10.1.2.1): icmp_seq=4 ttl=64 time=0.126 ms
64 bytes from rtr3-lan1 (10.1.2.1): icmp_seq=5 ttl=64 time=0.146 ms

--- rtr3-lan1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 0.100/0.134/0.168/0.023 ms
sc558bq@node4:~$ ping -c5 rtr2
PING rtr2-link1 (10.10.3.2) 56(84) bytes of data.
64 bytes from rtr2-link1 (10.10.3.2): icmp_seq=1 ttl=63 time=23.9 ms
64 bytes from rtr2-link1 (10.10.3.2): icmp_seq=2 ttl=63 time=10.3 ms
64 bytes from rtr2-link1 (10.10.3.2): icmp_seq=3 ttl=63 time=10.3 ms
64 bytes from rtr2-link1 (10.10.3.2): icmp_seq=4 ttl=63 time=10.3 ms
64 bytes from rtr2-link1 (10.10.3.2): icmp_seq=5 ttl=63 time=10.2 ms

--- rtr2-link1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 10.265/13.059/23.978/5.461 ms
sc558bq@node4:~$ ping -c5 node1
PING node1-lan0 (10.1.0.1) 56(84) bytes of data.
64 bytes from node1-lan0 (10.1.0.1): icmp_seq=1 ttl=62 time=23.4 ms
64 bytes from node1-lan0 (10.1.0.1): icmp_seq=2 ttl=62 time=10.4 ms
64 bytes from node1-lan0 (10.1.0.1): icmp_seq=3 ttl=62 time=10.4 ms
64 bytes from node1-lan0 (10.1.0.1): icmp_seq=4 ttl=62 time=10.3 ms
64 bytes from node1-lan0 (10.1.0.1): icmp_seq=5 ttl=62 time=10.3 ms

--- node1-lan0 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 10.351/13.010/23.449/5.221 ms
sc558bq@node4:~$ ping -c5 node2
PING node2-lan0 (10.1.0.2) 56(84) bytes of data.
64 bytes from node2-lan0 (10.1.0.2): icmp_seq=1 ttl=62 time=13.6 ms
64 bytes from node2-lan0 (10.1.0.2): icmp_seq=2 ttl=62 time=10.3 ms
64 bytes from node2-lan0 (10.1.0.2): icmp_seq=3 ttl=62 time=10.4 ms
64 bytes from node2-lan0 (10.1.0.2): icmp_seq=4 ttl=62 time=10.3 ms
64 bytes from node2-lan0 (10.1.0.2): icmp_seq=5 ttl=62 time=10.4 ms

--- node2-lan0 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 10.366/11.042/13.659/1.314 ms
sc558bq@node4:~$ ping -c5 rtr1
PING rtr1-link2 (10.10.2.2) 56(84) bytes of data.
64 bytes from rtr1-link2 (10.10.2.2): icmp_seq=1 ttl=63 time=10.3 ms
64 bytes from rtr1-link2 (10.10.2.2): icmp_seq=2 ttl=63 time=10.3 ms
64 bytes from rtr1-link2 (10.10.2.2): icmp_seq=3 ttl=63 time=10.3 ms
64 bytes from rtr1-link2 (10.10.2.2): icmp_seq=4 ttl=63 time=10.3 ms
```

```
64 bytes from rtr1-link2 (10.10.2.2): icmp_seq=5 ttl=63 time=10.2 ms

--- rtr1-link2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 10.203/10.310/10.349/0.139 ms
```

9. TRACEROUTE result

We tried to traceroute from node4 to every other node, and they worked.

```
sc558bq@node4:~$ traceroute rtr3
traceroute to rtr3 (10.1.2.1), 30 hops max, 60 byte packets
 1  rtr3-lan1 (10.1.2.1)  0.170 ms  0.165 ms  0.157 ms
sc558bq@node4:~$ traceroute rtr2
traceroute to rtr2 (10.10.3.2), 30 hops max, 60 byte packets
 1  rtr3-lan1 (10.1.2.1)  0.105 ms  0.097 ms  0.089 ms
 2  rtr2-link1 (10.10.3.2)  10.442 ms  10.436 ms  10.429 ms
sc558bq@node4:~$ traceroute rtr1
traceroute to rtr1 (10.10.2.2), 30 hops max, 60 byte packets
 1  rtr3-lan1 (10.1.2.1)  0.145 ms  0.137 ms  0.129 ms
 2  rtr1-link2 (10.10.2.2)  10.343 ms  10.376 ms  10.371 ms
sc558bq@node4:~$ traceroute node1
traceroute to node1 (10.1.0.1), 30 hops max, 60 byte packets
 1  rtr3-lan1 (10.1.2.1)  0.169 ms  0.162 ms  0.153 ms
 2  rtr1-link0 (10.10.1.1)  10.416 ms  10.419 ms  10.413 ms
 3  node1-lan0 (10.1.0.1)  10.479 ms  10.479 ms  10.472 ms
sc558bq@node4:~$ traceroute node2
traceroute to node2 (10.1.0.2), 30 hops max, 60 byte packets
 1  rtr3-lan1 (10.1.2.1)  0.166 ms  0.155 ms  0.147 ms
 2  rtr1-link0 (10.10.1.1)  10.312 ms  10.352 ms  10.350 ms
 3  node2-lan0 (10.1.0.2)  10.424 ms  10.428 ms  10.420 ms
sc558bq@node4:~$ traceroute rtr3
traceroute to rtr3 (10.1.2.1), 30 hops max, 60 byte packets
 1  rtr3-lan1 (10.1.2.1)  0.109 ms  0.102 ms  0.093 ms
sc558bq@node4:~$
```

10. IPERF result

UDP(Min packet size): we used 64 byte packets, avg. throughput is 55.08 Mbps, 112.8 Kbps. As using min packet size, there are way more packets injected, which might cause congestion and finally lead to UDP packet loss. This would have great reduced the throughput in bits/second.

```
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -l 64 -p 12345 -b
1000M
------------------------------------------------------------
Client connecting to node4, UDP port 12345
Sending 64 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 49938 connected with 10.1.2.4 port 12345
```

```
[  3]  0.0-10.0 sec    199 MBytes    167 Mbits/sec
[  3] Sent 3263367 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  63.8 MBytes  53.5 Mbits/sec  0.009 ms 2218654/3263366 (68%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -l 64 -p 12345 -b
1000M
------------------------------------------------------------
Client connecting to node4, UDP port 12345
Sending 64 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 48867 connected with 10.1.2.4 port 12345
[  3]  0.0-10.0 sec    156 MBytes    131 Mbits/sec
[  3] Sent 2551100 datagrams
[  3] Server Report:
[  3]  0.0-10.2 sec  69.8 MBytes  57.1 Mbits/sec  15.593 ms 1407855/2551098 (55%)
[  3]  0.0-10.2 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -l 64 -p 12345 -b
1000M
------------------------------------------------------------
Client connecting to node4, UDP port 12345
Sending 64 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 59810 connected with 10.1.2.4 port 12345
[  3]  0.0-10.0 sec    160 MBytes    134 Mbits/sec
[  3] Sent 2622958 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  66.1 MBytes  55.4 Mbits/sec  0.006 ms 1539937/2622957 (59%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -l 64 -p 12345 -b
1000M
------------------------------------------------------------
Client connecting to node4, UDP port 12345
Sending 64 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 42129 connected with 10.1.2.4 port 12345
[  3]  0.0-10.0 sec    160 MBytes    134 Mbits/sec
[  3] Sent 2614993 datagrams
[  3] Server Report:
[  3]  0.0-10.2 sec  66.5 MBytes  54.4 Mbits/sec  15.595 ms 1525374/2614991 (58%)
[  3]  0.0-10.2 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -l 64 -p 12345 -b
1000M
------------------------------------------------------------
Client connecting to node4, UDP port 12345
Sending 64 byte datagrams
UDP buffer size:   110 KByte (default)
```

```
------------------------------------------------------------
[  3] local 10.1.0.1 port 48412 connected with 10.1.2.4 port 12345
[  3]  0.0-10.0 sec    158 MBytes    133 Mbits/sec
[  3] Sent 2589050 datagrams
[  3] Server Report:
[  3]  0.0-10.2 sec  67.2 MBytes  55.0 Mbits/sec  15.602 ms 1487473/2589048 (57%)
[  3]  0.0-10.2 sec  1 datagrams received out-of-order
```

UDP (Max packet size): we used default 1470 byte packets, throughput is 957Mbps on a 1Gbps link, which is approx. 85.33 Kpps.

Client:

```
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 10k
------------------------------------------------------------
WARNING: delay too large, reducing from 1.2 to 1.0 seconds.
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 55721 connected with 10.1.2.4 port 51717
write2 failed: Connection refused
[  3]  0.0- 1.0 sec  1.44 KBytes  11.8 Kbits/sec
[  3] Sent 1 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 1 tries of 250 ms.
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 10k
------------------------------------------------------------
WARNING: delay too large, reducing from 1.2 to 1.0 seconds.
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 58912 connected with 10.1.2.4 port 51717
[  3]  0.0-11.0 sec  15.8 KBytes  11.8 Kbits/sec
[  3] Sent 11 datagrams
[  3] Server Report:
[  3]  0.0-11.0 sec  15.8 KBytes  11.8 Kbits/sec  0.023 ms    0/   11 (0%)
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 100M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 48602 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec    120 MBytes    101 Mbits/sec
[  3] Sent 85471 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec    120 MBytes    101 Mbits/sec  0.026 ms    0/85471 (0%)
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 256M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
```

```
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 56452 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec    312 MBytes    261 Mbits/sec
[  3] Sent 222223 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec    312 MBytes    261 Mbits/sec  0.027 ms    0/222223 (0%)
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 256M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 51353 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec    312 MBytes    261 Mbits/sec
[  3] Sent 222223 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec    312 MBytes    261 Mbits/sec  0.039 ms    0/222223 (0%)
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 512M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 41540 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec    637 MBytes    535 Mbits/sec
[  3] Sent 454546 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec    637 MBytes    535 Mbits/sec  0.038 ms    0/454545 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 512M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 44646 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec    637 MBytes    535 Mbits/sec
[  3] Sent 454546 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec    637 MBytes    535 Mbits/sec  0.029 ms    0/454545 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 800M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 41553 connected with 10.1.2.4 port 51717
```

```
[  3]  0.0-10.0 sec  1001 MBytes    840 Mbits/sec
[  3] Sent 714286 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1001 MBytes    840 Mbits/sec  0.022 ms    0/714285 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 800M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 35886 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec  1001 MBytes    840 Mbits/sec
[  3] Sent 714286 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1001 MBytes    840 Mbits/sec  0.023 ms    0/714285 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 900M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 39263 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec  1.05 GBytes    905 Mbits/sec
[  3] Sent 769231 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.05 GBytes    905 Mbits/sec  0.024 ms    0/769230 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 900M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 32781 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec  1.05 GBytes    905 Mbits/sec
[  3] Sent 769231 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.05 GBytes    905 Mbits/sec  0.016 ms    0/769230 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 950M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 57772 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec
[  3] Sent 813814 datagrams
```

```
[  3] Server Report:
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec  0.014 ms    0/813814 (0%)
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 950M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 39430 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec
[  3] Sent 813810 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec  0.013 ms    0/813809 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 1000M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 44851 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec
[  3] Sent 813791 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec  0.021 ms    0/813790 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 1000M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 45583 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec
[  3] Sent 813794 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec  0.017 ms    0/813793 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
sc558bq@node1:~$ /usr/local/etc/emulab/emulab-iperf -c node4 -u -p 51717 -b 1100M
------------------------------------------------------------
Client connecting to node4, UDP port 51717
Sending 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.0.1 port 43155 connected with 10.1.2.4 port 51717
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec
[  3] Sent 813805 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec  0.015 ms    0/813805 (0%)
```

Server:

```
sc558bq@node4:~$ /usr/local/etc/emulab/emulab-iperf -s -u -p 51717
------------------------------------------------------------
Server listening on UDP port 51717
Receiving 1470 byte datagrams
UDP buffer size:   110 KByte (default)
------------------------------------------------------------
[  3] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 58912
[  3]  0.0-11.0 sec  15.8 KBytes  11.8 Kbits/sec  0.024 ms    0/   11 (0%)
[  4] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 48602
[  4]  0.0-10.0 sec    120 MBytes    101 Mbits/sec  0.026 ms    0/85471 (0%)
[  3] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 56452
[  3]  0.0-10.0 sec    312 MBytes    261 Mbits/sec  0.028 ms    0/222223 (0%)
[  4] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 51353
[  4]  0.0-10.0 sec    312 MBytes    261 Mbits/sec  0.040 ms    0/222223 (0%)
[  3] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 41540
[  3]  0.0-10.0 sec    637 MBytes    535 Mbits/sec  0.039 ms    0/454545 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
[  4] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 44646
[  4]  0.0-10.0 sec    637 MBytes    535 Mbits/sec  0.030 ms    0/454545 (0%)
[  4]  0.0-10.0 sec  1 datagrams received out-of-order
[  3] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 41553
[  3]  0.0-10.0 sec  1001 MBytes    840 Mbits/sec  0.023 ms    0/714285 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
[  4] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 35886
[  4]  0.0-10.0 sec  1001 MBytes    840 Mbits/sec  0.024 ms    0/714285 (0%)
[  4]  0.0-10.0 sec  1 datagrams received out-of-order
[  3] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 39263
[  3]  0.0-10.0 sec  1.05 GBytes    905 Mbits/sec  0.025 ms    0/769230 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
[  4] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 32781
[  4]  0.0-10.0 sec  1.05 GBytes    905 Mbits/sec  0.016 ms    0/769230 (0%)
[  4]  0.0-10.0 sec  1 datagrams received out-of-order
[  3] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 57772
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec  0.015 ms    0/813814 (0%)
[  4] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 39430
[  4]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec  0.014 ms    0/813809 (0%)
[  4]  0.0-10.0 sec  1 datagrams received out-of-order
[  3] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 44851
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec  0.022 ms    0/813790 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
[  4] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 45583
[  4]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec  0.017 ms    0/813793 (0%)
[  4]  0.0-10.0 sec  1 datagrams received out-of-order
[  3] local 10.1.2.4 port 51717 connected with 10.1.0.1 port 43155
[  3]  0.0-10.0 sec  1.11 GBytes    957 Mbits/sec  0.016 ms    0/813805 (0%)
```

TCP: (For reference)

```
Server:
sc558bq@node3:~$ /usr/local/etc/emulab/emulab-iperf -s -p 11224 -w 256M
------------------------------------------------------------
Server listening on TCP port 11224
TCP window size:   512 MByte (WARNING: requested   256 MByte)
------------------------------------------------------------
[  4] local 10.1.2.3 port 11224 connected with 10.1.0.1 port 58149
[  5] local 10.1.2.3 port 11224 connected with 10.1.0.1 port 58150
[  4]  0.0-163.0 sec    665 MBytes  34.2 Mbits/sec
[  5]  0.0-222.4 sec    528 MBytes  19.9 Mbits/sec
[SUM]  0.0-222.4 sec  1.16 GBytes  45.0 Mbits/sec

Client
sc558bq@node1:/tmp/lab8$ /usr/local/etc/emulab/emulab-iperf -c node3 -p 11224 -w 400M
------------------------------------------------------------
Client connecting to node3, TCP port 11224
TCP window size:   800 MByte (WARNING: requested   400 MByte)
------------------------------------------------------------
[  3] local 10.1.0.1 port 58148 connected with 10.1.2.3 port 11224
[  3]  0.0-10.0 sec    311 MBytes    261 Mbits/sec
sc558bq@node1:/tmp/lab8$ /usr/local/etc/emulab/emulab-iperf -c node3 -p 11224 -w 1024M
------------------------------------------------------------
Client connecting to node3, TCP port 11224
TCP window size: 1.86 GByte (WARNING: requested 1.00 GByte)
------------------------------------------------------------
[  3] local 10.1.0.1 port 58149 connected with 10.1.2.3 port 11224
[  3]  0.0-10.0 sec    665 MBytes    557 Mbits/sec
sc558bq@node1:/tmp/lab8$ /usr/local/etc/emulab/emulab-iperf -c node3 -p 11224 -w 512M
------------------------------------------------------------
Client connecting to node3, TCP port 11224
TCP window size: 1.00 GByte (WARNING: requested   512 MByte)
------------------------------------------------------------
[  3] local 10.1.0.1 port 58150 connected with 10.1.2.3 port 11224
[  3]  0.0-10.0 sec    528 MBytes    442 Mbits/sec
```

A brief explanation about Mega: There is a historical reason that network people uses base 10 metric rather than base 2, as IEEE 1541-2002 (adopted in 2005 and reaffirmed in 2008) defines that standard metric prefixes should not be used when referring to base 2 numbers, typical in computing. Network people therefore use base 10 in network metric measurement and testing. While file transfer people deals more with memory and address space, which is manufactured in size of base 2. Using base 2 is better for them to align file data into memory (no waste of cache lines for example), and therefore giving better performance.

11. CIDR

CIDR (Classless Inter Domain Routing) is important for reasons like we can customize the size of our Network based on number of addresses required. Also Masking has its importance while

routing packets. By allowing the network address to be variable and adding logic to gracefully handle the routing of these variable-size networks, *CIDR improved address allocation efficiency by allowing a more granular approach to network provisioning than the original concept of classes A, B, and C.* That's what CIDR stands for, Classless Inter-Domain Routing. Meaning, routing between networks that are not bound by the sizing limitations of Classes A, B, or C.

In our case by using CIDR we reduced the number of entries in routing table making it look small and thus reducing the processing time. Below is a part of our routing table, that shows an aggregated CIDR route to reach node1 and node2 via rtr1 on lan0.