BITS Pilani
Hyderabad Campus

Department Of Computer Science and Information Systems

Project Report in Advanced Computer Networks

Project Title: Empirical Analysis of ECMP and Least Utilized Path Routing Algorithms in Software Defined Networks

Under the Supervision of

Dr. Pragati Srivastava

Members

Kushal Chakraborty [2022H1030089H]

Mohammad Avesh Husain [2022H1030090H]

Aritra Kumar Dutta [2022H1030096H]

INTRODUCTION

Data canters require redundant connections to avoid network failures. However, using redundant connections was not helping as it was only used as a backup and inefficient. ECMP routing algorithm weights all links equally and chooses one path out of several possible equally weighted paths. Then the final path is chosen based on a hash function. Hash functions are also purely mathematical, during calculation for one of the equally weighted path Network traffic is not considered. Another way could be to search the network and calculate the utilization of links at specific frequencies and assign these utilizations as link weights. However, the network load can change quickly and get stale values. so, in order to Empirically analyse it is required to determine the ideal frequency of performing probing as well. This analysis also includes comparisons with ECMP and LUP.

ABSTRACT

Redundant connections in the network have great benefits, but they require good routing algorithms. One easy way is to use static routing technique in which path between two nodes of network is always fixed. But this static routing does consider any advantage provided by the redundant link. Another method is ECMP routing. Here all links have a weight of 1 and shortest path between two nodes in the network is calculated using Dijkstra's algorithm. There could be multiple paths between the source and destination with minimum weight. From those, one of the paths is selected using a hash function using Source IP, Destination IP as input. When a particular path for source and the destination are calculated for some port, it always remains the same. In short, the hash function is a purely mathematical function, not taking into account factors such as: Congestion in the network at a particular time. The proposed algorithm assigns the utilization of the link as the weight of each link. that's why, the shortest route chosen by Dijkstra is the least used route. But due to the virtue of network properties, it is always changing rapidly and the usage assigned may be required to get updated frequently. so, the frequency of how often active probes should be run to determine the link utilizations in the network. For this an empirical analysis which involves various experiments wherein the probing is done at various frequencies is done. In the same analysis, we also compare it with the LUP and ECMP routing algorithms.

PROBLEM STATEMENT

In order to prevent failures, data centres require networks with redundant links. There are other ways to pick the path for traffic from these redundant links, including giving each link in the network an identical weight, assigning varied weights based on link utilisation, etc. It is necessary to conduct an empirical investigation to determine which of these strategies works better in certain circumstances.

Methodology

ECMP algorithm

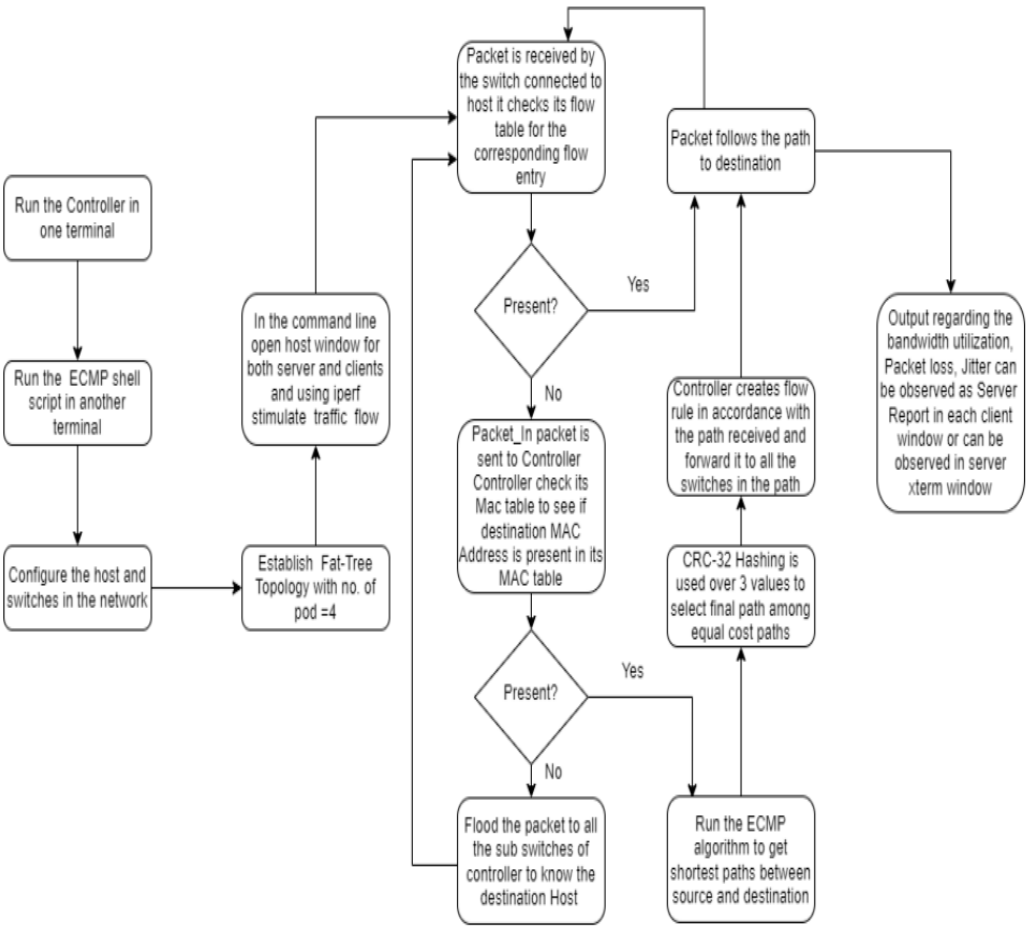
```
function MultipathDijkstra(Graph, source)
    for each vertex v in Graph:
        dist[v] ← INFINITY
        visited[v] ← FALSE
        previous[v] ← UNDEFINED
    end for
    dist[source] ← 0
    insert source into Q

    while Q is not empty do:
            u ← vertex in Q with smallest in dist[] and not yet visited;
            remove u from Q
            visited[u] ← true
            for each neighbour v of u:
                alt ← dist[u] + dist_between(u, v)
                if alt < dist[v]:
                    dist[v] ← alt
                    reset the previous[v]
                    add u into previous[v]
                end if
                else if alt == dist[v]:
                    add u into previous[v]
                if !visited[v] :
                    insert v into Q
                if
            end for

    end while
    return dist;
end function
```

Mutipath Dijkstra Algorithm

```
function ECMP_HASH(packet)
    hash_input ← [0] * 5
    if ipv4 is in packet :
        ip ← packet
        hash_input[0] ← unsigned ip[srcIP]
        hash_input[1] ← unsigned ip[dstIP]
        hash_input[2] ← unsigned ip[protocol]
        if TCP is in ip or UDP is in ip:
            l4 ← ip
            hash_input[3] ← l4[srcPort]
            hash_input[4] ← l4[dstPort]
            return crc32(hash_input)
    return 0
end function
```

Hashed ECMP Algorithm

ECMP Work Flow:



Least utilized path method:

Initializations:

1. Initialize all link weight as 1
2. Initialize last_port_stat (a mapping from port to last bytes received on the port) as 0.

Port statistics essentially tell us how many bytes have ever passed via a port. Therefore, each time we obtain port statistics, we must deduct them from the prior bytes in order to obtain the bytes transferred over the past f seconds.

Algorithm 1: Active prober (probing frequency f) Executed every f seconds

a. for every switch:
    a. for every port:
        i. Send port stats request

Algorithm 2: Link weight updater (probing frequency f) Executed on reception of port stats at controller Stats are received from each switch as a list. Each entry in the list corresponds to the ports in the switch.

1) for stat in stats:

a) bytes ← stat.bytes - last_port_stat[stat.port]
b) last_port_stat[stat.port] ← stat.bytes
c) utilization ← bytes / f
d) link_weight [stat.switch][stat.port.connected_switch] ← utilization / 1

Experimental setup:

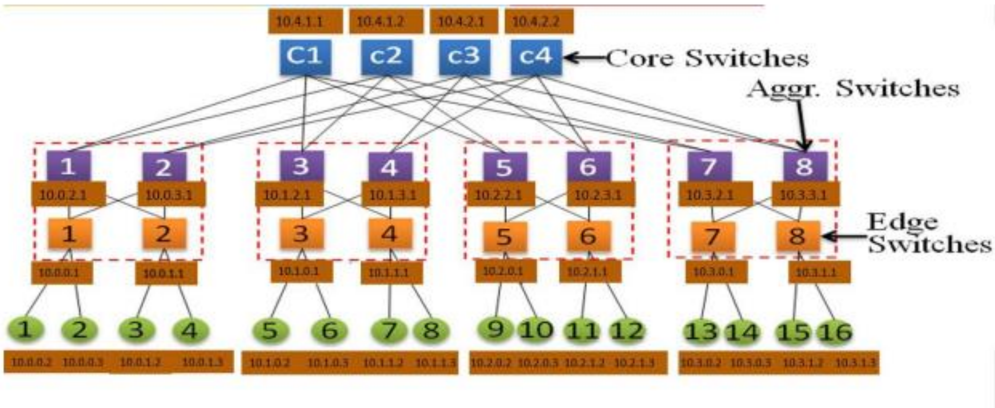


Fig.1 Fat Tree Topology

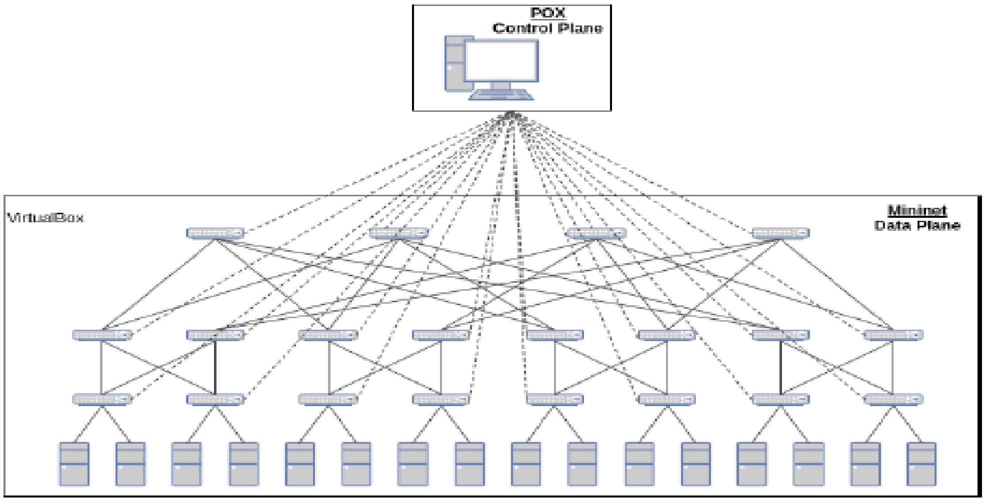
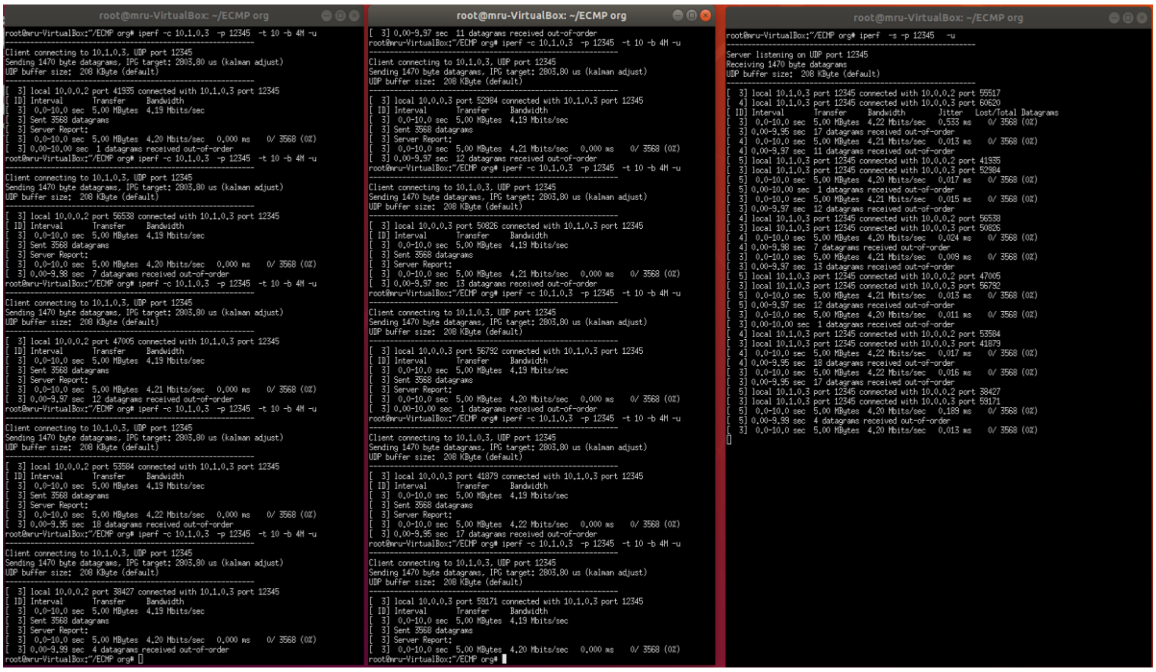


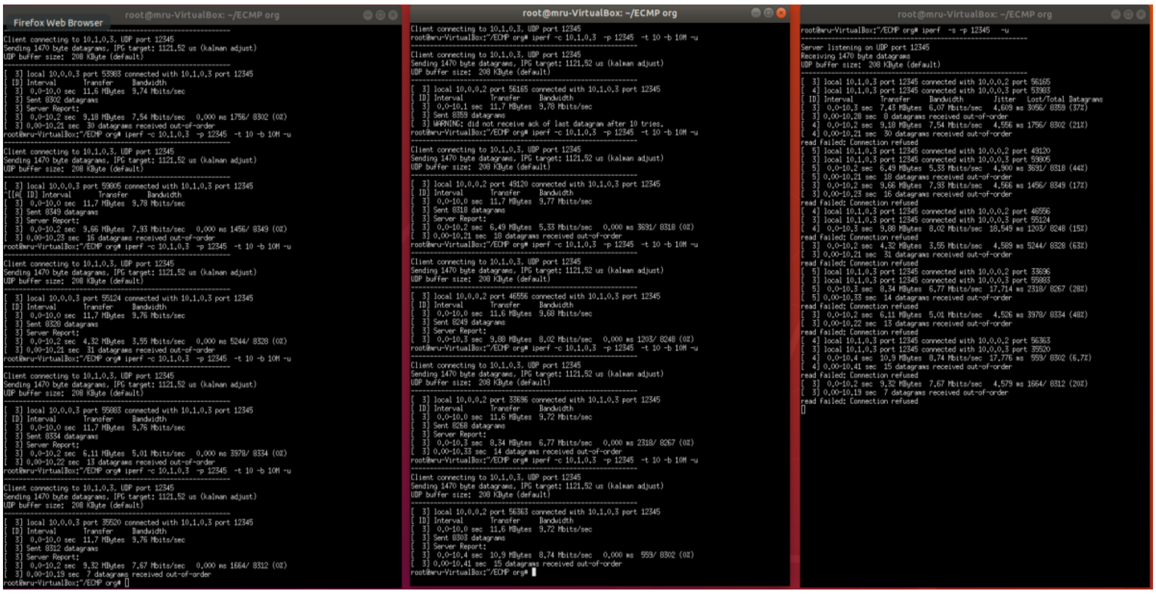
Fig.2 System Design

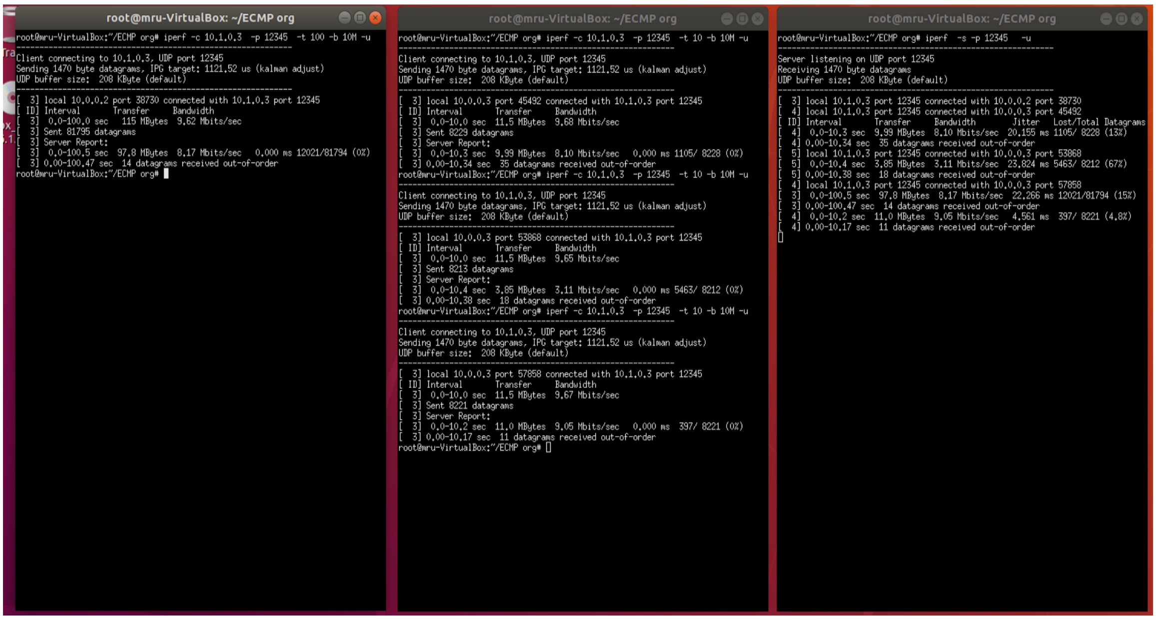
The three algorithms were applied to the fat-tree topology mentioned above. Utilizations calculated at frequency of 10 sec were used to test the least used path algorithm. For parallel, long, and short flows, it was compared how much packet loss, throughput, and jitter there was.

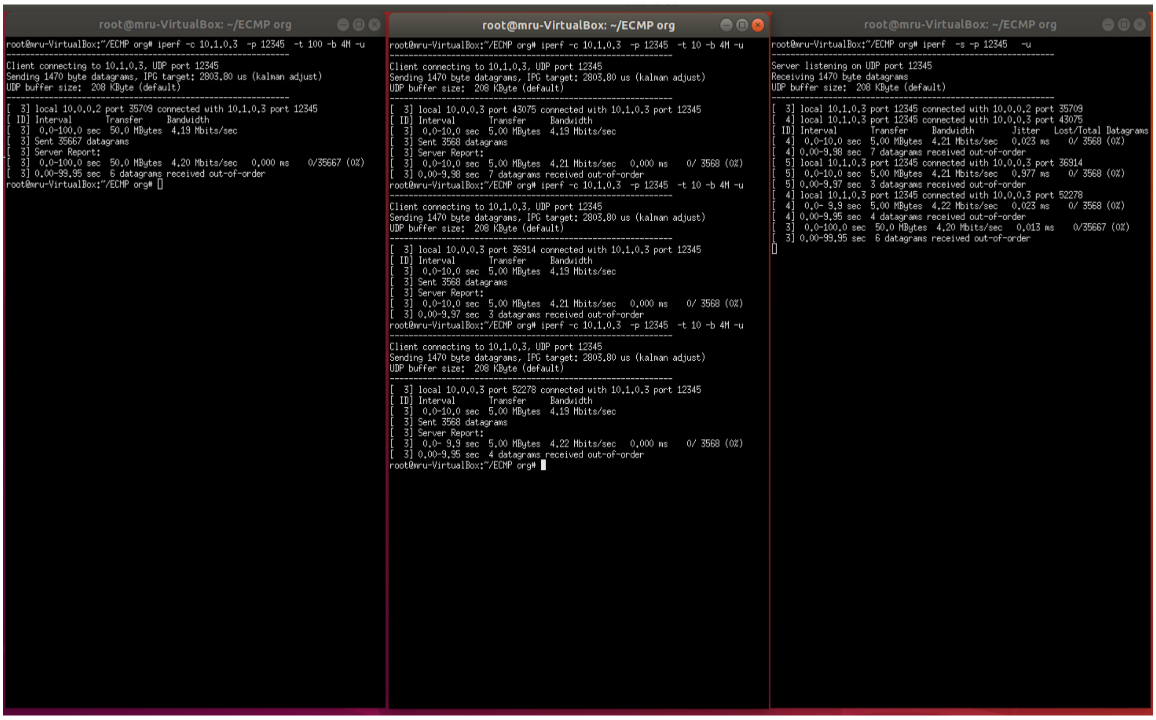
Outputs

ECMP:

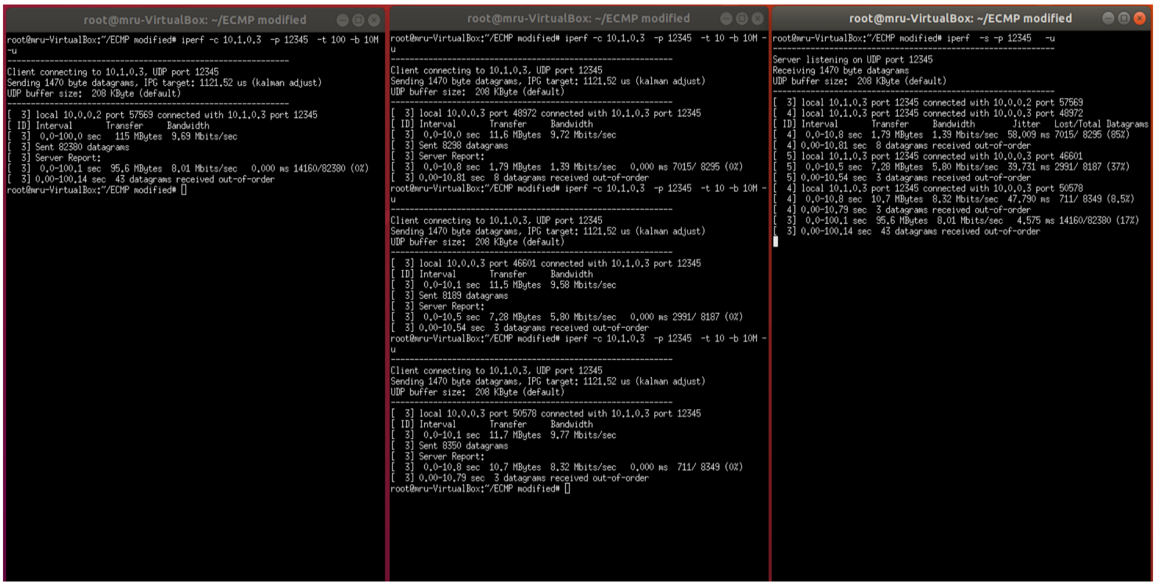


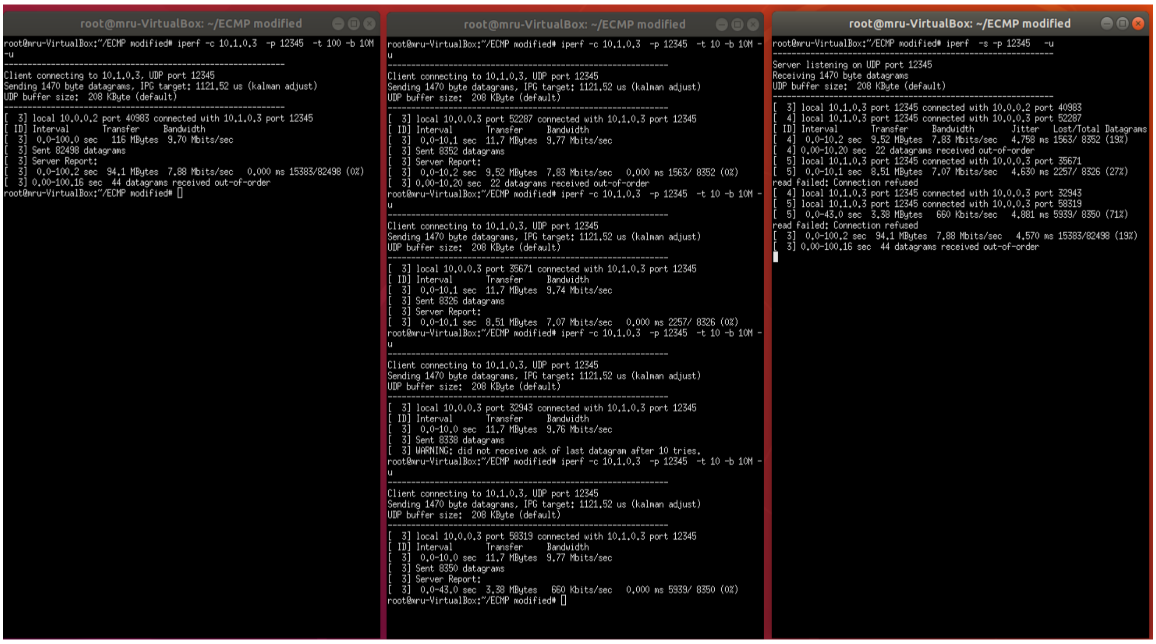


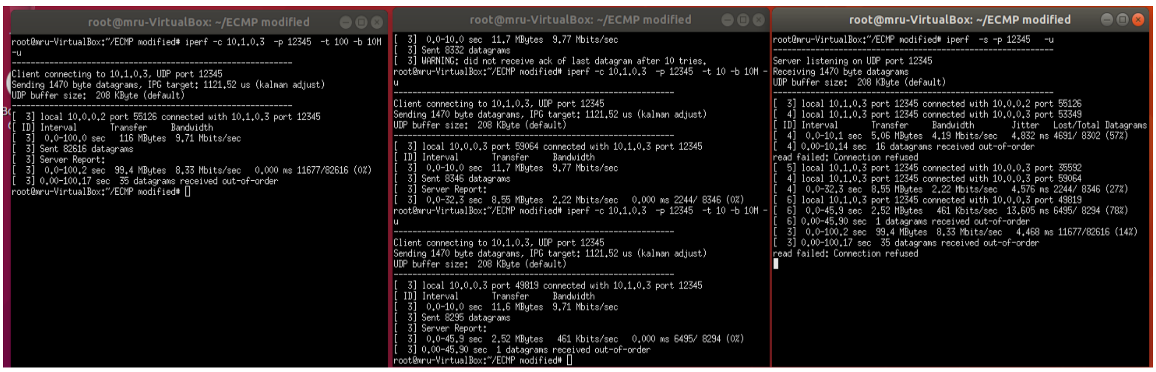

root@mru-VirtualBox: ~/ECMP org
root@mru-VirtualBox:~/ECMP org# iperf -c 10.1.0.3 -p 12345 -t 100 -b 10M -u
Client connecting to 10.1.0.3, UDP port 12345
Sending 1470 byte datagrams, IPG target: 1121.52 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[3] local 10.0.0.2 port 38730 connected with 10.1.0.3 port 12345
[ID] Interval Transfer Bandwidth
[3] 0.0-100.0 sec 115 MBytes 9.62 Mbits/sec
[3] Sent 81795 datagrams
[3] Server Report:
[3] 0.0-100.5 sec 97.8 MBytes 8.17 Mbits/sec 0.000 ms 12021/81794 (0%)
[3] 0.00-100.47 sec 14 datagrams received out-of-order
root@mru-VirtualBox: ~/ECMP org
root@mru-VirtualBox:~/ECMP org# iperf -c 10.1.0.3 -p 12345 -t 10 -b 10M -u
Client connecting to 10.1.0.3, UDP port 12345
Sending 1470 byte datagrams, IPG target: 1121.52 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[3] local 10.0.0.3 port 45492 connected with 10.1.0.3 port 12345
[ID] Interval Transfer Bandwidth
[3] 0.0-10.0 sec 11.5 MBytes 9.68 Mbits/sec
[3] Sent 8229 datagrams
[3] Server Report:
[3] 0.0-10.3 sec 9.99 MBytes 8.10 Mbits/sec 0.000 ms 1105/ 8228 (0%)
[3] 0.00-10.34 sec 35 datagrams received out-of-order
root@mru-VirtualBox:~/ECMP org# iperf -c 10.1.0.3 -p 12345 -t 10 -b 10M -u
Client connecting to 10.1.0.3, UDP port 12345
Sending 1470 byte datagrams, IPG target: 1121.52 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[3] local 10.0.0.3 port 53868 connected with 10.1.0.3 port 12345
[ID] Interval Transfer Bandwidth
[3] 0.0-10.0 sec 11.5 MBytes 9.65 Mbits/sec
[3] Sent 8213 datagrams
[3] Server Report:
[3] 0.0-10.4 sec 3.85 MBytes 3.11 Mbits/sec 0.000 ms 5463/ 8212 (0%)
[3] 0.00-10.38 sec 18 datagrams received out-of-order
root@mru-VirtualBox:~/ECMP org# iperf -c 10.1.0.3 -p 12345 -t 10 -b 10M -u
Client connecting to 10.1.0.3, UDP port 12345
Sending 1470 byte datagrams, IPG target: 1121.52 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[3] local 10.0.0.3 port 57858 connected with 10.1.0.3 port 12345
[ID] Interval Transfer Bandwidth
[3] 0.0-10.0 sec 11.5 MBytes 9.67 Mbits/sec
[3] Sent 8221 datagrams
[3] Server Report:
[3] 0.0-10.2 sec 11.0 MBytes 9.05 Mbits/sec 0.000 ms 397/ 8221 (0%)
[3] 0.00-10.17 sec 11 datagrams received out-of-order
root@mru-VirtualBox: ~/ECMP org
root@mru-VirtualBox:~/ECMP org# iperf -s -p 12345 -u
Server listening on UDP port 12345
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
[3] local 10.1.0.3 port 12345 connected with 10.0.0.2 port 38730
[4] local 10.1.0.3 port 12345 connected with 10.0.0.3 port 45492
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[4] 0.0-10.3 sec 9.99 MBytes 8.10 Mbits/sec 20.155 ms 1105/ 8228 (13%)
[4] 0.00-10.34 sec 35 datagrams received out-of-order
[5] local 10.1.0.3 port 12345 connected with 10.0.0.3 port 53868
[5] 0.0-10.4 sec 3.85 MBytes 3.11 Mbits/sec 23.824 ms 5463/ 8212 (67%)
[5] 0.00-10.38 sec 18 datagrams received out-of-order
[4] local 10.1.0.3 port 12345 connected with 10.0.0.3 port 57858
[3] 0.0-100.5 sec 97.8 MBytes 8.17 Mbits/sec 22.266 ms 12021/81794 (15%)
[3] 0.00-100.47 sec 14 datagrams received out-of-order
[4] 0.0-10.2 sec 11.0 MBytes 9.05 Mbits/sec 4.561 ms 397/ 8221 (4.8%)
[4] 0.00-10.17 sec 11 datagrams received out-of-order

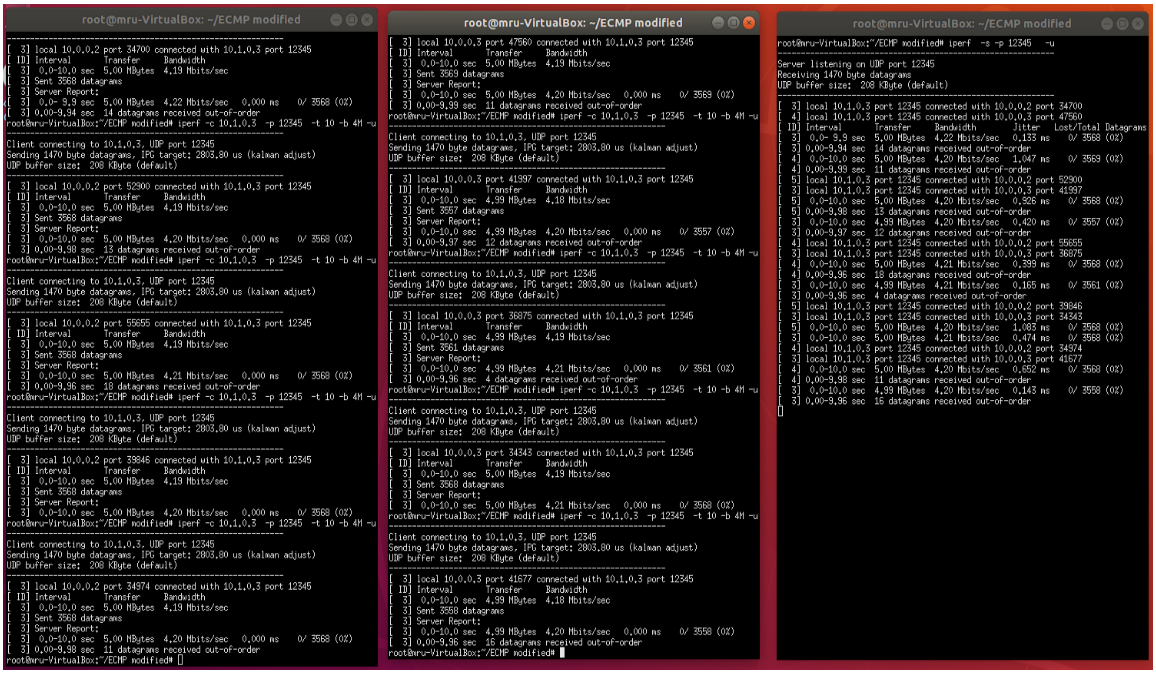

root@mru-VirtualBox: ~/ECMP org
root@mru-VirtualBox:~/ECMP org# iperf -c 10.1.0.3 -p 12345 -t 100 -b 4M -u
Client connecting to 10.1.0.3, UDP port 12345
Sending 1470 byte datagrams, IPG target: 2803.80 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[3] local 10.0.0.2 port 35709 connected with 10.1.0.3 port 12345
[ID] Interval Transfer Bandwidth
[3] 0.0-100.0 sec 50.0 MBytes 4.19 Mbits/sec
[3] Sent 35667 datagrams
[3] Server Report:
[3] 0.0-100.0 sec 50.0 MBytes 4.20 Mbits/sec 0.000 ms 0/35667 (0%)
[3] 0.00-99.95 sec 6 datagrams received out-of-order
root@mru-VirtualBox: ~/ECMP org
root@mru-VirtualBox:~/ECMP org# iperf -c 10.1.0.3 -p 12345 -t 10 -b 4M -u
Client connecting to 10.1.0.3, UDP port 12345
Sending 1470 byte datagrams, IPG target: 2803.80 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[3] local 10.0.0.3 port 43075 connected with 10.1.0.3 port 12345
[ID] Interval Transfer Bandwidth
[3] 0.0-10.0 sec 5.00 MBytes 4.19 Mbits/sec
[3] Sent 3568 datagrams
[3] Server Report:
[3] 0.0-10.0 sec 5.00 MBytes 4.21 Mbits/sec 0.000 ms 0/ 3568 (0%)
[3] 0.00-9.98 sec 7 datagrams received out-of-order
root@mru-VirtualBox:~/ECMP org# iperf -c 10.1.0.3 -p 12345 -t 10 -b 4M -u
Client connecting to 10.1.0.3, UDP port 12345
Sending 1470 byte datagrams, IPG target: 2803.80 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[3] local 10.0.0.3 port 36914 connected with 10.1.0.3 port 12345
[ID] Interval Transfer Bandwidth
[3] 0.0-10.0 sec 5.00 MBytes 4.19 Mbits/sec
[3] Sent 3568 datagrams
[3] Server Report:
[3] 0.0-10.0 sec 5.00 MBytes 4.21 Mbits/sec 0.000 ms 0/ 3568 (0%)
[3] 0.00-9.97 sec 3 datagrams received out-of-order
root@mru-VirtualBox:~/ECMP org# iperf -c 10.1.0.3 -p 12345 -t 10 -b 4M -u
Client connecting to 10.1.0.3, UDP port 12345
Sending 1470 byte datagrams, IPG target: 2803.80 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[3] local 10.0.0.3 port 52278 connected with 10.1.0.3 port 12345
[ID] Interval Transfer Bandwidth
[3] 0.0-10.0 sec 5.00 MBytes 4.19 Mbits/sec
[3] Sent 3568 datagrams
[3] Server Report:
[3] 0.0- 9.9 sec 5.00 MBytes 4.22 Mbits/sec 0.000 ms 0/ 3568 (0%)
[3] 0.00-9.95 sec 4 datagrams received out-of-order
root@mru-VirtualBox: ~/ECMP org
root@mru-VirtualBox:~/ECMP org# iperf -s -p 12345 -u
Server listening on UDP port 12345
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
[3] local 10.1.0.3 port 12345 connected with 10.0.0.2 port 35709
[4] local 10.1.0.3 port 12345 connected with 10.0.0.3 port 43075
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[4] 0.0-10.0 sec 5.00 MBytes 4.21 Mbits/sec 0.023 ms 0/ 3568 (0%)
[4] 0.00-9.98 sec 7 datagrams received out-of-order
[5] local 10.1.0.3 port 12345 connected with 10.0.0.3 port 36914
[5] 0.0-10.0 sec 5.00 MBytes 4.21 Mbits/sec 0.977 ms 0/ 3568 (0%)
[5] 0.00-9.97 sec 3 datagrams received out-of-order
[4] local 10.1.0.3 port 12345 connected with 10.0.0.3 port 52278
[4] 0.0- 9.9 sec 5.00 MBytes 4.22 Mbits/sec 0.023 ms 0/ 3568 (0%)
[4] 0.00-9.95 sec 4 datagrams received out-of-order
[3] 0.0-100.0 sec 50.0 MBytes 4.20 Mbits/sec 0.013 ms 0/35667 (0%)
[3] 0.00-99.95 sec 6 datagrams received out-of-order

LUP(least Utilized Path)



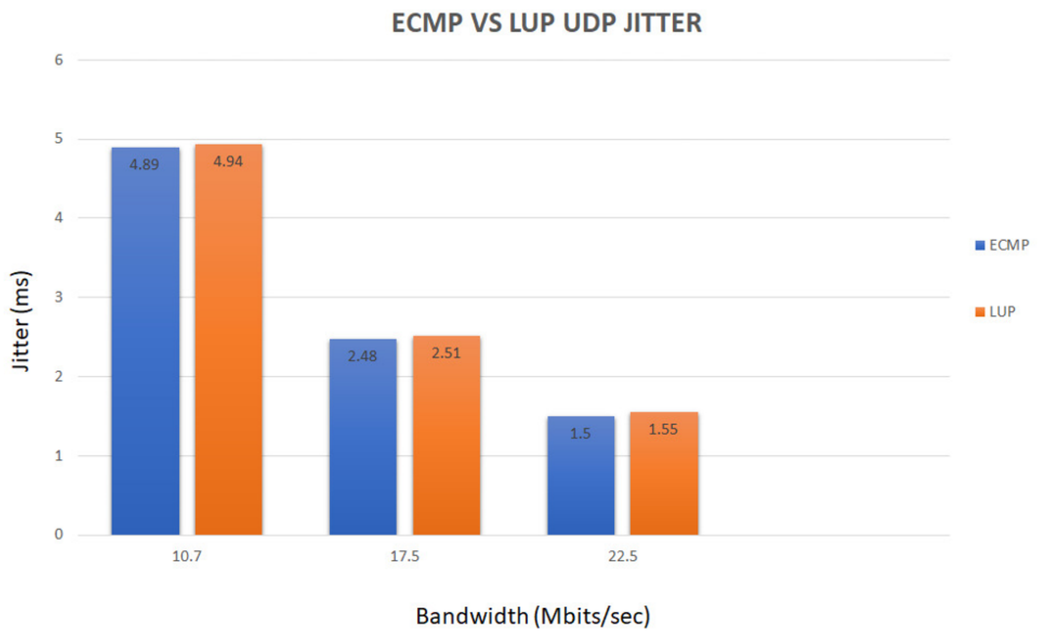





root@mru-VirtualBox: ~/ECMP modified
root@mru-VirtualBox: ~/ECMP modified
root@mru-VirtualBox: ~/ECMP modified

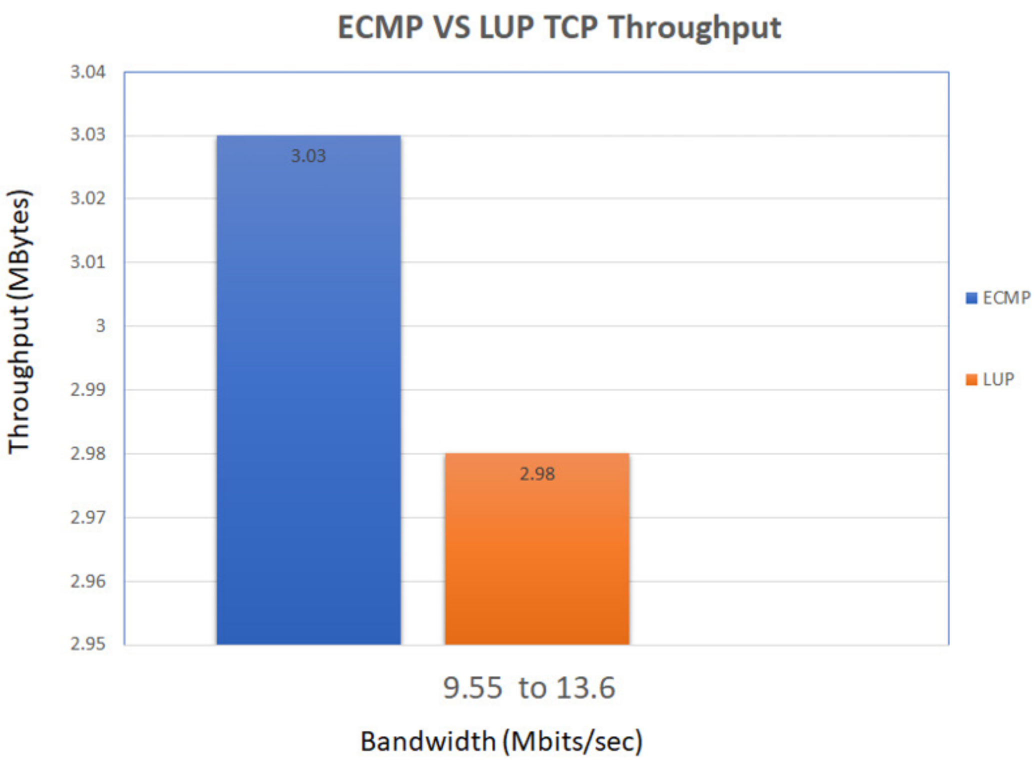
Results

a) Jitter



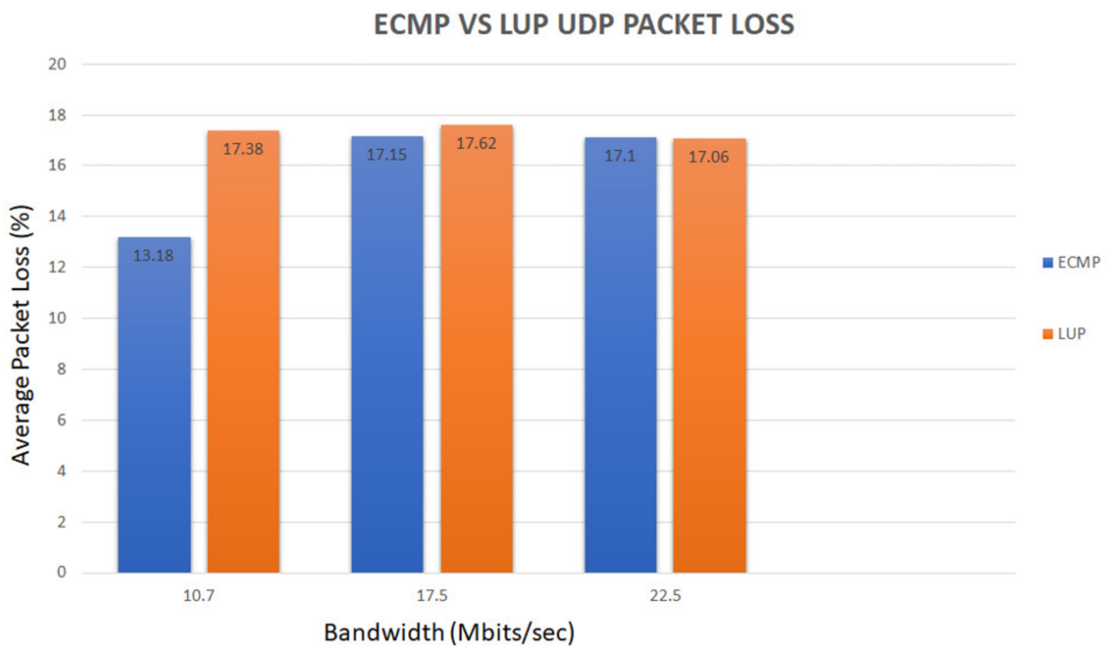
Jitter is defined as the variation in delay. As can be seen from the above graphs that in terms of jitter the performance of ECMP and Least Utilized Path is approximately same for three different bandwidths of 10.7 Mbps, 17.5 Mbps and 22.5 Mbps.

b) Throughput



Throughput is defined as the amount of data that is sent successfully per unit time. As we can see from the above graphs that for the bandwidth in the range of 9.55 to 13.6 Mbps the throughput of ECMP is better than the Least Utilized Path.

c) Packet Loss



Packet Loss is given in percentage which refers that in UDP connection the number of datagrams which are lost out of the total number of datagrams which are sent from sender to receiver. From the above graph it can be seen that although ECMP performs slightly better than UDP but still we can consider their performance to be approximately same in terms of packet loss.