

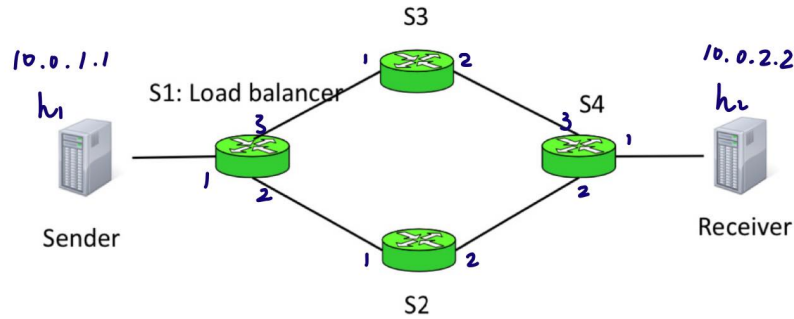
COMP436 HW2

Paul Gao pg22 S01271705

File descriptions and instructions to run the program can be found in README.

Milestone 1

Task 1



This is the topology graph for my network. We have 2 hosts and 4 switches. Sender host is h1 (10.0.1.1). Receiver host is h2 (10.0.2.2). Switches and their ports are shown in the graph as well.

Task 2

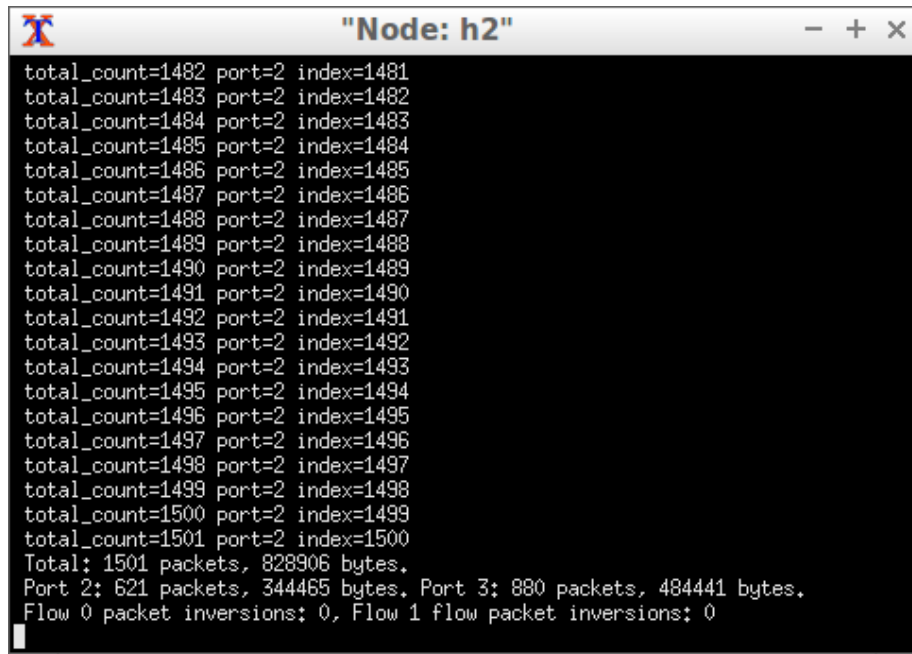
Applying ECMP load balancing results in the packets to be sent out through port 2 or 3 of Switch 1. A packet is tagged with a special IP destination 0.0.0.0 to notify the switch that it is on the first hop. Then, the switch set the destination to 10.0.2.2 and send it to the rest of the switches to do destination-based forwarding.

Task 3

I added a new 'query' protocol, which is send after all of the packets are processed and collects statistics from S1. The statistics are stored in registers in the switch.

Milestone 2

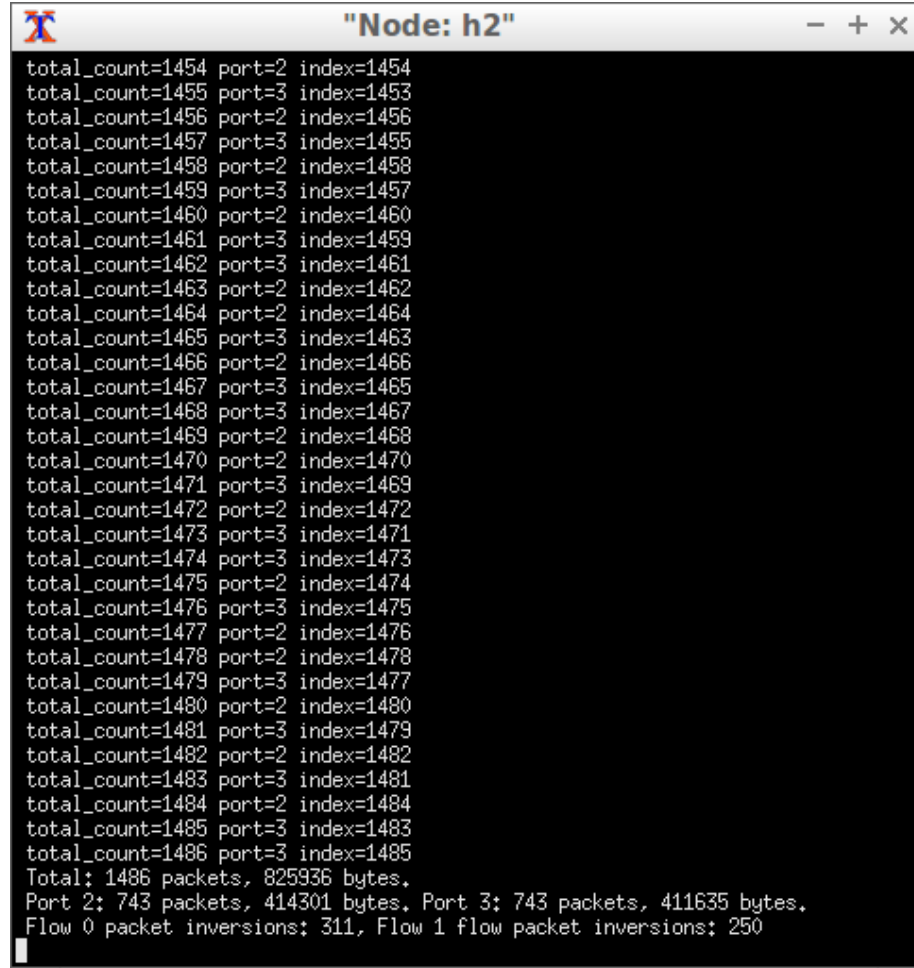
Task 1



```
total_count=1482 port=2 index=1481
total_count=1483 port=2 index=1482
total_count=1484 port=2 index=1483
total_count=1485 port=2 index=1484
total_count=1486 port=2 index=1485
total_count=1487 port=2 index=1486
total_count=1488 port=2 index=1487
total_count=1489 port=2 index=1488
total_count=1490 port=2 index=1489
total_count=1491 port=2 index=1490
total_count=1492 port=2 index=1491
total_count=1493 port=2 index=1492
total_count=1494 port=2 index=1493
total_count=1495 port=2 index=1494
total_count=1496 port=2 index=1495
total_count=1497 port=2 index=1496
total_count=1498 port=2 index=1497
total_count=1499 port=2 index=1498
total_count=1500 port=2 index=1499
total_count=1501 port=2 index=1500
Total: 1501 packets, 828906 bytes.
Port 2: 621 packets, 344465 bytes, Port 3: 880 packets, 484441 bytes.
Flow 0 packet inversions: 0, Flow 1 flow packet inversions: 0
```

I generated random flows as defined by the five tuples, with different sizes. Specifically, I generated 100 flows with 10-20 packets randomly for each flow, and each packet is of a random size. ECMP load balancing produces the result above. "Port 2" and "Port 3" are port 2 and 3 of Switch 1 of which the packets came out S1 from. Since packets of the same flow are sent to the same port of S1, we can see that the packets numbers and the traffic are not balanced very well. However, there are no out-of-order packets as expected.

Task 2



```
total_count=1454 port=2 index=1454
total_count=1455 port=3 index=1453
total_count=1456 port=2 index=1456
total_count=1457 port=3 index=1455
total_count=1458 port=2 index=1458
total_count=1459 port=3 index=1457
total_count=1460 port=2 index=1460
total_count=1461 port=3 index=1459
total_count=1462 port=3 index=1461
total_count=1463 port=2 index=1462
total_count=1464 port=2 index=1464
total_count=1465 port=3 index=1463
total_count=1466 port=2 index=1466
total_count=1467 port=3 index=1465
total_count=1468 port=3 index=1467
total_count=1469 port=2 index=1468
total_count=1470 port=2 index=1470
total_count=1471 port=3 index=1469
total_count=1472 port=2 index=1472
total_count=1473 port=3 index=1471
total_count=1474 port=3 index=1473
total_count=1475 port=2 index=1474
total_count=1476 port=3 index=1475
total_count=1477 port=2 index=1476
total_count=1478 port=2 index=1478
total_count=1479 port=3 index=1477
total_count=1480 port=2 index=1480
total_count=1481 port=3 index=1479
total_count=1482 port=2 index=1482
total_count=1483 port=3 index=1481
total_count=1484 port=2 index=1484
total_count=1485 port=3 index=1483
total_count=1486 port=3 index=1485
Total: 1486 packets, 825936 bytes.
Port 2: 743 packets, 414301 bytes, Port 3: 743 packets, 411635 bytes.
Flow 0 packet inversions: 311, Flow 1 flow packet inversions: 250
```

Similarly, I generated 100 flows with 10-20 packets randomly for each flow, and each packet is of a random size. With per-packet load balancing (according to packet numbers per port), it produces the result above. We can see that the packet numbers are evenly distributed between the two ports. The traffic is also balanced better. However, we can observe out-of-order packets.

Task 3

I used inversions to quantify the out-of-order packets. In a list of packets a , $a[i]$ and $a[j]$ form an inversion if $a[i] > a[j]$ and $i < j$.

In my receiver output, index is the original order that the packets are sent. We can see that ECMP LB preserved the order, but per-packet LB did not. From the result image in Task 2, we can see the total packet inversions for each flow. I configured the link latencies to observe the inversions. Specifically, S1-S2 has

a latency of 1ms, and S1-S3 has a latency of 50ms.