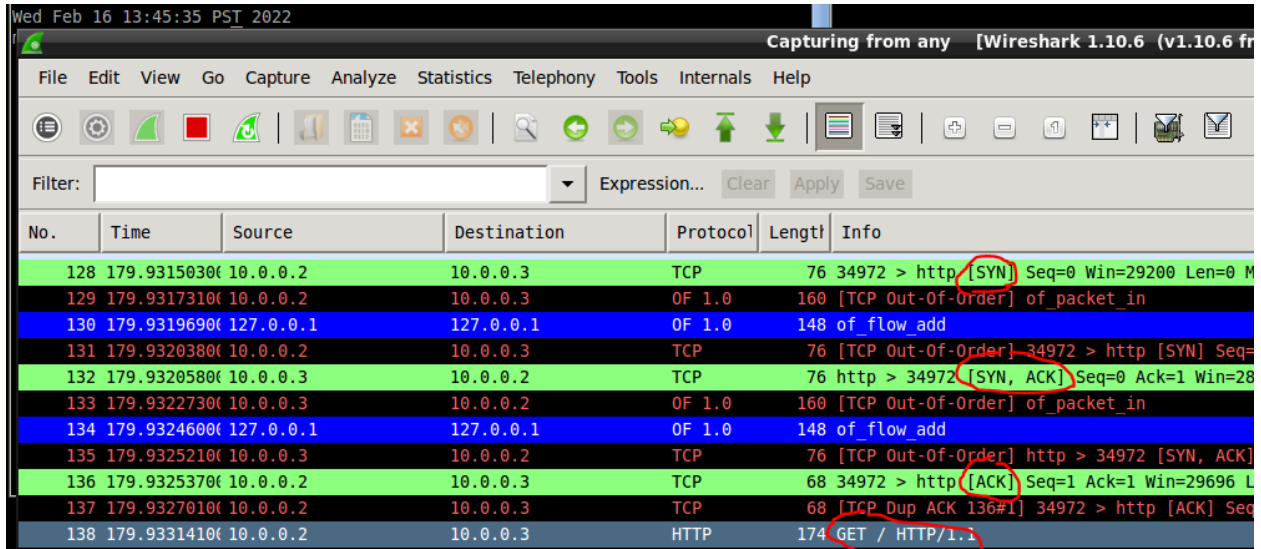


## .HTTP Request-Response Generation

1) [10pts] Include a screenshot showing the Wireshark transaction between the client and server. Do not use display filters and show all protocols you see in your screenshot.



Wed Feb 16 13:45:35 PST 2022

Capturing from any [Wireshark 1.10.6 (v1.10.6 fr

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
128	179.931503000	10.0.0.2	10.0.0.3	TCP	76	34972 > http [SYN] Seq=0 Win=29200 Len=0 M
129	179.931731000	10.0.0.2	10.0.0.3	OF 1.0	160	[TCP Out-Of-Order] of_packet_in
130	179.931969000	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
131	179.932038000	10.0.0.2	10.0.0.3	TCP	76	[TCP Out-Of-Order] 34972 > http [SYN] Seq=
132	179.932058000	10.0.0.3	10.0.0.2	TCP	76	http > 34972 [SYN, ACK] Seq=0 Ack=1 Win=28
133	179.932273000	10.0.0.3	10.0.0.2	OF 1.0	160	[TCP Out-Of-Order] of_packet_in
134	179.932460000	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
135	179.932521000	10.0.0.3	10.0.0.2	TCP	76	[TCP Out-Of-Order] http > 34972 [SYN, ACK]
136	179.932537000	10.0.0.2	10.0.0.3	TCP	68	34972 > http [ACK] Seq=1 Ack=1 Win=29696 L
137	179.932701000	10.0.0.2	10.0.0.3	TCP	68	[TCP Dup ACK 136#1] 34972 > http [ACK] Seq
138	179.933141000	10.0.0.2	10.0.0.3	HTTP	174	GET / HTTP/1.1

2) [20pts] Detail all the steps taken by the client to retrieve the file referencing the packets in your screenshot. Reference specific packets numbers from your screenshot or annotate your screenshot in support of your answer.

In lines 128,132 and 136 we see the TCP three point handshake and then on line 138 we see the HTTP GET request. The MAC addresses are also found using the ARP protocol.

## Load Balancer

1) [10pts] Include a screenshot of the output from the load balancer.

```
Fri Feb 11 11:47:21 PST 2022
mininet@mininet-vm: ~
File Edit Tabs Help
INFO:openflow.of_01:[00-00-00-00-00-01 1] disconnected
INFO:core:Down.
mininet@mininet-vm:~$ ~/pox/pox.py log.level --DEBUG misc.ip_loadbalancer --ip=10.0.0.10 --servers=10.0.0.1,10.0.0.2
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.2.0 (carp) going up...
DEBUG:core:Running on CPython (2.7.6/Nov 23 2017 15:50:55)
DEBUG:core:Platform is Linux-3.13.0-106-generic-i686-athlon-with-Ubuntu-14.04-trusty
INFO:core:POX 0.2.0 (carp) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:iplb:IP Load Balancer Ready.
INFO:iplb:Load Balancing on [00-00-00-00-00-01 1]
INFO:iplb.00-00-00-00-00-00-01:Server 10.0.0.1 up
DEBUG:iplb.00-00-00-00-00-00-01:Directing traffic to 10.0.0.1
INFO:iplb.00-00-00-00-00-00-01:Server 10.0.0.2 up
DEBUG:iplb.00-00-00-00-00-00-01:Directing traffic to 10.0.0.2
DEBUG:iplb.00-00-00-00-00-00-01:Directing traffic to 10.0.0.2
DEBUG:iplb.00-00-00-00-00-00-01:Directing traffic to 10.0.0.1
DEBUG:iplb.00-00-00-00-00-00-01:Directing traffic to 10.0.0.2
DEBUG:iplb.00-00-00-00-00-00-01:Directing traffic to 10.0.0.2
DEBUG:iplb.00-00-00-00-00-00-01:Directing traffic to 10.0.0.2
```

**2) [10pts] In your own words, describe the behavior of the load balancer.**

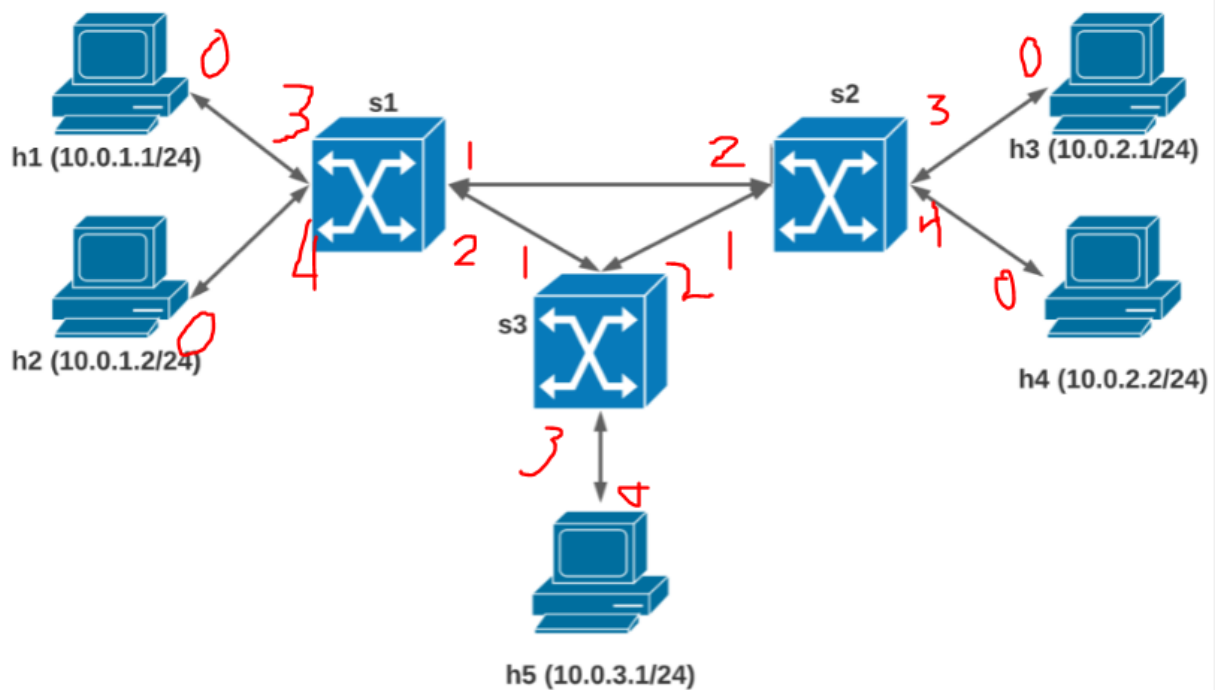
The load balancer is directing traffic to the website hosted by host 1 (10.0.0.1) and host 2 (10.0.0.2) so that neither server becomes overwhelmed. In the above output we see that most of the traffic is going to 10.0.0.2. I think this is because the traffic is light and that the load balancer has determined that it can keep directing traffic to 10.0.0.2 without it becoming overwhelmed. I would anticipate that if we simulated larger amounts of traffic that the load balancer would start diverting more traffic towards host 1 (10.0.0.1)

**3) [10pts] How might a load balancer increase the performance of a content delivery system serving a large number of users?**

A load balancer might reduce delay for users requesting content by more efficiently using the resources of a network. By evenly distributing traffic between servers hosting the same content, we can ensure that limited resources like the queues within the routers and servers that limit how quickly users can access content are being used efficiently and that one server isn't getting lots of traffic while another is getting none.

**Router**

**1) [10pts] Implement the topology above from the skeleton file. Annotate the figure with the ports associated with each link.**



2) [10pts] Verify Rule 1 with pingall. Are the results what you expect and why? Include a screenshot.

```
*** Ping: testing ping reachability
h1 -> h2 X X h5
h2 -> h1 X X h5
h3 -> X X X X
h4 -> X X X X
h5 -> h1 h2 X X
*** Results: 70% dropped (6/20 received)
mininet>
mininet@mininet-vm:~$ date
Wed Feb 16 12:55:28 PST 2022
```

The results are as expected. Since subnets 10.0.1 and 10.0.2 are able to send and receive ICMP packets it makes sense that all packets that are not between these subnets would be dropped.

3) [10pts] Verify Rule 2 with iperf. Are the results what you expect and why? Include a screenshot of an iperf between nodes belonging to each pair of subnets (5 pairs total).

```

Wed Feb 16 13:00:08 PST 2022
mininet@mininet-vm:~$ sudo python ~/lab3_topo_skel.py
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['47.3 Gbits/sec', '47.3 Gbits/sec']
mininet> iperf h3 h4
*** Iperf: testing TCP bandwidth between h3 and h4
*** Results: ['56.0 Gbits/sec', '56.1 Gbits/sec']
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['16.7 Gbits/sec', '16.7 Gbits/sec']
mininet> iperf h3 h1
*** Iperf: testing TCP bandwidth between h3 and h1
*** Results: ['16.5 Gbits/sec', '16.5 Gbits/sec']
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
^C
Interrupt
mininet> iperf h3 h5
*** Iperf: testing TCP bandwidth between h3 and h5
^C
Interrupt
mininet> iperf h5 h1
*** Iperf: testing TCP bandwidth between h5 and h1
^C
Interrupt
mininet> iperf h5 h3
*** Iperf: testing TCP bandwidth between h5 and h3
^C
Interrupt

```

The results are as expected. Subnet 10.0.1 and 10.0.2 are allowed to send TCP packets to each other so h1 to h3 and h3 to h1 working makes sense. Hosts within a subnet are also allowed to send TCP packets to each other so h1 to h2 and h3 to h4 working makes sense. No TCP packets are allowed to be sent to or from subnet 10.0.3 so the h1 h5 connection and the h5 h3 connection not working makes sense.

**4) [10pts] The topology skeleton has been configured to use static ARP. This means that hosts have cached IP-to-MAC mappings for all other hosts. What would happen if we instead flooded ARP? Why?**

ARP does not include authentication mechanisms to validate ARP messages so any device in a network, regardless of what is allowed by our `do_router` function would be able to answer an ARP request. This would make our network insecure because if for example h1 wants the mac address of h3 an attacker could answer with the wrong mac address of h3.

