Project 1: Part 2

Bradly T Booth

Student ID: 919346279

non_persistent_http_BradlyTBooth_919346279.py

persistent_http_BradlyTBooth_919346279.py

20220508

Terminal Output:

```
brad@betsyU:~/ECS_152a/images$ python3 non_persistent_http_BradlyTBooth_919346279.py
******************************************************
HTTP Client Version: Non Persistent HTTP
Total Page Load Time:     19.88148784637451 seconds
Average Request Delay:    0.05843038333904075 seconds
Above the Fold Load Time: 0.3381638526916504 seconds
Requests per Second:      17.114383696535526
******************************************************
brad@betsyU:~/ECS_152a/images$ python3 persisten_http_BradlyTBooth_919346279.py
******************************************************
HTTP Client Version: Persistent HTTP
Total Page Load Time: 16.225026607513428 seconds
Average Request Delay: 0.04754911442480144 seconds
Above the Fold Load Time: 0.25310516357421875 seconds
Request per Second:       21.03088589760157 seconds
******************************************************
brad@betsyU:~/ECS_152a/images$
```

Question 1:

a) Non-Persistent: Used 339 sockets. 1 for the HTML file and 338, 1 for each image.

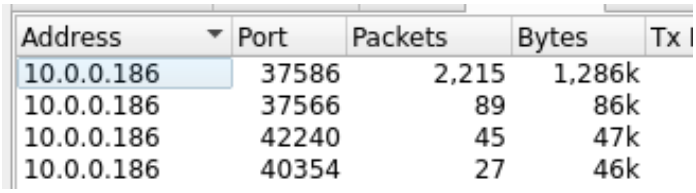
Persistent: Used 4 sockets. 1 socket for server 173.230.149.18. Since, this was a persistent connection, the socket didn't need to close. 3 sockets, 1 each for the sockets hosted on external servers.

b) Non-Persistent: I didn't bind the sockets to any port, so the program automatically determined port numbers. For each TCP connection, the program had the capability to use a different port. Then the number of ports I used was up to 339 ports. One for each connection made.

Using WireShark statistics, there indeed a great many number of ports used during the nonpersistent connection. Too many to count at the very least.

Persistent: I used 4 ports. 1 for server we made the initial GET request to. This connection stayed open while the program was running, so the port never changed. Then 1 socket for each image from the 3 external servers.

WireShark Image of Persistent Ports Used:

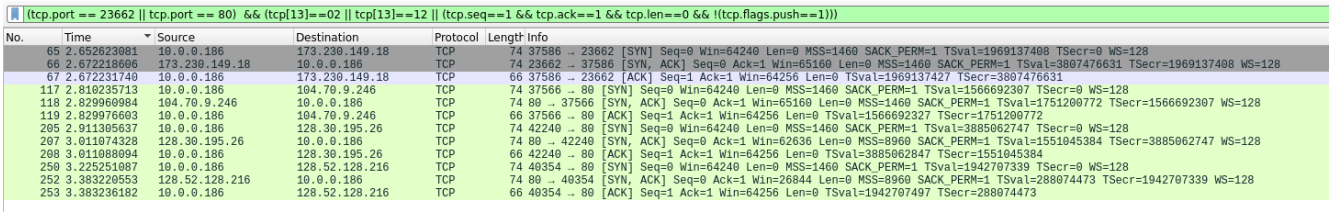


Address	Port	Packets	Bytes	Tx
10.0.0.186	37586	2,215	1,286k	
10.0.0.186	37566	89	86k	
10.0.0.186	42240	45	47k	
10.0.0.186	40354	27	46k	

Question 2:

a) The program will connect to 4 servers.

WireShark:

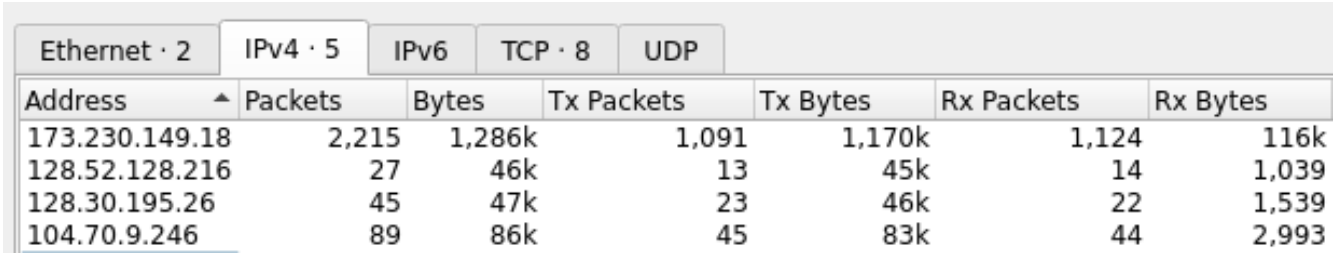


(tcp.port == 23662 || tcp.port == 80) && (tcp[13]==02 || tcp[13]==12 || (tcp.seq==1 && tcp.ack==1 && tcp.len==0 && !(tcp.flags.push==1)))

No.	Time	Source	Destination	Protocol	Length	Info
65	2.652623081	10.0.0.186	173.230.149.18	TCP	74	37586 → 23662 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1969137408 TSecr=0 WS=128
66	2.672218606	173.230.149.18	10.0.0.186	TCP	74	23662 → 37586 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3807476631 TSecr=1969137408 WS=128
67	2.672231740	10.0.0.186	173.230.149.18	TCP	66	37586 → 23662 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1969137427 TSecr=3807476631
117	2.810235713	10.0.0.186	104.70.9.246	TCP	74	37566 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1566692307 TSecr=0 WS=128
118	2.829960984	104.70.9.246	10.0.0.186	TCP	74	80 → 37566 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1751200772 TSecr=1566692307 WS=128
119	2.829976603	10.0.0.186	104.70.9.246	TCP	66	37566 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1566692327 TSecr=1751200772
205	2.911305637	10.0.0.186	128.30.195.26	TCP	74	42240 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3885062747 TSecr=0 WS=128
207	3.011074328	128.30.195.26	10.0.0.186	TCP	74	80 → 42240 [SYN, ACK] Seq=0 Ack=1 Win=62636 Len=0 MSS=8960 SACK_PERM=1 TSval=1551045384 TSecr=3885062747 WS=128
208	3.011088094	10.0.0.186	128.30.195.26	TCP	66	42240 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3885062847 TSecr=1551045384
250	3.225251087	10.0.0.186	128.52.128.216	TCP	74	40354 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1942707339 TSecr=0 WS=128
252	3.383220553	128.52.128.216	10.0.0.186	TCP	74	80 → 40354 [SYN, ACK] Seq=0 Ack=1 Win=26844 Len=0 MSS=8960 SACK_PERM=1 TSval=288074473 TSecr=1942707339 WS=128
253	3.383236182	10.0.0.186	128.52.128.216	TCP	66	40354 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1942707497 TSecr=288074473

Above is a screen shot of the 3-way handshake to each of the 4 servers. This of course is in persistent http. In non-persistent there would be a lot more than 4 3-way handshakes.

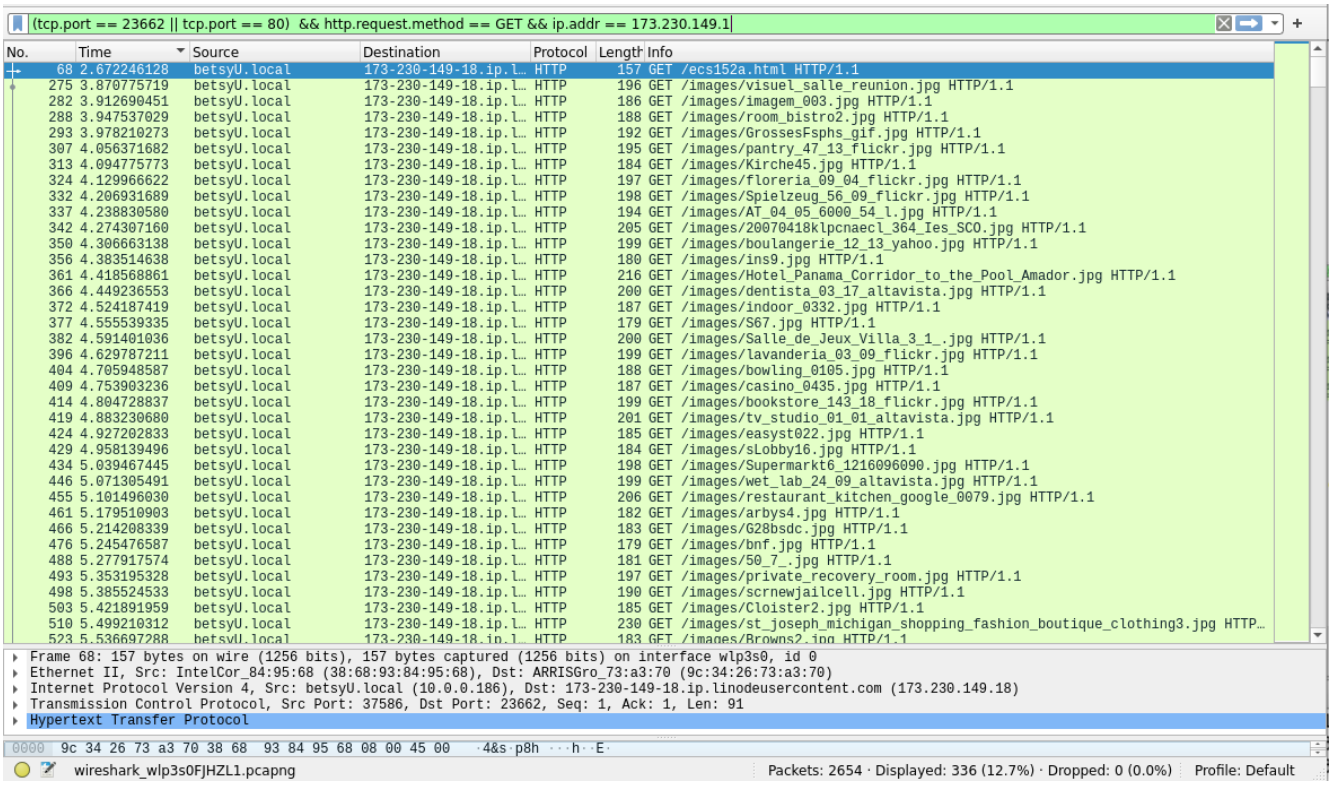
b) 4 Servers:
Host:



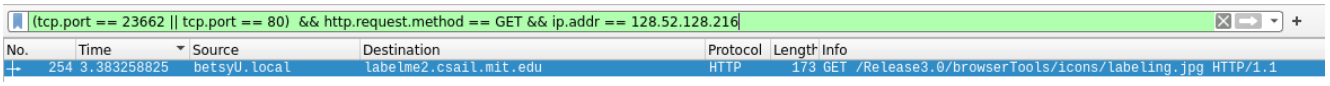
Ethernet · 2 | IPv4 · 5 | IPv6 | TCP · 8 | UDP

Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
173.230.149.18	2,215	1,286k	1,091	1,170k	1,124	116k
128.52.128.216	27	46k	13	45k	14	1,039
128.30.195.26	45	47k	23	46k	22	1,539
104.70.9.246	89	86k	45	83k	44	2,993

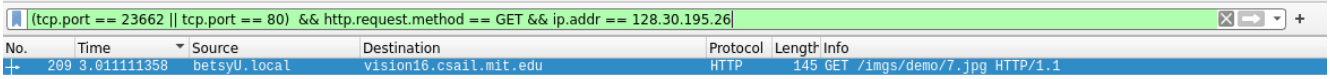
Host: 173-230-149-18.ip.linodeusercontent.com, IPv4 = 173.230.149.18,
# of GET = 336



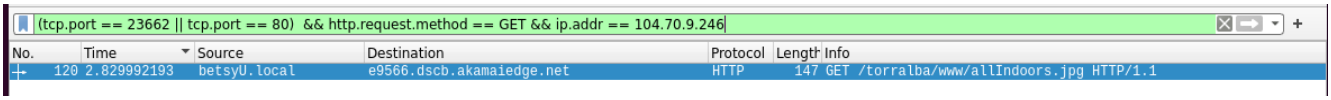
Host: labelme2.csail.mit.edu, IPv4: 128.52.128.216, # of GET = 1



Host: vision16.csail.mit.edu, IPv4: 128.30.195.26, # of GET = 1



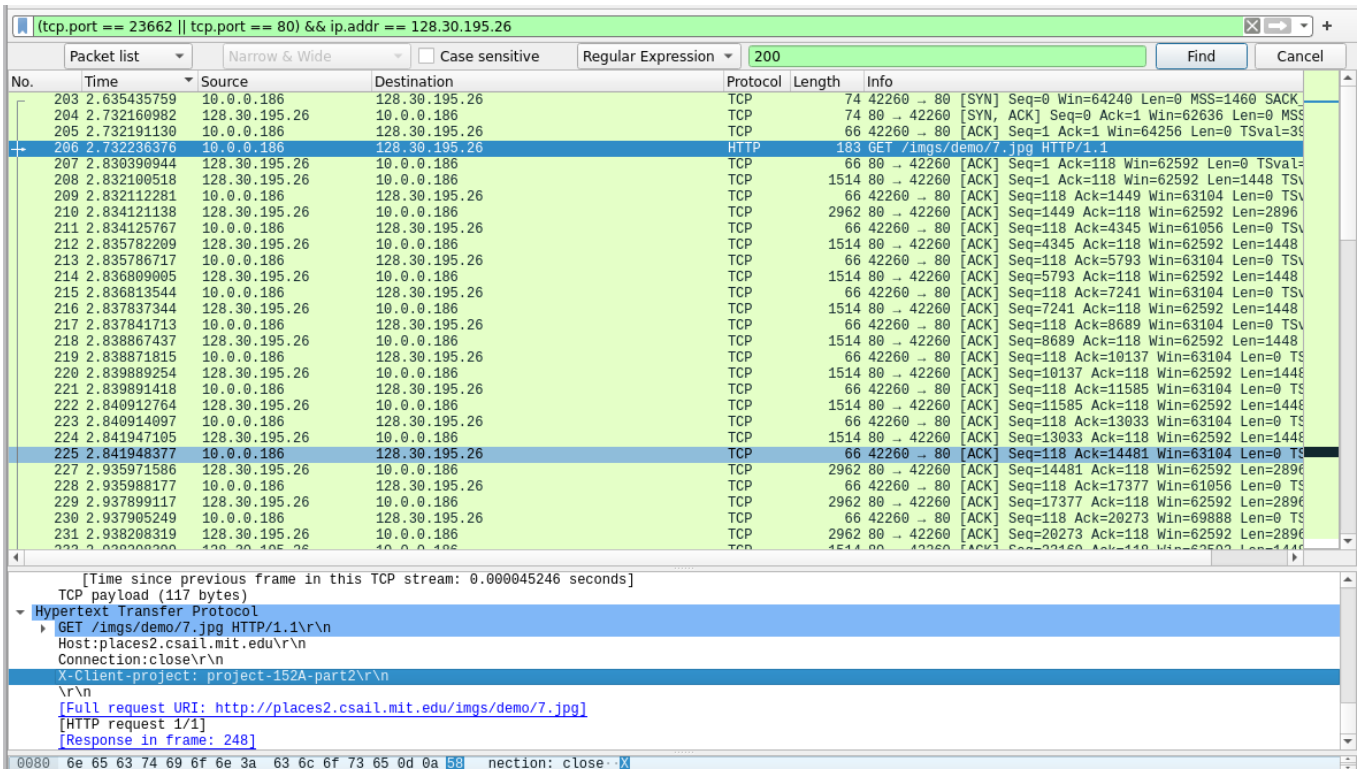
Host: e9566.dscb.akamaiedge.net, IPv4: 104.70.9.246 # of GET = 1



c) Total Number of Images Downloaded: 338

d) 1.2 MB

e) Nothing happens. The header gets ignored.



The highlighted packet is a GET request sent to 128.30.195.26 with the special header in the request. The header is highlighted. The GET was accepted, and the image was sent in response. I could not find the status code, but I assume it was 200 since the image was sent.

Question 3

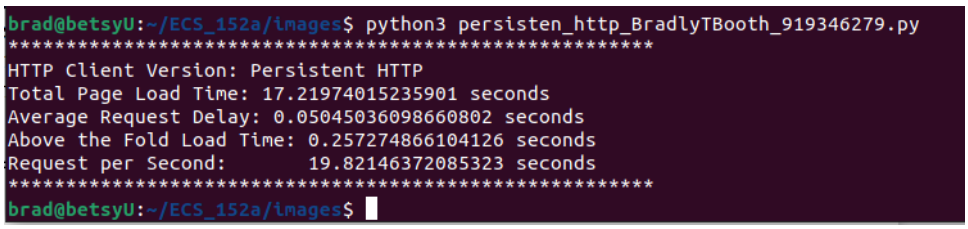
a) Above the fold load time was .34 seconds for non-persistent and .25 seconds for persistent connections. Resources needed to load this were the HTML file and allindoors.jpg. Justification for why only these 2 resources is in screenshot below.

b) Total Page Load Time for non-persistent was roughly 20 seconds, whereas for persistent it was roughly 16 seconds. Non-persistent has a higher PLT. This is because it had to reconnect to the server every time it needed to request an image. One way to improve the total page load time, could be to have multiple threads opening multiple sockets making GET request. For both flavors of HTTP, this could drastically increase the performance.

c) Well, my persistent HTTP program gets stuck when removing the connection header. It does download the html file but then doesn't do anything after. However, removing connection from GET requests in my non-persistent program, it works fine. But, this doesn't help me. I have connection closed statement after each GET request. I also tried printing out the header, and the header does not return connection header when I give none. Based on this information, I am going to

assume that the connection is default to HTTP/1.1 which is a persistent connection.

e)



Changing the buffer size to 256 did increase the page load time, but not by much. From this application, I would argue that a buffer of 4096 is better. In practice, I am not sure which is better.