📦 Curl! license MIT

A simple program that replicates Curl

This project is a simple command line program which fetches HTML pages based on the urls provided by users. The program uses the socket library to process low level http requests to webservers.

What you need

You will need python3 installed and the `socket` python library installed as dependecies.

Building this project

Once all dependecies are installed, you will also need the files:

- HTTPoutput.html
- Log.csv

To run the program:

```
$ python3 sicarbonMyCurl [-h] url [hostname]
```

Documentation:

```
Curl Replication Program

-------------------------------

 A Curl Replication for

 users to request HTML

 documents from hosts.



positional arguments:

 url http://hostname[ip]:[port]/[args]

 hostname Optional hostname argument
```

```
optional arguments:

 -h, --help show this help message and exit

 -l List Header Information.
```

❗ Unsupported

- HTTPS
- Chunk Encoding
- Redirection

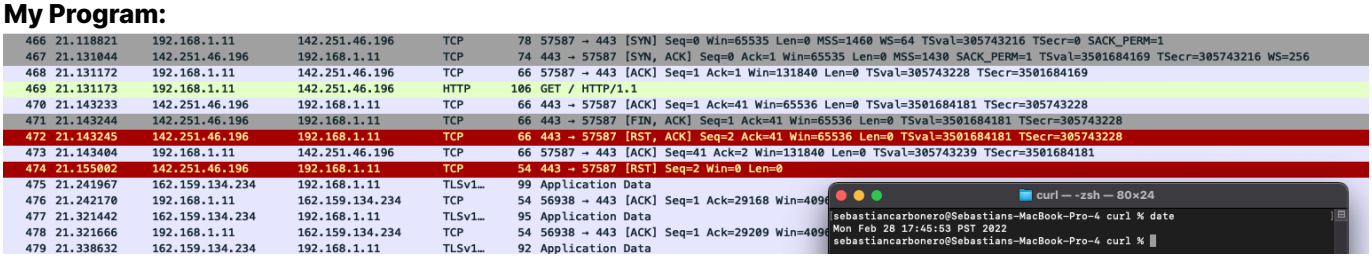
📋 Log.csv

Unsuccessful and Successful URLs

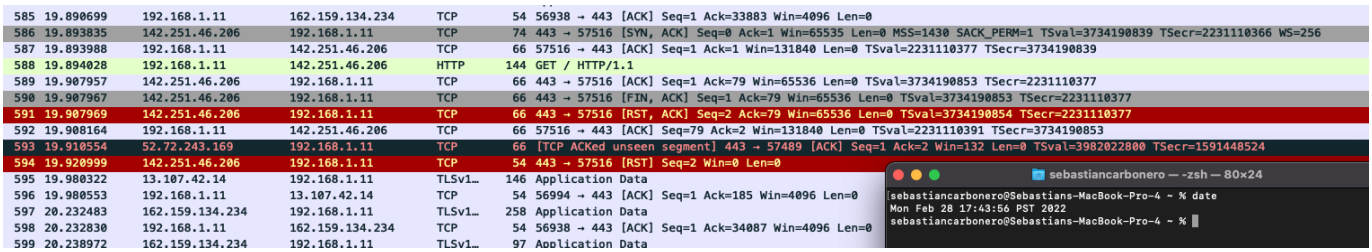
```
Unsuccessful, 54, http://www.google.com:443, www.google.com, 192.168.1.11, 142.251.46.228, 50817, 443, [Errno 54] Connection reset by peer

Unsuccessful, 301, http://isebas.us:80, isebas.us, 192.168.1.11, 149.248.9.165, 50931, 80, HTTP/1.1 301 Moved Permanently

Success, 200, http://example.com, example.com, 192.168.1.11, 93.184.216.34, 50963, 80, HTTP/1.1 200 OK

Success, 200, http://pudim.com.br/, pudim.com.br, 192.168.1.11, 54.207.20.104, 51122, 80, HTTP/1.1 200 OK

Unsuccessful, 404, http://www.example.com/anyname.html, www.example.com, 192.168.1.11, 93.184.216.34, 51143, 80, HTTP/1.1 404 Not Found
```

Google.com with Port 443

My Program:



Curl:



585	19.890699	192.168.1.11	162.159.134.234	TCP	54	56938 → 443 [ACK] Seq=1 Ack=33883 Win=4096 Len=0
586	19.893835	142.251.46.206	192.168.1.11	TCP	74	443 → 57516 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 TSval=3734190839 TSecr=2231110366 WS=256
587	19.893988	192.168.1.11	142.251.46.206	TCP	66	57516 → 443 [ACK] Seq=1 Ack=1 Win=131840 Len=0 TSval=2231110377 TSecr=3734190839
588	19.894028	192.168.1.11	142.251.46.206	HTTP	144	GET / HTTP/1.1
589	19.907957	142.251.46.206	192.168.1.11	TCP	66	443 → 57516 [ACK] Seq=1 Ack=79 Win=65536 Len=0 TSval=3734190853 TSecr=2231110377
590	19.907967	142.251.46.206	192.168.1.11	TCP	66	443 → 57516 [FIN, ACK] Seq=1 Ack=79 Win=65536 Len=0 TSval=3734190853 TSecr=2231110377
591	19.907969	142.251.46.206	192.168.1.11	TCP	66	443 → 57516 [RST, ACK] Seq=2 Ack=79 Win=65536 Len=0 TSval=3734190854 TSecr=2231110377
592	19.908164	192.168.1.11	142.251.46.206	TCP	66	57516 → 443 [ACK] Seq=79 Ack=2 Win=131840 Len=0 TSval=2231110391 TSecr=3734190853
593	19.910554	52.72.243.169	192.168.1.11	TCP	66	[TCP ACKed unseen segment] 443 → 57489 [ACK] Seq=1 Ack=2 Win=132 Len=0 TSval=3982022800 TSecr=1591448524
594	19.920999	142.251.46.206	192.168.1.11	TCP	54	443 → 57516 [RST] Seq=2 Win=0 Len=0
595	19.980322	13.107.42.14	192.168.1.11	TLSv1…	146	Application Data
596	19.980553	192.168.1.11	13.107.42.14	TCP	54	56994 → 443 [ACK] Seq=1 Ack=185 Win=4096 Len=0
597	20.232483	162.159.134.234	192.168.1.11	TLSv1…	258	Application Data
598	20.232830	192.168.1.11	162.159.134.234	TCP	54	56938 → 443 [ACK] Seq=1 Ack=34087 Win=4096 Len=0
599	20.238972	162.159.134.234	192.168.1.11	TLSv1…	97	Application Data

```
sebastiancarbonero — -zsh — 80×24
sebastiancarbonero@Sebastians-MacBook-Pro-4 ~ % date
Mon Feb 28 17:43:56 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 ~ %
```

As we can see from the screenshot, a TCP connection was fully established. But we can see that after the HTTP GET request was sent, the first response had a data length of 0. So since we got a data length of 0, no data will ever be incoming once once the program had a non-positive reply on receive. Therefore, we got a `[Errno 54] Connection reset by peer`.

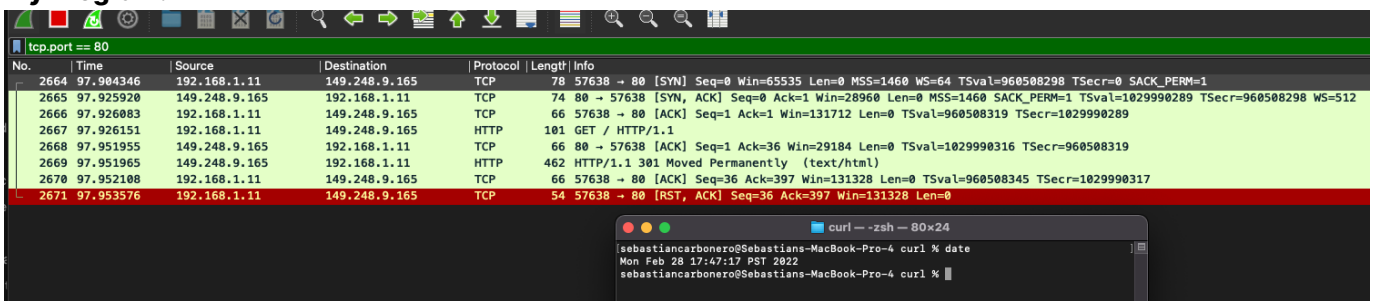
We can also see that from the screenshot, there are [RST] packets sent from the client to the server. This is due to the client ending the connection before it recieves all of the data. RST acks basically imply that the client no longer needs data from the server and cuts the connection.

We can also see that there is no difference between the wireshark output and my programs output.

---

lsebas.us with port 80

My Program:

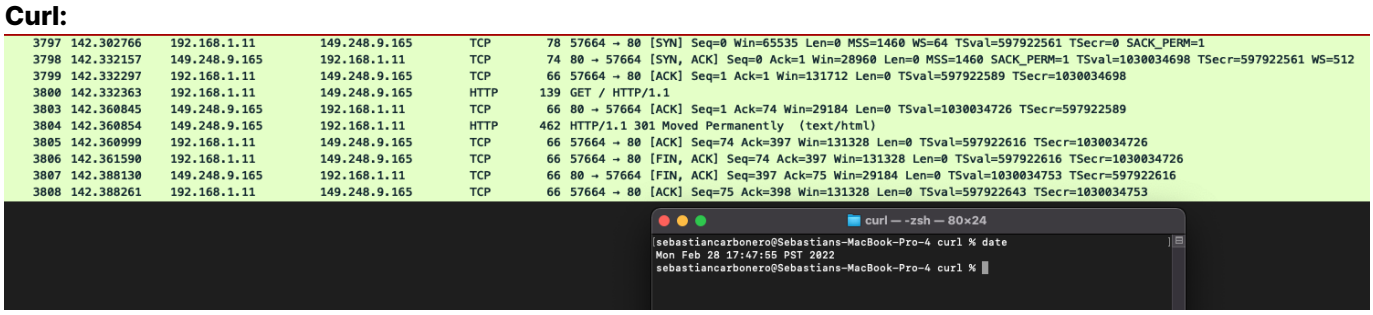


tcp.port == 80

No.	Time	Source	Destination	Protocol	Length	Info
2664	97.904346	192.168.1.11	149.248.9.165	TCP	78	57638 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=960508298 TSecr=0 SACK_PERM=1
2665	97.925920	149.248.9.165	192.168.1.11	TCP	74	80 → 57638 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1029990289 TSecr=960508298 WS=512
2666	97.926083	192.168.1.11	149.248.9.165	TCP	66	57638 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=960508319 TSecr=1029990289
2667	97.926151	192.168.1.11	149.248.9.165	HTTP	101	GET / HTTP/1.1
2668	97.951955	149.248.9.165	192.168.1.11	TCP	66	80 → 57638 [ACK] Seq=1 Ack=36 Win=29184 Len=0 TSval=1029990316 TSecr=960508319
2669	97.951965	149.248.9.165	192.168.1.11	HTTP	462	HTTP/1.1 301 Moved Permanently (text/html)
2670	97.952108	192.168.1.11	149.248.9.165	TCP	66	57638 → 80 [ACK] Seq=36 Ack=397 Win=131328 Len=0 TSval=960508345 TSecr=1029990317
2671	97.953576	192.168.1.11	149.248.9.165	TCP	54	57638 → 80 [RST, ACK] Seq=36 Ack=397 Win=131328 Len=0

```
curl — -zsh — 80×24
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl % date
Mon Feb 28 17:47:17 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl %
```

Curl:



3797	142.302766	192.168.1.11	149.248.9.165	TCP	78	57664 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=597922561 TSecr=0 SACK_PERM=1
3798	142.332157	149.248.9.165	192.168.1.11	TCP	74	80 → 57664 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1030034698 TSecr=597922561 WS=512
3799	142.332297	192.168.1.11	149.248.9.165	TCP	66	57664 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=597922589 TSecr=1030034698
3800	142.332363	192.168.1.11	149.248.9.165	HTTP	139	GET / HTTP/1.1
3803	142.360845	149.248.9.165	192.168.1.11	TCP	66	80 → 57664 [ACK] Seq=1 Ack=74 Win=29184 Len=0 TSval=1030034726 TSecr=597922589
3804	142.360854	149.248.9.165	192.168.1.11	HTTP	462	HTTP/1.1 301 Moved Permanently (text/html)
3805	142.360999	192.168.1.11	149.248.9.165	TCP	66	57664 → 80 [ACK] Seq=74 Ack=397 Win=131328 Len=0 TSval=597922616 TSecr=1030034726
3806	142.361590	192.168.1.11	149.248.9.165	TCP	66	57664 → 80 [FIN, ACK] Seq=74 Ack=397 Win=131328 Len=0 TSval=597922616 TSecr=1030034726
3807	142.388130	149.248.9.165	192.168.1.11	TCP	66	80 → 57664 [FIN, ACK] Seq=397 Ack=75 Win=29184 Len=0 TSval=1030034753 TSecr=597922616
3808	142.388261	192.168.1.11	149.248.9.165	TCP	66	57664 → 80 [ACK] Seq=75 Ack=398 Win=131328 Len=0 TSval=597922643 TSecr=1030034753

```
curl — -zsh — 80×24
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl % date
Mon Feb 28 17:47:55 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl %
```

As we can see from the screenshot, a TCP connection was fully established. But since the website was hosted under HTTPS, there was a redirection. Since my program closes the socket when ever there is no `HTTP/1,1 200 OK` in the header, the program will cut the connection early, resulting in another [RST] packet.

We can also notice that after each HTTP status line, the client always sends a [ACK] back to the server, which acknowledges that the client has recieved the message.

We can also notice that Curl goes right ahead to receive the content from the server, but my program stops the connection and does not proceed to receive the content. As a result, this was a design change in my end

so that way the program does not have to waste time to recieve the content and just show a error message to the user on terminal.

Example.com with port 80

My Program

tcp.port == 80

No.	Time	Source	Destination	Protocol	Length	Info
404	17.800555	192.168.1.11	93.184.216.34	TCP	78	57713 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2974829915 TSecr=0 SACK_PERM=1
405	17.814465	93.184.216.34	192.168.1.11	TCP	74	80 → 57713 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=1787380028 TSecr=2974829915 WS=512
406	17.814621	192.168.1.11	93.184.216.34	TCP	66	57713 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2974829928 TSecr=1787380028
407	17.814657	192.168.1.11	93.184.216.34	HTTP	103	GET / HTTP/1.1
408	17.828415	93.184.216.34	192.168.1.11	TCP	66	80 → 57713 [ACK] Seq=1 Ack=38 Win=65536 Len=0 TSval=1787380042 TSecr=2974829928
409	17.828423	93.184.216.34	192.168.1.11	TCP	1514	80 → 57713 [ACK] Seq=1 Ack=38 Win=65536 Len=1448 TSval=1787380042 TSecr=2974829928 [TCP segment of a reassembled PDU]
410	17.828425	93.184.216.34	192.168.1.11	HTTP	225	HTTP/1.1 200 OK (text/html)
411	17.828584	192.168.1.11	93.184.216.34	TCP	66	57713 → 80 [ACK] Seq=38 Ack=1608 Win=130112 Len=0 TSval=2974829941 TSecr=1787380042
412	17.830661	192.168.1.11	93.184.216.34	TCP	66	57713 → 80 [FIN, ACK] Seq=38 Ack=1608 Win=131072 Len=0 TSval=2974829943 TSecr=1787380042
413	17.847400	93.184.216.34	192.168.1.11	TCP	66	80 → 57713 [FIN, ACK] Seq=1608 Ack=39 Win=65536 Len=0 TSval=1787380061 TSecr=2974829943
414	17.847547	192.168.1.11	93.184.216.34	TCP	66	57713 → 80 [ACK] Seq=39 Ack=1609 Win=131072 Len=0 TSval=2974829959 TSecr=1787380061

curl — -zsh — 80×24

```
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl % date
Mon Feb 28 17:49:21 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl %
```

Curl:

tcp.port == 80

No.	Time	Source	Destination	Protocol	Length	Info
84	5.877530	192.168.1.11	93.184.216.34	TCP	78	57738 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2586934530 TSecr=0 SACK_PERM=1
85	5.894115	93.184.216.34	192.168.1.11	TCP	74	80 → 57738 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=1596344978 TSecr=2586934530 WS=512
86	5.894247	192.168.1.11	93.184.216.34	TCP	66	57738 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2586934546 TSecr=1596344978
87	5.894308	192.168.1.11	93.184.216.34	HTTP	141	GET / HTTP/1.1
88	5.912368	93.184.216.34	192.168.1.11	TCP	66	80 → 57738 [ACK] Seq=1 Ack=76 Win=65536 Len=0 TSval=1596344996 TSecr=2586934546
89	5.912378	93.184.216.34	192.168.1.11	TCP	1514	80 → 57738 [ACK] Seq=1 Ack=76 Win=65536 Len=1448 TSval=1596344998 TSecr=2586934546 [TCP segment of a reassembled PDU]
90	5.912382	93.184.216.34	192.168.1.11	HTTP	209	HTTP/1.1 200 OK (text/html)
91	5.912569	192.168.1.11	93.184.216.34	TCP	66	57738 → 80 [ACK] Seq=76 Ack=1592 Win=130176 Len=0 TSval=2586934564 TSecr=1596344998
92	5.912833	192.168.1.11	93.184.216.34	TCP	66	57738 → 80 [FIN, ACK] Seq=76 Ack=1592 Win=131072 Len=0 TSval=2586934564 TSecr=1596344998
93	5.926734	93.184.216.34	192.168.1.11	TCP	66	80 → 57738 [FIN, ACK] Seq=1592 Ack=77 Win=65536 Len=0 TSval=1596345011 TSecr=2586934564
94	5.926919	192.168.1.11	93.184.216.34	TCP	66	57738 → 80 [ACK] Seq=77 Ack=1593 Win=131072 Len=0 TSval=2586934578 TSecr=1596345011

curl — -zsh — 80×24

```
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl % date
Mon Feb 28 17:50:05 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl %
```

The TCP connection was fully established as there was a successful http get reqeuest from the client. We can see that once there was a successful HTTP GET request, my program proceeds to receive the bytes from the server. Then once all bytes were received, my program stops the connection with a [FIN, ACK] packet and the server responds with a [FIN, ACK] as well. Then the program closes the socket cleanly and exits the program successfully.

We can also see that my programs output is relatively similar to the Curl output in wireshark. This shows that both my program and wireshark are taking the same steps programatically.

Pudim.com with port 80

My Program:

tcp.port == 80

No.	Time	Source	Destination	Protocol	Length	Info
531	25.670046	192.168.1.11	54.207.20.104	TCP	78	57784 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1019847700 TSecr=0 SACK_PERM=1
538	25.860495	54.207.20.104	192.168.1.11	TCP	74	80 → 57784 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1460 SACK_PERM=1 TSval=2605263599 TSecr=1019847700 WS=128
539	25.860732	192.168.1.11	54.207.20.104	TCP	66	57784 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=1019847888 TSecr=2605263599
540	25.860897	192.168.1.11	54.207.20.104	HTTP	104	GET / HTTP/1.1
549	26.055443	54.207.20.104	192.168.1.11	TCP	66	80 → 57784 [ACK] Seq=1 Ack=39 Win=26880 Len=0 TSval=2605263647 TSecr=1019847888
550	26.055454	54.207.20.104	192.168.1.11	HTTP	1192	HTTP/1.1 200 OK (text/html)
551	26.055642	192.168.1.11	54.207.20.104	TCP	66	57784 → 80 [ACK] Seq=39 Ack=1127 Win=130624 Len=0 TSval=1019848080 TSecr=2605263648
552	26.060530	192.168.1.11	54.207.20.104	TCP	66	57784 → 80 [FIN, ACK] Seq=39 Ack=1127 Win=131072 Len=0 TSval=1019848084 TSecr=2605263648
555	26.251460	54.207.20.104	192.168.1.11	TCP	66	80 → 57784 [FIN, ACK] Seq=1127 Ack=40 Win=26880 Len=0 TSval=2605263697 TSecr=1019848084
556	26.251612	192.168.1.11	54.207.20.104	TCP	66	57784 → 80 [ACK] Seq=40 Ack=1128 Win=131072 Len=0 TSval=1019848271 TSecr=2605263697

curl — -zsh — 80×24

```
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl % date
Mon Feb 28 17:51:12 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl %
```

Curl:

tcp.port == 80

No.	Time	Source	Destination	Protocol	Length	Info
189	7.531893	192.168.1.11	54.207.20.104	TCP	78	57805 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2164893427 TSecr=0 SACK_PERM=1
192	7.716130	54.207.20.104	192.168.1.11	TCP	74	80 → 57805 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1460 SACK_PERM=1 TSval=2605272039 TSecr=2164893427 WS=128
193	7.716336	192.168.1.11	54.207.20.104	TCP	66	57805 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2164893610 TSecr=2605272039
194	7.716628	192.168.1.11	54.207.20.104	HTTP	142	GET / HTTP/1.1
199	7.905416	54.207.20.104	192.168.1.11	TCP	66	80 → 57805 [ACK] Seq=1 Ack=77 Win=26880 Len=0 TSval=2605272087 TSecr=2164893610
200	7.905424	54.207.20.104	192.168.1.11	TCP	66	[TCP Dup ACK 199#1] 80 → 57805 [ACK] Seq=1 Ack=77 Win=26880 Len=0 TSval=2605272087 TSecr=2164893610
201	7.906086	54.207.20.104	192.168.1.11	HTTP	1192	HTTP/1.1 200 OK (text/html)
202	7.906169	192.168.1.11	54.207.20.104	TCP	66	57805 → 80 [ACK] Seq=77 Ack=1127 Win=130624 Len=0 TSval=2164893797 TSecr=2605272087
203	7.906772	192.168.1.11	54.207.20.104	TCP	66	57805 → 80 [FIN, ACK] Seq=77 Ack=1127 Win=131072 Len=0 TSval=2164893798 TSecr=2605272087
206	8.095574	54.207.20.104	192.168.1.11	TCP	66	80 → 57805 [FIN, ACK] Seq=1127 Ack=78 Win=26880 Len=0 TSval=2605272134 TSecr=2164893798
207	8.095747	192.168.1.11	54.207.20.104	TCP	66	57805 → 80 [ACK] Seq=78 Ack=1128 Win=131072 Len=0 TSval=2164893986 TSecr=2605272134

curl — -zsh — 80×24

```
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl % date
Mon Feb 28 17:51:43 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl %
```

The TCP connection was fully established as there was a successful http get reqeuest from the client. My Program takes the same steps as described by the last GET request with Example.com.

We can also see that the output from my program and Curl are exactly the same except for port numbers obviously.

Example.com/anyname.html with port 80

My Program

1280	58.673016	192.168.1.11	93.184.216.34	TCP	78	57859 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3122965807 TSecr=0 SACK_PERM=1
1284	58.693012	93.184.216.34	192.168.1.11	TCP	74	80 → 57859 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=938497215 TSecr=3122965807 WS=512
1285	58.693147	192.168.1.11	93.184.216.34	TCP	66	57859 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=3122965826 TSecr=938497215
1286	58.693199	192.168.1.11	93.184.216.34	HTTP	115	GET /anyname.html HTTP/1.1
1287	58.706384	93.184.216.34	192.168.1.11	TCP	66	80 → 57859 [ACK] Seq=1 Ack=50 Win=65536 Len=0 TSval=938497229 TSecr=3122965826
1288	58.706393	93.184.216.34	192.168.1.11	TCP	1514	80 → 57859 [ACK] Seq=1 Ack=50 Win=65536 Len=1448 TSval=938497229 TSecr=3122965826 [TCP segment of a reassembled PDU]
1289	58.706396	93.184.216.34	192.168.1.11	HTTP	216	HTTP/1.1 404 Not Found (text/html)
1290	58.706515	192.168.1.11	93.184.216.34	TCP	66	57859 → 80 [ACK] Seq=50 Ack=1599 Win=130112 Len=0 TSval=3122965839 TSecr=938497229
1291	58.707982	192.168.1.11	93.184.216.34	TCP	54	57859 → 80 [RST, ACK] Seq=50 Ack=1599 Win=130112 Len=0

curl — -zsh — 80×24

```
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl % date
Mon Feb 28 17:53:21 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl %
```

Curl

tcp.port == 80

No.	Time	Source	Destination	Protocol	Length	Info
214	13.446510	192.168.1.11	93.184.216.34	TCP	78	57833 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1247505638 TSecr=0 SACK_PERM=1
215	13.460513	93.184.216.34	192.168.1.11	TCP	74	80 → 57833 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=93641321 TSecr=1247505638 WS=512
216	13.460650	192.168.1.11	93.184.216.34	TCP	66	57833 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=1247505653 TSecr=93641321
217	13.460717	192.168.1.11	93.184.216.34	HTTP	153	GET /anyname.html HTTP/1.1
218	13.479780	93.184.216.34	192.168.1.11	TCP	66	80 → 57833 [ACK] Seq=1 Ack=88 Win=65536 Len=0 TSval=93641341 TSecr=1247505653
219	13.479788	93.184.216.34	192.168.1.11	TCP	1514	80 → 57833 [ACK] Seq=1 Ack=88 Win=65536 Len=1448 TSval=93641341 TSecr=1247505653 [TCP segment of a reassembled PDU]
220	13.479790	93.184.216.34	192.168.1.11	HTTP	216	HTTP/1.1 404 Not Found (text/html)
221	13.479923	192.168.1.11	93.184.216.34	TCP	66	57833 → 80 [ACK] Seq=88 Ack=1599 Win=130112 Len=0 TSval=1247505672 TSecr=93641341
222	13.480288	192.168.1.11	93.184.216.34	TCP	66	57833 → 80 [FIN, ACK] Seq=88 Ack=1599 Win=131072 Len=0 TSval=1247505672 TSecr=93641341
223	13.498743	93.184.216.34	192.168.1.11	TCP	66	80 → 57833 [FIN, ACK] Seq=1599 Ack=89 Win=65536 Len=0 TSval=93641360 TSecr=1247505672
224	13.498866	192.168.1.11	93.184.216.34	TCP	66	57833 → 80 [ACK] Seq=89 Ack=1600 Win=131072 Len=0 TSval=1247505690 TSecr=93641360

curl — -zsh — 80×24

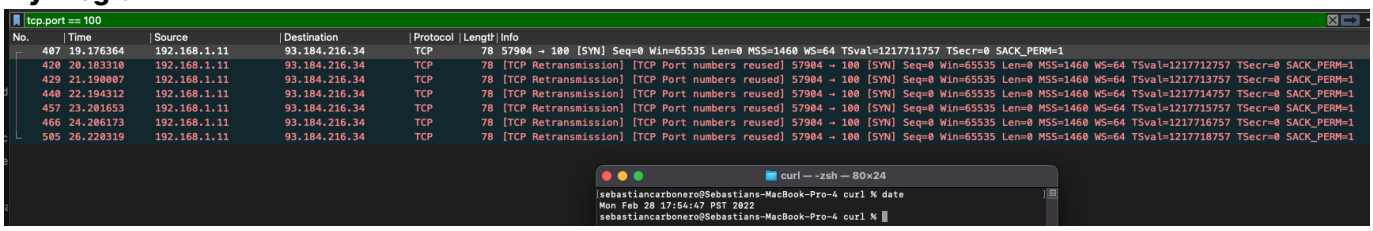
```
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl % date
Mon Feb 28 17:52:34 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl %
```

The TCP connection was fully established as there was no successful GET request and we received a `HTTP/1.1 404 NOT FOUND` error stating that there was no content found under the URL.

We can see that there are both `404 NOT FOUND` errors within both of the programs, but my program closes out before receiving the content page. But something very interesting happens. As curl proceeds to download the page, the page that is returned in the home page. This is because instead of the server sending a default 404 content page, it sends it's home page.

Example.com with port 100

My Program



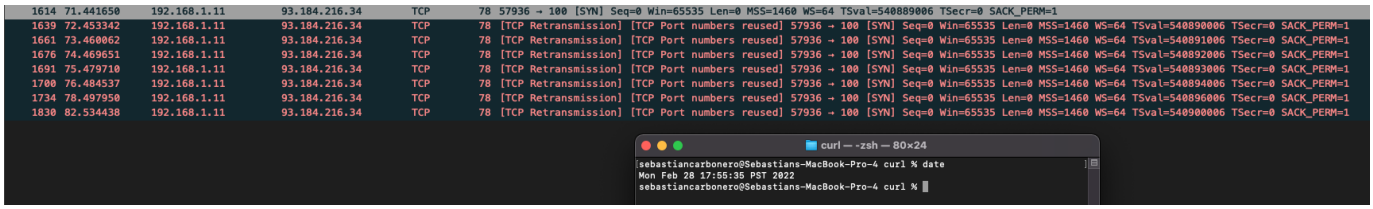
tcp.port == 100

No.	Time	Source	Destination	Protocol	Length	Info
407	19.176364	192.168.1.11	93.184.216.34	TCP	78	57904 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1217711757 TSecr=0 SACK_PERM=1
420	20.183310	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57904 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1217712757 TSecr=0 SACK_PERM=1
429	21.190007	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57904 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1217713757 TSecr=0 SACK_PERM=1
440	22.194312	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57904 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1217714757 TSecr=0 SACK_PERM=1
457	23.201653	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57904 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1217715757 TSecr=0 SACK_PERM=1
466	24.206173	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57904 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1217716757 TSecr=0 SACK_PERM=1
505	26.220319	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57904 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1217718757 TSecr=0 SACK_PERM=1

curl — -zsh — 80×24

```
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl % date
Mon Feb 28 17:54:47 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl %
```

Curl:



1614	71.441650	192.168.1.11	93.184.216.34	TCP	78	57936 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=540889006 TSecr=0 SACK_PERM=1
1639	72.453342	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57936 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=540890006 TSecr=0 SACK_PERM=1
1661	73.460062	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57936 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=540891006 TSecr=0 SACK_PERM=1
1676	74.469651	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57936 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=540892006 TSecr=0 SACK_PERM=1
1691	75.479710	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57936 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=540893006 TSecr=0 SACK_PERM=1
1700	76.484537	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57936 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=540894006 TSecr=0 SACK_PERM=1
1734	78.497950	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57936 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=540896006 TSecr=0 SACK_PERM=1
1830	82.534438	192.168.1.11	93.184.216.34	TCP	78	[TCP Retransmission] [TCP Port numbers reused] 57936 → 100 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=540900006 TSecr=0 SACK_PERM=1

curl — -zsh — 80×24

```
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl % date
Mon Feb 28 17:55:35 PST 2022
sebastiancarbonero@Sebastians-MacBook-Pro-4 curl %
```

As we can see from this example, there was no TCP connection established. We can see that the client initally starts out with a [SYN] packet, but never gets a response back. Therefore, the client keeps on retransmitting this packet until something is received. But since there is a timeout on my program, the client stops sending and closes the socket.

We can also see that curl takes the same steps as my program and then they both finally time out.