

Programming Assignment: Webserver

Author: Anup Bharadwaj

P538- Computer Networks

Write-up:

Persistent connections keep the socket open to process the requests from the client and hence are expected to be relatively more efficient compared to non-persistent connections where the client has to undergo a TCP handshake for every request. Therefore, non-persistent connections take 2 RTT more to send a request and get the response from the server.

In our client and the server application, one to eight 1.3 MB files are requested by the client using both persistent and non-persistent connections. The table below summarizes the time taken under each scenario. To perform the analysis, I have taken a text file named “xl.txt” which is 1.3 MB in size.

Number of Files * File size (MB)	Persistent Connections (Time in seconds)	Non-persistent Connections(Time in seconds)
1.3	0.55	0.57
2.6	2.07	1.5
3.9	3.57	2.04
5.2	4.78	3
6.5	8.61	4.2
7.8	8.74	4.9
9.1	10.76	5.9
10.4	11.86	6.8

Several test runs have been made and the average of the runs made in each scenario has been tabulated above. For non-persistent connections, the sum of the time taken to request each file has been taken. However for persistent connections, the time taken to fetch all the files at once is recorded. Therefore, the overall time taken for non-persistent connections could be much longer when we consider the time taken to send each requests after server closing the socket every time. Hence, even though the time taken in the case of persistent connections appears to be longer, it includes the overall time taken to fetch the contents of the files requested in order and hence is more efficient as it avoids a TCP handshake for every request.

It takes almost the same amount of time to request and receive a 1 MB file in both persistent and non-persistent connections. From the above table, it took 0.55 and 0.56 seconds for persistent and non-persistent connections respectively to request and receive a 1.3 MB file from the server.

Capturing on Wireless Network Connection [Wireshark 1.12.8 (v1.12.8-0-g5b6e543 from master-1.12)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
50	5.86831600	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
51	5.86841600	129.79.247.5	192.168.0.106	SSH	454	Server: Encrypted packet (len=400)
52	5.86844400	192.168.0.106	129.79.247.5	TCP	54	60268-22 [ACK] Seq=105 Ack=2505 win=3073 Len=0
53	5.86868300	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
54	5.86874400	129.79.247.5	192.168.0.106	SSH	326	Server: Encrypted packet (len=272)
55	5.86878900	192.168.0.106	129.79.247.5	TCP	54	60268-22 [ACK] Seq=105 Ack=4237 win=3073 Len=0
56	5.86892000	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
57	5.86896100	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
58	5.86900500	192.168.0.106	129.79.247.5	TCP	54	60268-22 [ACK] Seq=105 Ack=7157 win=3073 Len=0
59	5.87515300	129.79.247.5	192.168.0.106	SSH	234	Server: Encrypted packet (len=180)
60	5.92783100	192.168.0.106	129.79.247.5	TCP	54	60265-22 [ACK] Seq=1 Ack=573 win=256 Len=0
61	5.92919800	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
62	5.97670000	129.79.247.5	192.168.0.106	SSH	234	Server: [TCP Retransmission] , Encrypted packet (len=180)
63	5.97681300	192.168.0.106	129.79.247.5	TCP	66	[TCP Dup Ack 60#1] 60265-22 [ACK] Seq=1 Ack=573 win=256 Len=0 SLE=393 SRE=573
64	5.97964800	192.168.0.106	129.79.247.5	TCP	54	60268-22 [ACK] Seq=105 Ack=8617 win=3073 Len=0
65	5.99726600	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
66	5.99812900	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
67	5.99824000	192.168.0.106	129.79.247.5	TCP	54	60268-22 [ACK] Seq=105 Ack=11537 win=3073 Len=0
68	5.99853600	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
69	5.99862000	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
70	5.99869400	192.168.0.106	129.79.247.5	TCP	54	60268-22 [ACK] Seq=105 Ack=14457 win=3073 Len=0
71	5.99890500	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
72	5.99896600	129.79.247.5	192.168.0.106	SSH	1514	Server: Encrypted packet (len=1460)
73	5.99907100	192.168.0.106	129.79.247.5	TCP	54	60268-22 [ACK] Seq=105 Ack=17377 win=3073 Len=0

Frame 73: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Interface id: 0 (\Device\NPF_{F82C062F-596E-4557-A851-032E3D4F7BA8})
Encapsulation type: Ethernet (1)
Arrival time: Oct 17, 2015 19:30:05.312788000 US Eastern Daylight Time
[Time shift for this packet: 0.000000000 seconds]
Epoch time: 1445124605.312788000 seconds
[Time delta from previous captured frame: 0.000075000 seconds]
[Time delta from previous displayed frame: 0.000075000 seconds]
[Time since reference or first frame: 5.999071000 seconds]
Frame Number: 73
Frame Length: 54 bytes (432 bits)
Capture Length: 54 bytes (432 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:tcp]

0000 30 b5 c2 49 0a 52 ac 72 89 46 86 b8 08 00 45 00 0...I.R.r.F....E.
0010 00 28 27 be 40 00 80 06 99 aa c0 a8 00 6a 81 4f .(.0... ..).0
0020 f7 05 ab 6c 00 29 63 49 f3 57 6a 78 a3 50 10jC I.WjX.P.
0030 0c 01 3b 85 00 00

Wireless Network Connection: <live capture...> Packets: 2719 - Displayed: 2719 (100.0%) Profile: Default

Ask me anything 7:33 PM 10/17/2015

Multi-threaded server:

The multi-threaded server application was much more efficient in handling both persistent and non-persistent connections compared to the single threaded server application. Since each thread was handling requests in parallel, the client was able to receive the response with at least 20-30% lesser time compared to a single threaded system.

Scenario:

Three client applications were made to request for a 1.3 MB file at the same time to a single threaded server application. The second and the third client applications could only receive files after the first client application finished its transaction. The overall time taken for the server to respond to all the three clients was 8.27 seconds. However, with a multi-threaded server, it took just 3.18 seconds to do the same task. The performance of the multi-threaded server was a little better with persistent connections in the case where the client was sequentially requesting for files. Had it been a case where the client was requesting for multiple files simultaneously, even better performance could be seen with multi-threading.

Client and server using UDP:

The time taken to receive a response in a UDP server and client application was much faster than TCP. In one of the test runs conducted, the server was able to push contents in 0.5 seconds to the client. But the data transfer using UDP is not reliable like TCP as it is always prone to packet loss. However, I did not experience any packet loss as the application was run on the silo server having 100 MBPS switched Ethernet with very low congestion.