PCD - H1 - Technical Report

Pasat Tudor Cosmin

1. Introduction

This technical report explores the performance analysis of data transfer protocols, with a focus on UDP (User Datagram Protocol) and TCP (Transmission Control Protocol). The primary objective is to measure the time required for transferring varying amounts of data across different message sizes and transmission mechanisms.

1.1 Homework objectives

The task entails designing a program capable of measuring data transfer time for different message sizes (1 to 65535 bytes) and amounts of data using UDP and TCP protocols. Additionally, the program must support both streaming and stop-and-wait transmission mechanisms.

1.2 Implementation requirements

- **Supported Protocols:** UDP and TCP protocols must be supported as parameters for both client and server components.
- **Message Size:** The program should facilitate data transfer across a range of message sizes to assess performance implications.
- **Transmission Mechanisms:** Two transmission mechanisms, streaming and stop-and-wait, should be implemented for comprehensive analysis.

1.3 Output requirements

- **Server Output:** The server will print the protocol used, number of messages read, and bytes read after each session.
- **Client Output:** Upon completion, the client will display transmission time, number of sent messages, and total bytes sent.

2. Experiment

The experiments were done on a local host, using a laptop with Intel Core I5 processor and MacOS Ventura 13.4.1. I ran all the pairs, protocol and mechanism, 100 times and computed the minimum, maximum and the average values for each of the statistics presented in section **1.3**.

For the stop and wait mechanism, I added to the client a timeout of 10 seconds. If he doesn't receive the acknowledgement until the timer runs out, the client will not move to the next package and will try to send the same one until the acknowledgement is received.

2.1 TCP

I used for both of the mechanisms a buffer of 65535 bytes.

2.1.1 Streaming

Size	Messages			Bytes			Transmission time		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
500 MB	8001	8001.45	8036	524288000	524288000	524288000	0.18s	0.22s	0.5s
1GB	16385	16385.13	16388	1073741824	1073741824	1073741824	0.36s	0.44s	0.97s
2GB	32769	32769.39	32776	2147483648	2147483648	2147483648	0.63s	0.96s	2.64s

Table 1: TCPS Client statistics

Size	Messages			Bytes		
	Min	Avg	Max	Min	Avg	Max
500MB	8156	8289.31	9675	524288000	524288000	524288000
1GB	16533	16727.15	17229	1073741824	1073741824	1073741824
2GB	33121	33692.17	39712	2147483648	2147483648	2147483648

Table 2: TCPS Server statistics

2.1.2 Stop and wait

Size	Messages			Bytes			Transmission time		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
500 MB	8001	8001	8001	52428 8000	52428 8000	52428 8000	0.18s	0.20s	0.42s
1GB	16385	16385	16385	10737 41824	10737 41824	10737 41824	0.38s	0.48s	1.29s
2GB	32769	32769	32769	21474 83648	21474 83648	21474 83648	0.73s	1.08s	3.21s

Table 3: TCPSW Client statistics

Size	Messages			Bytes		
	Min	Avg	Max	Min	Avg	Max
500MB	8070	8119.47	8240	524288000	524288000	524288000
1GB	16542	16716.57	17210	1073741824	1073741824	1073741824
2GB	33013	33374.60	34246	2147483648	2147483648	2147483648

Table 4: TCPSW Server statistics

2.2 UDP

I used for both of the mechanisms a buffer of 9216 bytes.

2.2.1 Streaming

Size	Messages			Bytes			Transmission time		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
500 MB	56889	56889	56889	524288000	524288000	524288000	0.76	1.16	0.92
1GB	116509	116509	116509	1073741824	1073741824	1073741824	1.66s	1.91s	3.45s
2GB	233017	233017	233017	2147483648	2147483648	2147483648	3.09s	3.76s	7.89s

Table 5: UDPS Client statistics

Size	Messages			Bytes		
	Min	Avg	Max	Min	Avg	Max
500MB	56738	56883.16	56889	522896384	524234178.56	524288000
1GB	116047	116501.64	116509	1069484032	1073673994.24	1073741824
2GB	231733	232985.95	233017	2135650304	2147197491.2	2147483648

Table 6: UDPS Server statistics

2.2.2 Stop and wait

Size	Messages			Bytes			Transmission time		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
500 MB	56889	56889	56889	524288000	524288000	524288000	2.02s	2.18s	3.11s
1GB	116509	116509	116509	1073741824	1073741824	1073741824	4.18s	4.53s	7.35s
2GB	233017	233017	233017	2147483648	2147483648	2147483648	8.35s	9.37s	13.34s

Table 7: UDPSW Client statistics

Size	Messages			Bytes		
	Min	Avg	Max	Min	Avg	Max
500MB	56889	56889	56889	524288000	524288000	524288000
1GB	116509	116509	116509	1073741824	1073741824	1073741824
2GB	233017	233017	233017	2147483648	2147483648	2147483648

Table 8: UDPSW Server statistics

3. Discussion

We can see from the tables above that the UDP protocol, using the streaming mechanism, is the worst when considering the number of bytes that actually get to the server. There are packages of bytes lost even for the smaller amounts of data.

As for the other 3, all the data gets to the server, but the transmission times are significantly different, between the 2 TCP programs and the UDP one. This is mostly because I used a much smaller buffer size for the UDP, in order to avoid getting more expensive getting lost.

The TCP with stop and wait mechanism has a higher latency than the streaming version, because the client needs to wait for acknowledgement before sending the next package.

4. Conclusion

In the end, we can say that one protocol or mechanism is better than another. All have their advantages and disadvantages, and we need to analyse the use case to find which is best suited for it.

5. References

- https://profs.info.uaic.ro/~adria/teach/courses/pcd
- https://profs.info.uaic.ro/~adria/teach/courses/pcd/homework/PCD_Homework1.pdf
- https://docs.python.org/3.11/library/socket.html#module-socket

- https://pythontic.com/modules/socket/introduction
- https://pythontic.com/modules/socket/udp-client-server-example
- https://forums.developer.apple.com/forums/thread/74655#:~:text=By%20default%20macOS%20has%20limited,following%20command%20in%20the%20terminal.