Assignment 1

Shreyansh Singh

2020CS10385

PSP Network

Design Decision

I have used 4n ports. 2n of which are used by servers and 2n by client side. For each port a UDP and a TCP socket is bound to it, thus I have in total 8n sockets. There are n threads on server side, each thread has 2 TCP and 2 UDP sockets. 1 TCP socket is for sending data when client request and other is used to receive data when requested. 1 UDP is used for requesting client and other for receiving requests.

Client side has 2n threads and each thread has a UDP and a TCP socket. Each client has 2 threads, of which first thread request packet from server (ask_query function) (uses UDP for requesting and TCP for receiving data) and other receives request from server (ans_query function) (uses UDP to listen request and send chunk via TCP).

Analysis

1. Average RTT of all chunks is 0.003806581986801965 seconds on sending small file to 5 clients using first method. The second method takes 0.002591614212308611 seconds.

The first method has more RTT. This is expected since the bottleneck is transferring chunks through socket and UDP is much faster than TCP. Hence the second part takes less time since it transfers data via UDP whereas first part uses TCP for transferring data.

2. Average RTT for each chunk across all clients: for first part:

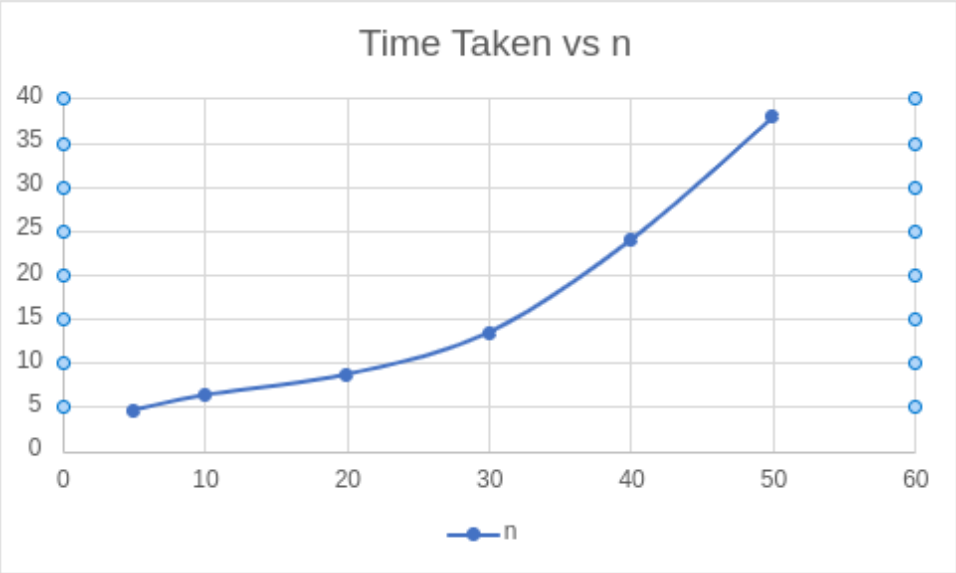
{0 : 0.001589, 1 : 0.0013315, 2 : 0.0042945, 3 : 0.0048679, 4 : 0.0022846, 5 : 0.0029015, 6 : 0.002562, 7 : 0.002276, 8 : 0.002884, 9 : 0.0034105, 10 : 0.003237, 11 : 0.0034356, 12 : 0.0027680, 13 : 0.004369, 14 : 0.0037920, 15 : 0.005416, 16 : 0.0020444, 17 : 0.003847, 18 : 0.0022614, 19 : 0.002944, 20 : 0.0025653, 21 : 0.003262, 22 : 0.005016, 23 : 0.0014024, 24 : 0.0014853, 25 : 0.011107, 26 : 0.004227, 27 : 0.003961, 28 : 0.0020563, 29 : 0.0024002, 30 : 0.0017392, 31 : 0.001723, 32 : 0.012443, 33 : 0.0027000, 34 : 0.001169, 35 : 0.011186, 36 : 0.0040978, 37 : 0.0018864, 38 : 0.022878, 39 : 0.00303, 40 : 0.003109, 41 : 0.0021898, 42 : 0.0012630, 43 : 0.03413, 44 : 0.001270, 45 : 0.0034606, 46 : 0.017300, 47 : 0.00278, 48 : 0.003822, 49 : 0.0020188, 50 : 0.0021278, 51 : 0.00376, 52 : 0.0020301, 53 : 0.0022220, 54 : 0.0029605, 55 : 0.009490, 56 : 0.011258, 57 : 0.004138, 58 : 0.002180, 59 : 0.004892, 60 : 0.004341, 61 : 0.012265, 62 : 0.001544, 63 : 0.0020903, 64 : 0.0022166, 65 : 0.002551, 66 : 0.003393, 67 : 0.002549, 68 : 0.0036853, 69 : 0.0040835, 70 : 0.004585, 71 : 0.003253, 72 : 0.0036269, 73 : 0.002472, 74 : 0.0019866, 75 : 0.003414, 76 : 0.003852, 77 : 0.0025081, 78 : 0.003631, 79 : 0.0028842, 80 : 0.011287, 81 : 0.003821, 82 : 0.01114, 83 : 0.0037294, 84 : 0.0019568, 85 : 0.002595, 86 : 0.001417, 87 : 0.01170, 88 : 0.002461, 89 : 0.002033, 90 : 0.0033098, 91 : 0.0024360, 92 : 0.00349, 93 : 0.0017780, 94 : 0.0024133, 95 : 0.003198, 96 : 0.0027441, 97 : 0.0036847, 98 : 0.0032716, 99 : 0.0024372, 100 : 0.002275, 101 : 0.0020879, 102 : 0.0018107, 103 : 0.0031894, 104 : 0.011157, 105 : 0.011176, 106 : 0.0037807, 107 : 0.00423, 108 : 0.01117, 109 : 0.003657, 110 : 0.003361, 111 : 0.0035065}

Average RTT for each chunk across all clients: for second part:

{0 : 0.002532422, 1 : 0.002537953, 2 : 0.0025352, 3 : 0.002559554, 4 : 0.002535706, 5 : 0.00253105, 6 : 0.002544653, 7 : 0.002540248, 8 : 0.00254651, 9 : 0.002534192, 10 : 0.0025356, 11 : 0.002530974, 12 : 0.002535319, 13 : 0.002536958, 14 : 0.002546620, 15 : 0.002532762, 16 : 0.002535051, 17 : 0.002535450, 18 : 0.002520912, 19 : 0.002537810, 20 : 0.002531158, 21 : 0.00249311, 22 : 0.002549093, 23 : 0.002541643, 24 : 0.002529489, 25 : 0.00253530, 26 : 0.002539110, 27 : 0.002538490, 28 : 0.00254088, 29 : 0.00253388, 30 : 0.00254603, 31 : 0.00253232, 32 : 0.00253449, 33 : 0.002537155, 34 : 0.002541679, 35 : 0.0025329, 36 : 0.002536857, 37 : 0.002542406, 38 : 0.002531760, 39 : 0.002532911, 40 : 0.002539014, 41 : 0.0025375, 42 : 0.002534502, 43 : 0.002535718, 44 : 0.00255676, 45 : 0.002530872, 46 : 0.002532136, 47 : 0.00253814, 48 : 0.00253929, 49 : 0.002538371, 50 : 0.00253532, 51 : 0.002531647, 52 : 0.002532321, 53 : 0.002543401, 54 : 0.002546751, 55 : 0.002546995, 56 : 0.002536559, 57 : 0.002536070, 58 : 0.002537244, 59 : 0.00252832, 60 : 0.002557611, 61 : 0.00253874, 62 : 0.002531635, 63 : 0.002543777, 64 : 0.002551054, 65 : 0.002539497, 66 : 0.002538317, 67 : 0.002538043, 68 : 0.00253458, 69 : 0.00254793, 70 : 0.00251634, 71 : 0.00253205, 72 : 0.00253519, 73 : 0.00254798, 74 : 0.002532809, 75 : 0.002532917, 76 : 0.002531552, 77 : 0.002534604, 78 : 0.0025379, 79 : 0.002539908, 80 : 0.00253317, 81 : 0.002567350, 82 : 0.002538329, 83 : 0.002541399, 84 : 0.002535796, 85 : 0.002553302, 86 : 0.00253355, 87 : 0.002535384, 88 : 0.002548640, 89 : 0.002537751, 90 : 0.00254102, 91 : 0.002536314, 92 : 0.002531415, 93 : 0.00254142, 94 : 0.002538889, 95 : 0.002550601, 96 : 0.002542275, 97 : 0.00253171, 98 : 0.002546620, 99 : 0.00254148, 100 : 0.002532333, 101 : 0.002536433, 102 : 0.002539765, 103 : 0.00252546, 104 : 0.00253645, 105 : 0.00254777, 106 : 0.002538526, 107 : 0.002534621, 108 : 0.002541762, 109 : 0.002532333, 110 : 0.002529913, 111 : 0.002519190}

No chunk has significantly higher RTT since in my implementation the clients request random packets.

3. The total time taken increases as value of n increases. Yes, this trend is expected because as the value of n increases the number of packets each client has decreases thus each client has to make a greater number of requests hence the number of broadcasts increases. The time taken by each broadcast also increases since, the probability that any client has data decreases.



4. Not able to run large file. I was unable to resolve the errors I got.

5. Sequential method will take more time than random method since in sequential all clients try server threads in a sequential manner and thus chances that a thread is busy is higher than in the random case where each client select a server thread at random.

Food For Thought

1. The main advantage is that server need not to store whole file which will increase efficiency since server does not have to create a new copy every item to send data to client. Thus, without much storage at server we can scale this for multiple files.
2. This network is more efficient for clients since in traditional P2P system each client would have to handle requests from many clients simultaneously but in this network each client request from server and handles request only from server. This will significantly reduce the computation at client side.
3. Presently each chunk is associated with a chunk number but in this system each chunk will be associated with a chunk number and file number. In this network initially server will send equal chunks to all clients and then each client could request chunks at random of the file it wants.