# Assignment 1 report

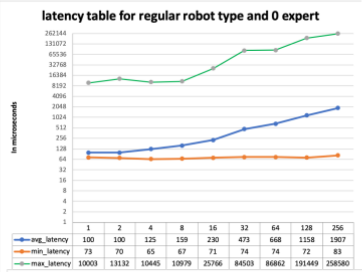
Software design:

In this assignment, we are asked to implement a basic client-server program as the basis for network program including distributed systems. Following through the instructions in this assignment, I firstly implement the basic communication classes that encapsulate the socket-level code including client socket class and server socket class. These two classes are very essential and needed for later works, since these two classes have many useful socket functionalities such as creation of socket, binding, connection, listen, send, receive and so on. Then, implementing the server stub and client stub become easier by taking a connected socket for server stub and passing server's IP address and port number to make a connection for client. Since server stub and client stub will send and receive robot information and order information, my next step is to implement order and robot Info classes. Since we will need marshal and unmarshal information when sending through network and receiving at local, I implement marshal and unmarshal in order and robot Info classes so that it can easily be used whenever order and robot info are sent or received.

Then I start implementing client program. It will firstly parse the input arguments including IP address, port number, customer number, order number and robot type. Since arguments are passed as string, I need transfer them to int when necessary. Since stoi won't do error handling, I create a help function to wrap stoi with a try catch to handle the possible error. After getting all required inputs, it will create a thread with a client stub for each customer, and for each customer thread it will simply send order and receive returned robot information. We also need take down elapsed time for each order, therefore we need a lock here since the latency record are shared and manipulated by many customer threads. After making all orders, stub will be disconnected and customer threads will will join and the whole client program ends. We also need take down the whole elapsed time here.

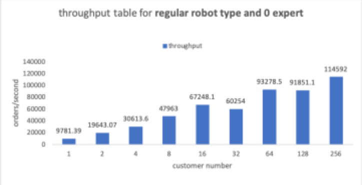
Then the server program is the most challenging one. For server program without the special robot, I firstly parse the inputs as usual. Then, the server program needs to create a sock to accept client's connection. Once connection is made, it will create an engineer thread for each connection and for each engineer thread it has the method to process orders by simply sending shipping information. Once expert thread receives that the client disconnected, it will break the loop. Later, we add the special robot type and only experts can handle the special robot type. We need parse this new argument. I create an expert thread pool to keep track of all experts' availability. Moreover, I need a "pair" structure to represent that an engineer assigns the special task to an expert, therefore I map the engineer id with expert id together. Once engineer thread receives a special order, it will push the special order to a shared queue list and will notify the expert thread pool that a new order comes in. The engineer thread will then be locked and it will check if there's available experts will be assigned, otherwise it will be waiting for the signal that an expert can be assigned for this special order. On the other hand, the expert thread pool will be locked until there's an available expert and it receives the signal that a new order is coming in. Expert thread pool will further pop the first order from the queue and assign an expert to this order and corresponding engineer who notifies this order (mapping them together). Once there's an expert assigned and engineer thread receives the signal, it will process the order.

That is basically my software design. I think the server program is the main challenging and I also spent lots of time learning how to build socket in C++. I think all these works help me learn more about these basics and help me get into the further implementations like distributed systems. Moreover, I try to handle the situation that when customer make special order but there is no expert in server. I try to make server to send an error message to the client and disconnected, but I don't know how the client can receive the error message. As a result, I simply make the connection die and notify the server that customer made the special order but failed. Hopefully I can get it done when I know more about socket program.

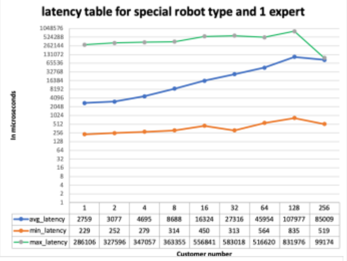


| | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|---|---|
| avg_latency | 100 | 100 | 125 | 159 | 230 | 473 | 668 | 1158 | 1907 |
| min_latency | 73 | 70 | 65 | 67 | 71 | 74 | 74 | 72 | 83 |
| max_latency | 10003 | 13132 | 10445 | 10979 | 25766 | 84503 | 86862 | 191449 | 258580 |

This is the graph of the latency table for regular robot and 0 expert. It can be observed that when there are more customers, the average latency becomes worse. The minimum latency looks stable, but the maximum become worse and worse when there are more customers.

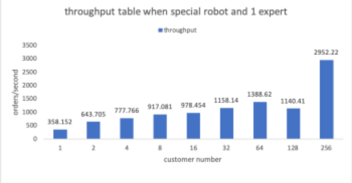


This is the throughput table for regular robot and 0 expert. The throughput gets larger when there are more customers.

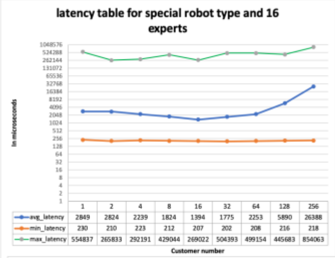


| | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|---|---|
| avg_latency | 2759 | 3077 | 4695 | 8688 | 16324 | 27316 | 45954 | 107977 | 85009 |
| min_latency | 229 | 252 | 279 | 314 | 450 | 313 | 564 | 835 | 519 |
| max_latency | 286106 | 327596 | 347057 | 363355 | 556841 | 583018 | 516620 | 831976 | 99174 |

This is the latency table for special robot and 1 expert. It can be observed that overall latency becomes larger than latency in the previous experiment. The latency grows dramatically when more customers order, since there is only one expert can process the order.

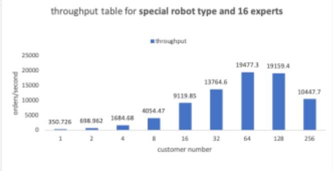


This is throughput table when special order made and 1 expert. The throughput is overall lower than the throughput from experiment 1.

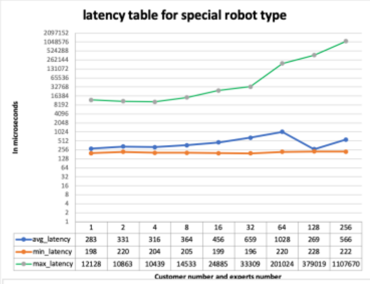


| | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|---|---|
| avg_latency | 2849 | 2824 | 2239 | 1824 | 1394 | 1775 | 2253 | 5890 | 26388 |
| min_latency | 230 | 210 | 223 | 212 | 207 | 202 | 208 | 216 | 218 |
| max_latency | 554837 | 265833 | 292191 | 429044 | 269022 | 504393 | 499154 | 445683 | 854063 |

This is the latency table for special robot and 16 experts. It can be observed that the average latency is stable until 128 and 256 customers. Minimum latency looks stable. Maximum latencies are very high.

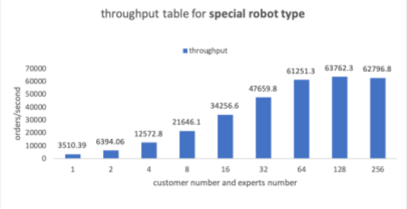


This is the throughput table for special robot and 16 experts. The throughput grows until when there are 256 customers. It also can be observed that the throughput is a lot higher when there are 16, 32, 64 and 128 customers. It gets worse when there are 256 customers. Perhaps 16 experts are good for 32 to 128 customers.



| | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|---|---|
| avg_latency | 283 | 331 | 316 | 364 | 456 | 659 | 1028 | 269 | 566 |
| min_latency | 198 | 220 | 204 | 205 | 199 | 196 | 220 | 228 | 222 |
| max_latency | 12128 | 10863 | 10439 | 14533 | 24885 | 33309 | 201024 | 379019 | 1107670 |

This is the latency table for special robot and customer number is same as the expert number. The average latency looks stable except when 64 customers. The minimum latency looks stable as well. The maximum latency grows and grows rapidly when customers and experts are more than 32.



This is the throughput table for special robot, and customer number is same as the expert number. The throughput is overall higher than all experiment before. The throughput grows when more customers and experts. It can be noticed the growth becomes stable when 64, 128 and 256 customers and experts.