## Implementation

For both problems, I use my own implementation of the client side; but it's similar to the given example solution to Problem3.

For the client side, I would take the user keyed in hostname, get the host according to the hostname, then try to send data.

Noticing that the time needed for sending the same amount of data using same configuration at different point of time can be different, for every different configuration of the Data Unit Size, I would repeat it 10 times and calculate the average.

The different between problem 3 and problem 4 is that, for problem 3 I send out all the package, then wait for the acknowledgement; while for problem 4, every time after I send out the package, I'll wait for the acknowledgement. The same goes for the server side: for problem 3 I send acknowledge after receiving all the packages; while for problem 4 I send out acknowledge every time after I receive a new package.

## Experiment Setup

I did the experiment in my lab. My laptop is connected to NUS wireless network, functions as a server; while my PC is connected to LAN within NUS, functions as a client.

For all the experiment I tested with sending the same file every time. The file size is 59793 byte.

The result & discussion is as shown below.

## Result & Discussion

Here's the data from the Lab3 - Send all without waiting for ack; and Lab4 - using stop&wait protocol.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Unit Size (Byte) | Lab 3 - Sending Time (ms) | Lab 3 - Through-Put (kB/s) | Lab 4 - Sending Time (ms) | Lab 4 Through-Put (kB/s) |
| 100 | 24.302 | 2460.465403 | 1823.604 | 32.788363 |
| 200 | 23.201 | 2577.140074 | 857.649 | 69.717324 |
| 300 | 23.101 | 2588.284693 | 632.736 | 94.499143 |
| 400 | 24.402 | 2450.382149 | 453.626 | 131.811199 |
| 500 | 27.802 | 2150.695821 | 390.322 | 153.188866 |
| 600 | 31.102 | 1922.487051 | 295.717 | 202.196832 |
| 700 | 25.101 | 2382.058371 | 273.815 | 218.369669 |
| 800 | 24.101 | 2480.90352 | 239.714 | 249.435055 |
| 900 | 20.401 | 2930.857008 | 210.912 | 283.497248 |
| 1000 | 19.701 | 3035.054414 | 191.711 | 311.891499 |

For problem 3, the sending time fluctuates between 20~30 ms, and the throughput fluctuates between 2000 ~ 3000 kB/s; and for problem 4, the sending time is inverse proportional to the unit package size, while the throughput is direct proportional to the unit package size.

***The result can be interpreted this way:***

Every time a package is sent out, there're some overhead added. The overhead fluctuates. Under the setup of problem 3 (send out all without ack), this overhead is big comparing with the sending time (around 20ms). Therefore for problem 3 the through put rate fluctuates because of this added overhead.

For lab4, the overhead mainly comes from the "wait" part of the stop&wait protocol, and it dominates in the overall time. Since the [***total\_number\_of\_package\_sent***] = ***file\_size / unit\_package\_size***, and the overall is linear to [***total\_number\_of\_package\_sent***]. Therefore, as unit package size increase linearly, the time\_cost decrease inverse proportionally, and the throughput increase linearly.