Thanh Ngo
Naftaly Minsky
Software Engineering
September 18, 2017

Project 1: Report

1. Description and Results

In this project, I measure the average latency of the TCP/IP networking protocol by sending and receiving various amounts of bytes with different configurations. There are two variables which are the message's size and the server address.

The programs used are:

- A server program that actively waits for a client message, an array of zero bytes delimited by a new line character. Then, it writes the same message back to the client.
- A client program which sends an array of zero bytes to the server, waits for the server's response, and calculate the overall time for that to occur.

There are three byte sizes being sent end-to-end which are 10 bytes, 1000 bytes or 10000 bytes. Two different types of server processes are put under the test which are a server process on the same host or one on another machine separated by a network (basic.cs.rutgers.edu specifically). I ran the client program 10 times for each configuration (byte sizes and host locations), and the results are documented in the tables below.

Noted that the documented interval includes both the latency of the network (if there is any), system calls, and the processing time on both sides which are considerably insignificant comparing to the system calls, and network latency.

Table 1. Latency for sending bytes between processes on the same machine.

	10 bytes	1000 bytes	10000 bytes
Trial #1	2ms	1ms	2ms
Trial #2	1ms	1ms	2ms
Trial #3	2ms	1ms	2ms
Trial #4	3ms	3ms	3ms
Trial #5	1ms	2ms	3ms

	10 bytes	1000 bytes	10000 bytes
Trial #6	2ms	2ms	3ms
Trial #7	2ms	2ms	3ms
Trial #8	2ms	2ms	2ms
Trial #9	1ms	2ms	2ms
Trial #10	1ms	1ms	2ms
Average	2ms	2ms	2ms

Table 2. Network Latency for sending bytes between processes on 2 network-segregated machines (the server process is on basic.cs.rutgers.edu machine and the client is my local machine)

	10 bytes	1000 bytes	10000 bytes
Trial #1	7ms	7ms	14ms
Trial #2	7ms	5ms	11ms
Trial #3	5ms	5ms	16ms
Trial #4	4ms	4ms	11ms
Trial #5	5ms	4ms	11ms
Trial #6	4ms	5ms	11ms
Trial #7	12ms	4ms	25ms
Trial #8	3ms	6ms	10ms
Trial #9	9ms	4ms	17ms
Trial #10	5ms	5ms	20ms
Average	6ms	5ms	15ms

2. Observation

Within a single host, the latency is small, and it varies from 1milliseconds to 3 milliseconds with different byte sizes. The average latency is 2 milliseconds for all message sizes. The anomaly in terms of latency does not exist.

Sending data beyond a single host introduces a significant latency. For 10 bytes, it takes 6 milliseconds on average to exchange 10 bytes which makes it three-fold longer than localhost communications. Similarly, on average, exchanging 1000 bytes takes 5 milliseconds, and exchanging 10000 bytes takes 15 milliseconds.

Besides that, with communication over the network, there are certain anomalies in the results which makes the data inconsistent overall. In trial #9, exchanging 10 bytes takes 12 milliseconds which doubles the average. While in trial #7, exchanging 10000 bytes takes 25 milliseconds, a 50% increase from the average.

On network-separated processes, there is one abnormal phenomenon; transferring 1000 bytes actually takes 5 milliseconds which is less than 6 milliseconds, the amount of time to exchange 10 bytes.

3. Conclusion

- IPC (Inter-process Communication) on a single machine using TCP/IP sockets has a significantly less latency than that on two different machines separated by a network.
- Communication over the network usually has certain anomalies (messages of the same size have a varying amount of time to transfer) due to the unstable nature of the network. For example, packets could be dropped along the way, or the message can be broken. Both of which requires retransmission.
- However, I was unable to explain how a larger message (1000 bytes) can be exchanged faster than a smaller one (10 bytes) in the networking environment.

4. Producing the Programs

1. Makefile

In the project folder, there is one makefile. The following targets can be made:

- all: produces both the client and the server
- client: the client program
- server: the server program
- clean: cleans up all the *.class file generated.

All compiled units are available within the folder containing this makefile such that we can run them right after compilation.

2. Server

After compilation for all target, the server program has the following formula:

java Server serving-port

Example:

```
$ java Server 12345
```

The server will not shut down until we send a Kill signal Ctrl+C.

3. Client

The client program has the following formula:

```
java Client server-address  server-serving-port  message-size-in-bytes
```

Example:

```
$ java Client localhost 12345 10

$ java Client basic.cs.rutgers.edu 12345 10000
```

4. Automation

I also wrote a bash script (./test.bash) to run the clients automatically and produce the results (10 iterations) for each byte size of the set 10-bytes, 100-bytes, 1000-bytes, and 10000-bytes. But before running the script, there should be a corresponding server waiting.

Formula:

```
./test.bash server-address port
```

Example:

```
$ ./test.bash localhost 12345
```

will produce the latency for a server on localhost, the client will run 10 times for each byte size.