CS 4513 Project 2

A Distributed Shell

Design

I created a command line flag to *server.c*, "*-e*", that allowed the program to bypass the call to *execvp*. This allowed me to measure the connection setup and tear-down time – I ran the server with this flag enabled and used the client to make repeated requests. Since the server would not actually execute the command specified to the client, I ran the client with the basic *ls* command.

The shell script I used to test the times measured the elapsed time using the *date* shell command - *start_time=`date +%N`*. Using the %N specifier allows for the measurement of time in nanoseconds. After executing the *rm* program, the script measured the time again - *end_time=`date+%N`*. The time taken by the program was obtained by the difference - *echo execution time was `expr $end_time - $start_time` nanoseconds*. I then converted this value to milliseconds.

I performed fifteen runs for calculating the connection latency.

To measure throughput, I created a text file on the server containing the entire text of the novel "Alice's Adventures in Wonderland" by Lewis Carroll. The text of the novel was obtained from Project Gutenberg.[1] I then ran the *cat alice.txt* command on the server using the client. I redirected the output of the client to a file.

I performed ten runs for calculating the throughput.

[1] The full text can be obtained at http://www.gutenberg.org/cache/epub/11/pg11.txt.

I performed this experiment on a laptop that was connected to the WPI Wireless network. I used the PuTTy software to connect to the CCC server (where I had developed my code), and to run the experiments using the shell. I was also running the Google Chrome browser, Microsoft Word, and WinSCP at the time on the laptop. The Microsoft Skydrive sync service was also running in the background, which may have contributed to a delay in the connection to the CCC server – since the service is built in to Windows 8, there was no way for me to turn it off.

Results

To measure latency, I ran the server with the *–e* flag and the *ls* command was sent using the distributed shell client.

Table 1: data on the connection latency

Run	Connection latency (nanoseconds)	Connection latency (milliseconds)
1	238725000	238.7
2	630725000	630.7
3	42667000	426.7
4	366524000	366.5
5	44855000	448.6
6	42870000	428.7
7	44359000	443.6
8	41811000	418.1
9	43346000	433.4
10	43965000	439.7

11	45037000	450.4
12	45133000	451.3
13	43451000	434.5
14	43471000	434.7
15	45324000	453.2
Mean	**117484200**	**433.3**
Standard Deviation	**170126149.6**	**77.0**

The throughput was measured by making a request for the file *alice.txt*. The size of the file was:

du alice.txt -b

167513 alice.txt

With the server running (without the –*e* option), I ran the command:

./dsh -s cccwork3.wpi.edu -c "cat alice.txt" > test

This program took 73997000 nanoseconds to execute, which is 74 ms. Thus, the throughput was 2.264 x 10^6 bytes per second (Bps). The full set of results are as in the table below-

Table 2: Results for throughput

Filename	**Size (bytes)**	**Time (milliseconds)**	**Throughput (10^6 Bps)**

alice.txt	167513	122.625	1.37
alice.txt	167513	94.224	1.78
alice.txt	167513	72.419	2.31
alice.txt	167513	99.146	1.69
alice.txt	167513	80.373	2.08
alice.txt	167513	70.668	2.37
alice.txt	167513	88.015	1.9
alice.txt	167513	68.531	2.44
alice.txt	167513	66.691	2.51
alice.txt	167513	75.56	2.22
Mean	**167513**	**83.8252**	**2.067**
Standard Deviation	**0**	**17.56504973**	**0.372**

Analysis

The average latency observed is 433.3 milliseconds. However, there is great variation observed in the results for the latency. I hypothesize that the variation in time is due to network delays, competition for network resources on the host computer, and possible delays between caching and flushing to disk. It is also important to note that this is not a perfect measure of latency, since there is an *if* condition that is used to determine whether or not the *execvp* command should be executed, which could add to the delay.

The data on the throughput shows less variation, thereby indicating that the average throughput of 2.067×10^6 bytes per second (Bps) is a fairly good indicator.