CSE232: Assignment 3

Analysis of synchronous and asynchronous socket calls

Deadline: November 6, 2023

Note: This is a lengthy assignment. If you do not start by tomorrow you will not be able to finish it on time.

Warm-up

1. Watch this video to have a conceptual understanding of I/O multiplexing techniques such as select, poll, and epoll calls in Linux. This was also done in Tutorial 4.
2. Understand traditional, synchronous TCP socket programming from Tutorial 2.
3. The code for different I/O multiplexing techniques is provided here. Your task is to understand every line of the code. You may choose to use this code with modifications or write code from scratch for this assignment. We would want you to choose the latter.
4. Understand how to set up your system for TCP client/server experiments.
5. Please go through the FAQ before starting the assignment.

The objective of this assignment is to analyze the performance of synchronous and asynchronous TCP servers. Write the TCP server socket programs in C language only.

TCP client:
We have provided the concurrent TCP client program here, and its usage is explained in the README file.

The assignment comprises the following tasks.

1. Write the following **TCP server programs:**
    a. Concurrent server program with **multiple processes** (using fork system call) **[3]**
    b. Concurrent server program with **multiple threads** (using pthreads) **[3]**
    c. **Non-blocking** server programs that implement I/O multiplexing using
        i. select() system call **[5]**
        ii. poll() system call **[5]**
        iii. epoll API **[5]**

        All non-blocking designs must support 4000 concurrent connections (i.e., listener file descriptors).
2. **Server function:** The server program (all designs in (1)), after accepting the incoming request, does the following. **[points included in (1)]**
    a. Read the payload and cast it to a 64-bit unsigned integer, n.
    b. Write a function, fact(n), to compute the factorial of the “n”; if n>20, then calculate the factorial of 20.
    c. Send the factorial result as the response message back to the client.
3. For each server program in (1), perform separate experiments with **number of concurrent client connections equal to 500, 1000, and 3000**. **Each connection**

generates 50 requests. For each combination of **<server_prog_type> <num_concurrent_clients>**, do the following.

a. **Data collection:** You must capture all the packets communicated between the client and the server on the server's interface using a tool such as tcpdump or wireshark. **[2]**
b. **Write python scripts** that take the pcap file generated in (a), and computing the following metrics for each server experiment. **Show your observations in a plot**.
    i. Average throughput for each TCP flow (in bits per second) **[2.5]**
    ii. Average latency for each TCP flow (in milliseconds) **[2.5]**
c. Observe the following metrics. **Show your observations in a table and analyze/justify it.**
    i. Server process's CPU utilization (use "top" or "htop" command) **[2.5]**
    ii. Server process's memory utilization (use "time -v" command) **[2.5]**

Note:
Do not use any print statements (while sending/processing requests) when you are observing the metrics. Such I/O operations will slow down the process.

—----------