Homework 2

Due: Friday 11:59 PM, March 8, 2024

1. (6 points) Answer the following:
    a. Identify the architecture where a server with permanent IP address has always-on host capability to provide service. Provide an example of the application of this model.
    b. Identify the architecture where a server with temporary IP address host file services does not have the always-on-host capability to share files. Provide an examples application of this model.
    c. Which architecture do you find ideal in your day-to-day application usage? Give some examples.

2. (4 points) Differentiate between inter-process communication (IPC) in multi-machines versus the single machine using examples? What programming interface do you use for IPC between two hosts?

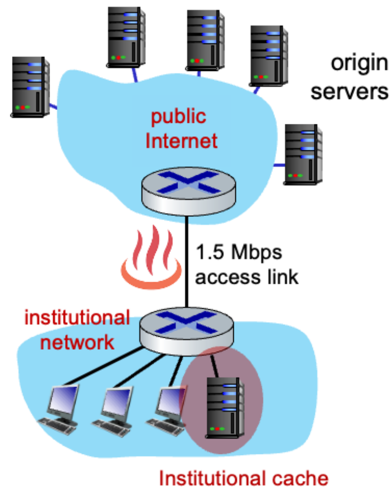
3. (5 points) Imagine you are designing a communication system for a real-time online gaming application. Players need to exchange information about their current positions, actions, and updates in the game environment. Additionally, the system should prioritize low latency to ensure a smooth gaming experience.

   Based on the given scenario, analyze the requirements of the online gaming application and propose the most suitable transport protocol for the communication between players. Justify your choice by considering factors such as reliability, latency, and the nature of the data being transmitted.

4. (5 points) Suppose that you have two browser applications open and active at the same time and that both applications are accessing the same server to retrieve HTTP documents at the same time. How does the server know how to tell the difference between the two applications (to send the correct document(s) to the correct application)?

5. (10 points) Suppose that we want to distribute a file with a size of $F = 10$ Gbits to $N = 100$ clients/peers. The server supports an upload rate of $u_s = 20$ Mbps while each client/peer has a download rate of $d_i = 3$ Mbps and an upload rate of u, where $u = 500$ Kbps or 800 Kbps. Show your calculations.
    a. Calculate the minimum distribution time for a client-server distribution using the two values of u given above.
    b. Calculate the minimum distribution time for a peer-to-peer distribution using the two values of u given above.

6. (10 Points)Consider the following simplified network diagram where there is an institutional network connected to the Internet:



Suppose that the average object size is 60,000 bits and that the average request rate from the institution's browsers to the origin server is 23 requests per second. Also, suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is two seconds on average. Model the total average response time as the sum of the average access delay and the average Internet delay. For the average access delay, use $\Delta/(1-\Delta\beta)$, where Δ is the average time required to send an object over the access link and β is the arrival rate of objects to the access link. You can assume that the HTTP request messages are negligibly small and thus create no realized traffic on the network or the access link. Show your calculations.

a. Find the total average response time.

b. Now, suppose a cache is installed in the institutional LAN (see figure). Suppose the miss rate is 0.4. Find the total response time.

7. (5 points) Suppose that you join BitTorrent as a new peer without possessing any chunks. Unfortunately, you cannot become a top-4 uploader for any of your peers since you do not have anything to upload. Describe how you will be able to get your first chunk. Be specific.

8. (5 points) Identify at least one reason why DNS uses UDP instead of TCP for its query and response messages. Justify your answer.

9. (55 points) UDP Pinger programming assignment: In this assignment, you will write two complete programs to support a client/server model using Linux sockets for a UDP "ping" utility, similar to the ping utility already available on ECS coding machines. You can either use Python of C programming language for this assignment

Server

- The server program will be called with one command-line argument, the port number being used, such as `python3 pingsvr.py 8001`. If the user calls the server program with too few or too many arguments, you will print out a usage statement and terminate the program.
- The server will set up a UDP socket on the Internet (i.e., INET) domain and then wait in an infinite loop listening for incoming UDP packets, specifically PING messages from a client. Your server should be able to support multiple clients at the same time (though no extra work is expected to support this requirement).
- *Packet Loss*: UDP provides applications with an unreliable transport service. Messages may get lost in the network due to a variety of reasons. Since packet loss is rare or even non-existent in typical campus networks, including Sac State's, the server in this project will inject artificial loss to simulate the effects of network packet loss. The server will simulate 30% packet loss through generation of a seeded, randomized integer that will determine whether a particular incoming PING message is lost or not.
- When a `PING` message comes in from a client and if the packet is not lost, the server will print the client message to the terminal and then send a `PONG` message back to the client. If the packet is determined to be lost, the server will print an appropriate message to the terminal and simply "eat" the message by not responding to the client.
- The server will remain "always on" until a user enters `Ctrl-C (^C)` to send an interrupt signal to the program to terminate.

Client

- The client program will be called with two command-line arguments, the hostname of the server and the port number being used, such as `python3 pingcli.py ecs-coding1.csus.edu 8001`. If the user calls the client program with too few or too many arguments, you will print out a usage statement and terminate the program. You will run this command on ecs-coding2.csus.edu.

- The client will send 10 automated `PING` messages to the server using a UDP socket, where automated means the message is built in the code, not entered from the keyboard. Because UDP is an unreliable protocol, a packet sent from the client to the server may be lost in the network, or vice versa. For this reason, the client cannot wait indefinitely for a reply to a `PING` message. You should get the client to wait up to one second for a reply – if no reply is received within one second, your client program should assume that the packet was lost during transmission across the network.

- Specifically, for each of the 10 `PING` messages, your client program should:
  - send the `PING` message using the UDP socket and print a status message.

  - if the response message is received from the server, calculate and print the round trip time (RTT) in milliseconds for each message; otherwise, print a status message that it timed out.

- After all of the `PING` messages have been sent (and responses received or timed out), the client program should report the following and then terminate:
  - the number of messages sent, the number of messages received, and the message loss rate (as a percentage);
  - the minimum, maximum, and average RTTs for all of the `PING` messages in milliseconds.

Your program should run on the INET domain using SOCK_DGRAM (i.e., UDP) sockets so that the server and the client execute on a different ECS machine. You will also need to make sure you are able to handle any error cases.

SAMPLE OUTPUT (user input shown in bold):

==> SERVER on ecs-coding1

```
[badruddoja@ecs-pa-coding1 TEST]$ python pingsvr.py
Usage: python3 pingsvr.py <port>
[badruddoja@ecs-pa-coding1 TEST]$ python pingsvr.py 8001
Server listening on port 8001
Packet loss - Message dropped.
Received from ('130.86.188.33', 50320): PING 2
Received from ('130.86.188.33', 50320): PING 3
Packet loss - Message dropped.
Received from ('130.86.188.33', 50320): PING 5
Received from ('130.86.188.33', 50320): PING 6
Received from ('130.86.188.33', 50320): PING 7
Packet loss - Message dropped.
Received from ('130.86.188.33', 50320): PING 9
Received from ('130.86.188.33', 50320): PING 10
```

==> CLIENT on ecs-coding1

```
[badruddoja@ecs-pa-coding1 TEST]$ python pingcli.py
Usage: python3 pingcli.py <server_host> <server_port>
[badruddoja@ecs-pa-coding1 TEST]$ python pingcli.py ecs-
coding1.csus.edu 8001
Request timed out for PING 1
Received from ('130.86.188.33', 8001): PONG - RTT: 3.72 ms
Received from ('130.86.188.33', 8001): PONG  - RTT: 1.40 ms
Request timed out for PING 4
Received from ('130.86.188.33', 8001): PONG - RTT: 1.32 ms
Received from ('130.86.188.33', 8001): PONG  - RTT: 0.51 ms
Received from ('130.86.188.33', 8001): PONG. - RTT: 0.88 ms
```

```
Request timed out for PING 8
Received from ('130.86.188.33', 8001): PONG.  - RTT: 1.58 ms
Received from ('130.86.188.33', 8001): PONG.  - RTT: 1.34 ms

Ping statistics:
  Packets: Sent = 10, Received = 7, Lost = 3 (70.00% loss)



RTT statistics:
 Minimum RTT: 0.51 ms, Maximum RTT: 3.72 ms, Average RTT: 1.54 ms
```

Note:

- Ideally, we should try the client and server programs in two different machines. Due to ECS server restrictions and rules, running the programs and getting the desired output may be impossible. Hence, you can try the code in a single machine to get the desired output.
- Maximum points will not exceed 100

Code Requirements:

- Your code should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, course section, date, and brief description), comments for each variable, and commented blocks of code.
- Your programs should be named “pingsvr.py” and “pingcli.py”, without the quotes, for the server and client code, respectively.
- Your program will be graded based largely on whether it works correctly on the ECS machines (e.g., ecs-coding1, ecs-coding2, …, ecs-coding3), so you should make sure that your program compiles and runs on a ECS machine.
- Please pay attention to the SAMPLE OUTPUT for how this program is expected to work. If you have any questions about this, please contact your instructor, assigned to this course to ensure you understand these directions.

Submission Files:

- You will submit a pdf file consisting of answers to questions 1-4.
- Additionally, you will submit pingsvr.py and pingcli.py separately on canvas.