Name-Sugandh Mishra

Reg-20204211

Sec-CSE C



Motilal Nehru National Institute of TechnologyAllahabad Prayagraj-211004 [India]

Department of Computer Science & Engineering

Programme Name: B.Tech Semester: VII Branch: Computer Science & Engg.

Course Code: CS17201 Course Name: Distributed Systems (Lab)

Lab Assignment 6

Lab Name of Experiment # Suppose you have two TCP servers for converting a lower case string to upper case string. You have to design a load balancer server that accept lower case string from client and check for the CPU utilization of both servers. Load balancer will transfer the string to the server having less CPU utilization. The load balancer will get upper case string from server and return to the clients. 2. Get CPU load Broker server TCP 1. string server 1 6. Compare client CPU load 4,5. CPU load 9. String TCP (upper case) 3. Get CPU load server 2 7. string 8. String (upper case)

Load_balancer.c----

```
#include <stdio.h>
#include <stdib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define LOAD_BALANCER_PORT 8888
#define SERVER_COUNT 2
#define SERVER1_IP "127.0.0.1"
#define SERVER1_PORT 8889
#define SERVER2_IP "127.0.0.1"
#define SERVER2_IP "127.0.0.1"
```

```
int main() {
   int loadBalancerSocket, serverSockets[SERVER_COUNT];
    struct sockaddr_in loadBalancerAddr, serverAddrs[SERVER_COUNT];
   loadBalancerSocket = socket(AF_INET, SOCK_STREAM, 0);
   // Initialize the load balancer address structure
   memset(&loadBalancerAddr, 0, sizeof(loadBalancerAddr));
   loadBalancerAddr.sin_family = AF_INET;
    loadBalancerAddr.sin_addr.s_addr = INADDR_ANY;
    loadBalancerAddr.sin_port = htons(LOAD_BALANCER_PORT);
    // Bind the load balancer socket
   bind(loadBalancerSocket, (struct sockaddr *)&loadBalancerAddr,
sizeof(loadBalancerAddr));
   listen(loadBalancerSocket, SERVER_COUNT);
    // Create sockets for Server 1 and Server 2
   serverSockets[0] = socket(AF_INET, SOCK_STREAM, 0);
    serverSockets[1] = socket(AF_INET, SOCK_STREAM, 0);
   // Initialize server addresses
   memset(&serverAddrs[0], 0, sizeof(serverAddrs[0]));
    serverAddrs[0].sin_family = AF_INET;
    serverAddrs[0].sin_addr.s_addr = inet_addr(SERVER1_IP);
    serverAddrs[0].sin_port = htons(SERVER1_PORT);
   memset(&serverAddrs[1], 0, sizeof(serverAddrs[1]));
    serverAddrs[1].sin_family = AF_INET;
    serverAddrs[1].sin_addr.s_addr = inet_addr(SERVER2_IP);
    serverAddrs[1].sin_port = htons(SERVER2_PORT);
   // Connect to Server 1 and Server 2
    connect(serverSockets[0], (struct sockaddr *)&serverAddrs[0], sizeof(serverAddrs[0]));
    connect(serverSockets[1], (struct sockaddr *)&serverAddrs[1], sizeof(serverAddrs[1]));
    int currentServer = 0; // Variable to keep track of the selected server
   while (1) {
       int clientSocket;
       // Accept an incoming connection from a client
        clientSocket = accept(loadBalancerSocket, NULL, NULL);
        printf("Accepted connection from a client.\n");
        char buffer[1024];
        ssize_t bytesRead;
        bytesRead = read(clientSocket, buffer, sizeof(buffer));
        printf("Received from client: %s\n", buffer);
```

```
// Forward the message to the selected server
       write(serverSockets[currentServer], buffer, bytesRead);
       printf("Forwarded to server %d: %s\n", currentServer + 1, buffer);
       char serverResponse[1024]; // Response buffer for server response
       ssize_t serverResponseBytes;
       // Receive the response from the server
       serverResponseBytes = read(serverSockets[currentServer], serverResponse,
sizeof(serverResponse));
       printf("Received from server %d: %s\n", currentServer + 1, serverResponse);
       // Send the response back to the client
       write(clientSocket, serverResponse, serverResponseBytes);
       printf("Sent response to client: %s\n", serverResponse);
       // Close the client socket
       close(clientSocket);
       // Switch to the other server in a round-robin fashion
       currentServer = (currentServer + 1) % SERVER_COUNT;
   // Close sockets and clean up (not reached in this simplified example)
   close(loadBalancerSocket);
   close(serverSockets[0]);
   close(serverSockets[1]);
    return 0;
```

Server1.c----

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <ctype.h>
#define SERVER1_PORT 8889
int main() {
    int serverSocket;
    struct sockaddr_in serverAddr;
    // Create a socket for Server 1
    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    // Initialize the server address structure
    memset(&serverAddr, 0, sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    serverAddr.sin_port = htons(SERVER1_PORT);
    // Bind the server socket
```

```
bind(serverSocket, (struct sockaddr *)&serverAddr, sizeof(serverAddr));
// Listen for incoming connections
listen(serverSocket, 5);
while (1) {
   int clientSocket;
    char buffer[1024];
    ssize_t bytesRead;
    // Accept an incoming connection from a load balancer
    clientSocket = accept(serverSocket, NULL, NULL);
    printf("Accepted connection from the load balancer.\n");
    // Receive the message from the load balancer
    bytesRead = read(clientSocket, buffer, sizeof(buffer));
    printf("Received from the load balancer: %s\n", buffer);
    // Process the message (e.g., convert to uppercase in this example)
    for (int i = 0; i < bytesRead; i++) {
        buffer[i] = toupper(buffer[i]);
    // Send the processed message back to the load balancer
    write(clientSocket, buffer, bytesRead);
    printf("Processed message: %s\n", buffer);
    // Close the client socket
    close(clientSocket);
// Close the server socket (not reached in this simplified example)
close(serverSocket);
return 0;
```

server2.c----

```
#include <stdio.h>
#include <stdib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <ctype.h>
#define SERVER2_PORT 8890

int main() {
    int serverSocket;
    struct sockaddr_in serverAddr;

    // Create a socket for Server 2
    serverSocket = socket(AF_INET, SOCK_STREAM, 0);

// Initialize the server address structure
    memset(&serverAddr, 0, sizeof(serverAddr));
```

```
serverAddr.sin_family = AF_INET;
serverAddr.sin_addr.s_addr = INADDR_ANY;
serverAddr.sin_port = htons(SERVER2_PORT);
// Bind the server socket
bind(serverSocket, (struct sockaddr *)&serverAddr, sizeof(serverAddr));
listen(serverSocket, 5);
while (1) {
   int clientSocket;
    char buffer[1024];
    ssize_t bytesRead;
    // Accept an incoming connection from a load balancer
    clientSocket = accept(serverSocket, NULL, NULL);
    printf("Accepted connection from the load balancer.\n");
    // Receive the message from the load balancer
    bytesRead = read(clientSocket, buffer, sizeof(buffer));
    printf("Received from the load balancer: %s\n", buffer);
    // Process the message (e.g., convert to lowercase in this example)
    for (int i = 0; i < bytesRead; i++) {
        buffer[i] = tolower(buffer[i]);
    write(clientSocket, buffer, bytesRead);
    printf("Processed message: %s\n", buffer);
    // Close the client socket
    close(clientSocket);
// Close the server socket (not reached in this simplified example)
close(serverSocket);
return 0;
```

client.c -----

```
#include <stdio.h>
#include <stdib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int clientSocket;
    struct sockaddr_in serverAddr;

// Create a socket for the client
    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
```

```
// Initialize the server address structure
  memset(&serverAddr, 0, sizeof(serverAddr));
   serverAddr.sin_family = AF_INET;
   serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); // Change to the load balancer's
   serverAddr.sin_port = htons(8888); // Use the load balancer's port
   // Connect to the load balancer
   connect(clientSocket, (struct sockaddr *)&serverAddr, sizeof(serverAddr));
   char message[1024];
  printf("Enter a message to send to the load balancer: ");
   fgets(message, sizeof(message), stdin);
  message[strcspn(message, "\n")] = '\0'; // Remove the newline character
  write(clientSocket, message, strlen(message));
   char buffer[1024];
   ssize_t bytesRead;
   // Receive the response from the load balancer
  bytesRead = read(clientSocket, buffer, sizeof(buffer));
  printf("Response from Load Balancer: %s\n", buffer);
  // Close the client socket
   close(clientSocket);
  return 0;
                                                                1: load_balancer, servε ∨ + □ ඕ
       PROBLEMS OUTPUT
                          TERMINAL
                                    DEBUG CONSOLE
宀
       Avinashs-MacBook-Air:di
                                Avinashs-MacBook-Air:a
                                                                                      Avinashs-MacBook-Air:as
                                                        Avinashs-MacBook-Air:as6 anu
                                                        rag$ gcc server2.c -o server
       s sys lab anurag$ cd as
                                s6 anurag$ gcc server1
                                                                                      6 anurag$ gcc client.c
                                .c -o server1
                                                                                      -o client
                                                                                     Avinashs-MacBook-Air:as 6 anurag$ ./client
       Avinashs-MacBook-Air:as
                                Avinashs-MacBook-Air:a
                                                        Avinashs-MacBook-Air:as6 anu
       6 anurag$ gcc load_bala
                                s6 anurag$ ./server1
                                                        rag$ ./server2
       ncer.c -o load_balancer
                                      Accepted connect
                                                        Accepted connection from the
                                                                                      Enter a message to send
                                                         load balancer.
       Avinashs-MacBook-Air:as
                                ion from the load bala
                                                                                      to the load balancer:
       6 anurag$ ./load_balanc
                                ncer.
                                                        Received from the load balan
                                                                                      hello i am sugandh
                                                                                      Response from Load Bala
ncer: HELLO I AM SUGAND
                                Received from the load
                                                        cer: reg 20204211
                                                        Processed message: reg 20204
       Accepted connection fro
                                 balancer: hello i am
       m a client.
                                sugandh
                                                        211
       Received from client: h
                                Processed message: HEL
                                                                                      Avinashs-MacBook-Air:as
                                LO I AM SUGANDH
       ello i am sugandh
                                                                                      6 anurag$ ./client
       Forwarded to server 1:
                                                                                      Enter a message to send
                                                                                      to the load balancer:
reg 20204211
       hello i am sugandh
       Received from server 1:
HELLO I AM SUGANDH
                                                                                      Response from Load Bala
                                                                                     ncer: reg 20204211
Avinashs-MacBook-Air:as
       Sent response to client
       : HELLO I AM SUGANDH
       Accepted connection fro
                                                                                      6 anurag$
       m a client.
       Received from client: r
       eg 20204211ugandh
       Forwarded to server 2:
       reg 20204211ugandh
(<u>P</u>)
       Received from server 2:
        reg 20204211UGANDH
      : reg 20204211UGANDH
       Sent response to client
                                                        Ln 31, Col 46 Spaces: 4 UTF-8 LF C P Go Live 👨 🗘
\otimes 0 \wedge 0
```