Practical - 9

Aim : Assume that processes communicate by send/receive messages, which are unreliable, e.g. messages may be lost during send/recv. After sending a message, a process expects a reply. If it does not receive a reply within ‘t’ seconds, it re-sends the same message again. If it receives a reply within ‘t’ seconds, it must not send the same message again. Design an algorithm for the sending process and implement it.

Code:
server.c :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <netinet/in.h>
#include <string.h>
#include <sys/socket.h>
#define PORT 8080

int server_fd, new_socket, valread;
struct sockaddr_in address;
int opt = 1;
int addrlen = sizeof(address);
int true = 1;
void start() {
	if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
		perror("Connection failed");
		exit(EXIT_FAILURE);
	}
	if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,sizeof(opt))) {
		perror("setsockopt");
		exit(EXIT_FAILURE);
	}
	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons(PORT);
	if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
		perror("bind failed");
		exit(EXIT_FAILURE);
	}
	if (listen(server_fd, 3) < 0) {
		perror("listen");
		exit(EXIT_FAILURE);
	}
}

int main(int argc, char const *argv[]) {
	char buffer[1024] = {0};
	char *hello = "Hello!";
	start();
	setsockopt(new_socket, SOL_SOCKET, SO_REUSEADDR, &true, sizeof(int));
	if ((new_socket = accept(server_fd, (struct sockaddr *)&address,(socklen_t *)&addrlen)) < 0) {
		perror("accept");
		exit(EXIT_FAILURE);
```

```
        }
        valread = read(new_socket, buffer, 1024);
        printf("%s\n", buffer);
        printf("\n[!] Sleeping for 4 seconds!\n");
        close(new_socket);
        sleep(3);
        if ((new_socket = accept(server_fd, (struct sockaddr *)&address,(socklen_t *)&addrlen)) < 0) {
                perror("accept");
                exit(EXIT_FAILURE);
        }
        // start();
        valread = read(new_socket, buffer, 1024);
        printf("Message received: %s\n", buffer);
        send(new_socket, hello, strlen(hello), 0);
        printf("Hello message sent\n");
        return 0;
}
```

client.c:

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <time.h>
#define PORT 8080

int main() {
        int sent = 0;
        int _received = 0;
        int valread = 0;
        char buffer[1024] = {0};
        // while not received,
        while (!_received) {
                int sock = 0;
                struct sockaddr_in serv_addr;
                char *hello = "Hello from client";
                if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
                        printf("\n Socket creation error \n");
                        return -1;
                }
                serv_addr.sin_family = AF_INET;
                serv_addr.sin_port = htons(PORT);
                if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
                        printf("\nInvalid address/ Address not supported \n");
                        return -1;
                }
                if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
                        printf("\nConnection Failed \n");
                        return -1;
                }
                time_t startTime = time(NULL);
```

```
            if (sent) {
                printf("\n[!] Sending again.\n");
            }
            send(sock, hello, strlen(hello), 0);
            printf("[+] Hello message sent\n");
            sent = 1;
            printf("[!] Timer for 2 seconds started.\n");
            valread = read(sock, buffer, 1024);
            if (valread) break;
            while (!valread) {
                if (time(NULL) - startTime > 2) {
                printf("[-] Timeout!");
                break;
                }
                valread = read(sock, buffer, 1024);
                if (valread) _received = 1;
            }
        }
        printf("[!] Message received >> %s\n\n", buffer);
        return 0;
}
```

Output:

```
bhishm@BhishmDaslaniya: /media/bhishm/Projects/Internals_of_Operating_System_Lab/Practi
bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/
Practical_9$ ./server
Hello from client

[!] Sleeping for 4 seconds!
Message received: Hello from client
Hello message sent
bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/
Practical_9$
```

```
bhishm@BhishmDaslaniya: /media/bhishm/Projects/Internals_of_Operating_System_Lab/Practi
bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/
Practical_9$ ./client
[+] Hello message sent
[!] Timer for 2 seconds started.
[-] Timeout!
[!] Sending again.
[+] Hello message sent
[!] Timer for 2 seconds started.
[!] Message received >> Hello!

bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/
Practical_9$
```