

**Computer Networks**  
**RA1911030010014**  
**Experiment - 4**  
**UDP Echo Client-Server Communication**

**Aim:** To create a UDP Echo Client-Server Communication.

**Server Code:**

```
/**
 * RA1911030010014 - UDP Server
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT 4040
#define MAXLINE 1024

int main()
{
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr, cliaddr;
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
    {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }
    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    if (bind(sockfd, (const struct sockaddr *)&servaddr,
              sizeof(servaddr)) < 0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    else
    {
        printf("Server listening on Port %d\n", PORT);
    }
    int len, n;
```

```
len = sizeof(cliaddr);
n = recvfrom(sockfd, (char *)buffer, MAXLINE,
             MSG_WAITALL, (struct sockaddr *)&cliaddr,
             &len);
buffer[n] = '\0';
printf("Client Sent : %s\n", buffer);
sendto(sockfd, (const char *)buffer, strlen(buffer),
        MSG_CONFIRM, (const struct sockaddr *)&cliaddr,
        len);
return 0;
}
```

### Client Code:

```
/**
 * RA1911030010014 - UDP Client
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

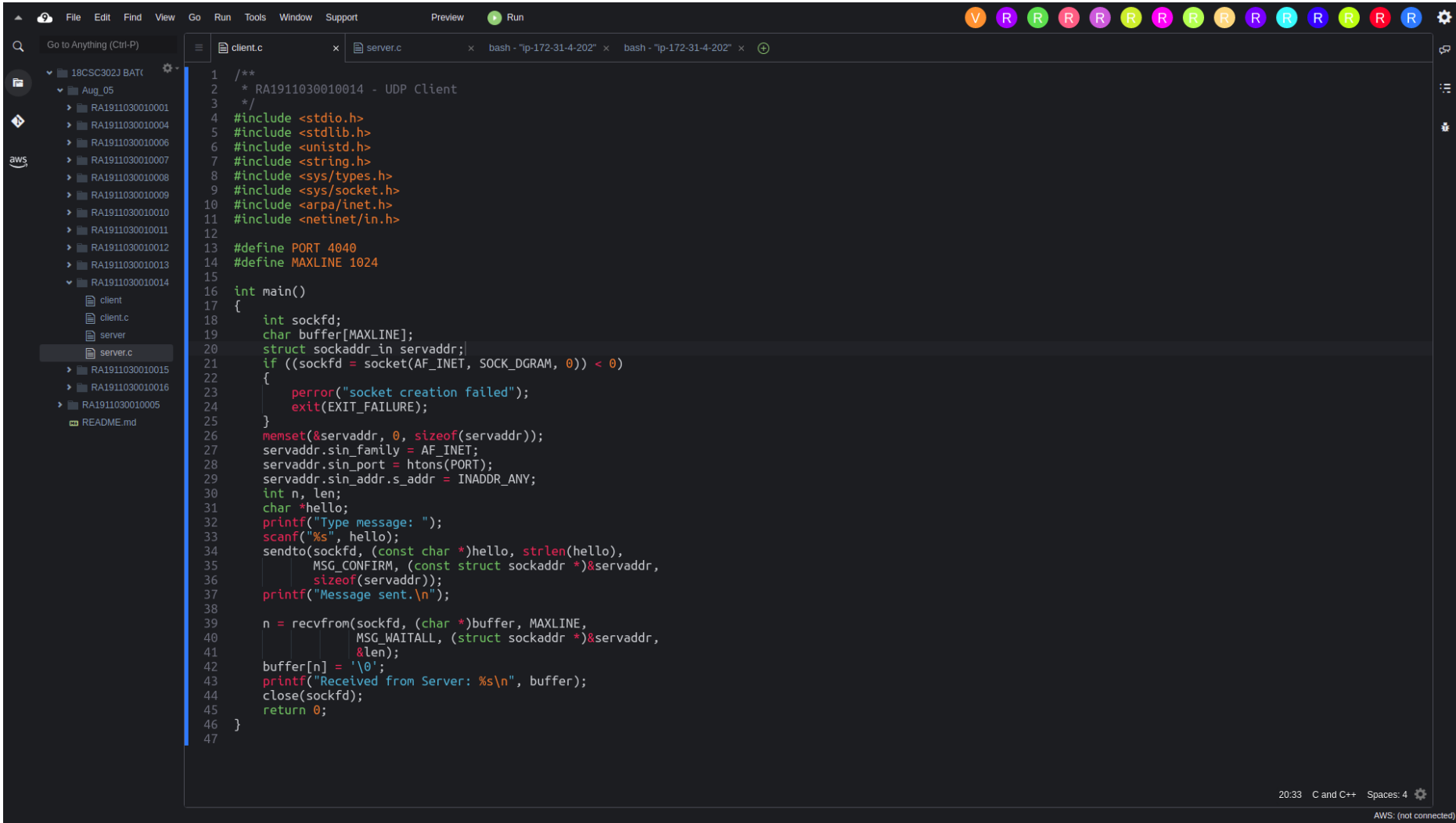
#define PORT 4040
#define MAXLINE 1024

int main()
{
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr;
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
    {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;
    int n, len;
    char *hello;
    printf("Type message: ");
    scanf("%s", hello);
    sendto(sockfd, (const char *)hello, strlen(hello),
           MSG_CONFIRM, (const struct sockaddr *)&servaddr,
```

```
        sizeof(servaddr));
printf("Message sent.\n");

n = recvfrom(sockfd, (char *)buffer, MAXLINE,
               MSG_WAITALL, (struct sockaddr *)&servaddr,
               &len);
buffer[n] = '\0';
printf("Received from Server: %s\n", buffer);
close(sockfd);
return 0;
}
```

Output:



18CSC302J BATC

Aug\_05

RA1911030010001

RA1911030010004

RA1911030010006

RA1911030010007

RA1911030010008

RA1911030010009

RA1911030010010

RA1911030010011

RA1911030010012

RA1911030010013

RA1911030010014

client

client.c

server

server.c

RA1911030010015

RA1911030010016

RA1911030010005

README.md

client.c

server.c

bash - "ip-172-31-4-202"

bash - "ip-172-31-4-202"

```
1  /**
2   * RA1911030010014 - UDP Server
3   */
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <unistd.h>
7  #include <string.h>
8  #include <sys/types.h>
9  #include <sys/socket.h>
10 #include <arpa/inet.h>
11 #include <netinet/in.h>
12
13 #define PORT 4040
14 #define MAXLINE 1024
15
16 int main()
17 {
18     int sockfd;
19     char buffer[MAXLINE];
20     struct sockaddr_in servaddr, cliaddr;
21     if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
22     {
23         perror("socket creation failed");
24         exit(EXIT_FAILURE);
25     }
26     memset(&servaddr, 0, sizeof(servaddr));
27     memset(&cliaddr, 0, sizeof(cliaddr));
28     servaddr.sin_family = AF_INET;
29     servaddr.sin_addr.s_addr = INADDR_ANY;
30     servaddr.sin_port = htons(PORT);
31
32     if (bind(sockfd, (const struct sockaddr *)&servaddr,
33             sizeof(servaddr)) < 0)
34     {
35         perror("bind failed");
36         exit(EXIT_FAILURE);
37     }
38     else
39     {
40         printf("Server listening on Port %d\n", PORT);
41     }
42     int len, n;
43     len = sizeof(cliaddr);
44     n = recvfrom(sockfd, (char *)buffer, MAXLINE,
45                 MSG_WAITALL, (struct sockaddr *)&cliaddr,
46                 &len);
47     buffer[n] = '\0';
48     printf("Client Sent : %s\n", buffer);
49     sendto(sockfd, (const char *)buffer, strlen(buffer),
50           MSG_CONFIRM, (const struct sockaddr *)&cliaddr,
51           len);
52
53 }
```

53.2 C and C++ Spaces: 4

AWS: (not connected)

18CSC302J BATC

Aug\_05

RA1911030010001

RA1911030010004

RA1911030010006

RA1911030010007

RA1911030010008

RA1911030010009

RA1911030010010

RA1911030010011

RA1911030010012

RA1911030010013

RA1911030010014

client

client.c

server

server.c

RA1911030010015

RA1911030010016

RA1911030010005

README.md

client.c

server.c

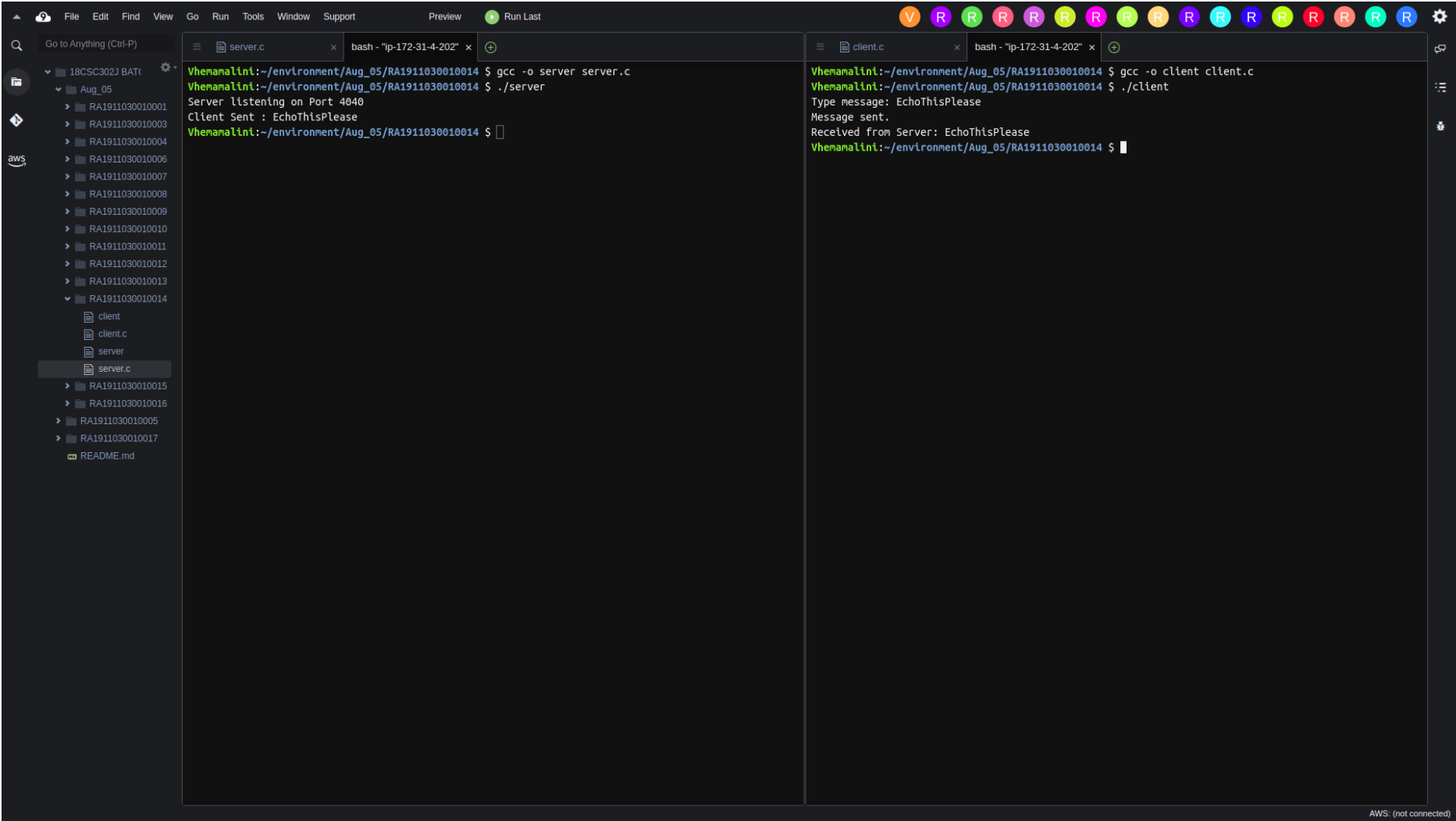
bash - "ip-172-31-4-202"

bash - "ip-172-31-4-202"

```
13 #define PORT 4040
14 #define MAXLINE 1024
15
16 int main()
17 {
18     int sockfd;
19     char buffer[MAXLINE];
20     struct sockaddr_in servaddr, cliaddr;
21     if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
22     {
23         perror("socket creation failed");
24         exit(EXIT_FAILURE);
25     }
26     memset(&servaddr, 0, sizeof(servaddr));
27     memset(&cliaddr, 0, sizeof(cliaddr));
28     servaddr.sin_family = AF_INET;
29     servaddr.sin_addr.s_addr = INADDR_ANY;
30     servaddr.sin_port = htons(PORT);
31
32     if (bind(sockfd, (const struct sockaddr *)&servaddr,
33             sizeof(servaddr)) < 0)
34     {
35         perror("bind failed");
36         exit(EXIT_FAILURE);
37     }
38     else
39     {
40         printf("Server listening on Port %d\n", PORT);
41     }
42     int len, n;
43     len = sizeof(cliaddr);
44     n = recvfrom(sockfd, (char *)buffer, MAXLINE,
45                 MSG_WAITALL, (struct sockaddr *)&cliaddr,
46                 &len);
47     buffer[n] = '\0';
48     printf("Client Sent : %s\n", buffer);
49     sendto(sockfd, (const char *)buffer, strlen(buffer),
50           MSG_CONFIRM, (const struct sockaddr *)&cliaddr,
51           len);
52     return 0;
53 }
```

53.2 C and C++ Spaces: 4

AWS: (not connected)



**Result:**

The required code for UDP Echo Client-Server Communication was written in the AWS Cloud9 environment and successfully compiled.