PRAKTIKUM 10
DATAGRAM SOCKET PROGRAMMING

I. Tujuan

1. Mahasiswa Mengenal konsep Client Server
2. Mahasiswa Mengenal protokol TCP/IP untuk Interaksi Client Server
3. Mahasiswa Mengenal konsep API
4. Mahasiswa Mengenal Datagram Socket
5. Mahasiswa Memahami Konsep Pembuatan Pemrograman Datagram Socket untuk Komunikasi Client-Server
6. Mahasiswa Mampu Membuat Pemrograman Datagram Socket untuk Komunikasi Client-Server

II. Dasar Teori

Datagram Socket

Pada sistem operasi linux ada banyak socket, tetapi ada 2 yang paling utama yaitu stream socket dan datagram socket. Stream socket digunakan untuk sistem komunikasi 2 arah dan menggunakan protokol TCP (Transmission Control Protocol). Contoh aplikasi yang menggunakan stream socket adalah **telnet** dan **HTTP** (web browser). TCP menjamin data terkirim secara urut dan bebas dari error, sedangkan IP (Internet Protocol) bertugas untuk mengatur lalu-lintas routing.

Jenis socket yang kedua yaitu datagram socket disebut juga connectionless socket sebab untuk interaksi client-server tidak harus selalu terhubung terus menerus. Jika client mengirimkan data ke server, data tersebut ada kemungkinan sampai ke server atau tidak. Untuk itu client menunggu sinyal ‘error free’ dari client. Jika client tidak menerima sinyal ‘error free’ dalam suatu kurun waktu, maka client akan mengirimkan lagi data tersebut. Contoh aplikasi yang menggunakan datagram socket adalah **tftp** dan **bootp**.

Gambar 3. Enkapsulasi data

Data yang dikirimkan melalui datagram socket akan melalui proses yang diberi nama enkapsulasi (data encapsulation). Data yang akan dikirimkan sebelumnya dibungkus dulu dengan sebuah header dari protokol yang pertama (misalnya TFTP), lalu dibungkus lagi dengan protokol berikutnya (misalnya UDP), lalu IP dan yang terakhir dibungkus dengan ethernet protocol pada physical layer.

Linux System Call

Pemrograman socket pada linux dapat memanfaatkan banyak system call yang telah tersedia, antara lain yaitu socket(), bind(), listen(), connect(), dll.

1. socket()

   Socket system call digunakan untuk mendapatkan file descriptor.

   Struktur dari socket() adalah sbb. :

```
    #include <sys/types.h>
    #include <sys/socket.h>

    int socket(int domain, int type, int protocol);
```

   Argumen :

   Domain diisi dengan “AF_NET”, type diisi dengan SOCK_STREAM atau SOCK_DGRAM, protocol di-set 0

2. bind()

   bind system call digunakan untuk memberi nomer port ke socket.

   Struktur dari bind() adalah sbb. :

```
 #include <sys/types.h>
 #include <sys/socket.h>

int bind(int sockfd, struct sockaddr *my_addr, int addrlen);
```

Argumen :

Sockfd : socket file descriptor yang dihasilkan dari fungsi socket()

My_addr : berisi alamat ip address, addrlen diisi sizeof(struc sockaddr)

3. send() dan recv()

Kedua system call tersebut digunakan untuk komunikasi data melalui stream socket. Send() dikirimkan untuk mengirimkan data ke remote host, dan recv() digunakan untuk menerima data.

Send() system call :

```
Int send(int sockfd, const void *msg, int len, int flags);
```

recv() system call :

```
Int recv(int sockfd, void *buf, int len, unsigned int
flags);
```

4. sendto() dan recvfrom()

Kedua system call ini mempunyai fungsi yang sama dengan send() dan recv(), bedanya adalah bahwa sendto() dan recvfrom() digunakan oleh datagram socket. Seperti yang dijelaskan pada bab section sebelumnya, bahwa datagram socket adalah connectionless protocol sehingga data yang dikirimkan harus dilengkapi dengan informasi alamat dari komputer yang dituju.

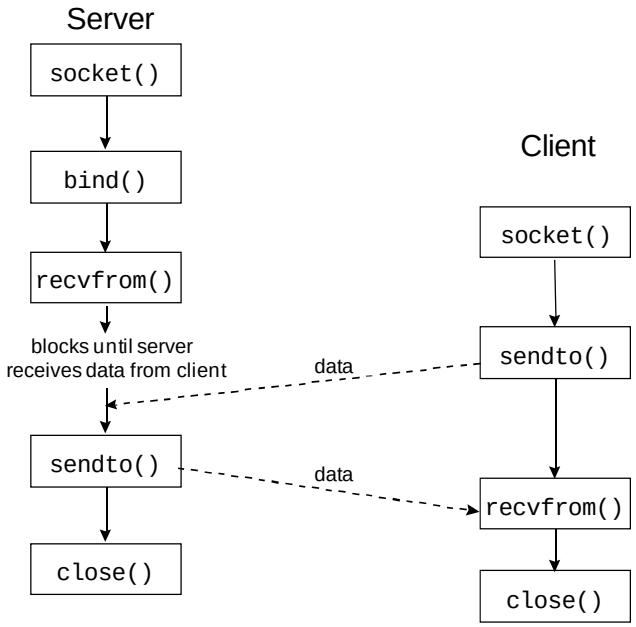
Struktur dari sendto() system call :

```
int sendto(int sockfd, const void *msg, int len, unsigned
int flags, const struct sockaddr *to, int tolen);
```

Struktur dari recvfrom() system call :

```
int recvfrom(int sockfd, void *buf, int len, unsigned int
flags, struct sockaddr *from, int *fromlen);
```

Algorithma Program Client-Server menggunakan Datagram Socket



Gambar 4. Algorithma pemrograman socket

Keterangan :

a. Socket procedure

Socket procedure digunakan untuk membuat socket dan mengembalikan satu nilai integer (descriptor)

Cara memanggilnya : descriptor = socket(protocolfamily, type, protocol)

b. Close procedure

Close procedure memerintahkan sistem untuk menghapus penggunaan socket. Cara memanggilnya : close(socket)

c. Bind procedure

Ketika dibuat, socket tidak mempunyai local address maupun remote address. Server menggunakan procedure bind untuk memberi suatu protokol alamat port pada server yang sedang menunggu sinyal masuk.

Cara memanggilnya : bind(socket, localaddress, addresslength)

d. Send Procedure

Baik server maupun client keduanya membutuhkan procedure ini untuk mengirim informasi. Biasanya, client mengirimkan sinyal request, kemudian server mengirimkan sinyal response. Jika socket sudah terhubung, procedure send dapat digunakan untuk mengirimkan data .

Cara memanggilnya : send(socket, data, length, flags)

e. Receive Procedure

Seperti halnya send procedure, baik client maupun server sama-sama membutuhkan procedure receive untuk menerima data .

Cara memanggilnya : receive(socket, buffer, length, flag)

Program client-server sederhana menggunakan datagram socket

1. # vim /home/client02.c

```
/*
** client02.c -- a datagram
*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define MYPORT 4950    // no port server yang digunakan

int main(int argc, char *argv[])
{
    int sockfd;
    struct sockaddr_in their_addr; // ip server
    struct hostent *he;
    int numbytes;

    if (argc != 3) {
        fprintf(stderr,"usage: client hostname message\n");
        exit(1);
    }

    if ((he=gethostbyname(argv[1])) == NULL) {  // dpt info tentang host
        perror("gethostbyname");
        exit(1);
    }

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
        perror("socket");
        exit(1);
    }
```

```
    their_addr.sin_family = AF_INET;     // host byte order
    their_addr.sin_port = htons(MYPORT); // host to network short
    their_addr.sin_addr = *((struct in_addr *)he->h_addr);
    memset(&(their_addr.sin_zero), '\0', 8); // set semua nilai ke 0

    if ((numbytes=sendto(sockfd, argv[2], strlen(argv[2]), 0,
        (struct sockaddr *)&their_addr, sizeof(struct sockaddr))) == -1) {
        perror("sendto");
        exit(1);
    }
    printf("Kirim %d byte ke %s\n", numbytes,
                                    inet_ntoa(their_addr.sin_addr));

    close(sockfd);
    return 0;
}
```

2. # vim /home/server02.c

```
/*
** server02.c -- a datagram sockets "server"
*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define MYPORT 4950    // no port server

#define MAXBUFLEN 100

int main(void)
{
    int sockfd;
    struct sockaddr_in my_addr;    // ip address server
    struct sockaddr_in their_addr; // ip address client
    int addr_len, numbytes;
    char buf[MAXBUFLEN];

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
        perror("socket");
        exit(1);
    }
```

```
    my_addr.sin_family = AF_INET;         // host byte order
    my_addr.sin_port = htons(MYPORT);  // host to network short
    my_addr.sin_addr.s_addr = INADDR_ANY; // ip address server
    memset(&(my_addr.sin_zero), '\0', 8); // set semua nilai menjadi 0

    if (bind(sockfd, (struct sockaddr *)&my_addr,
                                sizeof(struct sockaddr)) == -1) {
       perror("bind");
       exit(1);
    }

    addr_len = sizeof(struct sockaddr);
    if ((numbytes=recvfrom(sockfd,buf, MAXBUFLEN-1, 0,
                (struct sockaddr *)&their_addr, &addr_len)) == -1) {
       perror("recvfrom");
       exit(1);
    }

    printf("Mendapat paket dari : %s\n",inet_ntoa(their_addr.sin_addr));
    printf("Panjang paket : %d bytes \n",numbytes);
    buf[numbytes] = '\0';
    printf("Isi paket : \"%s\"\n",buf);
    close(sockfd);
    return 0;
}
```

III. Tugas Pendahuluan

1. Jelaskan secara singkat apa yang anda ketahui tentang UDP
2. Jelaskan perbedaan TCP dan UDP
3. Berikan contoh aplikasi –aplikasi yang menggunakan protokol UDP, dan jelaskan bagaimana kerja aplikasi tersebut.

IV. Percobaan

1. Tuliskan kembali program yang ditulis di kotak pada dasar teori
2. Kompilasi dan jalankan program diatas

   a. Pada sisi server

   ```
   # gcc –o server02 server02.c
   # ./server02
   ```

   b. Pada sisi client

   ```
   # gcc –o client02 client02.c
   ```

   # ./client <no_ip_server> <pesan_yg_dikirim_ke_server,tanpa_spasi>

3. Amati output yang dihasilkan dari program diatas.
4. Cek port yang digunakan oleh server

   ```
   # netstat –nlptu | grep server02
   ```
5. Cek port yang digunakan oleh client

   ```
   # tcpdump –nte
   ```

V. Laporan Resmi

1. Tulis hasil percobaan dan analisa hasilnya.
2. Berikan komentar tiap baris pada program tersebut apa maksud dan kegunaan perintah diatas bila dihubungkan dengan socket datagram.
3. Tambahkan program diatas yang memakai datagram socket yang bisa mengirimkan data dari server ke client (pesan diketikkan dari keyboard) sehingga terbentuk komunikasi 2 arah.