MODUL 12
KONSEP DASAR PEMPROGRAMAN SOCKET DATAGRAM

TUJUAN PEMBELAJARAN:

1. Mahasiswa Mengenal Datagram Socket
2. Mahasiswa Memahami Konsep Pembuatan Pemprograman Datagram Socket untuk Komunikasi Client-Server
3. Mahasiswa Mampu Membuat Pemprograman Datagram Socket untuk Komunikasi Client-Server

DASAR TEORI

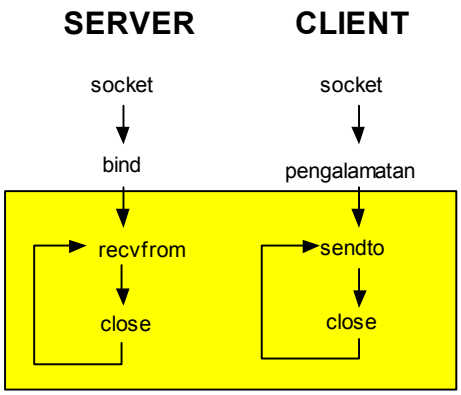
DatagramSocket

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 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.



Gambar . Algorithma pemrograman datagram socket

Pada socket datagram algoritmanya lebih sederhana, tidak membuthkan koneksi antara server dan client. Sehingga *system call/function* yang dibutuhkan lebih sedikit, yaitu *socket(), bind(), sendto()* dan *recvfrom()*. *sendto()* dan *recvfrom()* adalah fungsi khusus yang dipakai untuk mengirim dan menerima data pada *socket datagram*.

Secara garis besar langkah – langkah yang dilakukan pada client dan server pada *socket datagram* adalah sebagai berikut :

1. Langkah – langkah dasar di *client*:
   a. Membuka koneksi *client* ke *server*, yang di dalamnya adalah :
      - Membuat socket dengan perintah socket().
      - melakukan pengalamatan ke server.
   b. Melakukan komunikasi (mengirimkan data), dengan menggunakan perintah *sendto()*
   c. Menutup hubungan dengan perintah *close()*;
2. Langkah – langkah dasar di server :
   a. Membuat socket dengan perintah *socket()*
   b. Mengikatkan socket kepada sebuah alamat network dengan perintah *bind()*
   c. Melakukan komunikasi (menerima data), dengan menggunakan perintah *recvfrom()*

Struktur Pengalamatan

Struktur pengalamatan yang dipakai antara stream socket dan datagram socket tidak ada perbedaan.
Dibawah ini adalah structure yang dipakai.

```
struct sockaddr_in {
unsigned short sin_family; /* address family (always AF_INET)
*/
unsigned short sin_port; /* port num in network byte order */
struct in_addr sin_addr; /* IP addr in network byte order */
unsigned char sin_zero[8]; /* pad to sizeof(struct sockaddr) */
};
```

Contoh pemakaian struktur tersebut bisa dilihat pada tabel berikut ini:

```
int sockfd;
struct sockaddr_in their_addr; // connector's address information
struct hostent *he;
int numbytes;
```

Langkah – Langkah Program di *Client*

1. Berikut ini adalah prosedur pembukaan koneksi client ke server pada *hostname:port* tertentu. Di dalamnya termasuk membuat socket, melakukan pengalamatan ke server dan melakukan koneksi ke server dengan perintah connect(). adalah sebagai berikut :
   a. Membuat socket dengan perintah socket().

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

   Socket system call digunakan untuk mendapatkan file descriptor.
   `IF_INET` menunjukkan bahwa socket dihubungkan dengan protokol internet.
   `SOCK_DGRAM` menunjukkan bahwa program ini memakai *datagram socket/UDP*, yang berari *connectionless.*

   b. Selanjutnya setelah membuat socket melakukan pengalamatan ke server.

```
their_addr.sin_family = AF_INET;     // host byte order
      their_addr.sin_port = htons(MYPORT); // short, network byte order
      their_addr.sin_addr = *((struct in_addr *)he->h_addr);
      memset(&(their_addr.sin_zero), '\0', 8); // zero the rest of the struct
```

2. Melakukan komunikasi (mengirim data), dengan menggunakan perintah *sendto()*

```
if ((numbytes=sendto(sockfd, argv[2], strlen(argv[2]), 0,
             (struct   sockaddr   *)&their_addr,   sizeof(struct
sockaddr))) == -1) {
    perror("sendto");
    exit(1);
}
```

3. Menutup hubungan dengan perintah *close()*;

Langkah – langkah Program di *Server*

1. Melakukan prosedur pembukaan koneksi yang di dalamnya berupa langkah – langkah : membuat socket, mengikat socket, dan pengalamatan socket.

Langkah demi langkah membuat koneksi di server adalah sebagai berikut :

a. Membuat socket dengan perintah socket()

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

b. Melakukan pengalamatan.

```
 my_addr.sin_family = AF_INET;         // host byte order
 my_addr.sin_port = htons(MYPORT);     // short, network byte
order
 my_addr.sin_addr.s_addr = INADDR_ANY; // automatically fill
with my IP
 memset(&(my_addr.sin_zero), '\0', 8); // zero the rest of
the struct
```

c. Mengikatkan socket kepada sebuah alamat network dengan perintah *bind()*

```
if (bind(sockfd, (struct sockaddr *)&my_addr,

sizeof(struct sockaddr)) == -1) {
       perror("bind");
       exit(1);
}
```

bind system call digunakan untuk memberi nomer port ke socket.
Argumen :
Sockfd : socket file descriptor yang dihasilkan dari fungsi socket()
My_addr : berisi alamat ip address, addrlen diisi sizeof(struc sockaddr)

2. Menerima koneksi dengan perintah *recvfrom()*.

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

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.

PERCOBAAN

1. Denga memakai editor vi tuliskan kembali program di bawah ini. Ada dua bagian program, *client* dan *server*. Simpan sesuai dengan nama yang ada pada *comment* program

```
/*
     ** talker.c -- a datagram "client" demo
     */

     #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        //  the  port  users  will  be
connecting to

     int main(int argc, char *argv[])
     {
         int sockfd;
         struct sockaddr_in their_addr; // connector's address
information
         struct hostent *he;
         int numbytes;

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

         if  ((he=gethostbyname(argv[1]))  ==  NULL)  {   //  get
the host info
             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);   //   short,
network byte order
         their_addr.sin_addr   =   *((struct   in_addr   *)he-
>h_addr);
```

```
        memset(&(their_addr.sin_zero), '\0', 8); // zero the
rest of the struct

        if          ((numbytes=sendto(sockfd,          argv[2],
strlen(argv[2]), 0,
             (struct  sockaddr  *)&their_addr,  sizeof(struct
sockaddr))) == -1) {
            perror("sendto");
            exit(1);
        }

        printf("sent %d bytes to %s\n", numbytes,

inet_ntoa(their_addr.sin_addr));

        close(sockfd);

        return 0;
    }
```

```
/*
    ** listener.c -- a datagram sockets "server" demo
    */

    #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       //  the  port  users  will  be
connecting to

    #define MAXBUFLEN 100

    int main(void)
    {
        int sockfd;
        struct  sockaddr_in  my_addr;         //  my  address
information
        struct sockaddr_in their_addr; // connector's address
information
        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);         //  short,
network byte order
        my_addr.sin_addr.s_addr      =      INADDR_ANY;      //
automatically fill with my IP
        memset(&(my_addr.sin_zero),  '\0',  8);  //  zero  the
rest of the struct

        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("got                packet                from
%s\n",inet_ntoa(their_addr.sin_addr));
        printf("packet is %d bytes long\n",numbytes);
        buf[numbytes] = '\0';
        printf("packet contains \"%s\"\n",buf);

        close(sockfd);

        return 0;
    }
```

2. Jalankan program tersebut, output apa yang dihasilkan dari program tersebut.
3. Berikan komentar tiap baris pada program tersebut apa maksud dan kegunaan perintah diatas bila dihubungkan dengan socket datagram.
4. Buatlah program memakai datagram socket yang bisa mengirimkan data posisi jam client sekarang ke server.

LAPORAN RESMI

FORMAT LAPORAN RESMI

Nama dan NRP mahasiswa

Judul Percobaan : Konsep Dasar Pemprograman socket datagram

Dasar Teori :

Tugas Pendahuluan :

Hasil percobaan :

Daftar Pertanyaan

1. Berikan kesimpulan hasil praktikum yang anda lakukan.
2. Pada pemprograman socket datagram, untuk mengetahui data sampai atau tidak, client menunggu sinyal ‘error free’ dari server. Jika client tidak menerima sinyal ‘error free’ dalam suatu kurun waktu, maka client akan mengirimkan lagi data tersebut. Modifikasi program yang anda buat supaya bisa mengakomodasi ‘error free’ tersebut.