

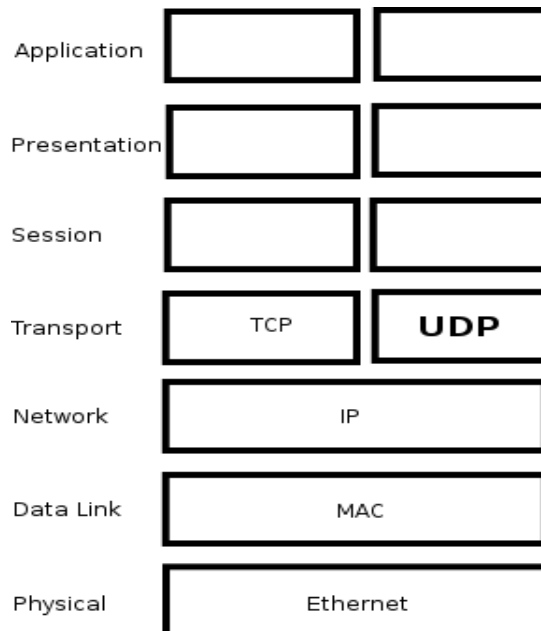
PERCOBAAN IV
Socket Programming
User Datagram Protocol (UDP)

1. TUJUAN

- Mahasiswa dapat memahami cara kerja protokol UDP
- Mahasiswa dapat membuat aplikasi client-server

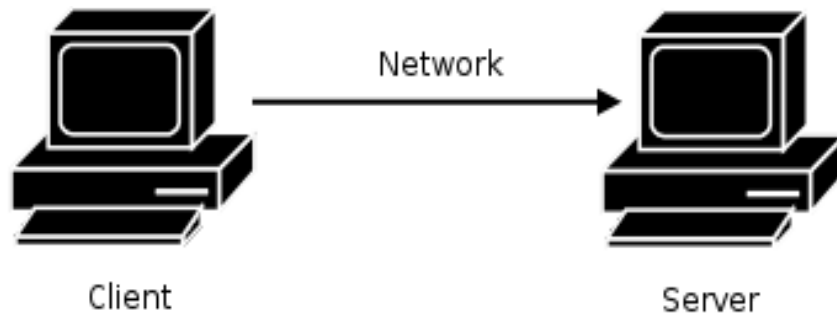
2. DASAR TEORI

UDP adalah suatu protokol pengiriman data yang berbasis Internet Protocol (IP) dan bersifat *connection-less oriented*. Pada OSI layer UDP berada pada layer transport yang fungsinya mengatur pengiriman suatu data dari client ke server.



Gb 1. UDP pada OSI Layer

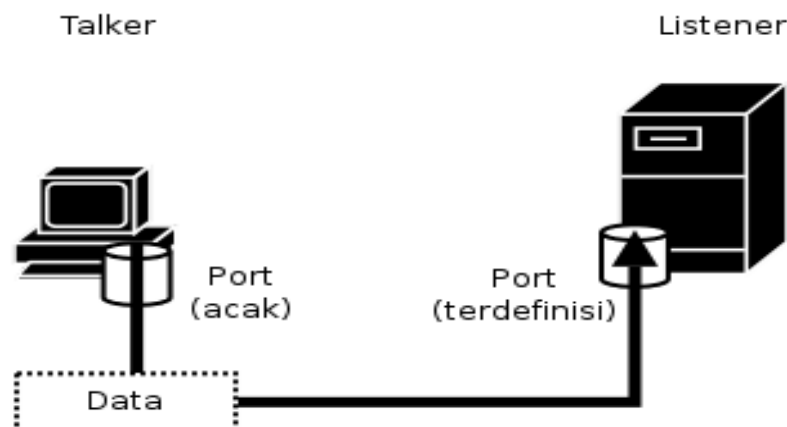
Model komunikasi data dengan client-server artinya pada saat pengiriman data, salah satu komputer ada yang bersifat client dan yang satu akan bersifat sebagai server.



Gb 2. Client - Server

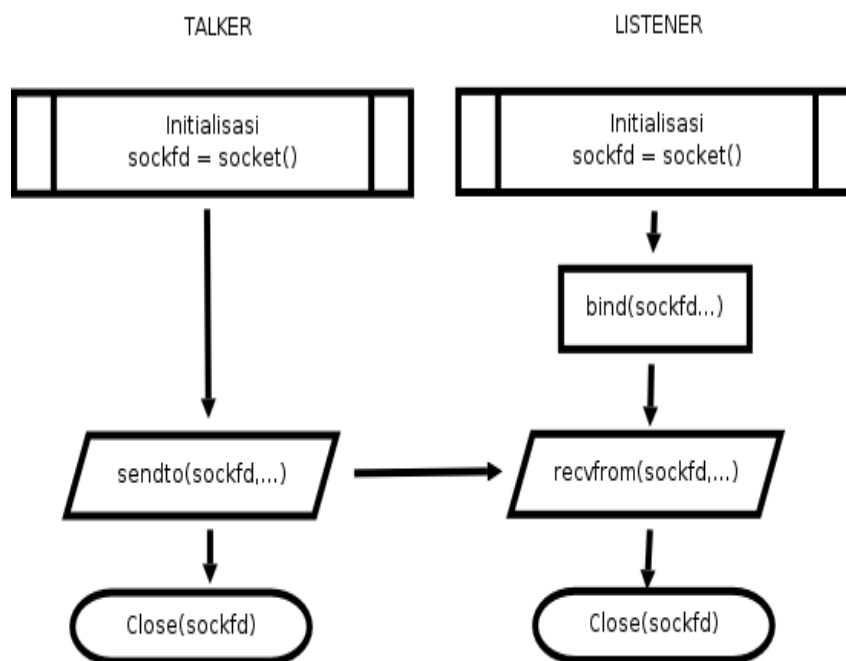
Untuk pengiriman datanya, pada masing-masing komputer (client-server) akan

menggunakan *port* dengan pendefinisian terlebih dahulu. Kemudian dari client akan mengirimkan data dari port pada PC-nya ke arah port pada PC servernya. Apabila port tersebut sudah digunakan oleh aplikasi lainnya maka akan terjadi error apabila aplikasi yang kita jalankan menggunakan port yang sama. Jumlah port yang ada 65535 digunakan sesuai dengan aplikasi yang sudah distandarkan.



Gb 3. Pengiriman data melalui PORT

Alur penggunaan socket programming untuk UDP seperti pada Gb. 4.



Gb 4. Alur socket programming pada UDP

3. PERALATAN

- PC (Linux OS)
- GCC
- UTP Cable
- Hub / Switch (optional)

4. LANGKAH PRAKTIKUM

1. Sebelum PC menyala, Sambungkan PC ke jaringan, apabila dihubungkan dengan switch / hub gunakan kabel UTP straight through apabila dengan PC langsung gunakan kabel crossover
2. Nyalakan PC hingga proses booting sempurna.

3. login dengan user “root” dan password “root” (isikan tanpa tanda petik)
4. jalankan perintah `ifconfig eth0` , kemudian catat IP addressnya.

contoh :

```
highway:~# ifconfig eth0
eth0 Link encap:Ethernet HWaddr 00:13:D4:CC:4E:2A
      inet addr:192.168.0.167 Bcast:192.168.0.255 Mask:255.255.255.0
      inet6 addr: fe80::213:d4ff:fecc:4e2a/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:5630764 errors:0 dropped:0 overruns:0 frame:0
      TX packets:2730184 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:2746138798 (2.5 GiB) TX bytes:404795862 (386.0 MiB)
      Base address:0xdc00 Memory:febc0000-febe0000
```

IP address kita adalah **192.168.0.167**

- **Tips untuk asisten :**

Apabila PC belum memiliki IP, asisten diharapkan memberikan IP dengan perintah :

```
# ifconfig eth0 192.168.0.*
```

Dimana * digantikan dengan angka 1 – 254, dengan syarat tidak ada IP yang sama antar PC

5. Buat direktori dengan nama kelas – group, contoh : 2D4TA-1

```
# cd
```

```
# mkdir 2D4TA-1
```

Tanda # tidak perlu ikut diketik !!!

6. Masuk ke direktori tersebut ...

```
# cd 2D4TA-1
```

7. Untuk memulai pengetikan program di linux, peserta menggunakan program “VIM”. Ketikkan source program `listener.c` atau `talker.c` dengan perintah :

```
# vim listener.c
```

atau

```
# vim talker.c
```

- Tips untuk asisten:

Peserta praktikum dibagi menjadi beberapa kelompok dan ditunjuk supaya ada yang memprogram *listener* dan ada yang memprogram *talker*

8. Tekan tombol “Ins” / “Insert” pada keyboard untuk memulai pengetikan hingga terlihat indikasi “--INSERT--” di layar bagian bawah kiri. Ketik sesuai dengan source pada lampiran.

9. Setelah semua source di ketik, simpan source tersebut dengan menekan tombol “ESC”, hingga indikator “--INSERT--” hilang, dilanjutkan dengan “:wq” (tanpa tanda petik).

10. Lakukan kompilasi program dengan cara :

Untuk program `listener.c`

```
# gcc -o listener listener.c
```

Untuk program `talker.c`

```
# gcc -o talker talker.c
```

Apabila terjadi error, lakukan pengecekan dengan membuka file source seperti pada langkah ke-7.

11. Jalankan program dengan perintah, sebagai berikut :

Untuk `listener` :

```
# ./listener
```

Untuk talker :

```
# ./talker 192.168.0.25 "percobaan pesan"
```

Dimana 192.168.0.25 adalah IP dari komputer yang melakukan pemrograman *listener*. Pesan yang dikirim adalah ***percobaan pesan***. Pada komputer yang menjalankan program listener akan tampil data text tersebut.

* Tips : Untuk mematikan program lakukan dengan menekan "Ctrl + C"

12. Lakukan pengiriman text tersebut dengan kondisi sebagai berikut, kemudian amati pada komputer tersebut dan apabila muncul error catat di laporan sementara !
 1. Program listener dijalankan di komputer A, pada komputer B kirim pesan dengan program talker ke komputer A.
 2. Matikan program listener pada komputer A, pada komputer B kirim pesan dengan program talker ke komputer A.

5. TUGAS

1. Lampirkan RFC yang berhubungan dengan protokol UDP (RFC 768)

6. REFERENSI

- RFC 768
- man socket, bind, sendto, recvfrom

LEMBAR ANALISA

Praktikum Komunikasi Data – 4

Tanggal praktikum :
Nama :
NRP :
Kelas :

IP Listener :
IP Talker :

<i>No</i>	<i>Listener</i>	<i>Talker</i>	<i>Pesan Error</i>
1.	Dijalankan	Dijalankan	
2.	Dimatikan	Dijalankan	

LAMPIRAN

```
/*
** Praktikum Komunikasi Data
** talker.c -- a datagram "client" demo
** by Sritrusta Sukaridhoto, ST
*/

#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;
}
```

```

/*
** Praktikum Komunikasi Data
** listener.c -- a datagram sockets "server" demo
** by Sritrusta Sukaridhoto, ST
*/

#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;
}

```