PRAKTIKUM
SOCKET PROGRAMMING (TCP dan UDP)

I. Tujuan

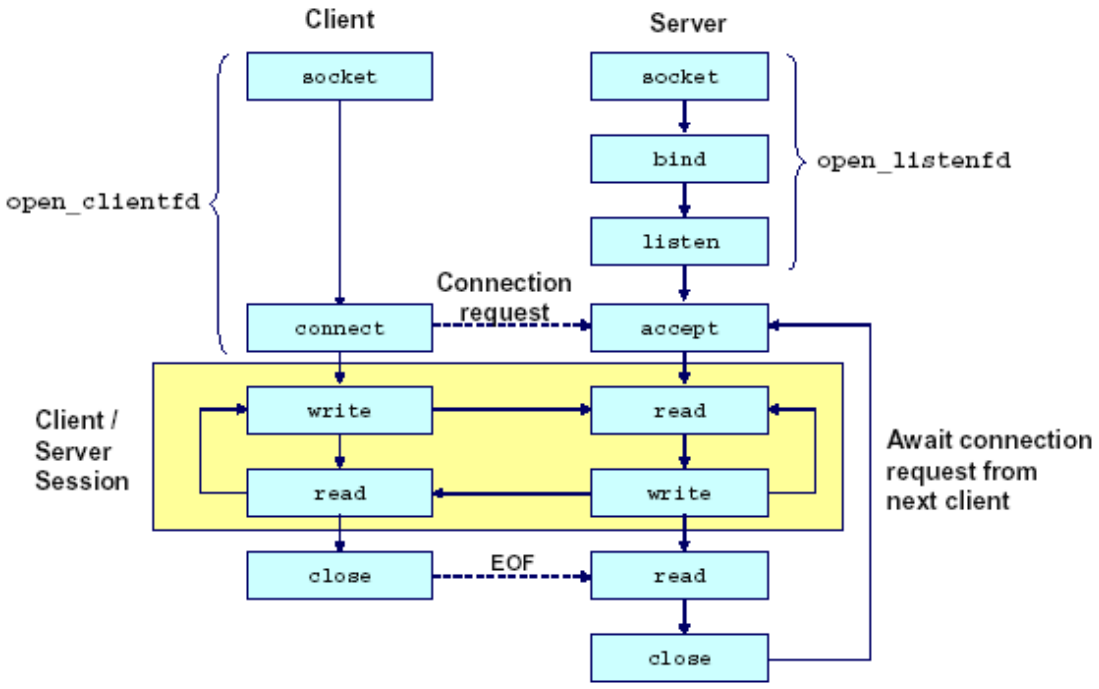
1. Mahasiswa memahami konsep aplikasi client server di jaringan.
2. Mahasiswa memahami konsep pemprograman socket.
3. Mahasiswa memahami jenis – jenis socket programming
4. Mahasiswa mampu membangun program socket sederhana

II. Peralatan Yang Dibutuhkan

1. Beberapa komputer yang berfungsi sebagai server.
2. Beberapa komputer yang berfungsi sebagai *client*.
3. *Hub/switch* sebagai penghubung jaringan.
4. Kabel jaringan secukupnya.

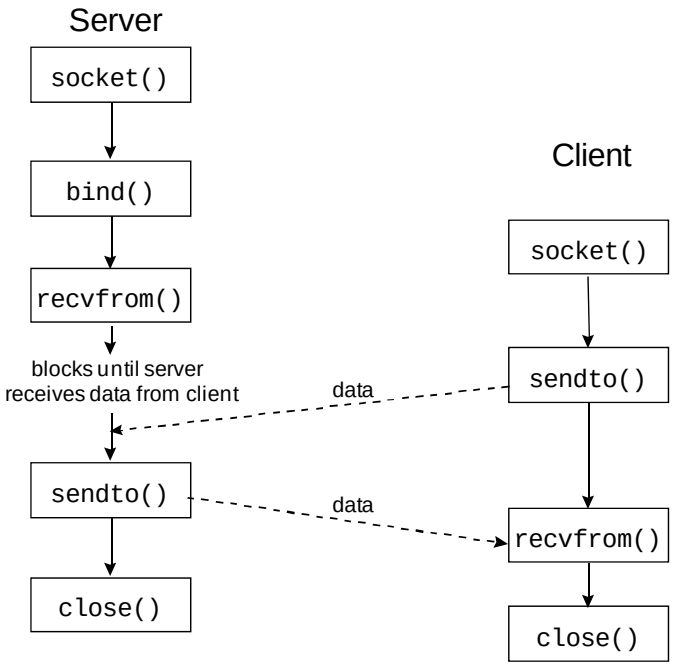
III. Dasar Teori

Setiap aplikasi di jaringan, transaksinya didasarkan pada konsep *client-server*. Sebuah *server* dan sebuah atau beberapa *client* yang meminta/*request* pelayanan ke server. Fungsi server sebagai pengatur *resource* yang ada, yang menyediakan pelayanan dengan memanfaatkan resource yang untuk kebutuhan client. Proses ini (proses *client-server*) bisa dijalankan pada sebuah komputer (komputer tunggal) atau bisa juga satu komputer berfungsi sebagai server dan sebuah atau beberapa komputer berfungsi sebagai client.



Gambar 1 Ilustrasi TCP socket

Algorithma Program Client-Server menggunakan Datagram Socket



Gambar 2. Ilustrasi UDP socket

IV. Tugas Pendahuluan

1. Jelaskan secara singkat tentang 3-way handshake di aplikasi TCP
2. Jelaskan perbedaan TCP dan UDP

V. Percobaan

A. SOCKET PROGRAMMING UDP

PROGRAM CLIENT

```
/*
** clientUDP.c -- program client sederhana menggunakan datagram socket
*/

#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
```

```
#define MAXBUFLEN 100

int main()
{
	char no[16], dt[30];

	printf("----- PROGRAM CHATTING -----\n");

	printf("To : ");
	scanf("%s", no);

	while(1){
		printf("Me : ");
		scanf("%s", dt);
		kirim(no, dt);
		terima();
	}
}

int kirim(char no[], char dt[])
{
	int sockfd;
	struct sockaddr_in my_addr;
	struct sockaddr_in their_addr;
	struct hostent *he;
	int addr_len, numbytes;

	if((he = gethostbyname(no)) == NULL)
	{
		perror("gethostbyname");
		exit(1);
	}

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

	their_addr.sin_family = AF_INET;
	their_addr.sin_port = htons(MYPORT);
	their_addr.sin_addr = *((struct in_addr *)he->h_addr);
	memset(&(their_addr.sin_zero), '\0', 8);

	if((numbytes=sendto(sockfd, dt,strlen(dt),0,(struct sockaddr *)&their_addr,sizeof(struct sockaddr)))
== -1)
	{
		perror("sendto");
```

```
		exit(1);
	}

	close(sockfd);
	return 0;
}

int terima()
{
	int sockfd;
	struct sockaddr_in my_addr;
	struct sockaddr_in their_addr;
	struct hostent *he;
	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;
	my_addr.sin_port = htons(MYPORT);
	my_addr.sin_addr.s_addr = INADDR_ANY;
	memset(&(my_addr.sin_zero),'\0',8);

	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);
	}
	buf[numbytes]='\0';
	printf("%s : \"%s\"\n", inet_ntoa(their_addr.sin_addr), buf);

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

PROGRAM SERVER

```
/*
** serverUDP.c -- program client sederhana menggunakan datagram socket
*/

#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

#define MAXBUFLEN 100
char  noip[50];              //untuk mendapatkan no ip dari client

int main()
{
        char no[16], dt[30];

        printf("----- PROGRAM CHATTING -----\n");

        terima();
        strcpy (no, noip);          //copy data dari variable noip ke variabel no
        while(1){
                printf("Me : ");
                scanf("%s", dt);
                kirim( dt);
                terima();
        }
}

int kirim( char dt[30])
{
        int sockfd;
        struct sockaddr_in my_addr;
        struct sockaddr_in their_addr;
        struct hostent *he;
        int addr_len, numbytes;

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

```
                exit(1);
        }

        their_addr.sin_family = AF_INET;
        their_addr.sin_port = htons(MYPORT);
        their_addr.sin_addr = *((struct in_addr *)he->h_addr);
        memset(&(their_addr.sin_zero), '\0', 8);

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

        close(sockfd);
        return 0;
}

int terima()
{
        int sockfd;
        struct sockaddr_in my_addr;
        struct sockaddr_in their_addr;
        struct hostent *he;
        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;
        my_addr.sin_port = htons(MYPORT);
        my_addr.sin_addr.s_addr = INADDR_ANY;
        memset(&(my_addr.sin_zero),'\0',8);

        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);
        }
        buf[numbytes]='\0';
        printf("%s : \"%s\"\n", inet_ntoa(their_addr.sin_addr), buf);
        strcpy (no, inet_ntoa(their_addr.sin_addr);              //copy data ke varibel no
        close(sockfd);
        return 0;
}
```

LANGKAH PERCOBAAN

1. Setelah selesai menulis dan menyimpan program, pastikan gcc sudah terinstall pada system operasi linux anda. Jalankan perintah :

   ```
   # dpkg –l | grep gcc
   ```

   Jika belum ter*install* lakukan *installasi* paket *gcc* beserta *library*nya.

   ```
   # apt-get install gcc gcc-4.3
   Jika standard library belum terinstall, maka diinstall juga :
   # apt-get install libc6-dev
   atau
   # apt-get install build-essential
   ```

2. Kompilasi dan jalankan program diatas
   a. Pada sisi server
      # gcc –o serverUDP serverUDP.c
      # ./serverUDP
   b. Pada sisi client
      # gcc –o clientUDP clientUDP.c
      # ./clientUDP

   ```
   NB: Sebelum menjalankan program diatas, jalankan dulu wireshark di
   sisi Client.
   ```

3. Amati output yang dihasilkan dari program diatas. Catat informasi yang dihasilkan oleh wireshark.
4. Cek port yang digunakan oleh server
   # netstat –nlptu | grep serverUDP

B. SOCKET PROGRAMMING TCP

PROGRAM CLIENT

```
/*
** clientTCP.c -- program client sederhana menggunakan stream socket
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

```
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#define PORT 3333                   // nomer port yang digunakan
#define MAXDATASIZE 100         // jumlah bytes maximal yang dikirimkan
int main()
{
int sockfd, numbytes;
char buf[MAXDATASIZE];
char nilai[MAXDATASIZE];
struct hostent *he;
struct sockaddr_in their_addr;   // informasi alamat server
char  no[50];
printf("To : ");
scanf("%s", no);

printf("||      Anda adalah Client      ||\n");
printf("|| silahkan memulai pembicaraan ||\n");

if ((he=gethostbyname(no)) == NULL) {   // mencari info tentang host
        perror("gethostbyname");
        exit(1);
}

if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        exit(1);
}
their_addr.sin_family = AF_INET;          // host byte order
their_addr.sin_port = htons(PORT);        // short, network byte order
their_addr.sin_addr = *((struct in_addr *)he->h_addr);
memset(&(their_addr.sin_zero), '\0', 8);            // lainnya diisi 0
if (connect(sockfd, (struct sockaddr *)&their_addr,
        sizeof(struct sockaddr)) == -1) {
        perror("connect");
        exit(1);
}

while(1){
        printf("Client : ");
        scanf("%s",nilai);                              //data yang akan dikirim ke server
        if (send(sockfd,nilai,50, 0)== -1){
                perror("send");
                exit(0);
        }
```

```
        numbytes=0;
        if ((numbytes=recv(sockfd, buf, MAXDATASIZE-1, 0)) == -1) {
                perror("recv");
                exit(0);
        }
        if(numbytes > 1){
                buf[numbytes] = '\0';
                printf("Server:  %s \n",buf);       //data dari server
        }
}
close(sockfd);              //menutup socket
return 0;
}
```

PROGRAM SERVER

```
/*
** serverTCP.c -- program server sederhana menggunakan stream socket
*/
#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 <sys/wait.h>
#include <signal.h>
#define MYPORT 3333              // nomer port yang digunakan
#define BACKLOG 10               // jumlah koneksi yang diperbolehkan
#define MAXDATASIZE 40           // jumlah bytes maximal yang dikirimkan

int main(void)
{
int sockfd, new_fd,numbytes;    // sock_fd ---> koneksi saat ini, new_fd ---->kon baru
struct sockaddr_in my_addr;     // ip address server
char buf[MAXDATASIZE],nilai[MAXDATASIZE];
struct sockaddr_in their_addr;  // ip address client
int sin_size;
struct sigaction sa;
int yes=1;
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        exit(1);
}
if (setsockopt(sockfd,SOL_SOCKET,SO_REUSEADDR,&yes,sizeof(int)) == -1)
{
```

```
        perror("setsockopt");
        exit(1);
}
my_addr.sin_family = AF_INET;             // host byte order atau (big endian)
my_addr.sin_port = htons(MYPORT);     // short, network byte order
my_addr.sin_addr.s_addr = INADDR_ANY;         // diisi dengan ip address server
memset(&(my_addr.sin_zero), '\0', 8);              // lainya diisi 0
if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr))== -1) {
        perror("bind");
        exit(1);
}
if (listen(sockfd, BACKLOG) == -1) {
        perror("listen");
        exit(1);
}
sin_size = sizeof(struct sockaddr_in);
if ((sockfd = accept(sockfd, (struct sockaddr *)&their_addr, &sin_size)) == -1) {
        perror("accept");
}
printf("||     Anda adalah Server     ||\n");
printf("Server mendapat koneksi dari %s\n",inet_ntoa(their_addr.sin_addr));


while(1){
        numbytes=0;
        if ((numbytes=recv(sockfd,buf, MAXDATASIZE-1, 0)) == -1){
                perror("recv");
        }
        if(numbytes != 0){
                buf[numbytes]=='\0';
                printf("Client :  %s \n" ,buf);
        }

        printf("Server : ");
        scanf("%s",nilai);
        if (send(sockfd, nilai, 50, 0) == -1){
                perror("send");
        }

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

LANGKAH PERCOBAAN

1. Lakukan kompilasi program client dan server, dengan cara :

```
# cd /home
# gcc –o clientTCP clientTCP.c
```

```
# gcc –o serverTCP serverTCP.c
```

2. Jalankan program server dan selanjutnya jalankan program client, dengan cara berikut ini :

```
#./serverTCP
#./clientTCP
```

NB: Sebelum menjalankan program diatas, jalankan dulu wireshark di sisi Client.

3. Amati output yang dihasilkan. Catat informasi yang dihasilkan oleh wireshark.
4. Catat proses terjadinya 3-way handshake dari wireshark.
5. Cek port yang digunakan oleh server

```
# netstat –nlptu | grep serverTCP
```

6. Editlah program untuk TCP diatas agar sama dengan program aplikasi untuk UDP. Buat dalam bentuk fungsi.