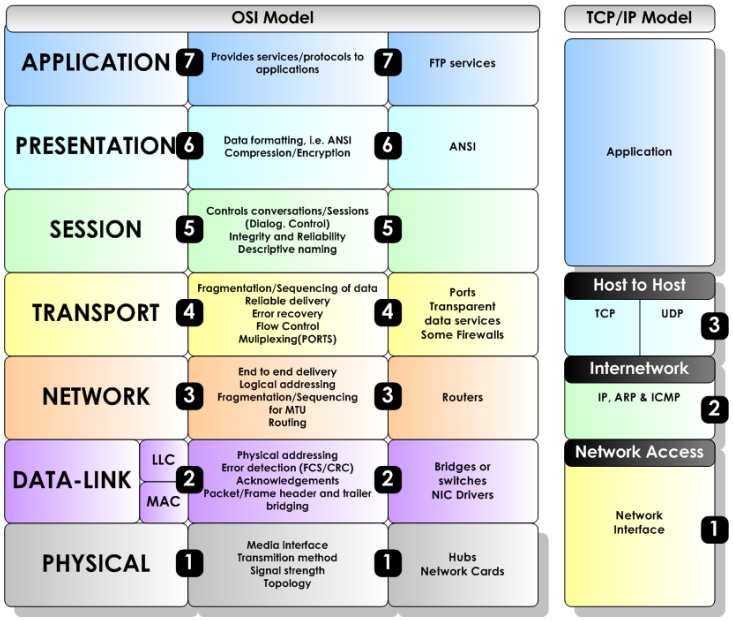
PERCOBAAN 9
Pemrograman Socket Transport Control Protocol (TCP)

A. TUJUAN

1. Mahasiswa dapat memahami cara kerja protokol TCP
2. Mahasiswa mampu memahami konsep aplikasi client-server
3. Mahasiswa memahami konsep pemrograman socket
4. Mahasiswa dapat membuat aplikasi client-server menggunakan protokol TCP

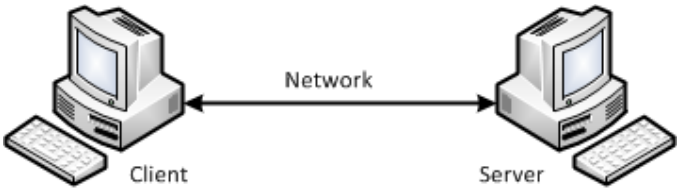
B. DASAR TEORI

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



Gambar 1. TCP pada OSI layer

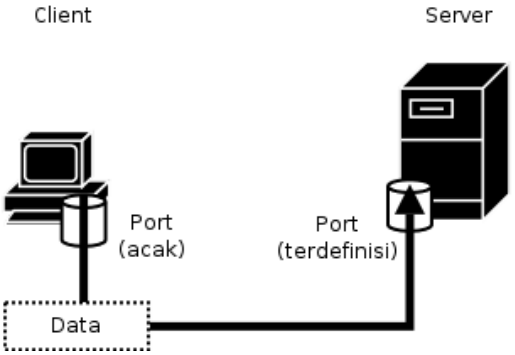
Model komunikasi data dengan clientserver artinya pada saat pengiriman data, salah satu komputer ada yang bersifat client dan yang satu akan bersifat sebagai server.



Gambar 2. Komunikasi client-server

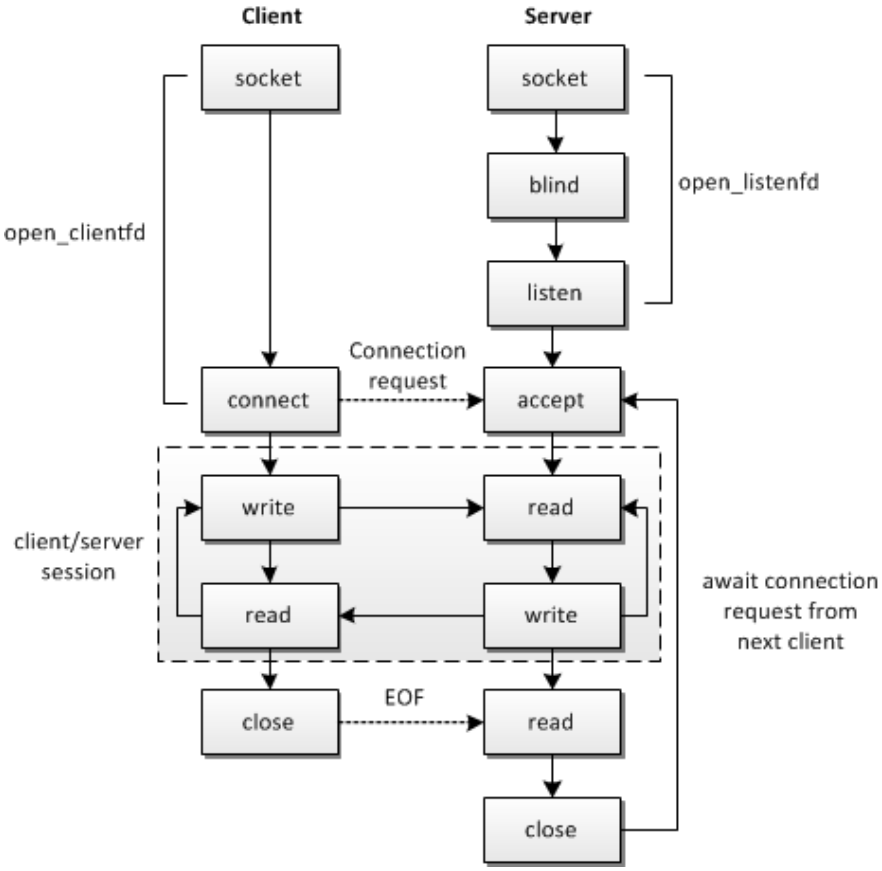
Untuk pengiriman datanya, pada masingmasing komputer (clientserver) akan menggunakan port dengan pendefinisian terlebih dahulu. Kemudian dari client akan mengirimkan data dari port pada PCnya ke arah port pada PC servernya. Apabila port tersebut sudah digunakan oleh aplikasi lainnya maka akan terjadi error apabila aplikasi yang kita jalankan

menggunakan port yang sama. Jumlah port yang ada 65535 digunakan sesuai dengan aplikasi yang sudah distandarkan.



Gambar 3. Pengiriman data melalui port

Alur penggunaan socket programming untuk TCP seperti gambar di bawah ini.



Gambar 4. Alur socket programming pada TCP

C. PERALATAN

1. PC (Linux OS)
2. gcc
3. Kabel UTP
4. Hub / Switch (optional)

D. PERCOBAAN

1. Hubungkan 2 PC, bisa menggunakan switch atau dihubungkan langsung menggunakan kabel cross. PC 1 sebagai server dan PC 2 sebagai client.
2. Catat kedua IP, dengan menjalankan perintah `ifconfig`.
   Server : (IP)
   Client : (IP)
3. Buat folder baru dengan menjalankan perintah di bawah ini,
   ```
   #cd
   #mkdir tcp
   #cd tcp
   ```
4. Tulis program server.c dan client.c di bawah ini dengan menggunakan editor linux misalkan gedit atau vim.

server.c

```
#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 3490 // the port users will be connecting to
#define BACKLOG 10 // how many pending connections queue will hold
#define MAXDATA 100000
void sigchld_handler(int s)
{
while(wait(NULL) > 0);
}
int main(void)
{
int sockfd, new_fd; // listen on sock_fd, new connection on new_fd
struct sockaddr_in my_addr; // my address information
struct sockaddr_in their_addr; // connector's address information
int sin_size, numbytes;
char buf[MAXDATA];
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
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);
}
if (listen(sockfd, BACKLOG) == 1)
{
perror("listen");
exit(1);
}
sa.sa_handler = sigchld_handler; // reap all dead processes
sigemptyset(&sa.sa_mask);
sa.sa_flags = SA_RESTART;
if (sigaction(SIGCHLD, &sa, NULL) == 1)
{
perror("sigaction");
exit(1);
}
printf("SERVER: siap menerima koneksi\n");
while(1) { // main accept() loop
sin_size = sizeof(struct sockaddr_in);
if ((new_fd = accept(sockfd, (struct sockaddr *)&their_addr,
&sin_size)) == 1)
{
perror("accept");
continue;
}
printf("SERVER: menerima koneksi dari
%s\n",inet_ntoa(their_addr.sin_addr));
if (!fork()) { // this is the child process
close(sockfd); // child doesn't need the listener
numbytes=recv(new_fd, buf, MAXDATA1,
0);
if (numbytes < 0) {
perror("recv");
exit(1);
}
printf("Data: %s\n", buf);
buf[numbytes]= '\0';
close(new_fd);
exit(0);
}
close(new_fd); // parent doesn't need this
}
return 0;
}
```

client.c

```
#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 3490  // the port client will be connecting to
#define MAXDATASIZE 100  // max number of bytes we can get at once
int main(int argc, char *argv[])
{
```

```
int sockfd;
struct hostent *he;
struct sockaddr_in their_addr; // connector's address information
if (argc != 3) {
fprintf(stderr,"Penggunaan: %s server pesan\n", argv[0]);
exit(1);
}
if ((he=gethostbyname(argv[1])) == NULL) {   // get the host info
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);
// zero the rest of the struct
if (connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct so
ckaddr)) == -1) {
perror("connect");
exit(1);
}
if ((send(sockfd, argv[2], strlen(argv[2]), 0)) == -1) {
perror("send");
exit(0);
}
printf("mengirimkan %s ke %s\n", argv[2], argv[1]);
close(sockfd);
return 0;
}
```

5. Setelah selesai menulis dan menyimpan program, pastikan gcc sudah terinstall pada sistem operasi linux anda. Jalankan perintah :
   ```
   # dpkg –l | grep gcc
   ```
   Jika belum terinstall lakukan installasi paket gcc beserta librarynya.
   ```
   # apt-get install gcc gcc-4.3
   ```
6. Jika standard library belum terinstall, maka diinstall juga :
   ```
   # apt-get install libc6-dev
   ```
   atau
   ```
   # apt-get install build-essential
   ```
7. Lakukan kompilasi program dengan cara :
   Untuk program server.c
   ```
   # gcc -o server server.c
   ```
   Untuk program client.c
   ```
   # gcc -o client client.c
   ```
8. Apabila terjadi error, lakukan pengecekan dengan membuka file source seperti pada langkah ke-4.
9. Jalankan aplikasi wireshark pada sisi client, capture paket data yang dikirim sesuai interface yang digunakan.
10. Jalankan program dengan perintah, sebagai berikut :
    Untuk server :
    ```
    # ./server
    ```
    Untuk client :
    ```
    # ./client 192.168.0.25 “Percobaan Pesan TCP”
    ```

Dimana 192.168.0.25 adalah IP dari komputer yang melakukan pemrograman server. Pesan yang dikirim adalah Percobaan Pesan TCP. Pada komputer yang menjalankan program server akan tampil data text tersebut.

Tips : Untuk mematikan program lakukan dengan menekan “Ctrl + C”

11. Pada aplikasi wireshark amati IP source, IP destination, protokol yang digunakan, port yang digunakan dan pesan yang dikirim.
12. Lakukan pengiriman text tersebut dengan kondisi sebagai berikut, kemudian amati pada komputer tersebut dan apabila muncul error catat di laporan sementara !
    a. Program server dijalankan di komputer A, pada komputer B kirim pesan dengan program client ke komputer A.
    b. Matikan program server pada komputer A, pada komputer B kirim pesan dengan program client ke komputer A.
13. Catat tampilan program dan proses pengiriman data pada sisi server dan client

Tabel Pengamatan

No	Server	Client	Pesan Error
1.	Dijalankan	Dijalankan	
2.	Dimatikan	Dijalankan	