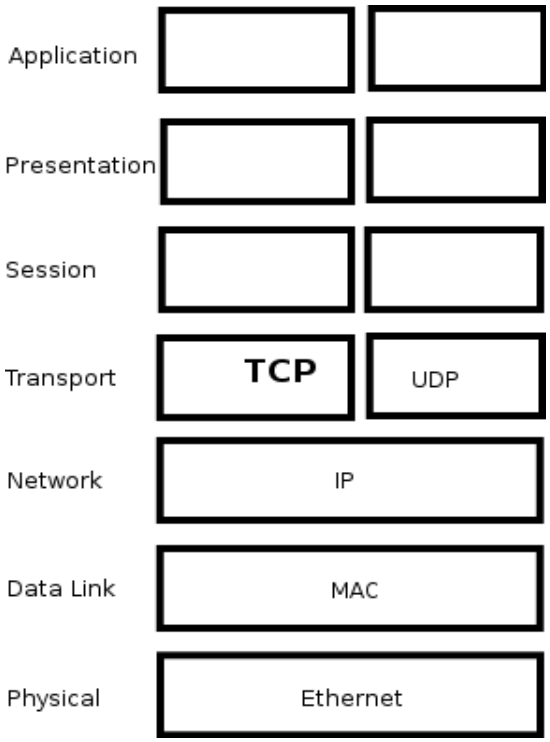
PERCOBAAN V
Socket Programming
Transport Control Protocol (TCP)

1. TUJUAN
   - Mahasiswa dapat memahami cara kerja protokol TCP
   - Mahasiswa dapat membuat aplikasi client-server

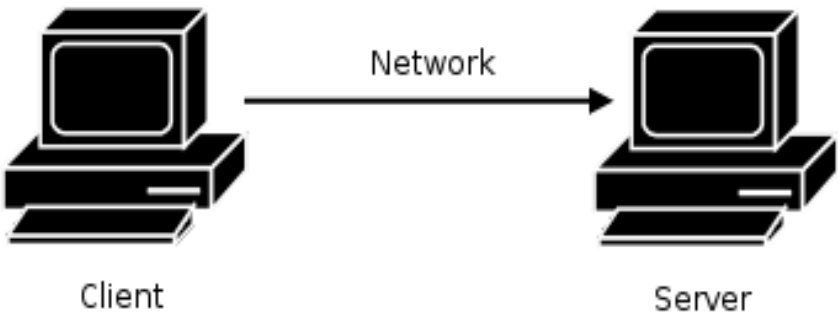
2. DASAR TEORI

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.



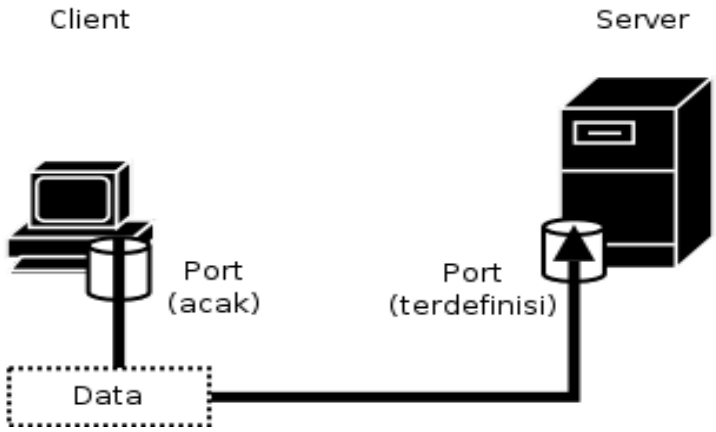
Gb 1. TCP pada OSI Layer

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.



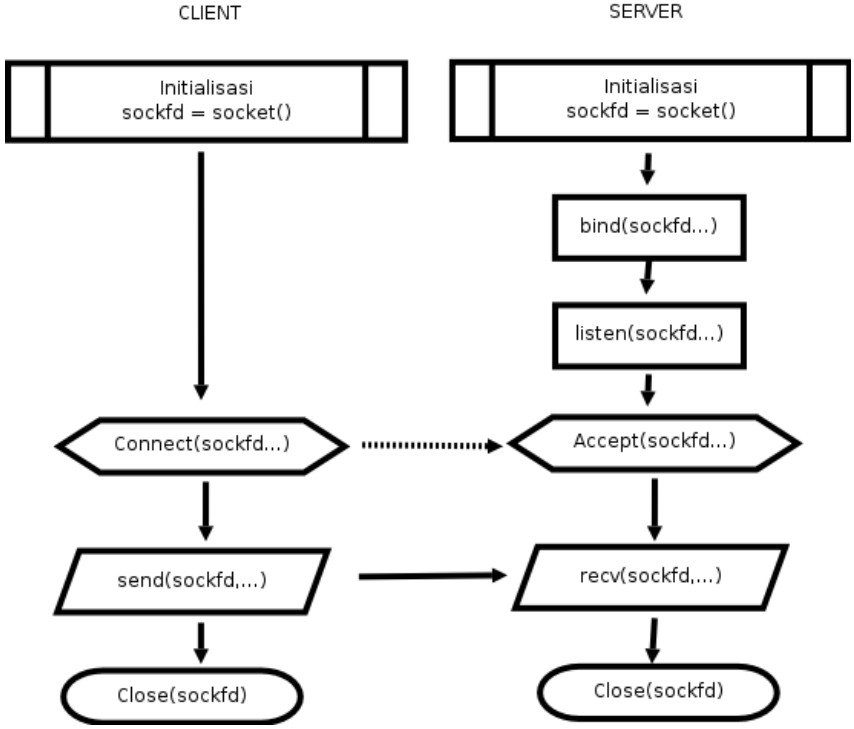
Gb 2. Client - Server

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 TCP seperti pada Gb. 4.



Gb 4. Alur socket programming pada TCP

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 straigh 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 client.c atau server.c dengan perintah :
   # vim client.c
   atau
   # vim server.c

- **Tips untuk asisten:**
  Peserta praktikum dibagi menjadi beberapa kelompok dan ditunjuk supaya ada yang memrogram *client* dan ada yang memrogram *server*

8. Tekan tombol "Ins" / "Insert" pada keyboard untuk memulai pengetikan hingga terlihar 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 server.c
    # gcc -o server server.c
    Untuk program client.c
    # gcc -o client client.c
    Apabila terjadi error, lakukan pengecekan dengan membuka file source seperti pada langkah ke-7.

11. Jalankan program dengan perintah, sebagai berikut :

    Untuk server :

        # ./server

    Untuk client :

        # ./client 192.168.0.25 “percobaan pesan”

    Dimana 192.168.0.25 adalah IP dari komputer yang melakukan pemrograman *server*. Pesan yang dikirim adalah ***percobaan pesan***. Pada komputer yang menjalankan program server 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 server dijalankan di komputer A, pada komputer B kirim pesan dengan program client ke komputer A.
    2. Matikan program server pada komputer A, pada komputer B kirim pesan dengan program client ke komputer A.

5. TUGAS
    1. Lampirkan RFC yang berhubungan dengan protokol TCP (RFC 793)

6. REFERENSI

- RFC 793
- man socket, bind, send, recv, ip, accept, listen, connect

LEMBAR ANALISA

Praktikum Komunikasi Data – 5

Tanggal praktikum :
Nama :
NRP :
Kelas :

IP Server :
IP Client :

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

LAMPIRAN

```
/*
** client.c -- a stream socket client for KOMDAT
** by Sritrusta Sukaridhoto, ST
*/
#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 sockaddr)) == -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;
}
```

```
/*
** server.c -- a stream socket server for KOMDAT
** 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 <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, MAXDATA-1, 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;
}
```