PERCOBAAN 5
Pemrograman Socket User Datagram Protocol (UDP)

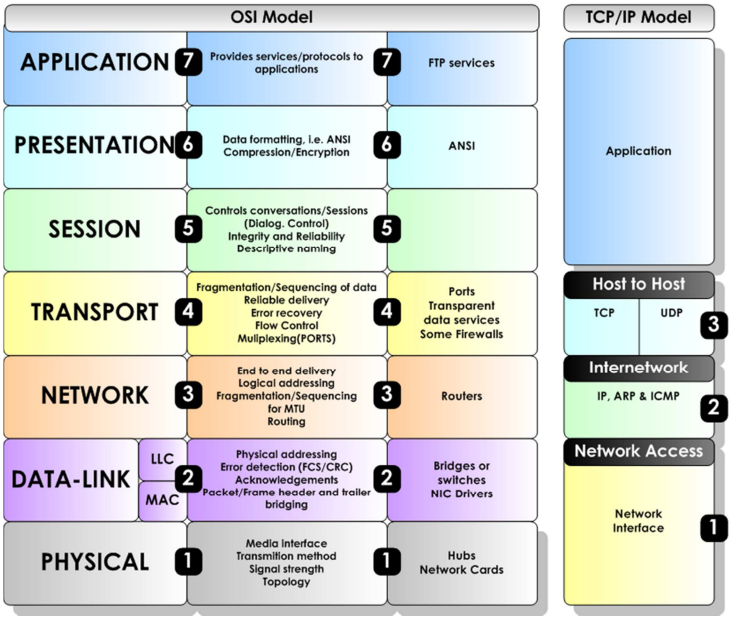
A. TUJUAN

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

B. DASAR TEORI

UDP adalah suatu protokol pengiriman data yang berbasis Internet Protocol (IP) dan bersifat *connection-less oriented*. Dengan kata lain UDP tidak ada jaminan paket akan dikirim pada urutan yang benar. Faktanya protokol ini tidak menjamin bahwa paket akan diterima semua. Selain itu UDP tidak melakukan pengecekan error atau pengurutan. Namun demikian, kurangnya UDP membuat lebih efisien dari pada TCP. UDP sangat berguna untuk pengiriman data dengan volume besar dan kecepatan tinggi. Seperti transmisi streaming audio dan video melalui internet. Pada kasus ini mekanisme acknowledgments, checksums, dan flow control pada TCP hanya menambah overhead pada transmisi.

Pada OSI layer UDP berada pada layer transport yang fungsinya mengatur pengiriman suatu data dari client ke server.

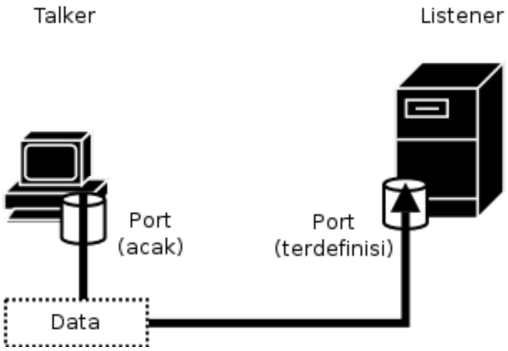


Gambar 1. UDP pada OSI layer

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.

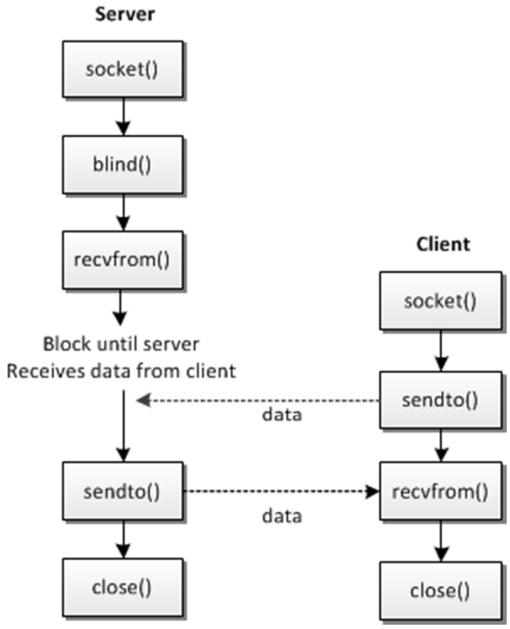
Gambar 2. Komunikasi client-server

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 client ke arah port pada PC server. 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.



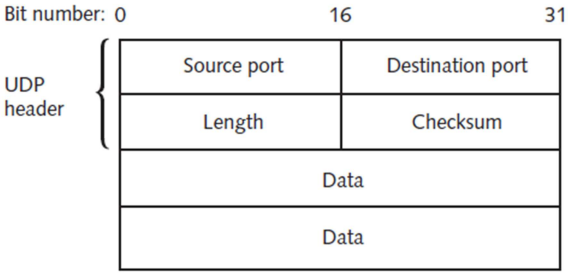
Gambar 3. Pengirim data melalui port

Alur penggunaan socket programming untuk UDP seperti pada gambar di bawah ini.



Gambar 4. Alur socket programming pada UDP

UDP header terdiri dari empat fields : Source port, Destination port, Length, dan Checksum. Penggunaan field checksum pada UDP adalah opsional. Gambar 5 menunjukan sebuah UDP segment.



Gambar 5. UDP segment

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 `talker` dan PC 2 sebagai `listener`.
2. Catat kedua IP, dengan menjalankan perintah `ifconfig`.
   Talker : (IP)
   Listener : (IP)
3. Buat folder baru dengan menjalankan perintah di bawah ini,
   ```
   #cd
   #mkdir udp
   #cd udp
   ```
4. Tulis program talker.c dan listener.c di bawah ini dengan menggunakan editor linux misalkan gedit atau vim.

   talker.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 <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

```
 #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, MAXBUFLEN1,0,(
 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;
}
```

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`
   Jika standard library belum terinstall, maka diinstall juga :
   `# apt-get install libc6-dev`
   atau
   `# apt-get install build-essential`
6. Lakukan kompilasi program dengan cara :
   Untuk program listerner.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-4.
7. Jalankan aplikasi wireshark pada sisi client, capture paket data yang dikirim sesuai interface yang digunakan.
8. Jalankan program dengan perintah, sebagai berikut :
   Untuk listener :
   `# ./listener`
   Untuk talker :
   `# ./talker 192.168.0.25 “Percobaan Pesan UDP”`
   Dimana 192.168.0.25 adalah IP dari komputer yang melakukan pemrograman listener. Pesan yang dikirim adalah percobaan pesan. Pada komputer yang menjalankan program listerner akan tampil data text tersebut.
   * Tips : Untuk mematikan program lakukan dengan menekan “Ctrl + C”
9. Catat hasil capture paket meliputi IP source, IP destination, protokol yang digunakan, source port, destination port, length, dan checksum.
10. Lakukan pengiriman text tersebut dengan kondisi sebagai berikut, kemudian amati pada komputer tersebut dan apabila muncul error catat di laporan sementara !
    - Program listener dijalankan di komputer A, pada komputer B kirim pesan dengan program talker ke komputer A.
    - Matikan program listener pada komputer A, pada komputer B kirim pesan dengan program talker ke komputer A.
11. Catat tampilan program dan proses pengiriman data pada sisi server dan client

Tabel Pengamatan

No	Listener	Talker	Pesan error
1.	dijalankan	dijalankan	
2.	dimatikan	dijalankan	