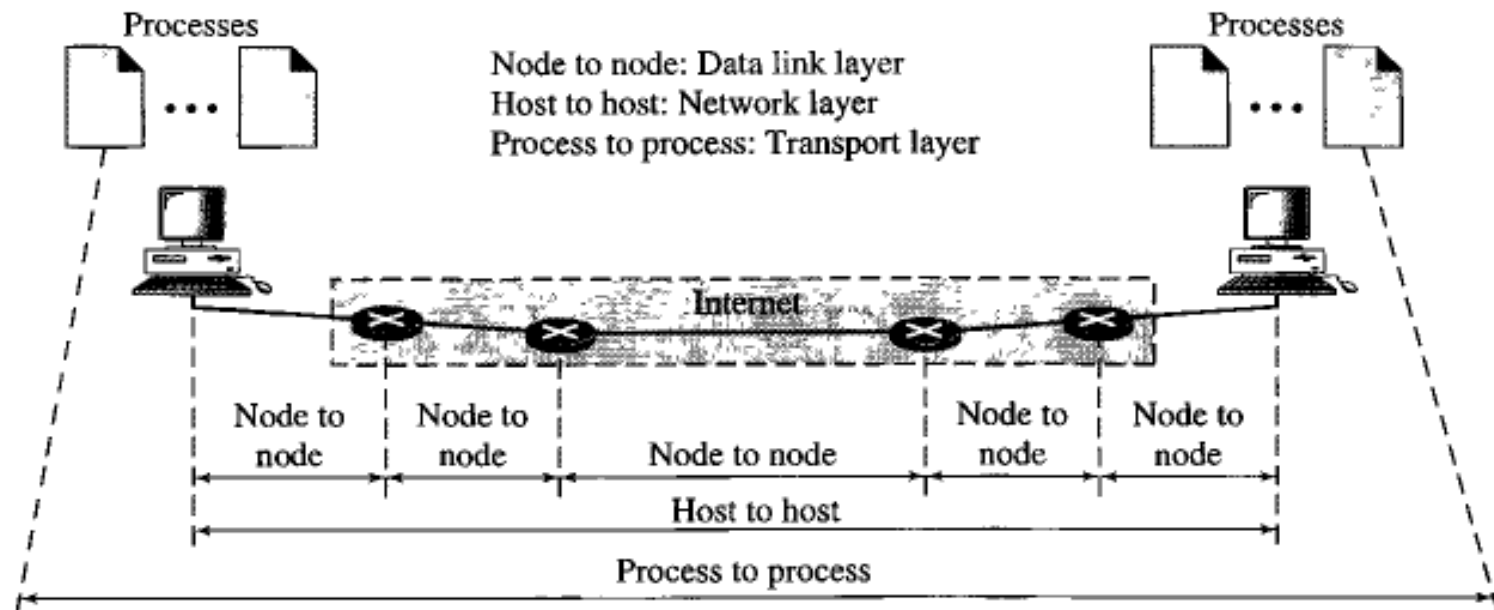




# Transport Layer




**Figure 23.1** *Types of data deliveries*






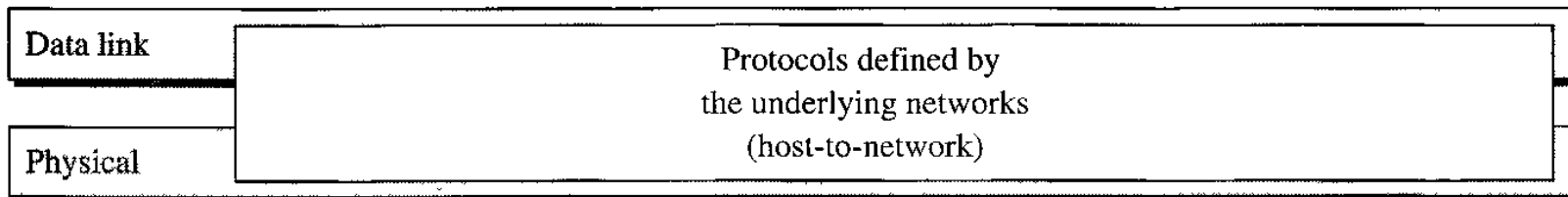
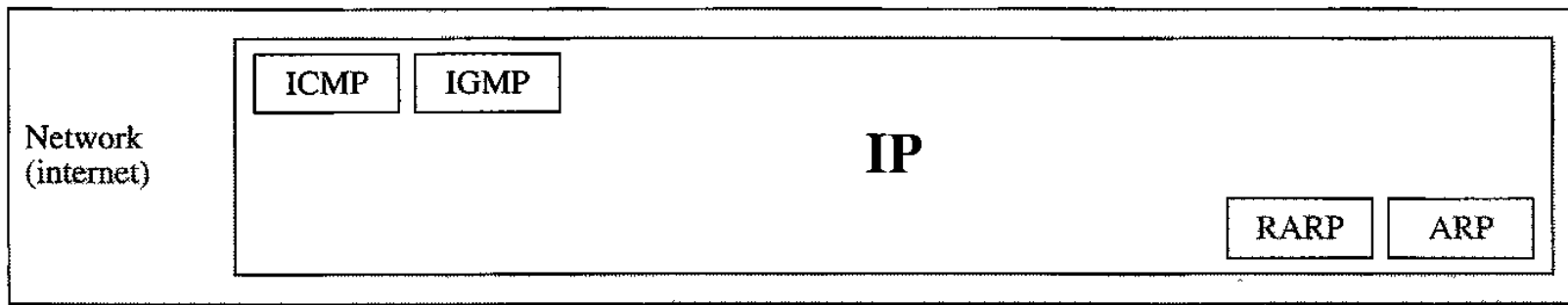
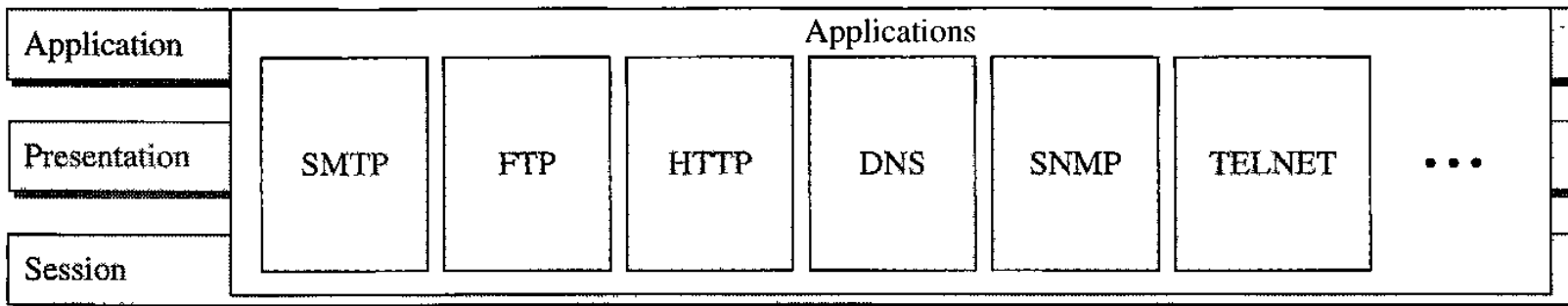
# Fungsi umum

- Memungkinkan multi aplikasi dapat dikomunikasikan melalui jaringan pada saat yang sama dalam single device.
  - Memastikan agar, jika diperlukan, data dapat diterima dengan handal dan oleh aplikasi yang benar.
  - Melibatkan mekanisme error handling.
- 



# Fungsi Lapis Transport

- Multiplexing dan demultiplexing
  - Segmentasi data
  - Menyediakan komunikasi logika antar proses aplikasi yang berjalan pada host yang berbeda.
  - Mengidentifikasi aplikasi yang berbeda berdasarkan nomor port.
  - Tracking the individual conversation.
- 



# Tipe Transfer Data

Komunikasi logika pada lapisan Transport dapat berbentuk :

- **connectionless** atau **connection-oriented**.
- **Reliable** atau **unreliable** :
  - **reliable**, jika data yang dikirim corrupt, bisa dikirim ulang untuk memastikan seluruh data mencapai tujuan. Lebih Lambat. Contoh TCP
  - **unreliable**, jika data yang dikirim corrupt, tidak dikirim ulang. Lebih cepat. Contoh : UDP
- **Stateful** atau **stateless**.
  - Pengiriman data stateful berarti informasi yang dimasukkan pada satu *request*, yang dikirimkan dari pengirim ke penerima, dapat dimodifikasi untuk *request* berikutnya.
  - Pengiriman stateless berarti informasi dalam satu request tidak dapat dikaitkan dengan *request* lainnya, sehingga tidak dapat digunakan untuk *request* lainnya.

## Transport Layer Protocols



- IP Telephony
- Streaming Video

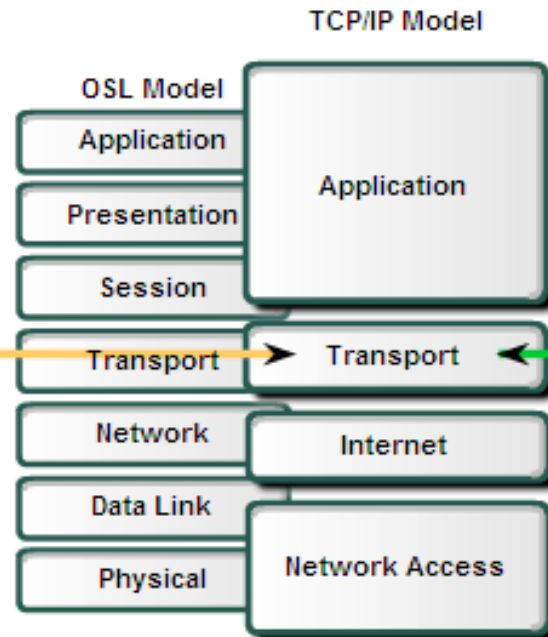
- SMTP/POP (Email)
- HTTP

### Required Protocol Properties

- Fast
- Low overhead
- Does not require acknowledgements
- Does not resend lost data
- Delivers data as it arrives

### Required Protocol Properties


- Reliable
- Acknowledge data
- Resend lost data
- Delivers data in order sent



Application developers choose the appropriate Transport Layer protocol based on the nature of the application.




Untuk alamat proses, yaitu port, berdasar standarisasi dari IANA, dapat dikelompokkan menjadi 3, yaitu

- **Well-known** port, yaitu 0 – 1023 → direserved untuk aplikasi dan servis (misal untuk HTTP : 80, POP3: 110, Telnet: 23 , SMTP : 25, IRC : 194, TFTP : 69)
  - **Registered** port, yaitu 1024 – 49151 → diassign untuk user process atau aplikasi → lebih ke 'install'. ( MS SQL : 1433, WAP: 2948, MSN Messenger : 1863)
  - **Dynamic/Private** port, 49152 – 65535 → inisialisasi connection, peer to peer file sharing program, dapat dipilih secara dinamik oleh client sebagai source port
- 





## Protokol Transport dan Layanannya

- Protokol transport adalah protokol end2end (hanya berada di end sistem, tidak ada di jaringan).
  - Protokol transport tergantung pada layanan dilakukan oleh protokol jaringan, sekaligus menambah kualitas layanan sistem
- 



# TCP dan UDP

2 jenis protokol yang banyak digunakan pada transport layer:

Transmission Control protocol (TCP)

User Datagram Protocol (UDP)

Protokol tambahan pada transport layer

Stream Control Transmission Protocol (SCTP)

Datagram Congestion Control Protocol (DCCP)

Fiber Channel Protocol (FCP)





# Layanan Transport Internet

Layanan pengiriman yang handal, berurutan, unicast (TCP)

Kendali aliran

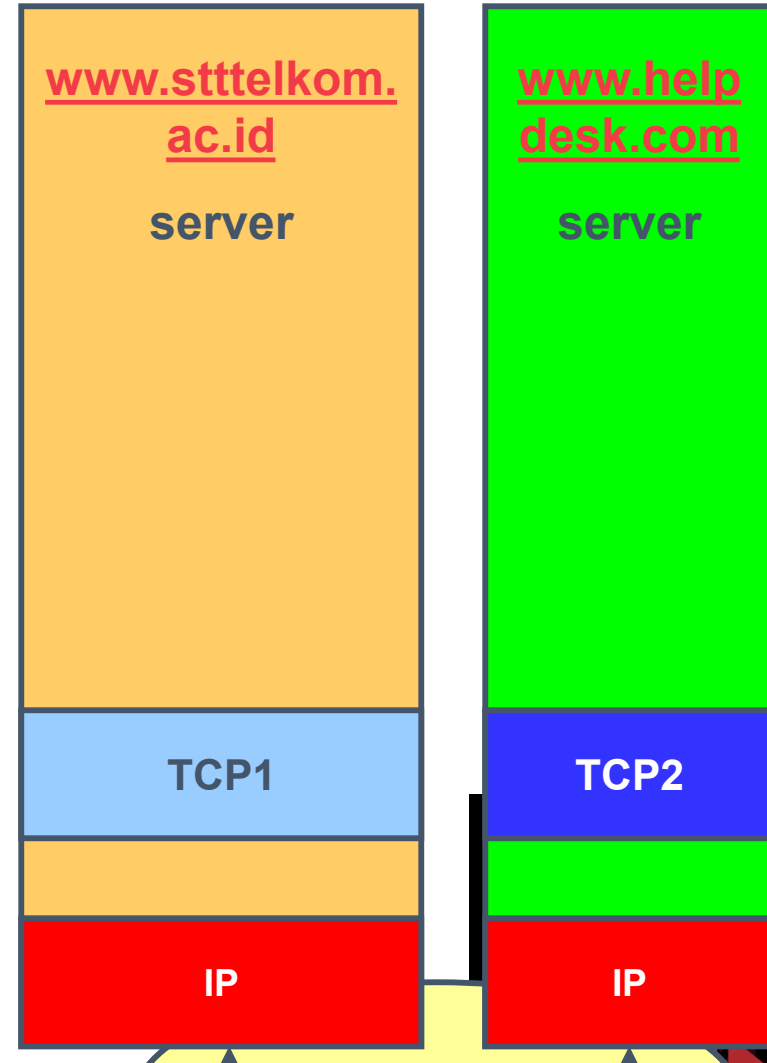
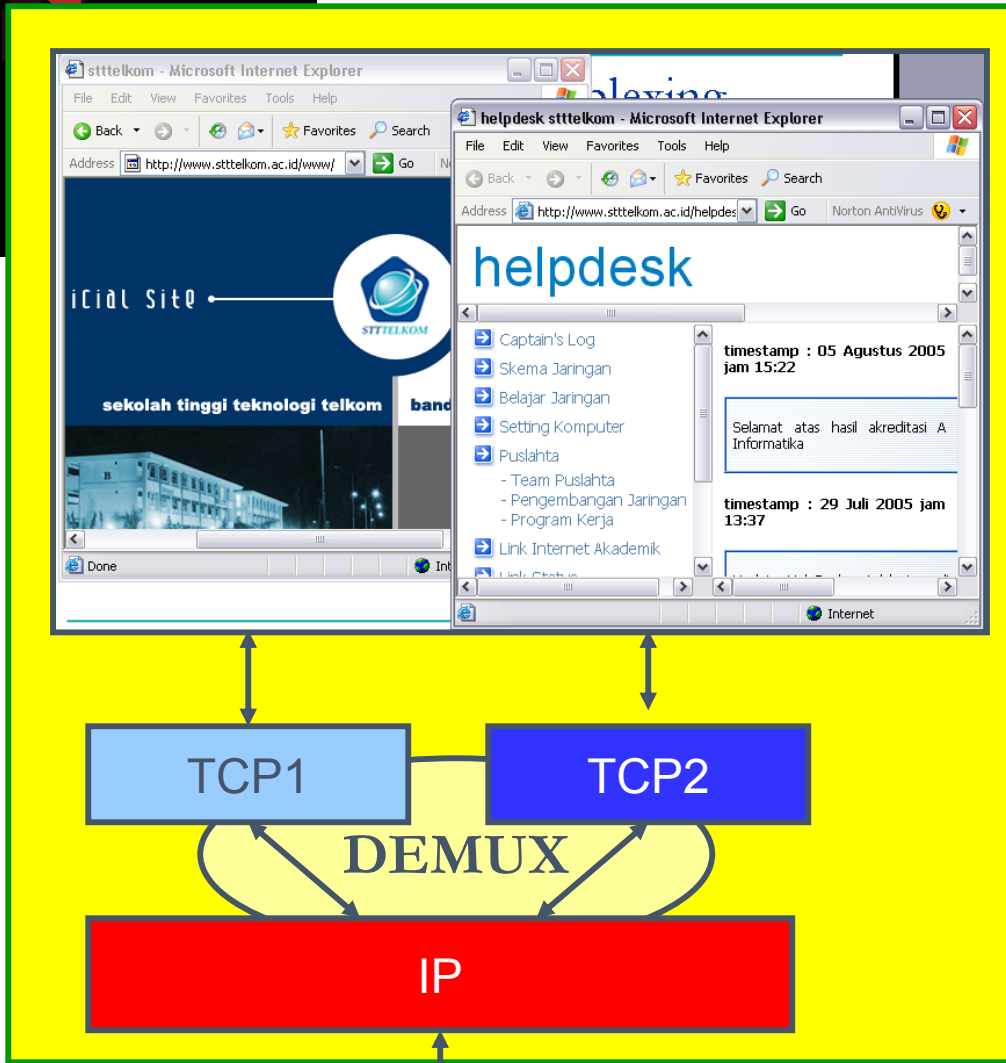
Kendali collision


Pembangunan hubungan

Layanan pengiriman yang tidak handal (best-effort), tidak berurutan, unicast atau multicast (UDP)



# Multiplexing dan Demultiplexing





**Multiplexing** : pengumpulan data dari banyak proses aplikasi kemudian membungkusnya dengan header

Berdasarkan alamat port pengirim dan penerima, alamat IP

Nomor port ada di setiap segmen TCP/UDP

Nomor port untuk aplikasi standar TCP/IP telah ditentukan

HTTP = 80

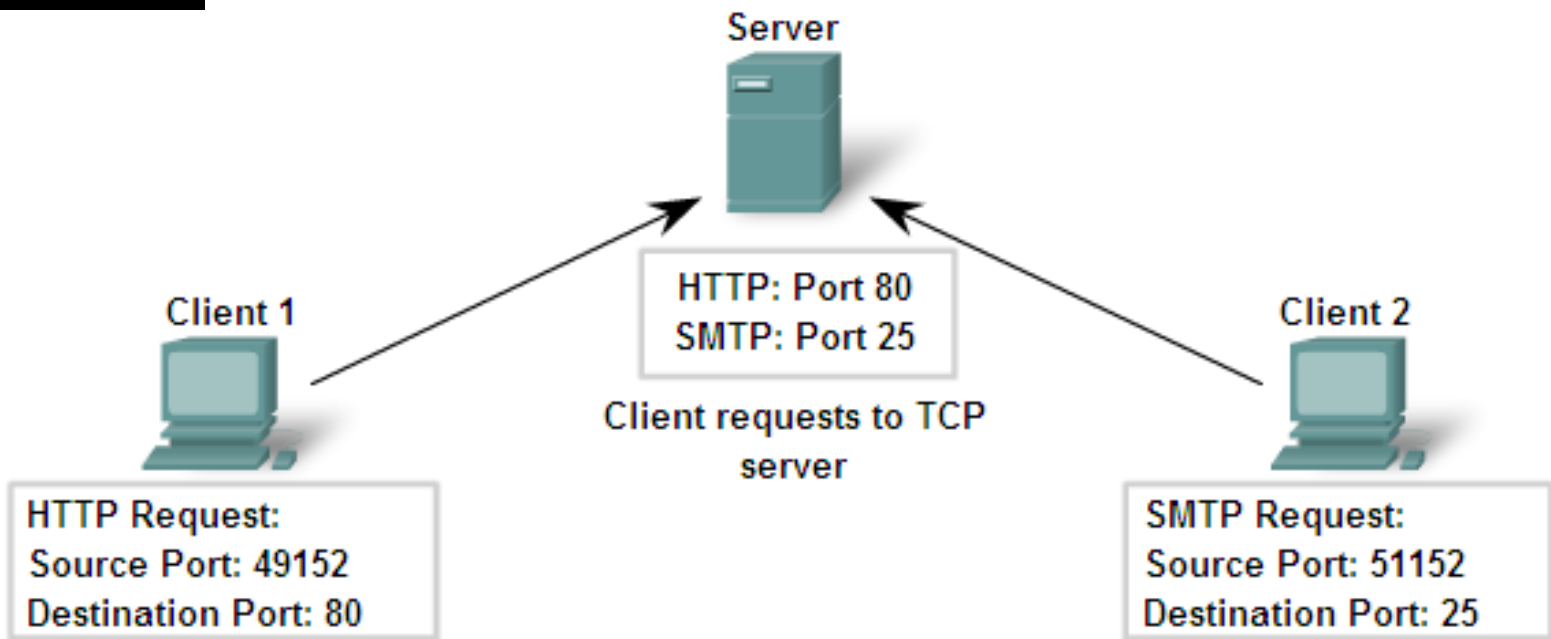
SMTP = 25

POP3 = 110

TELNET = 23

dll








# TCP

Berikut beberapa karakteristik TCP :

- point-to-point
  - reliable dan stateful
  - pipeline, menerapkan flow control sliding window untuk mengontrol kemacetan dan aliran datanya
  - terdapat buffer pengirim dan penerima
  - header min 20 byte, maksimum 60 byte
  - connection-oriented
  - aliran data yang dikontrol
- 

## TCP Segment

Bit (0)	Bit (15)	Bit (16)	Bit (31)
Source Port (16)		Destination Port (16)	
Sequence Number (32)			
Acknowledgement Number (32)			
Header Length (4) Reserved (6) Code Bits (6)		Window (16)	
Checksum (16)		Urgent (16)	
Options (0 or 32 if any)			
APPLICATION LAYER DATA (Size varies)			

↑  
20  
Bytes  
↓



Source port dan destination port ( $2^{16}$ ) : alamat lapis 4 (satu protokol lapis 3 bisa mendukung lebih dari 1 buah protokol lapis 4), misal : **10.14.64.54:80** berbeda port dengan **10.14.64.54:102**

Sequence number ( $2^{32}$ ) : nomor paket yang dikirim

Acknowledge number ( $2^{32}$ ): nomor acknowledge untuk paket telah diterima (nomor acknowledge merupakan nomor paket selanjutnya)

URG,ACK,PSH,RST,SEQ,FIN : bit-bit status pada paket.

URG = Urgent pointer field valid

ACK = Acknowledgment field valid

PSH = Deliver data on receipt of this segment

RST = Reset the sequence/acknowledgment numbers

SEQ = Sequence number valid

FIN = End of byte stream from sender



# Manajemen Koneksi TCP

Pada saat **Setup Koneksi** → **Three Way Handshake**


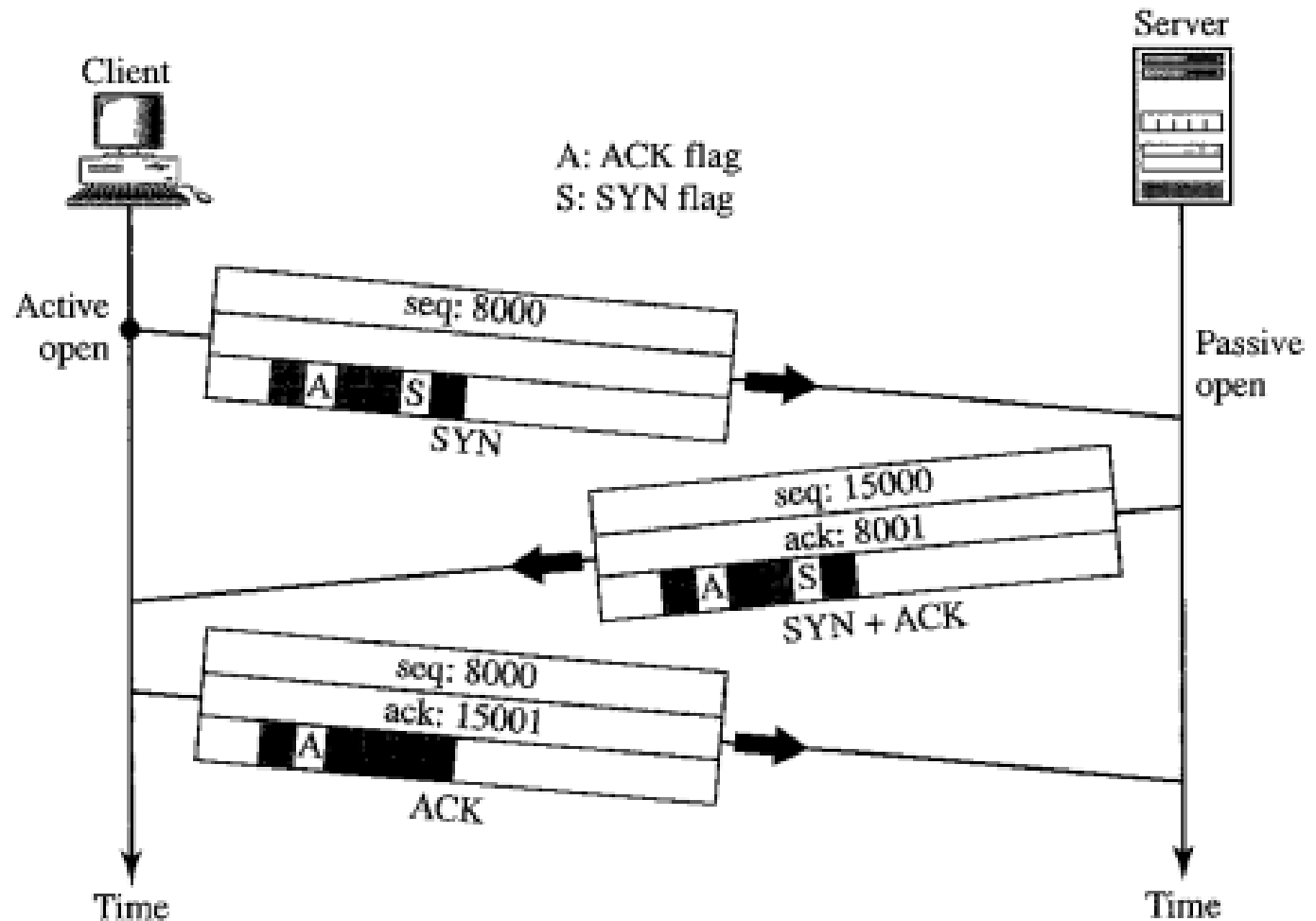

1. Membuka hubungan ke destination device
  2. Memverifikasi bahwa destination device aktif dan menerima request pada nomor port destination yang diminta client untuk memulai sesi.
  3. Menginformasikan kepada destination device bahwa siap membentuk hubungan pada port yang diminta
- 

Figure 23.18 *Connection establishment using three-way handshaking*

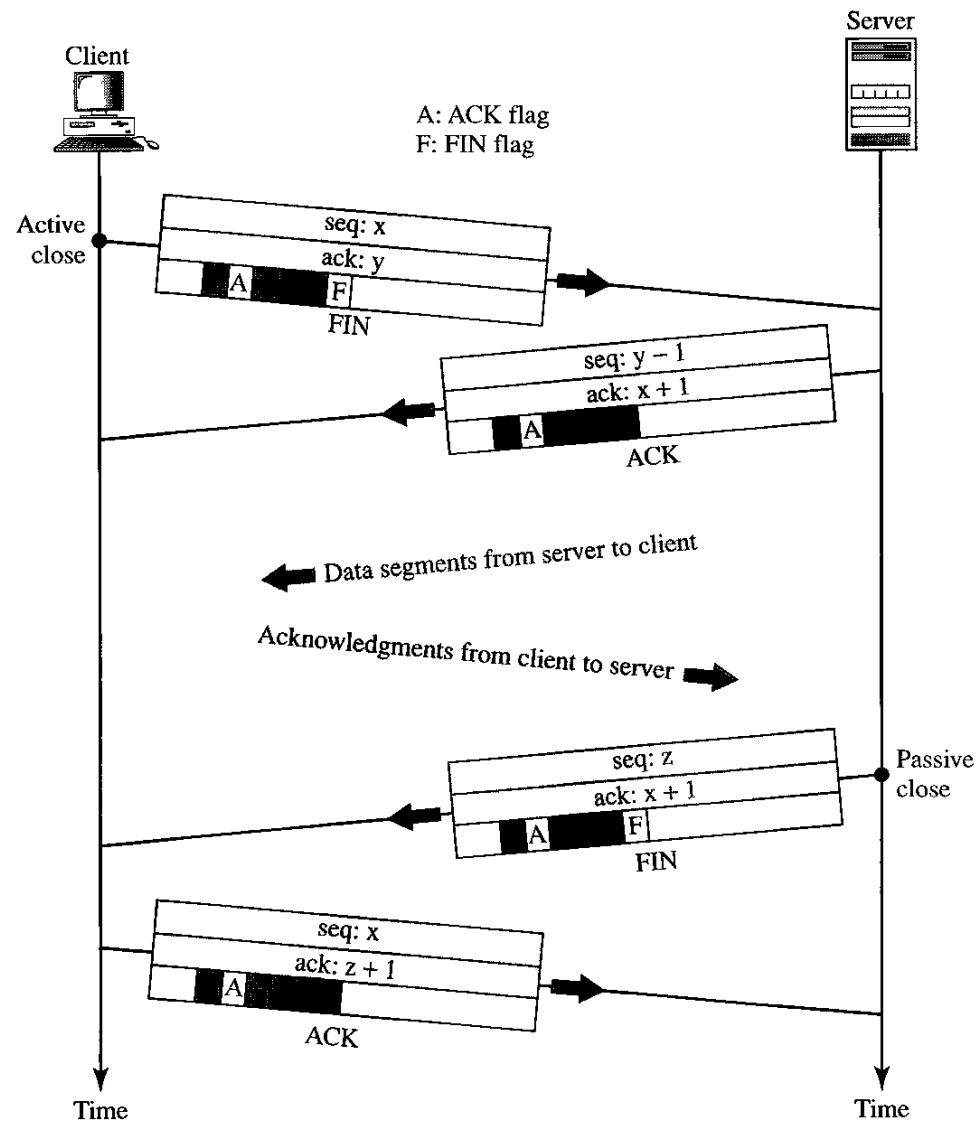
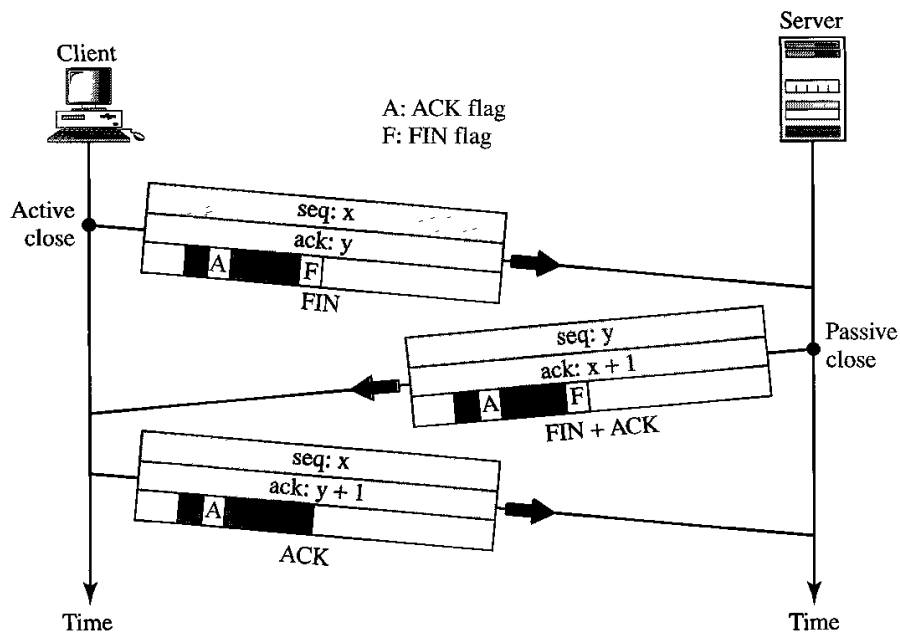




## Pada saat **Menutup Koneksi**

1. client mengirim kontrol TCP FIN ke Server
  2. server menerima FIN, dan membalas dengan ACK. Menutup koneksi dan mengirimkan FIN ke client.
  3. Client menerima FIN dan membalas ACK
  4. Server menerima ACK dan koneksi tertutup.
- 

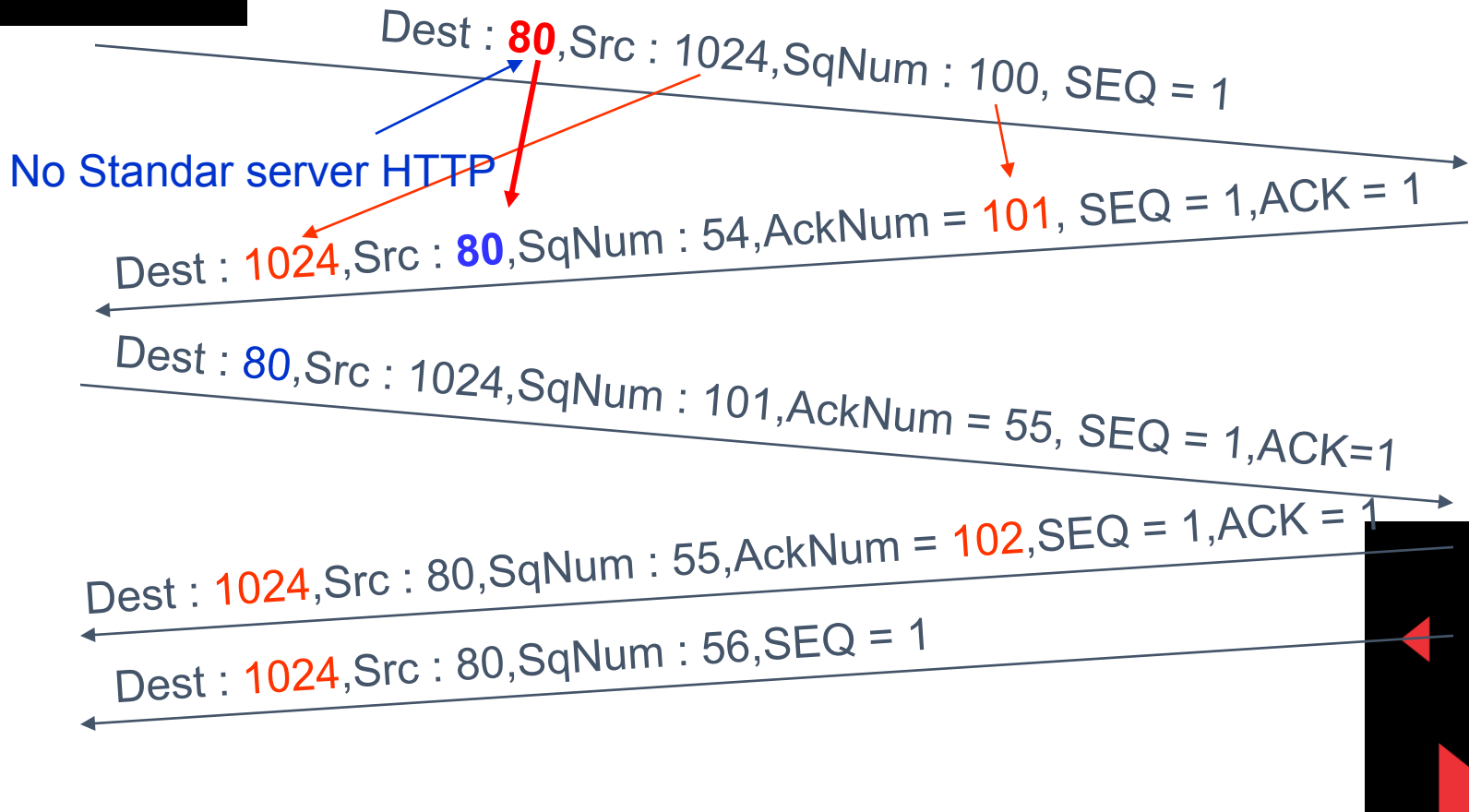
# Three way hand shake & half close



# Contoh komunikasi

server

host



host

server

Dest : 80, Src : 1024, SeqNum : 102, AckNum = 57, SEQ = 1, ACK=1

Dest : 1024, Src : 80, SeqNum : 57, AckNum = 103, SEQ = 1, ACK = 1

Dest : 1024, Src : 80, SeqNum : 58, SEQ = 1

Dest : 1024, Src : 80, SeqNum : 59, SEQ = 1

Dest : 1024, Src : 80, SeqNum : 60, SEQ = 1

Dest : 80, Src : 1024, SeqNum : 103, AckNum = 59, SEQ = 1, ACK=1

Dest : 1024, Src : 80, SeqNum : 59, AckNum = 104, SEQ = 1, ACK = 1, FIN=1

Pembubaran hubungan

error

t





# UDP

## (User Datagram Protocol)

Menyediakan layanan “best effort” sehingga segmen UDP bisa:

- Hilang

- Tidak berurutan diterima

### Connectionless

- Tidak ada handshaking antara host dan server


- Setiap segmen UDP ditangani secara independen








## Mengapa ada UDP?

- ✓ Tidak perlu adanya setup koneksi terlebih dahulu (hal ini dapat mengurangi delay)
  - ✓ sederhana, artinya antara penerima dan pengirim tidak perlu menjaga session atau status koneksi
  - ✓ ukuran header segment lebih kecil → 8 byte
  - ✓ tidak perlu kontrol kemacetan koneksi, artinya UDP dapat mengirimkan per segment tanpa dipengaruhi oleh kesibukan jaringan
- 



# Latar Belakang UDP (RFC768)

- Tidak ada pembangunan hubungan (menambah delay)
  - Header kecil (8 byte, dibanding TCP min 20 byte)
  - Tidak ada kendali kongesti : UDP bisa dikirim secepat mungkin
  - Umum digunakan untuk aplikasi dengan req:
    - Loss tolerant
    - Rate sensitive
- 



## Pengguna UDP:

Domain Name System (DNS)


Simple Network Management Protocol (SNMP)

Dynamic Host Configuration Protocol (DHCP)

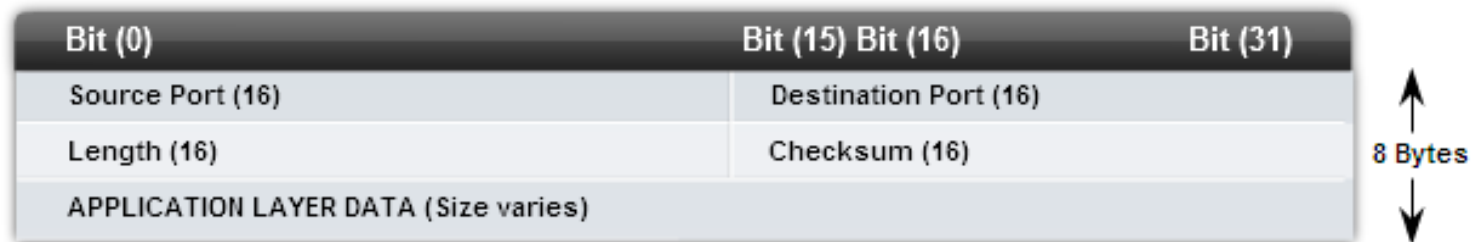
Trivial File Transfer Protocol (TFTP)

Online games

Jika diinginkan kehandalan yang tinggi tetapi tetap menggunakan UDP, maka harus ditambahkan fungsi kehandalan pada protokol lapis aplikasi atau lapis diatas UDP (disesuaikan dengan requirement aplikasi)



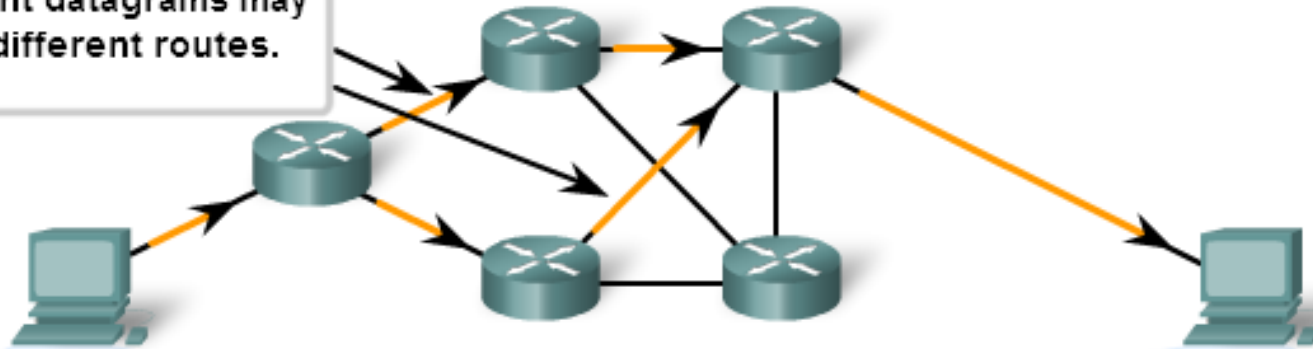
## UDP Datagram



- Source port = alamat host
- Destination port = alamat server
- Length = panjang total paket UDP
- Checksum = fungsi deteksi error sederhana

## UDP: Connectionless and Unreliable

Different datagrams may take different routes.



Data

Data is divided into datagrams.

Datagram 1

Datagram 2

Datagram 3

Datagram 4

Datagram 5

Datagram 6

Having taken different routes to the destination, datagrams arrive out of order.

Datagram 1

Datagram 2

Datagram 6

Datagram 5

Datagram 4

Out of order datagrams are not re-ordered.

Lost datagrams are not re-sent.



SEKIAN

