NETWORK PROGRAMMING

PERTEMUAN 1

PENGANTAR

M. Yusril Helmi Setyawan

OVERVIEW

- Network Computer – Basic Concept
- Introduction of Network Programming
- Port, Socket & Protocols

1. Network Computer : Basic Concept

Role of end devices:

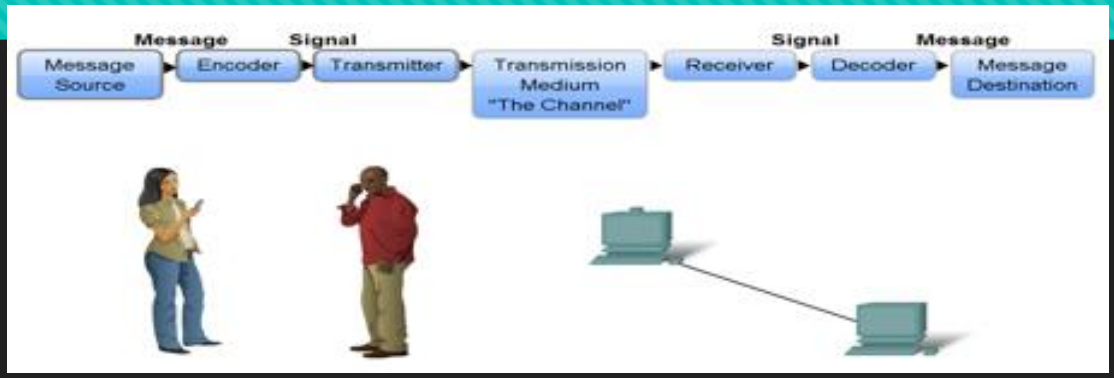
- client
- server
- both client and server

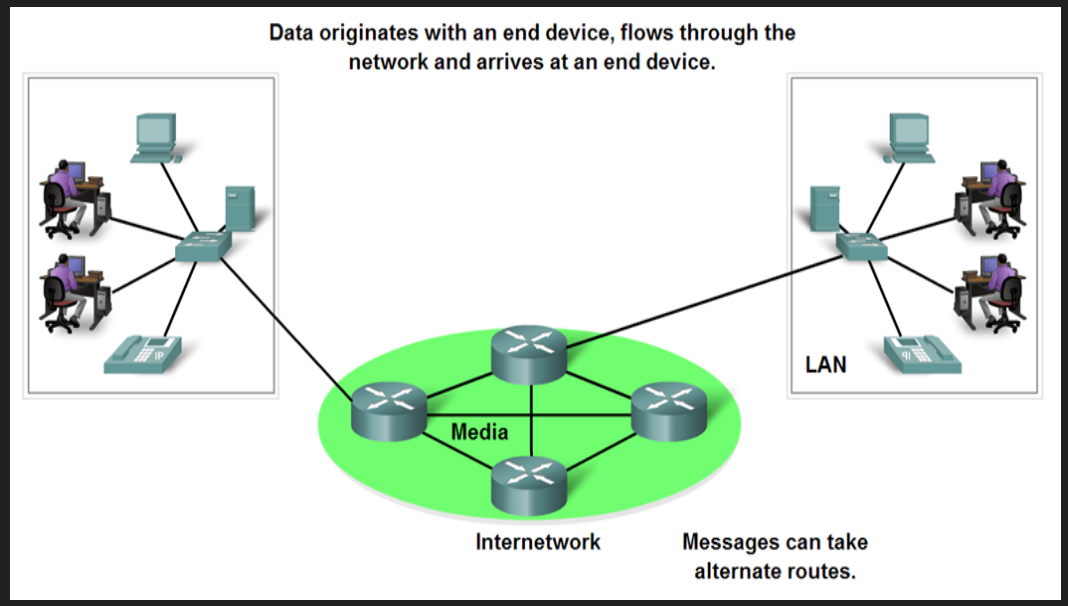
Network components

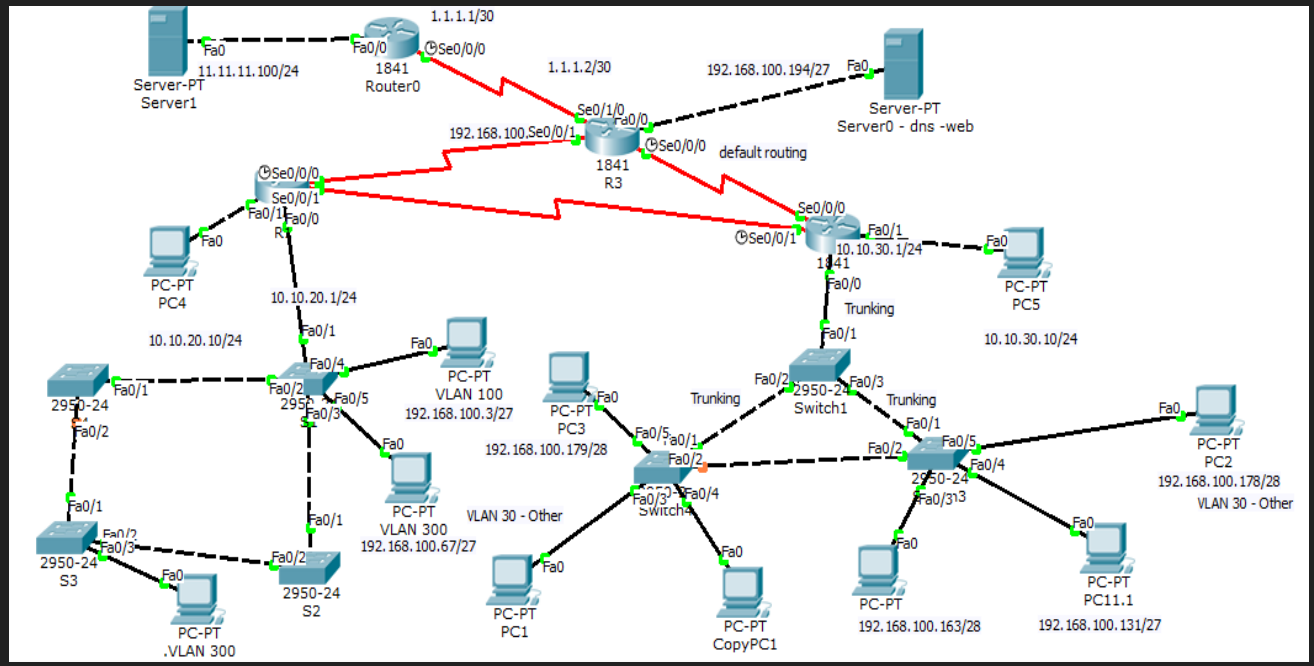
- hardware
- software

3 common elements of communication

- message source
- the channel
- message destination

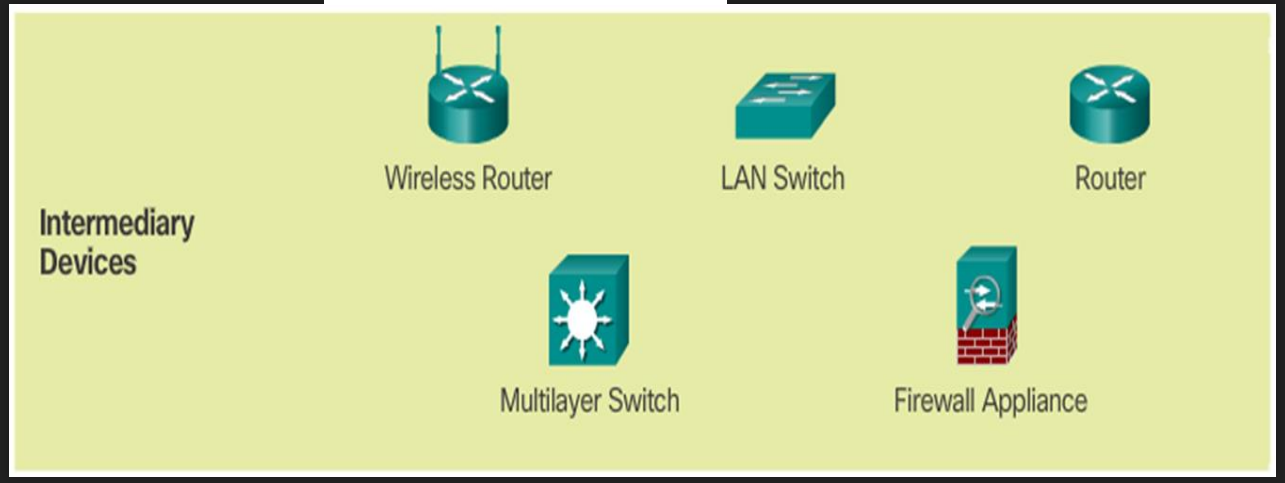






Network Structure

intermediary device

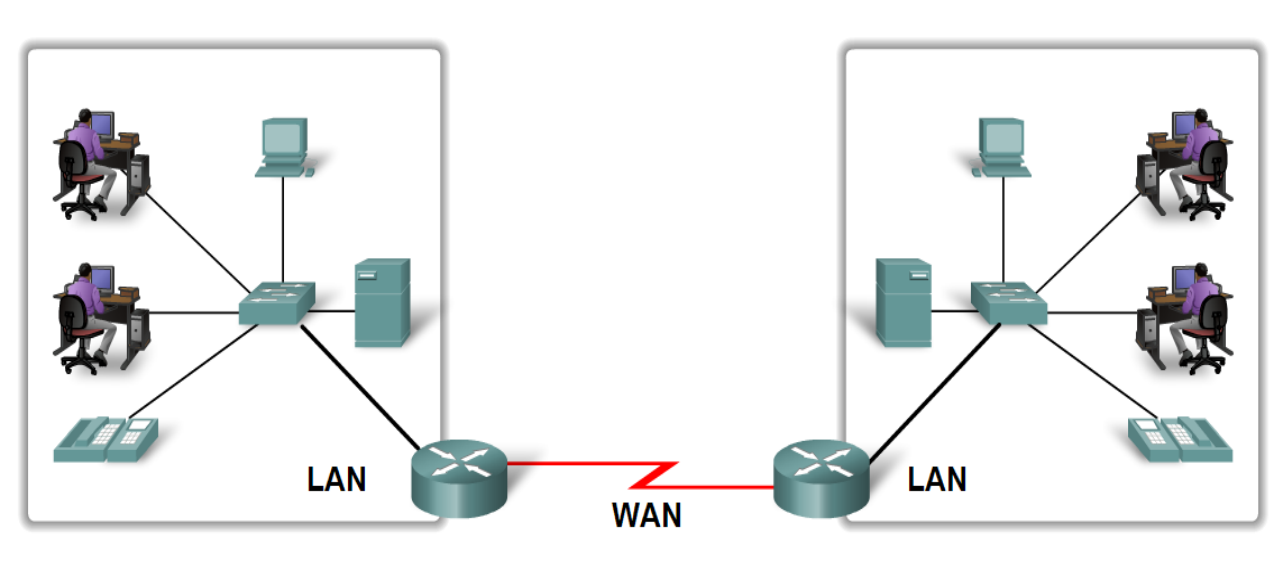


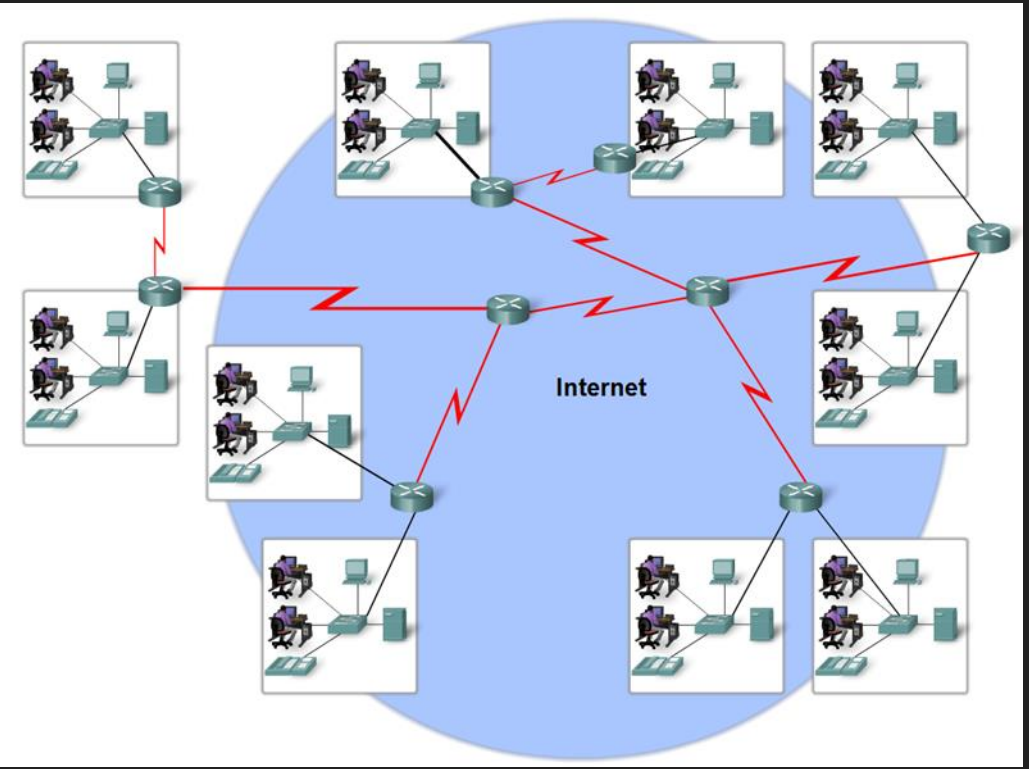
Network media

Intermediary network devices perform some or all of these functions:

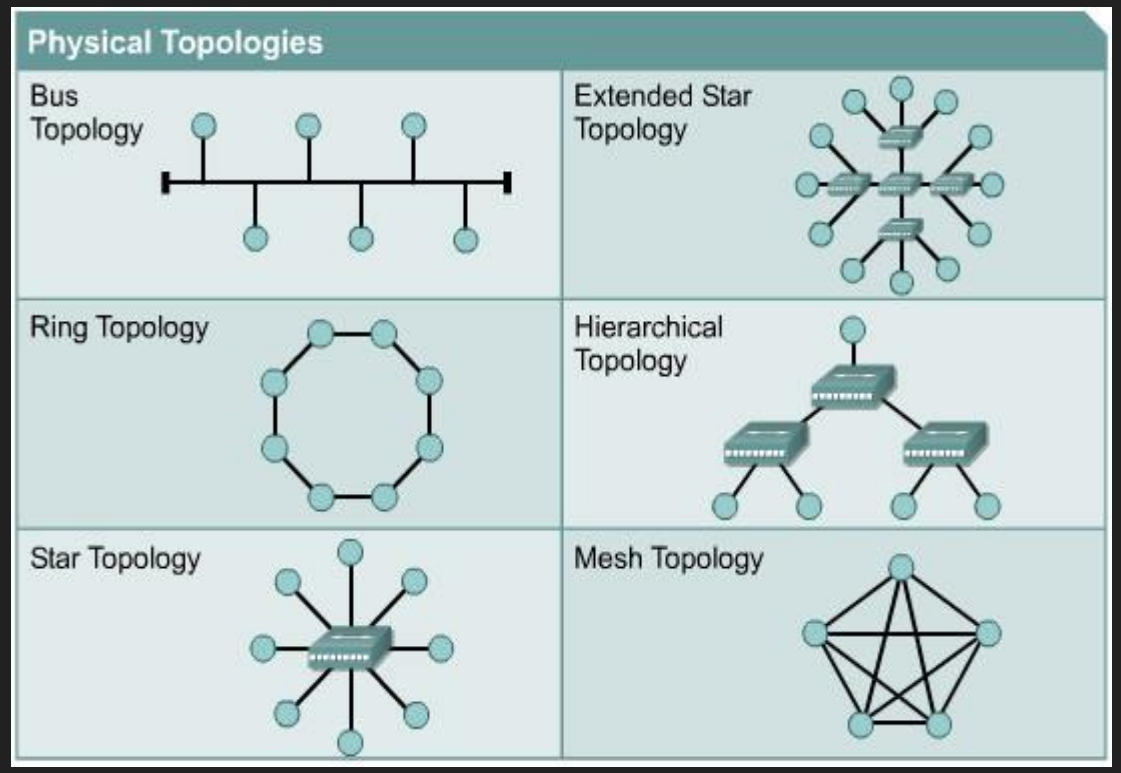
- Regenerate and retransmit data signals
- Maintain information about what pathways exist through the network and internetwork
- Notify other devices of errors and communication failures
- Direct data along alternate pathways when there is a link failure
- Classify and direct messages according to priorities
- Permit or deny the flow of data, based on security settings

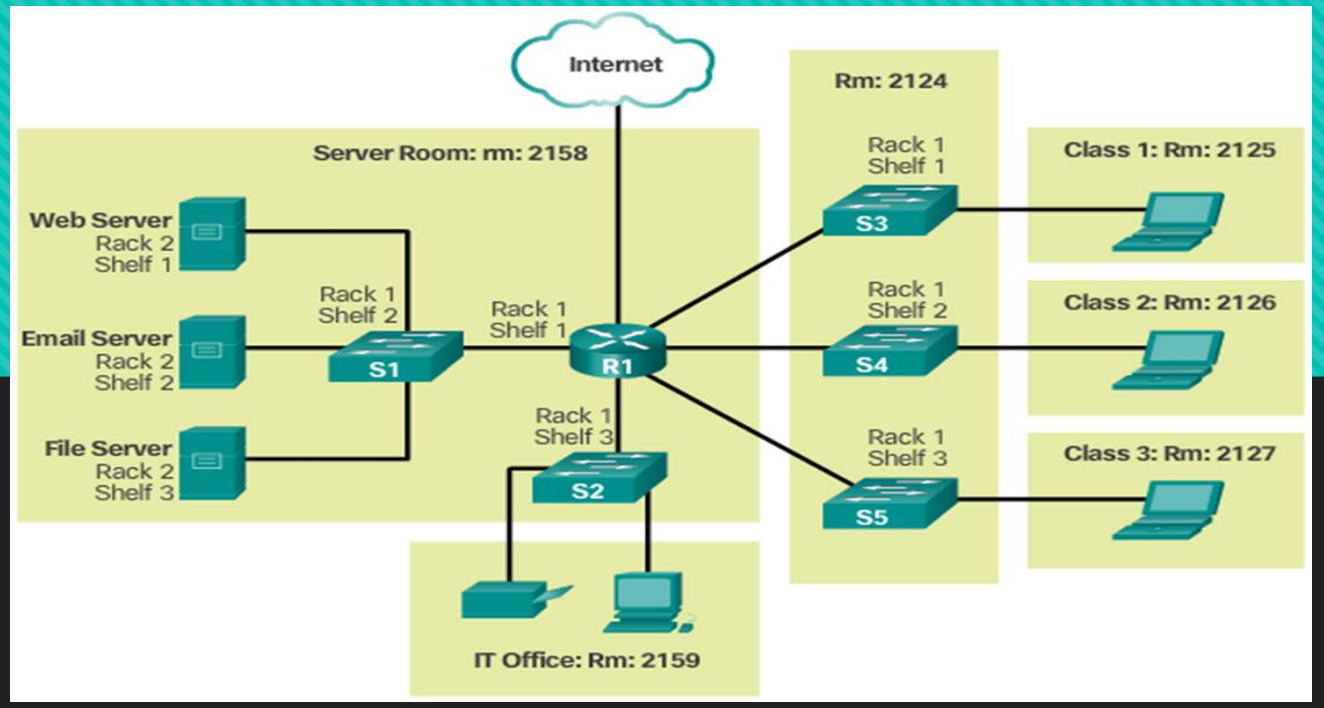
Network Type

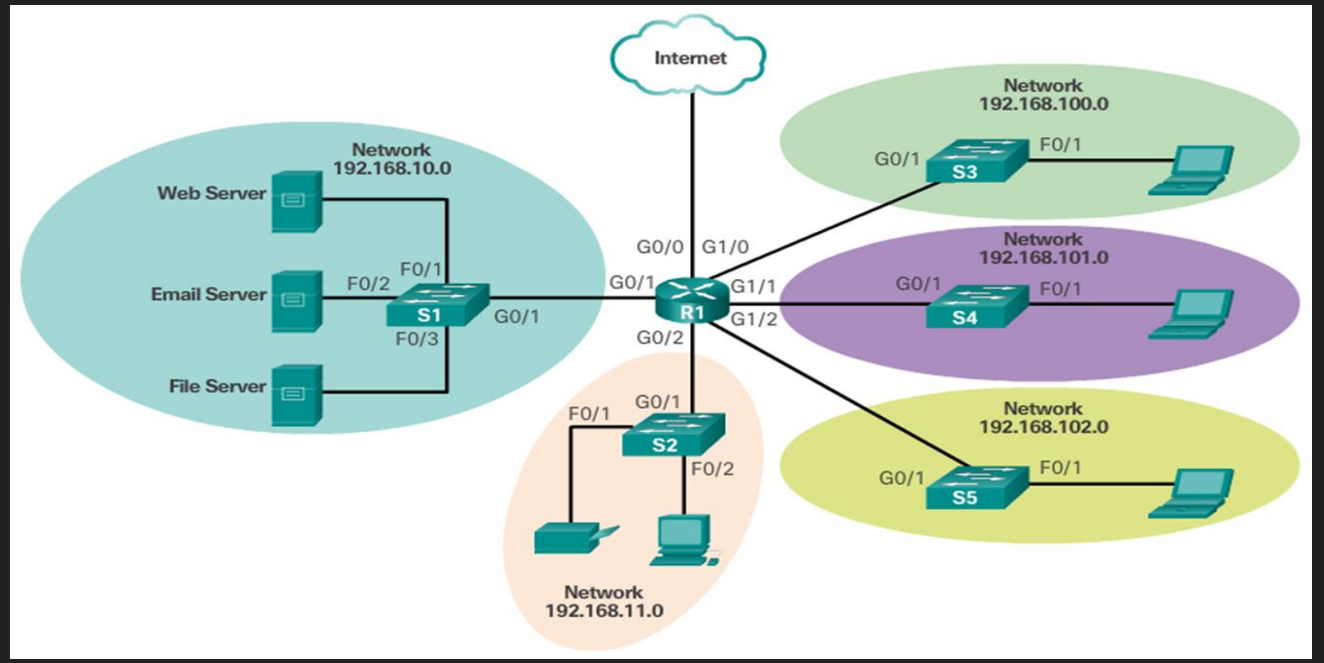




TOPOLOGY

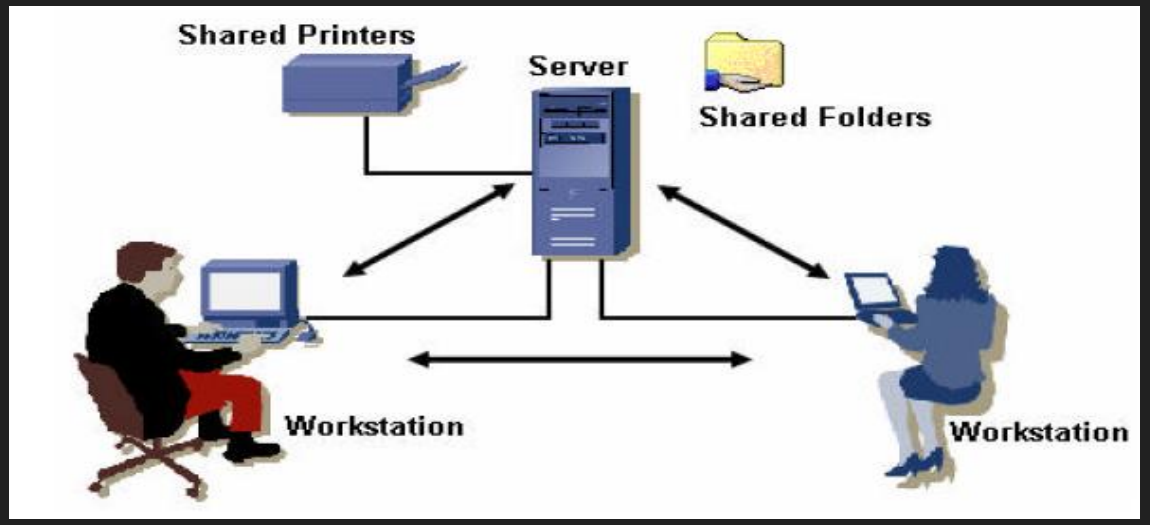

Physical Topologies
Bus Topology
Extended Star Topology
Ring Topology
Hierarchical Topology
Star Topology
Mesh Topology


Internet
Server Room: rm: 2158
Rm: 2124
Web Server
Rack 2
Shelf 1
Email Server
Rack 2
Shelf 2
File Server
Rack 2
Shelf 3
Rack 1
Shelf 2
S1
Rack 1
Shelf 1
R1
Rack 1
Shelf 3
S2
Rack 1
Shelf 1
S3
Rack 1
Shelf 2
S4
Rack 1
Shelf 3
S5
Class 1: Rm: 2125
Class 2: Rm: 2126
Class 3: Rm: 2127
IT Office: Rm: 2159

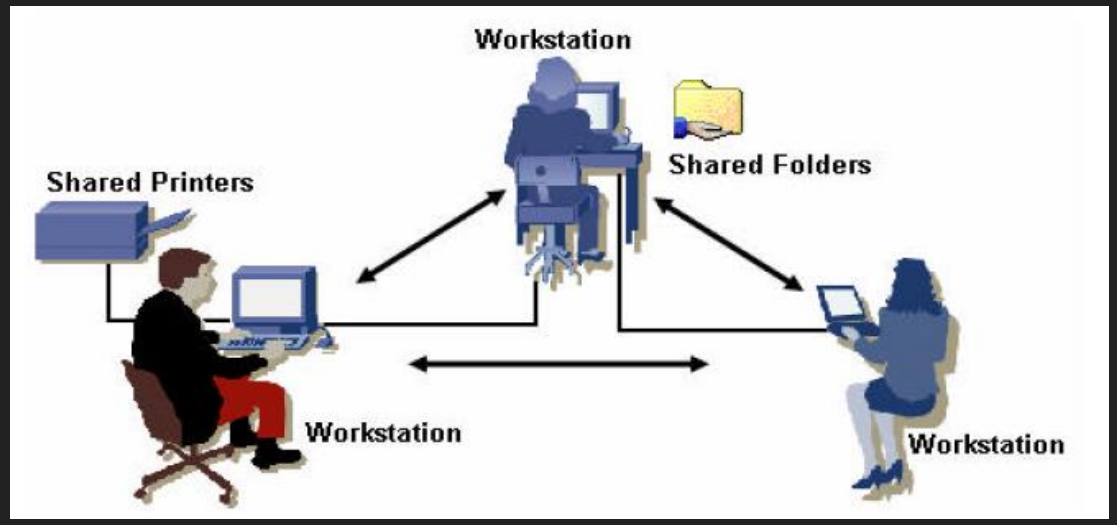

Internet
Network
192.168.10.0
Web Server
Email Server
File Server
F0/1
F0/2
F0/3
S1
G0/1
G0/0
G1/0
G0/1
R1
G1/1
G1/2
G0/2
Network
192.168.100.0
G0/1
S3
F0/1
Network
192.168.101.0
G0/1
S4
F0/1
Network
192.168.102.0
G0/1
S5
F0/1
F0/1
G0/1
S2
F0/2
Network
192.168.11.0

Network Models

Client – Server



Peer to Peer



Network Protocols

- Network protocols are used to allow devices to communicate successfully

Protocols provide:

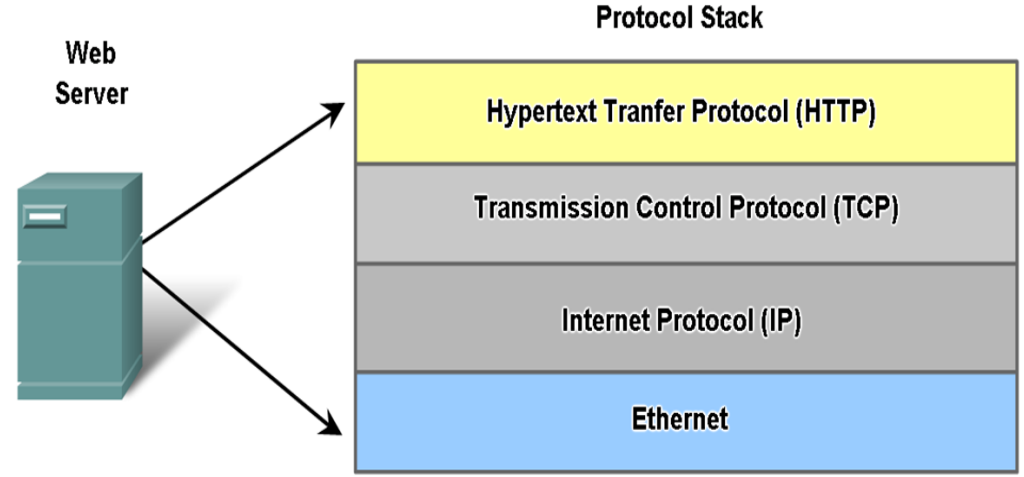
The format or structure of the message

The process by which networking devices share information about pathways to other networks

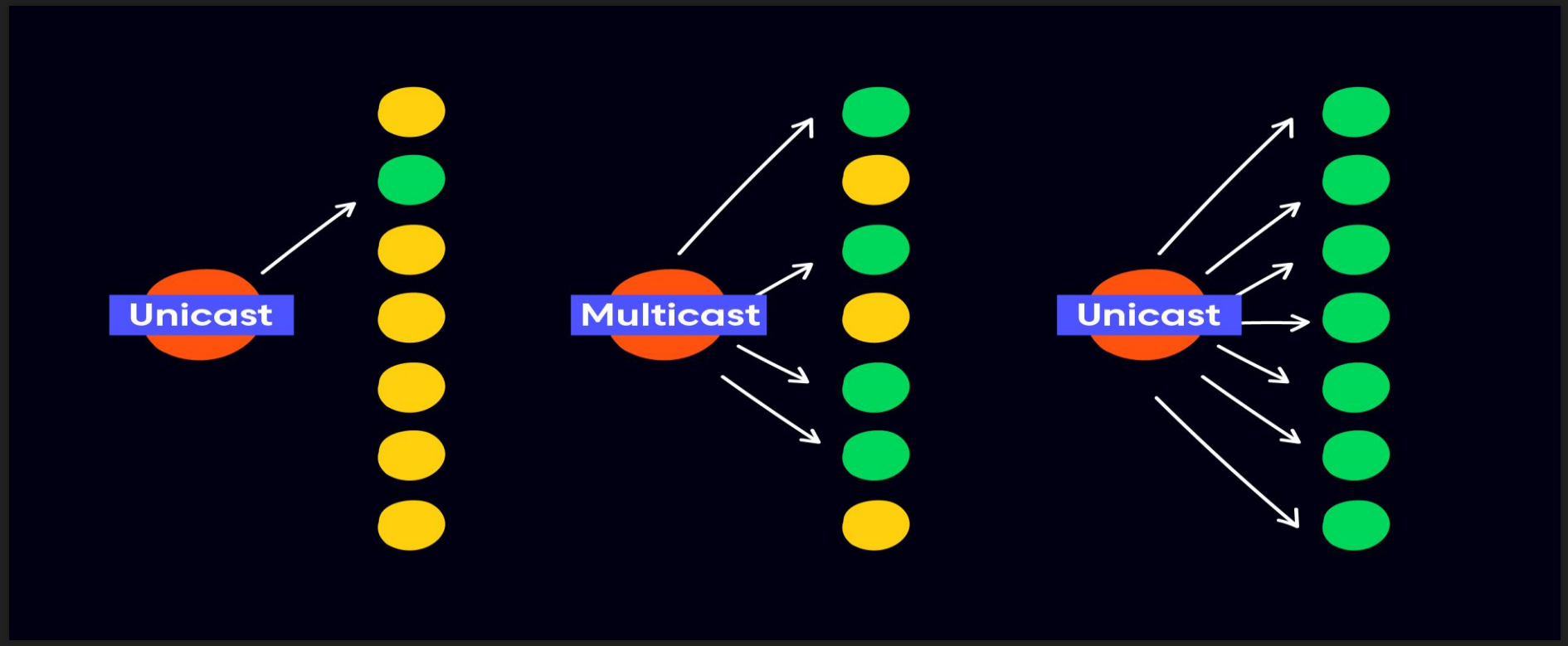
How and when error and system messages are passed between devices

The setting up and termination of data transfer sessions

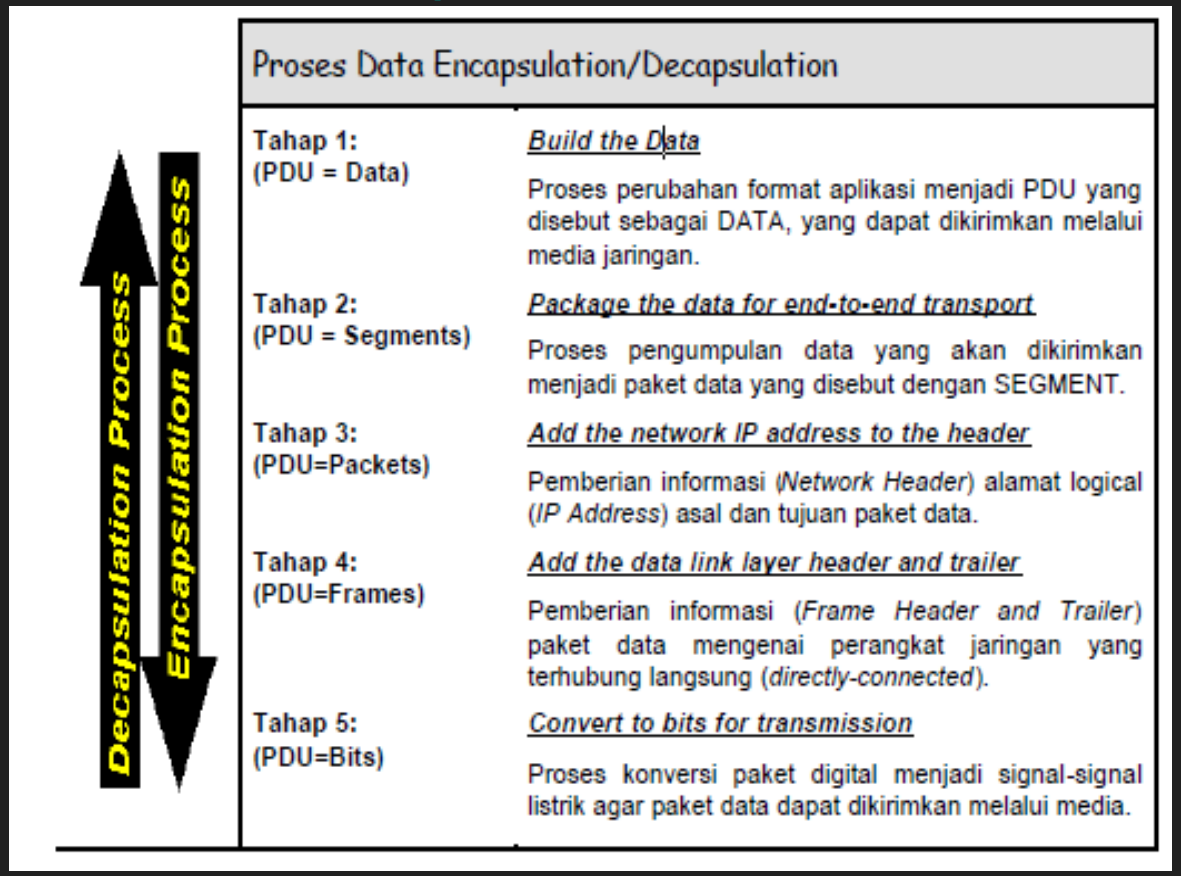
Service	Protocol/Rule
World Wide Web (WWW)	HTTP (Hypertext Transfer Protocol)
E-mail	SMTP (Simple Mail Transport Protocol) POP (Post Office Protocol)
Instant Message (AIM, Jabber, ICQ)	XMPP (Extensible Messaging **and** Presence Protocol)
	OSCAR (Open System For Communication in Realtime)
IP Telephony	SIP (session initiation protocol)

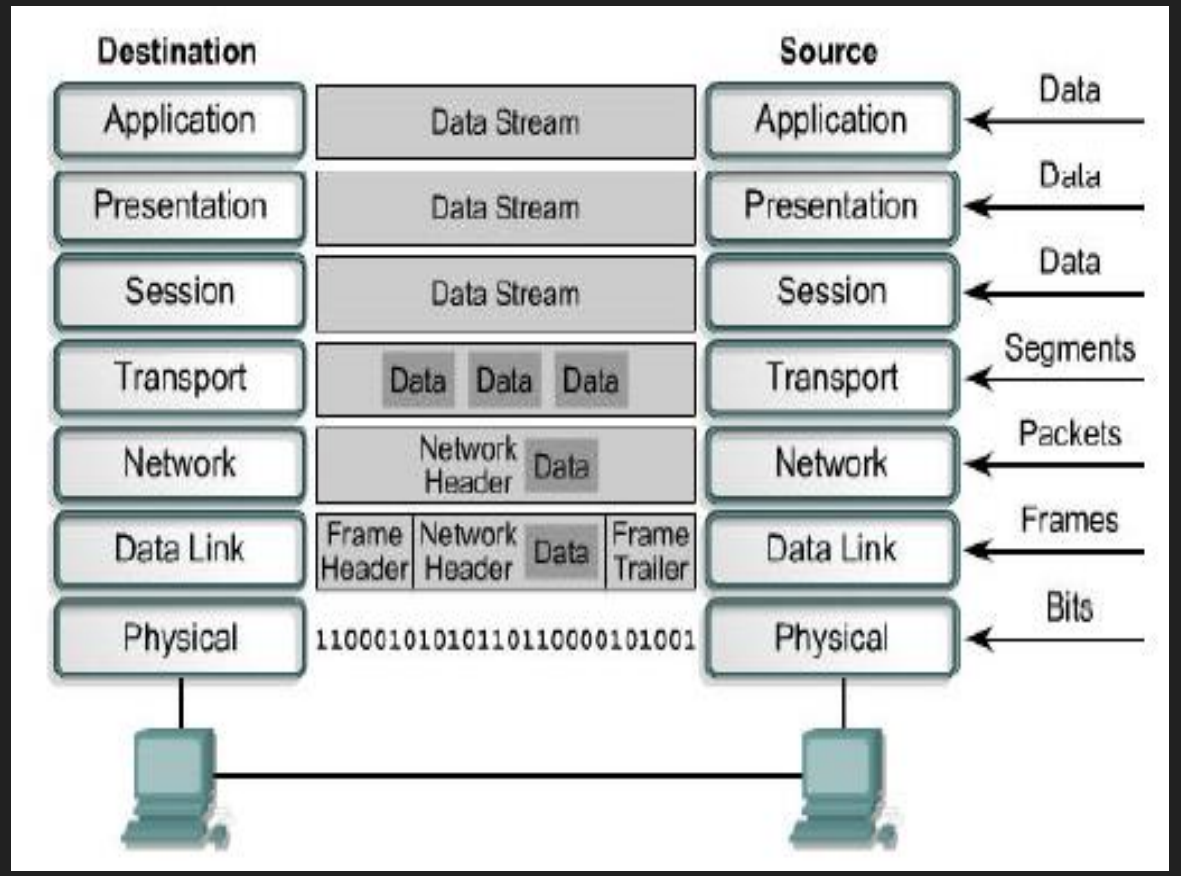


Transmission Technology

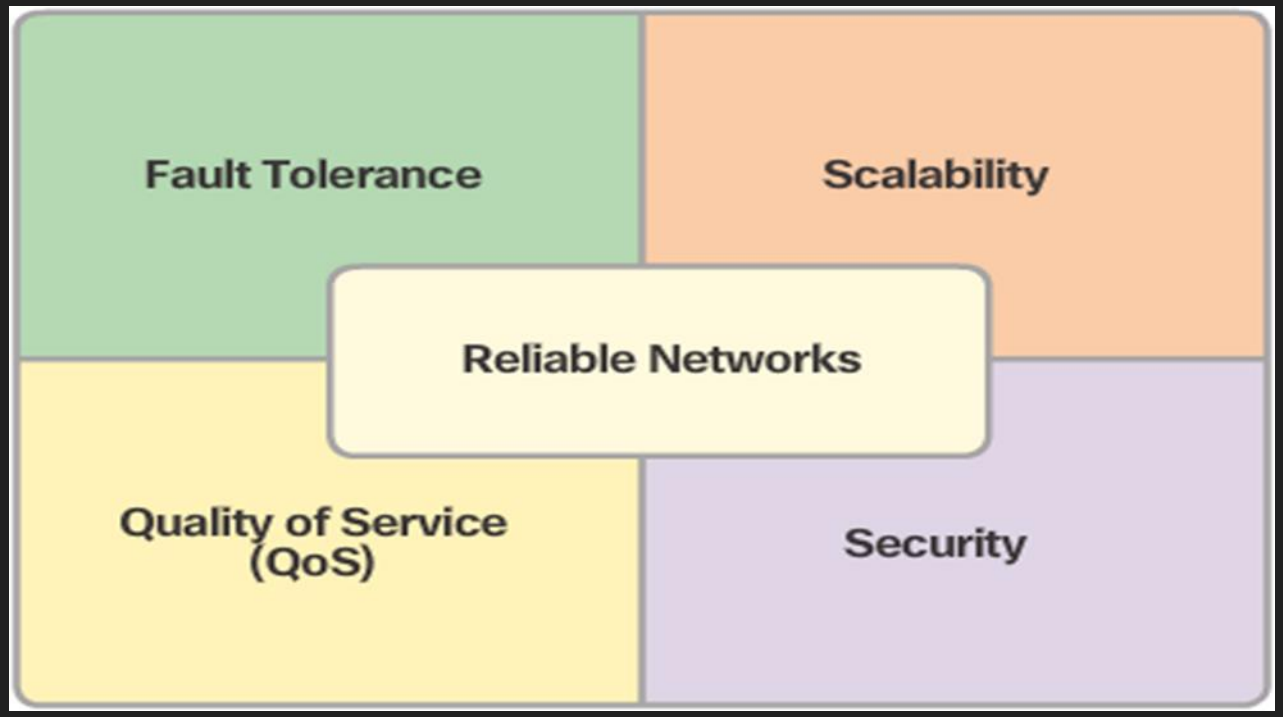


Data Communication & Process





Network Architecture



- Latency (jumlah waktu)
- Delay (waktu tunda)
- Jitter (variasi delay)
- Troughput (bandwidth actual)

2. Introduction of Network Programming

- Network programming adalah cabang ilmu komputer yang memungkinkan dua atau lebih sistem komputer untuk berkomunikasi dan berbagi sumber daya melalui jaringan komputer.
- Melalui network programming, berbagai aplikasi dan layanan seperti email, media sosial, perbankan online, dan e-commerce dapat diakses dan digunakan oleh pengguna di seluruh dunia.
- Ketika kita menulis sebuah program/aplikasi yang berkomunikasi melalui jaringan, program yang kita buat itu berada pada layer Application

Library

Beberapa lib populer yang digunakan untuk pemrograman jaringan dengan Python :

Socket : Pustaka standar Python untuk pemrograman soket. Dapat digunakan untuk membuat koneksi TCP/UDP, client-server applications, dll

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("0.0.0.0", 8080))
s.listen(1)
conn, addr = s.accept()
```

Requests: pustaka untuk membuat HTTP requests

```
import requests
response = requests.get('https://www.example.com')
print(response.text)
```

Library (2)

Scapy: Pustaka untuk manipulasi paket jaringan. Berguna untuk task-task seperti sniffing, packet injection, dll.

```
from scapy.all import *
packets = sniff(count=10)
print(packets)
```

Paramiko:Untuk SSHv2 protocol, yang memungkinkan kita untuk melakukan koneksi SSH ke server lain dan menjalankan perintah

```
import paramiko
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect('example.com', username='user', password='pass')
stdin, stdout, stderr = ssh.exec_command('ls')
print(stdout.read())
```

socket ( )

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("0.0.0.0", 8080))
s.listen(1)
conn, addr = s.accept()
```

Fungsi socket() digunakan untuk membuat objek socket, yang merupakan *endpoint* untuk mengirim atau menerima data di dalam sebuah jaringan komputer. Pada dasarnya, socket memberikan cara bagi proses di dua mesin yang berbeda (atau bahkan pada mesin yang sama) untuk berkomunikasi satu sama lain menggunakan protokol jaringan seperti TCP atau UDP.

Untuk aplikasi server, dapat digunakan untuk membuat socket, mengikatnya ke alamat dan port tertentu dengan bind(), lalu mulai mendengarkan koneksi masuk dengan listen().

Untuk aplikasi klien, Anda akan membuat socket dan kemudian menghubungkannya ke server dengan connect().

Parameter

Fungsi socket() dapat menerima beberapa parameter, tetapi dua yang paling umum adalah:

- family: Ini menentukan domain alamat dari socket. Beberapa contoh termasuk:
  - socket.AF_INET: Untuk alamat IPv4.
  - socket.AF_INET6: Untuk alamat IPv6.
- type: Ini menentukan tipe socket. Beberapa contoh termasuk:
  - socket.SOCK_STREAM: Untuk socket TCP.
  - socket.SOCK_DGRAM: Untuk socket UDP.

Fungsi ini mengembalikan objek socket yang dapat digunakan untuk berinteraksi dengan sistem jaringan lain. Dengan objek socket ini, kita dapat melakukan operasi seperti bind(), listen(), accept(), connect(), send(), dan recv().

bind( )

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("0.0.0.0", 8080))
s.listen(1)
conn, addr = s.accept()
```

Fungsi bind() digunakan untuk mengikat socket ke alamat IP dan port tertentu pada komputer lokal. Proses pengikatan ini memungkinkan socket untuk memiliki identitas tetap di jaringan sehingga sumber lain dapat berkomunikasi dengannya.

Fungsi bind() biasanya memerlukan satu parameter utama, yaitu sebuah *tuple* yang berisi alamat IP dan port. Misalnya: ('0.0.0.0', 8080).

- '0.0.0.0': Alamat IP ini memungkinkan socket mendengarkan semua interface jaringan pada komputer. Juga dapat mengikat socket ke alamat IP spesifik jika Anda memiliki beberapa interface jaringan dan hanya ingin mendengarkan pada salah satunya.
- 8080: Ini adalah port di mana socket mendengarkan. Juga bisa memilih port lain sesuai kebutuhan, tetapi perlu diperhatikan bahwa port di bawah 1024 biasanya dikenal sebagai "well-known ports" dan mungkin memerlukan hak istimewa untuk mengikatnya

Fungsi bind() biasanya digunakan pada sisi server dari aplikasi jaringan. Sebelum server dapat mulai mendengarkan koneksi masuk, ia perlu mengikat socket ke alamat dan port tertentu sehingga klien tahu di mana harus menghubungi.

Listen( )

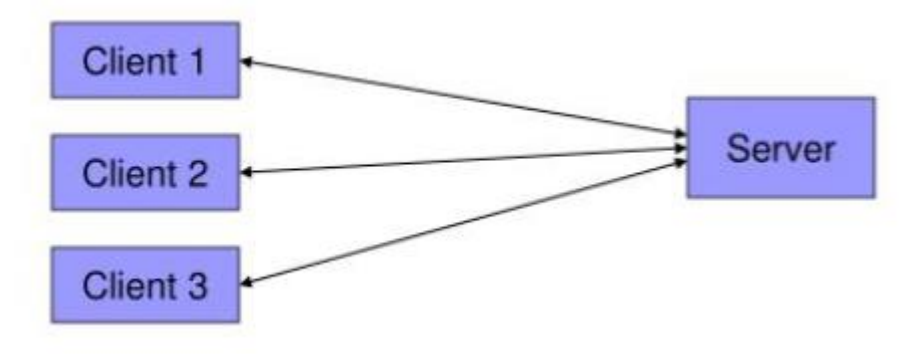
```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("0.0.0.0", 8080))
s.listen(1)
conn, addr = s.accept()
```

Fungsi listen() digunakan pada objek socket (biasanya pada sisi server) untuk memungkinkannya mulai menerima koneksi masuk dari klien. Setelah sebuah socket telah diikat ke alamat tertentu menggunakan fungsi bind(), Anda dapat memanggil listen() untuk membuat socket tersebut beroperasi dalam mode mendengarkan (listening mode).

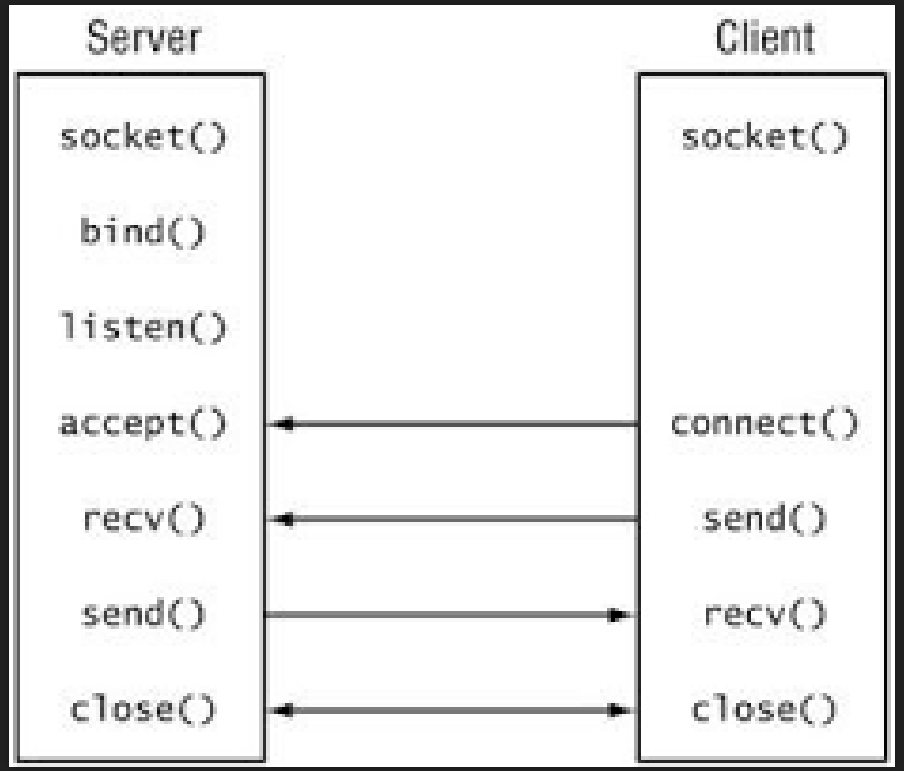
Fungsi listen() biasanya menerima satu parameter opsional yaitu 'backlog' . Ini menentukan jumlah koneksi yang tidak aktif yang diperbolehkan sampai sistem operasi mulai menolak koneksi masuk yang baru. Dalam kata lain, ini adalah panjang dari antrian koneksi yang menunggu untuk di-accept() oleh server (jumlah maksimal koneksi yang bisa menunggu di antrian sebelum di-accept)

- Setelah memanggil listen() pada socket, socket tersebut siap untuk menerima koneksi masuk.
- Biasanya akan menggunakan fungsi accept() setelah listen() untuk menerima koneksi masuk tersebut.

Client-Server Model



- One side of communication is client, and the other side is server
- Server waits for a client request to arrive
- Server processes the client request and sends the response back to the client
- Iterative or concurrent



Aplikasi Echo Server

Echo_server

```python
import socket

def start_echo_server(host, port):
    # Membuat socket dengan protokol TCP
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(5)  # Mendengarkan maksimal 5 koneksi dalam antrian
    print(f"Echo server started on {host}:{port}")

    while True:
        client_socket, client_address = server_socket.accept()
        print(f"Accepted connection from {client_address}")

        data = client_socket.recv(1024)
        while data:
            print(f"Received: {data.decode('utf-8')}")
            client_socket.send(data)  # Echo/mengirim kembali data ke klien
            data = client_socket.recv(1024)

        client_socket.close()
        print(f"Connection from {client_address} closed")

if __name__ == "__main__":
    start_echo_server("0.0.0.0", 8080)
```

A. Inisialisasi Socket

AF_INET = IPv4
SOCK_STREAM = TCP

B. Binding & Listening

0.0.0.0 = semua interface di mesin lokal
5 = jumlah maksimal koneksi yang bisa menunggu di antrian sebelum di-accept

C. Menerima Koneksi

Dalam loop tak terbatas (while True), server selalu siap menerima koneksi dengan perintah **client_socket, client_address = server_socket.accept().**

Fungsi **accept()** akan menghentikan eksekusi program sampai ada koneksi yang masuk. Setelah ada koneksi, fungsi ini mengembalikan dua nilai: socket klien yang terhubung dan alamat klien.

D. Komunikasi dengan Klien

1024 = jumlah byte maksimum yang dapat diterima dalam satu kali penerimaan.

E. Menutup Koneksi

Setelah selesai berkomunikasi dengan klien, koneksi dengan klien ditutup dengan perintah client_socket.close()

Aplikasi Echo Client

Echo_client

```python
import socket

def start_echo_client(host, port, message):
    # Membuat socket dengan protokol TCP
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((host, port))

    # Mengirim pesan ke server
    client_socket.send(message.encode('utf-8'))

    # Menerima respons dari server
    data = client_socket.recv(1024)
    print(f"Received from server: {data.decode('utf-8')}")

    client_socket.close()

if __name__ == "__main__":
    start_echo_client("127.0.0.1", 8080, "Hello, Echo Server!")
```

- **Inisialisasi Socket:**

  Seperti server, klien juga membuat socket TCP/IP dengan socket.socket(socket.AF_INET, socket.SOCK_STREAM).

- **Menghubungkan ke Server:**

  Dengan client_socket.connect((host, port)), klien berusaha menghubungkan diri ke server yang berjalan pada alamat IP dan port tertentu.

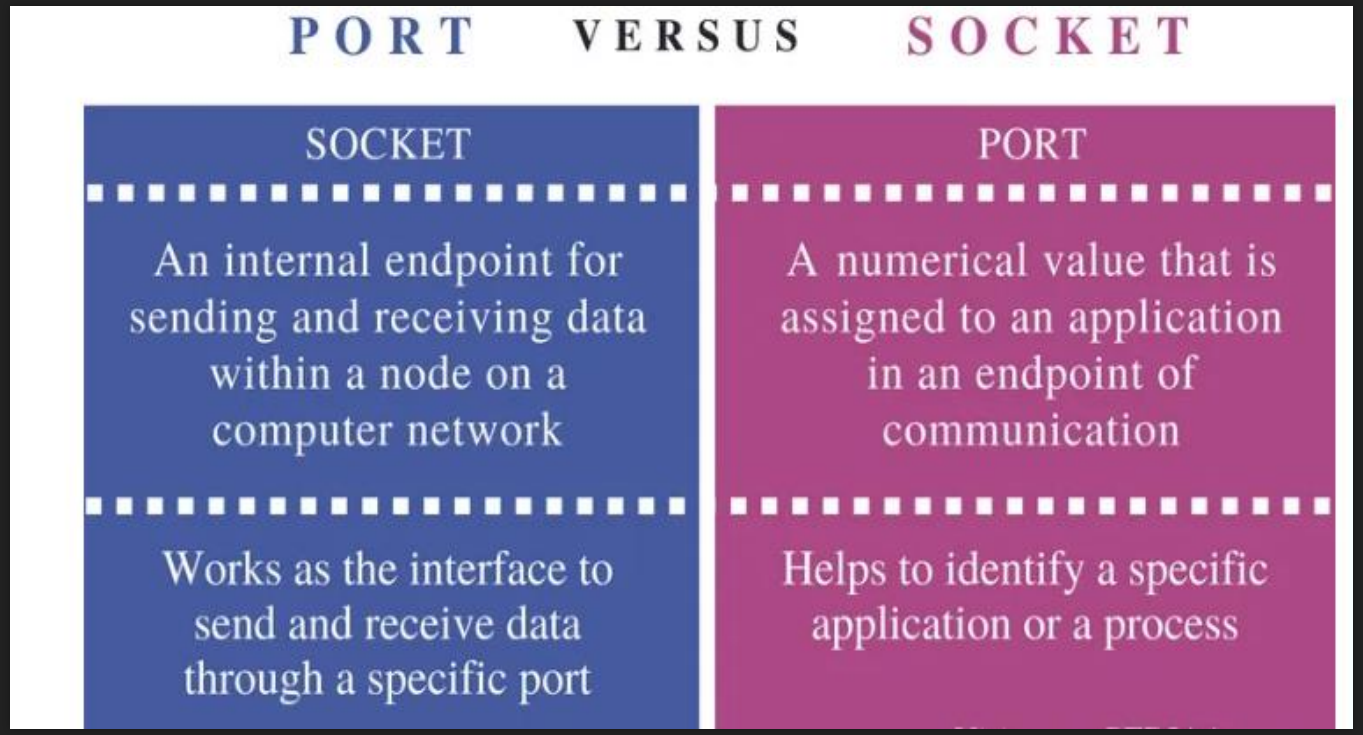
- **Mengirim dan Menerima Data:**

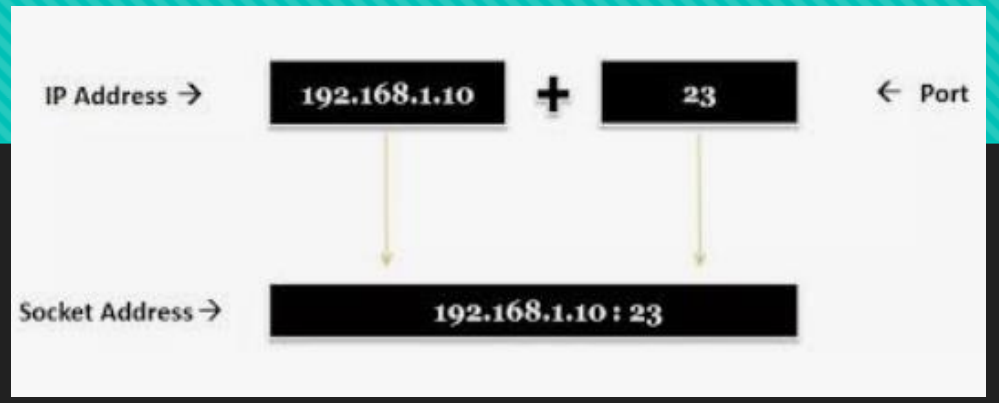
  Setelah terhubung, klien mengirim pesan ke server dengan perintah client_socket.send(message.encode('utf-8')). Kemudian, klien menunggu respons dari server dengan perintah data = client_socket.recv(1024) dan mencetak respons tersebut.

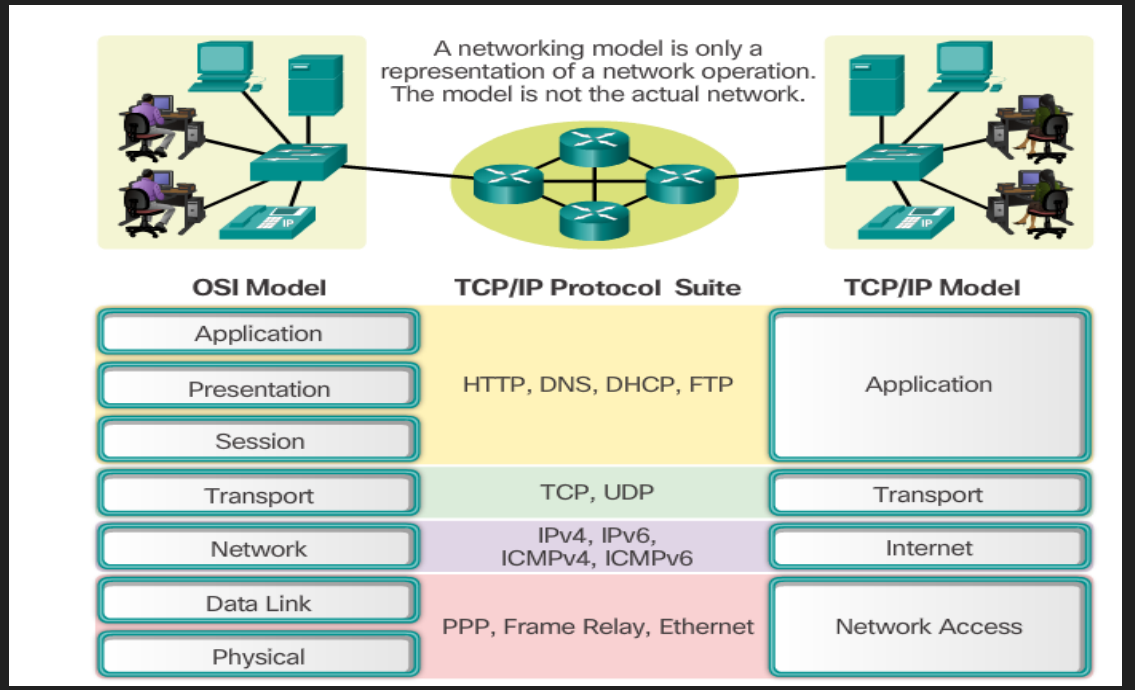
- **Menutup Koneksi:**

  Setelah selesai berkomunikasi dengan server, klien menutup koneksi dengan perintah client_socket.close()

3. Port, Socket & Protocol







Diskusi Kelompok

Tujuan :

a) Memahami definisi dan konsep dasar network programming.
b) Mengetahui pentingnya network programming dalam pengembangan aplikasi dan layanan berbasis jaringan.
c) Memahami konsep socket dan protocol dalam network programming.
d) Mengetahui perbedaan antara socket dan protocol.
e) Mengetahui kriteria dalam memilih bahasa pemrograman untuk network programming.
f) Memahami kelebihan dan kekurangan beberapa bahasa pemrograman dalam konteks network programming.

Kegiatan & Pembahasan Kelompok :

1. Diskusikan dalam kelompok mengenai aplikasi dan layanan apa saja yang memanfaatkan network programming.
2. Identifikasi dan jelaskan manfaat dari network programming pada aplikasi dan layanan tersebut.
3. Buatlah tabel yang membandingkan socket dan protocol berdasarkan karakteristik dan fungsinya.
4. Diskusikan dalam kelompok mengenai peran socket dan protocol dalam sebuah aplikasi jaringan.
5. Buatlah daftar kriteria yang perlu dipertimbangkan saat memilih bahasa pemrograman untuk network programming.
6. Bandingkan kelebihan dan kekurangan dari setidaknya tiga bahasa pemrograman yang populer digunakan dalam network programming
7. Bagaimana pengaruh network programming terhadap perkembangan teknologi informasi saat ini?
8. Mengapa pemahaman tentang socket dan protocol penting dalam network programming?

Terima Kasih