> Processes: A program in execution

Consists of: 1) Program code (possibly shared)

2) A set of data

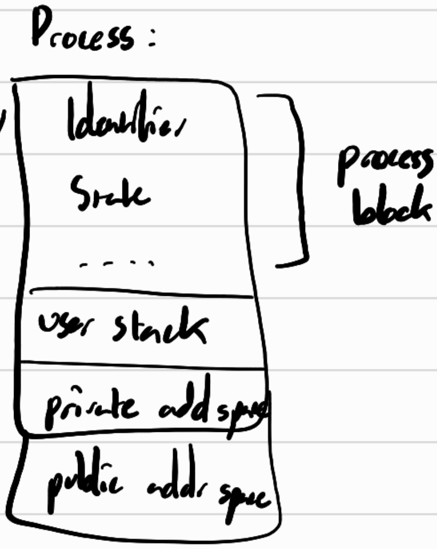
3) Attributes describing state of process

↳ - Identifier
- State
- Priority
- Program Counter
- Memory Pointer
- Context Data
- I/O status information
- Accounting Information

(Usually kept in PCB)
process control block

> Process Control Block (PCB)

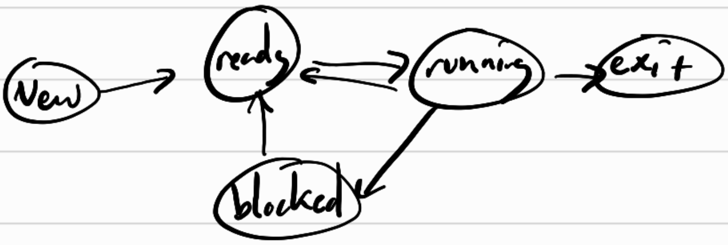
↳ - Contains process elements
- Created and managed by OS
- Support for multiple processes
- PCB usually unique, not interchangable b/w OS



> States

Different Models:

- 5 state model:



> Networks TCP / UDP

- Most widely used protocols for internet network traffic.
- Previously expensive + slow. Now adequate for user broadband, underperform for host-to-host.

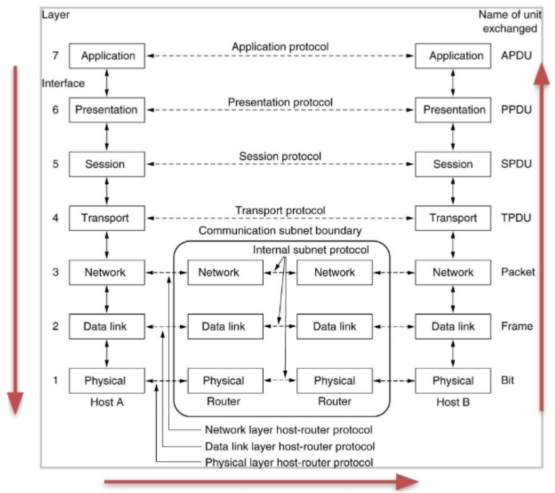
> Open Systems Interconnection (OSI) Model

↳ Provide set of design standards for manufacturers to communicate w/ each other

↳ Conceptual framework dividing network communications into 7 layers.

- Layers each have well-defined function + different abstraction
- As message flows down stack → encapsulation added
  ↳ add data to packet header
- Message flows up → encapsulation removed

travel down protocol stack

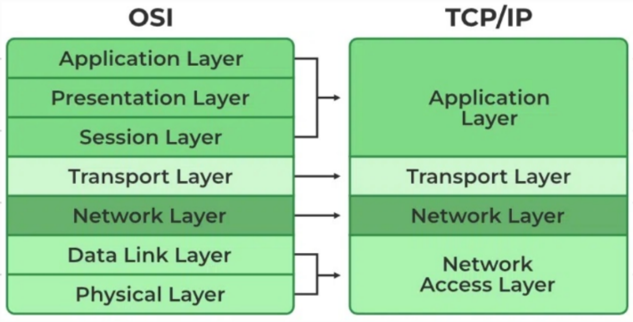


Message Travelling Across Network

go up protocol stack

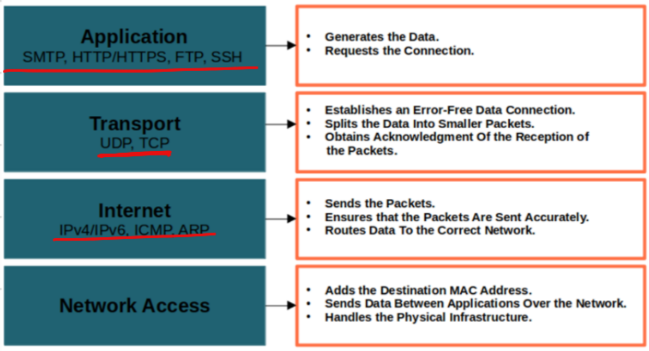
↳ 3 key concepts
1) Services
2) Interfaces
3) Protocols

> TCP/IP Model : Set of protocols governing flow of network info



Internet

link



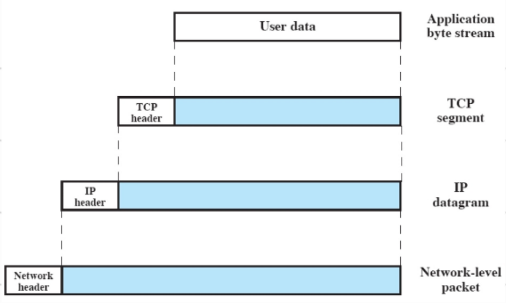
Application
SMTP, HTTP/HTTPS, FTP, SSH
- Generates the Data.
- Requests the Connection.

Transport
UDP, TCP
- Establishes an Error-Free Data Connection.
- Splits the Data Into Smaller Packets.
- Obtains Acknowledgment Of the Reception of the Packets.

Internet
IPv4/IPv6, ICMP, ARP
- Sends the Packets.
- Ensures that the Packets Are Sent Accurately.
- Routes Data To the Correct Network.

Network Access
- Adds the Destination MAC Address.
- Sends Data Between Applications Over the Network.
- Handles the Physical Infrastructure.

↳ TCP/IP loosely follows OSI
- OSI more finely grained



TCP-IP encapsulation

> TCP/UDP + IP Stack

- TCP: - keeps track of segments transmitted. Assigns numbers.
  - Reliable Delivery in Sequence (FIFO/pipe)
  - Not OS dependent, not suit LAN
  - Slow and takes more bandwidth
    - ↳ Does not load page if something missing
  - ex) www., emails, ssh, stream videos, file transfer protocol

- UDP: - Simple message requests, less control over flow/error
  - Used in realtime applications which cannot tolerate uneven delays b/w messages recieved.
  - Does not require connection
  - Unreliable (no mechanism to track sequence of data)
  - Can't tell if data transfer is successful
  - ex) Online games, live-streaming, DNS queries

> Middleware: - Software b/w OS and application, enable communication
  - Provides API interface for user program running in a process

> Stream oriented IPC performance
  - Latency b/w messages
  - Throughput: Bandwidth/capacity per unit time.

>Work

1) 200 + 8 = 208 bytes

+

520

+

820 × 3 = 4096 bytes

+

908

2) 15 msg/s × 16 processes × 3 kB = 720 kB/s

A1

1) 32818 bytes 40928

2) 8 process × 7 messages/s × 5 kB Ⓔ

3) 8820

4) 32 × 31 × 2 = 1984 kB/s