SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELIGENCE & MACHINE LEARNING

23AMT302- COMPUTER NETWORK AND SECURITY

UNIT 1 – Introduction and Application Layer

Prepared by
A.catherine
AP/AIML

Why Sockets? Networking Fundamentals

IP Addresses

Unique identifiers for network devices.

IPv4 is 32-bit, IPv6 is 128-bit.

Port Numbers

Logical identifiers for applications.

Range from 0 to 65535.

OSI Model

Sockets operate at the Transport Layer (Layer 4).

Think of an IP address as a building, and a port number as an apartment.

Socket Types: TCP (Stream) vs. UDP (Datagram)

TCP Sockets

- Connection-oriented.
- Reliable, guarantees delivery.
- Stream-based data flow.
- Used by HTTP, HTTPS, FTP.

UDP Sockets

- Connectionless, 'send and forget'.
- Unreliable, no delivery guarantee.
- Datagram-based packets.
- Used by DNS, online gaming, streaming.

The Client-Server Model

1

Server

Listens for incoming connections.

Awaits requests on a specific port.

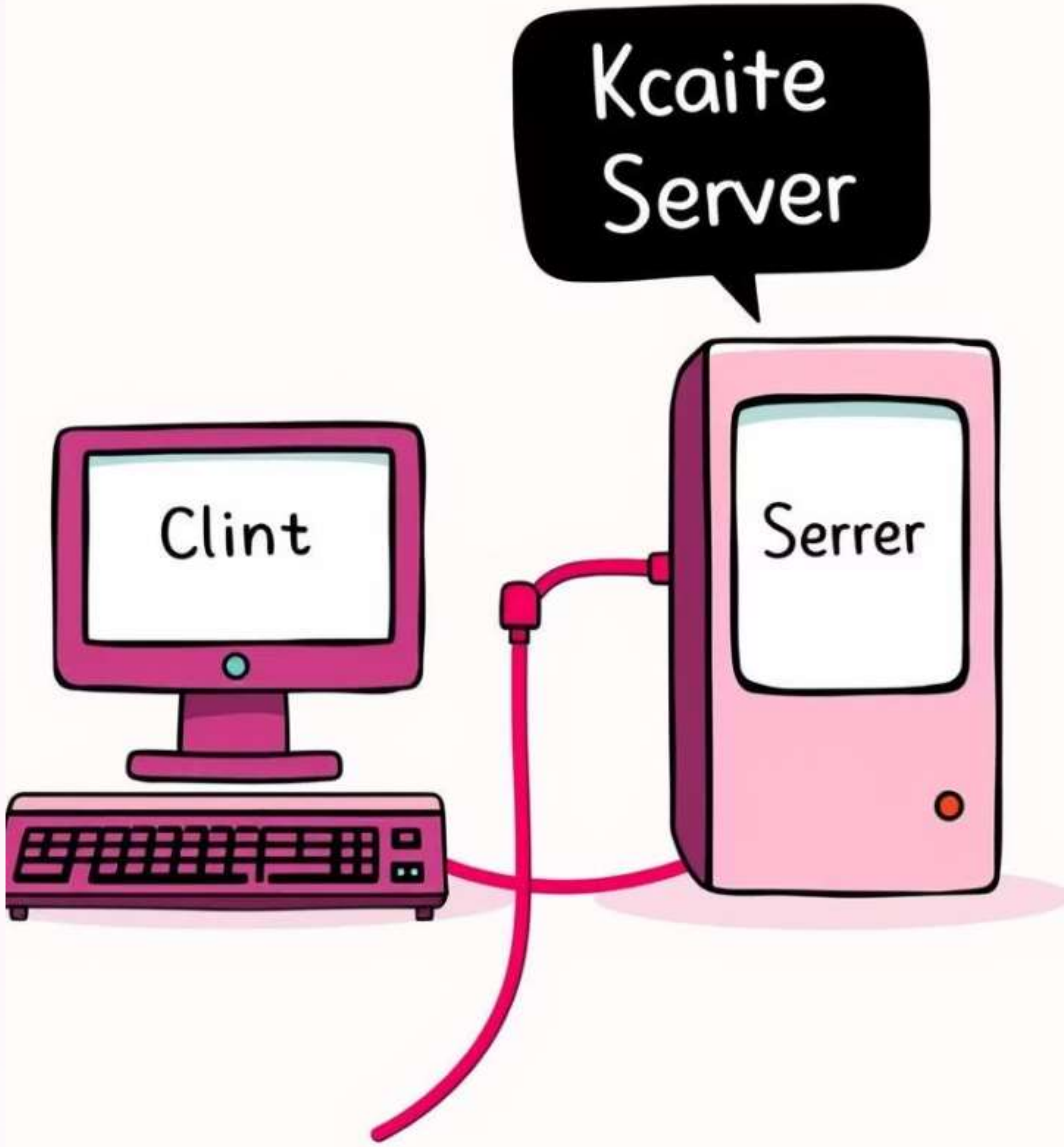
2

Client

Initiates connection to a server.

Sends requests and receives responses.

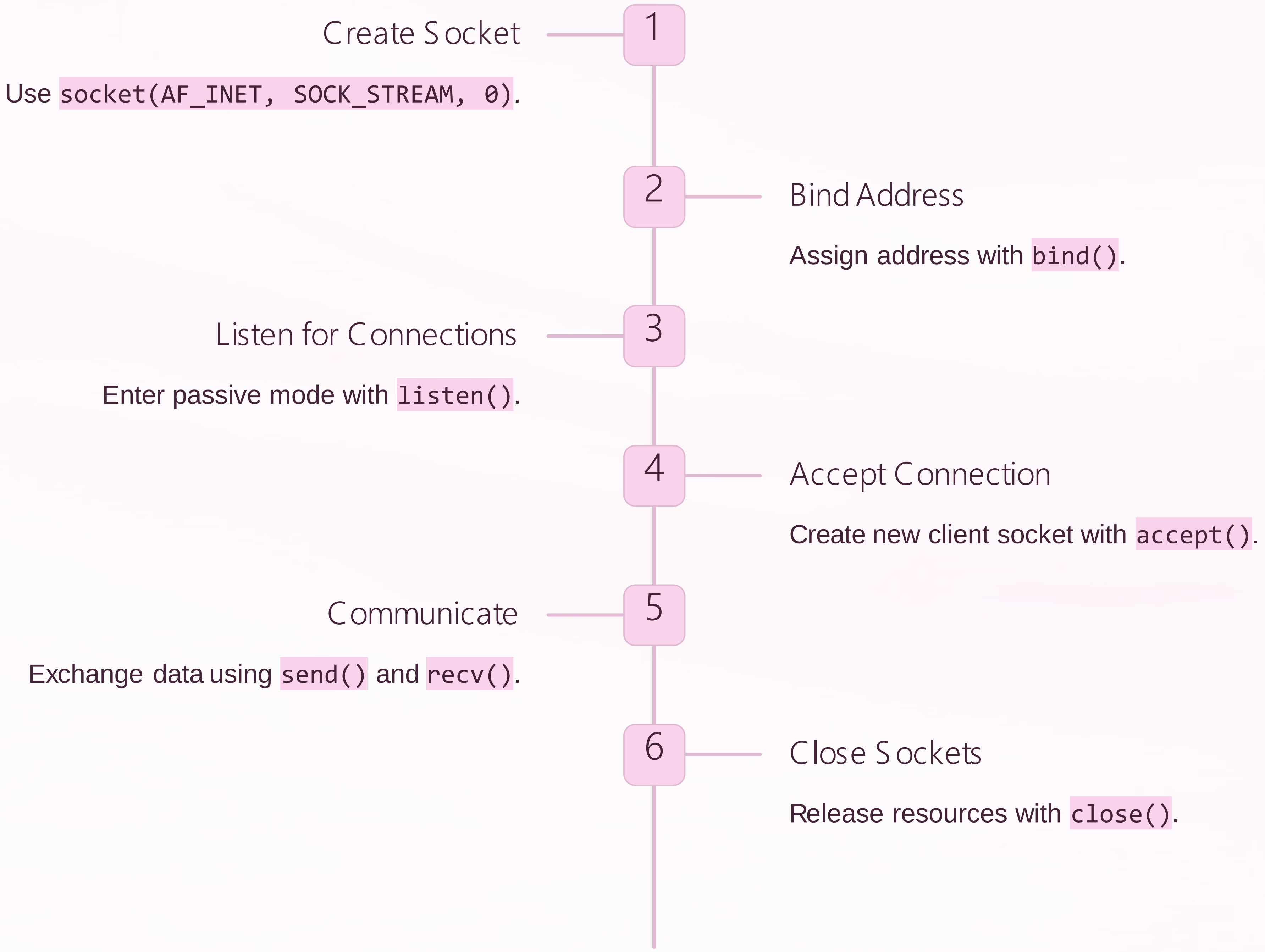
The client sends requests, and the server processes and responds.



Key Socket API Functions (POSIX Standard)

- **socket():** Creates a new socket descriptor.
- **bind():** Assigns an IP address and port.
- **listen():** Puts a server socket into listening mode.
- **accept():** Accepts a new client connection.
- **connect():** Initiates a remote connection.
- **send() / recv():** Transmits and receives data.
- **close():** Terminates connection and releases resources.

TCP Socket Workflow: Server Side



1 Create Socket

Use `socket(AF_INET, SOCK_STREAM, 0)`.

2 Bind Address

Assign address with `bind()`.

3 Listen for Connections

Enter passive mode with `listen()`.

4 Accept Connection

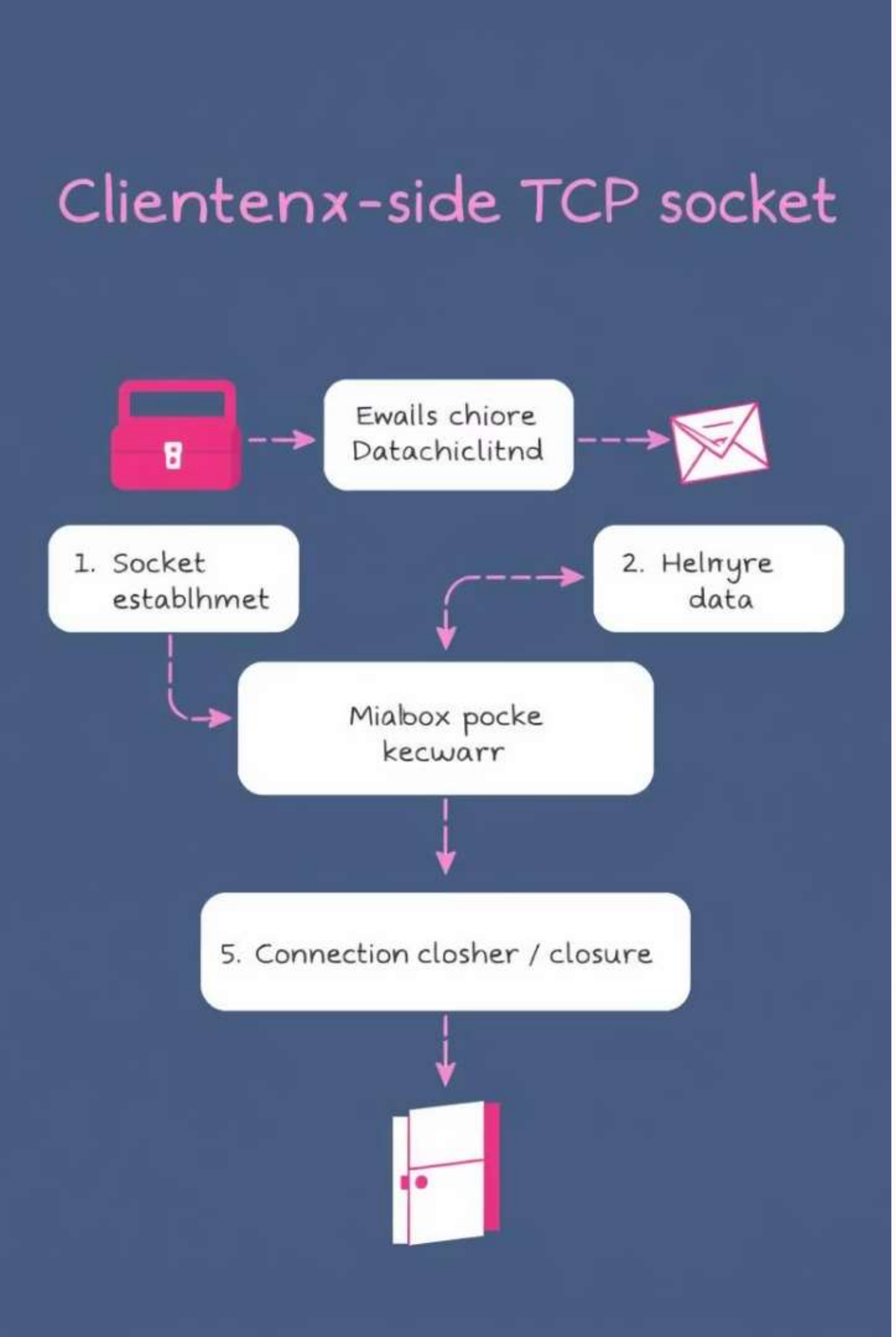
Create new client socket with `accept()`.

5 Communicate

Exchange data using `send()` and `recv()`.

6 Close Sockets

Release resources with `close()`.


Clientenx-side TCP socket
Ewails chiore Datachiclitnd
1. Socket establhmet
2. Helnyre data
Miabox pocke kecwarr
5. Connection closher / closure

TCP Socket Workflow: Client Side

Create Socket

Call `socket(AF_INET, SOCK_STREAM, 0)`.

Connect to Server

Initiate connection using `connect()`.

Communicate

Send and receive data with `send()` and `recv()`.

Close Socket

Terminate connection and free resources with `close()`.

UDP Socket Workflow

Sender Workflow

1. Create Socket: `socket(AF_INET, SOCK_DGRAM, 0)`.
2. Send Data: `sendto()` to destination.
3. Close: `close()` the socket.

Receiver Workflow

1. Create Socket: `socket(AF_INET, SOCK_DGRAM, 0)`.
2. Bind Address: `bind()` to local address.
3. Receive Data: `recvfrom()` from sender.
4. Close: `close()` the socket.

UDP is connectionless, so no explicit connection is established.

Real-World Applications of Sockets

Web Browsing

Uses TCP sockets (Port 80/443) for web servers.

Email

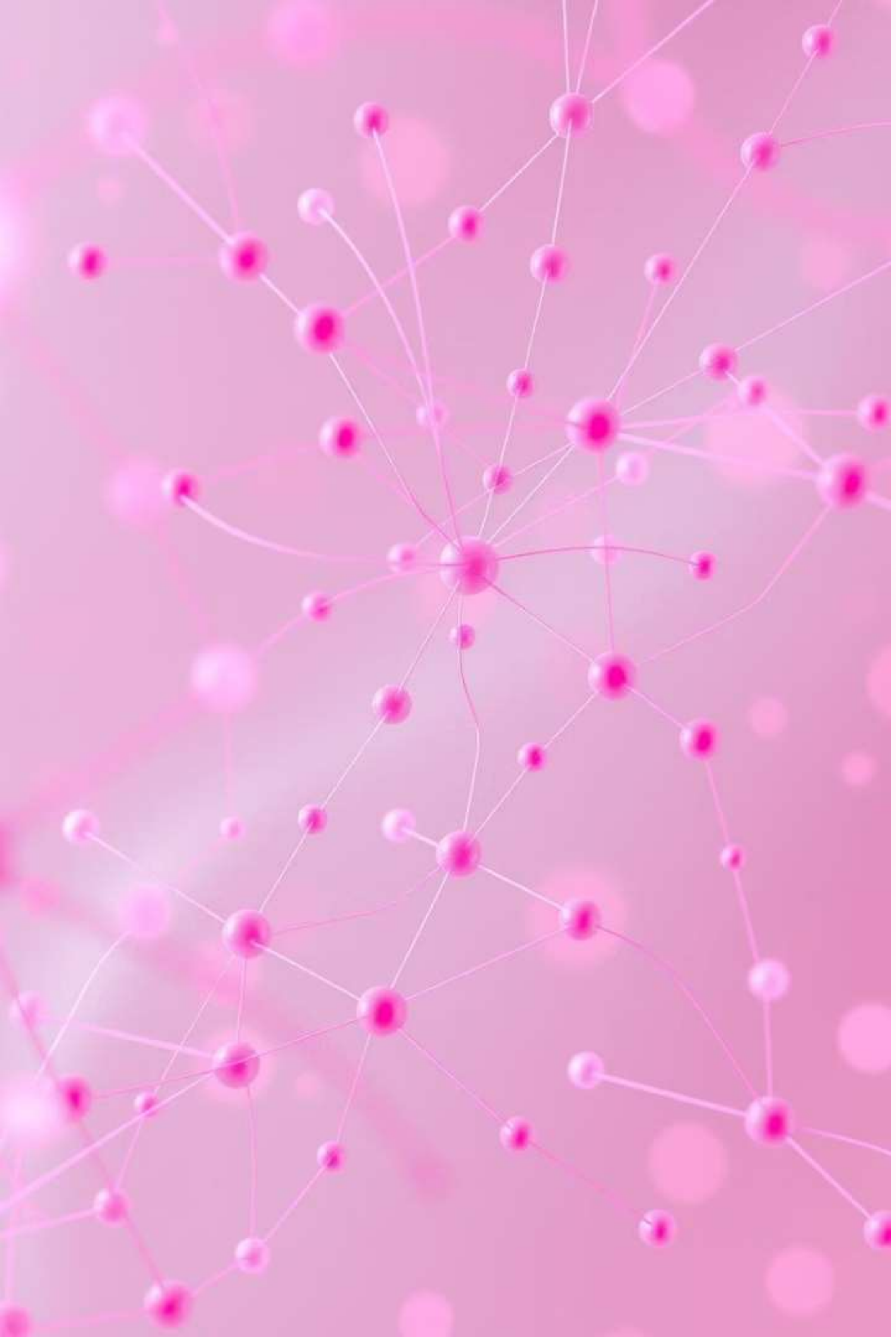
TCP for sending (SMTP: 25) and receiving (POP3: 110, IMAP: 143).

Online Gaming

Often uses UDP for fast updates, TCP for critical data.

DNS

Primarily uses UDP (Port 53) for efficient hostname lookups.



Conclusion

Fundamental Blocks

Sockets are the core of network communication.

Protocol Choice

TCP for reliability, UDP for speed.

Versatility

Essential for all networked applications.

Future Evolution

Sockets continue to adapt with new protocols.