# Java Networking

Mastering Sockets, TCP/IP, and Event Handling in Java

# Core Concepts

Understanding the foundations of network communication.

# TCP vs. UDP: Choosing the Protocol

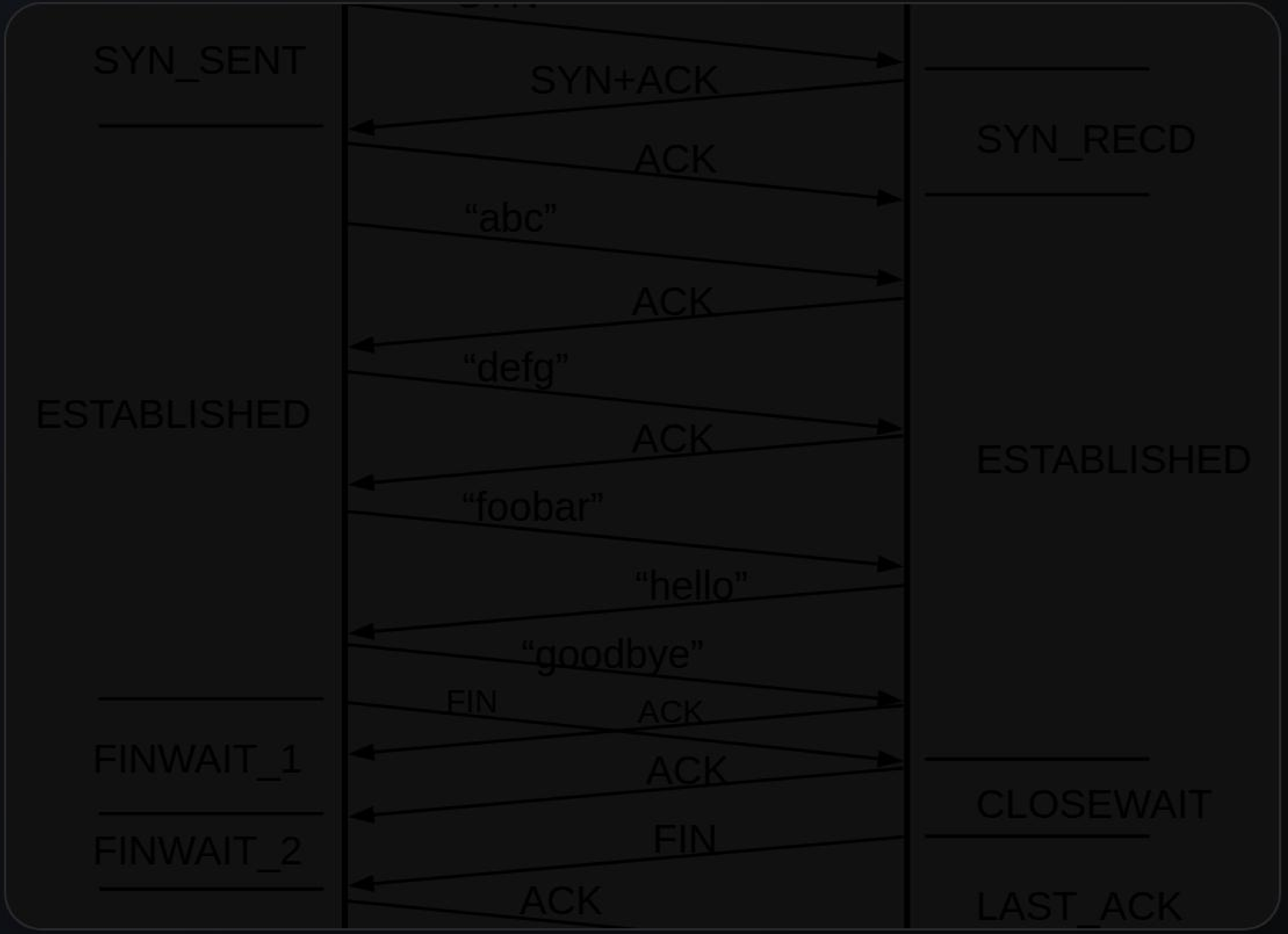
| Feature | TCP (ServerSocket) | UDP (DatagramSocket) |
|---|---|---|
| **Connection** | Connection-Oriented (Handshake) | Connectionless (Fire & Forget) |
| **Reliability** | Guaranteed Delivery, Ordered | Unreliable, Unordered |
| **Speed** | Slower (Overhead) | Faster (Low Overhead) |
| **Use Case** | Web, Email, File Transfer | Streaming, Gaming, VoIP |

# The TCP Protocol

## Reliable Streams

TCP (Transmission Control Protocol) establishes a virtual "pipe" between client and server. It ensures that data arrives intact and in the correct order.

- ✓ **ServerSocket:** Listens for incoming connections.
- ✓ **Socket:** Represents the endpoint for communication.
- ✓ **Handshake:** SYN, SYN-ACK, ACK process.



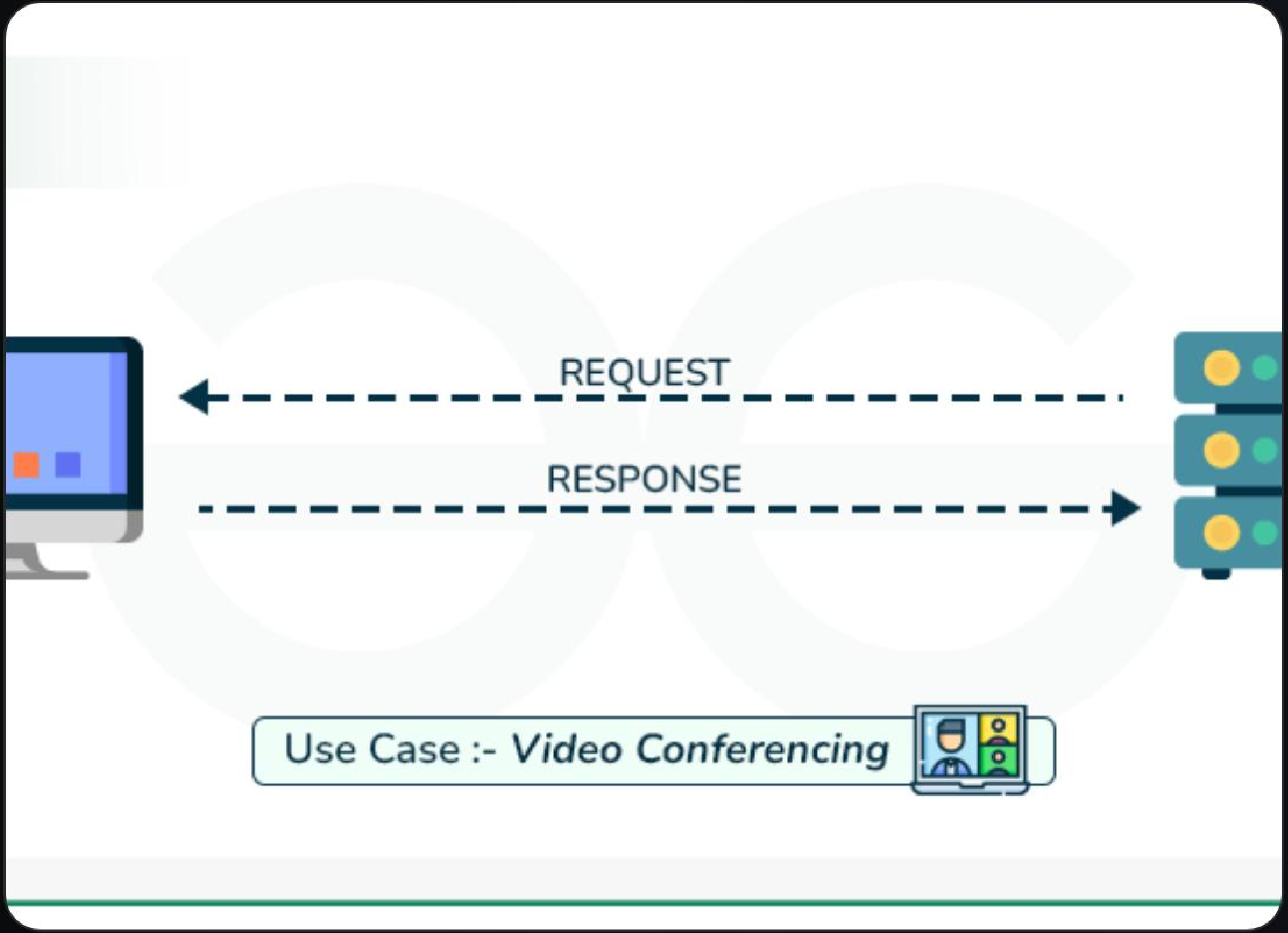
## Blocking I/O Model

The `accept()` method blocks the current thread until a connection is made. This often requires multithreading to handle multiple clients.

```
try (ServerSocket server = new ServerSocket(8080)) { // Blocks until client connects Socket client = server.accept(); // Inp
```

Once connected, `InputStream` and `OutputStream` are used for byte-level

# The UDP Protocol



## Fast & Stateless

UDP (User Datagram Protocol) sends independent packets (datagrams) without establishing a connection. It's efficient but risky.

- **DatagramSocket:** Used for both sending and receiving.
- **DatagramPacket:** The container for data and address.
- **No Guarantee:** Packets may be lost or arrive out of order.

# UDP Sender & Receiver

## Packet Handling

Unlike TCP streams, UDP requires you to manually package data into arrays of bytes. You must specify the destination IP and port for every packet sent.

```
// Receiver DatagramSocket socket = new DatagramSocket(9000); byte[] buf = new byte[256]; DatagramPacket packet = new DatagramPa
```

# Event Handling

Managing asynchronous data and connection states.

# Network Events

### Connection

Triggered when a client successfully connects (TCP) or a server starts listening. Handled via `accept()` return.

### Data Reception

The most common event. Triggered when bytes are available in the input stream or a packet arrives.

### Exceptions

Network timeouts, disconnects, or unreachable hosts. Must be caught to prevent server crashes.

### Write Completion

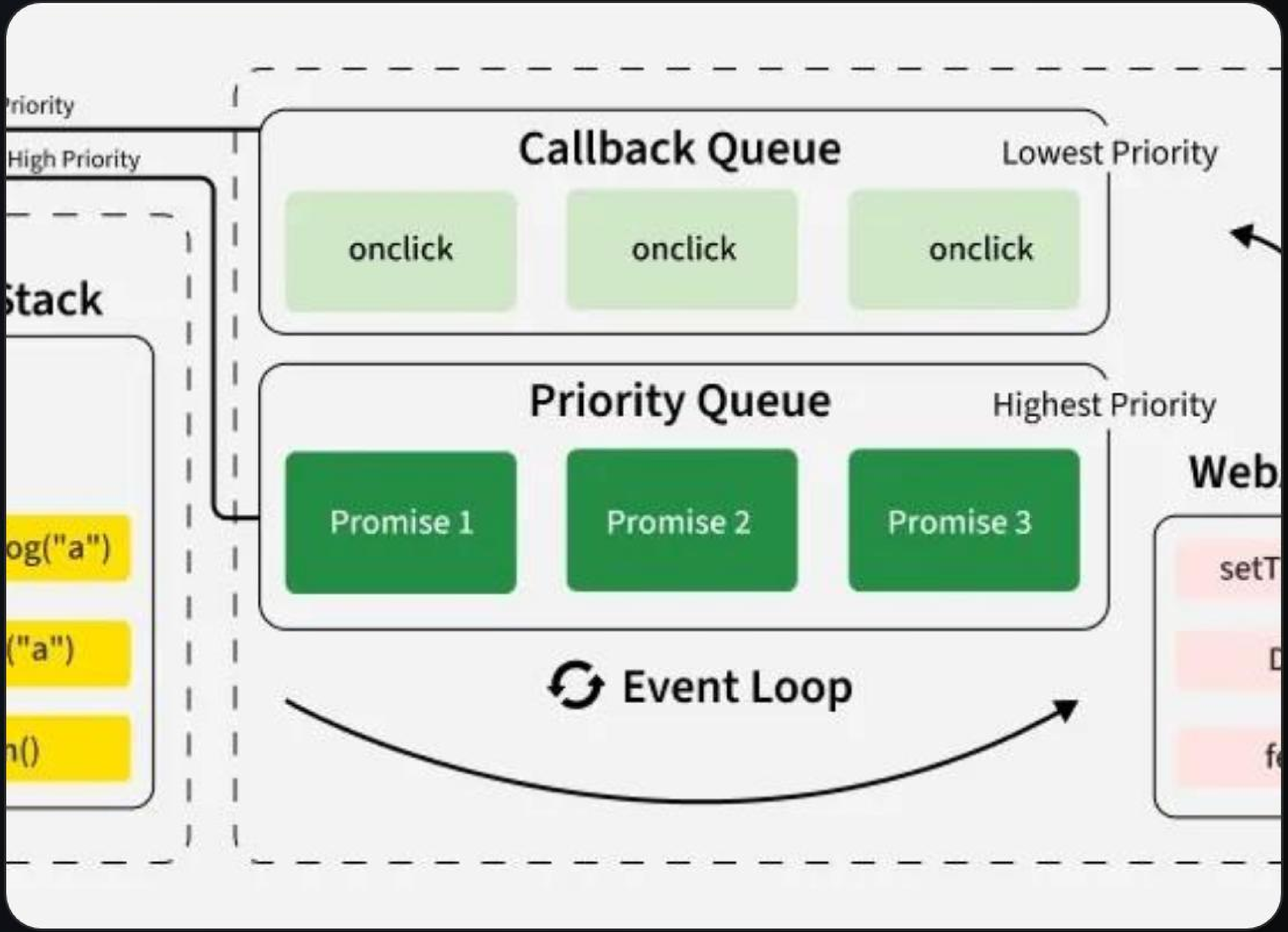
Ensuring data has effectively left the buffer. Critical in high-load non-blocking systems.

# The Event-Driven Model

## From Blocking to Async

In standard Java Networking, "events" are often handled by assigning a dedicated thread to each connection. The thread waits (blocks) for an event.

For cleaner architecture, we wrap this in an **Observer Pattern**. A listener interface defines methods like `onMessage()`, decoupling the network logic from the business logic.



# Implementing Callbacks

## The Listener Interface

Define a simple interface to abstract the low-level socket operations. The networking thread calls these methods when specific states occur.

```
public interface NetworkListener { void onConnect(Socket client); void onMessage(String message); void onError(Exception e); }
```

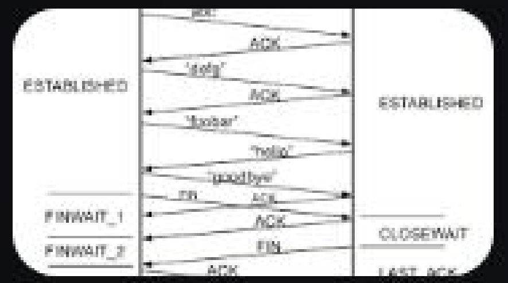
# Best Practices

- **Resource Management:** Always close sockets and streams using try-with-resources blocks to prevent memory leaks.
- **Threading:** Never handle network I/O on the main UI thread. Use `ExecutorService` for thread pooling.
- **Timeouts:** Set `setSoTimeout()` to prevent threads from blocking indefinitely if a peer vanishes.
- **Non-Blocking I/O:** For high-performance servers handling thousands of connections, consider `java.nio` (New I/O) over standard IO.

# Questions?

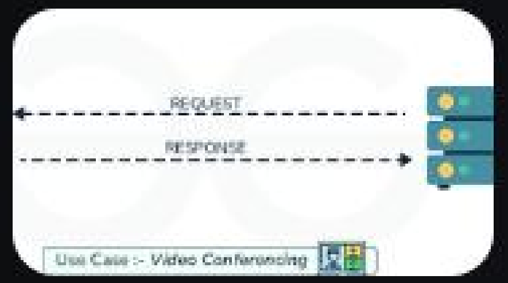
Thank you for your attention.

# Image Sources



https://intronetworks.cs.luc.edu/current/uhtml/_images/tcp_ladder_states.svg

Source: intronetworks.cs.luc.edu

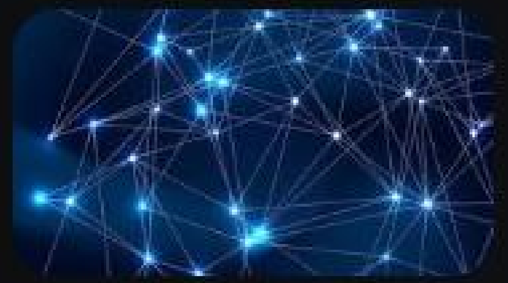


https://media.geeksforgeeks.org/wp-content/uploads/20240226104348/UDP-gif.gif

Source: www.geeksforgeeks.org

https://media.geeksforgeeks.org/wp-content/uploads/20250208123836185275/Event-Loop-in-JavaScript.jpg

Source: www.geeksforgeeks.org



https://img.freepik.com/premium-vector/abstract-digital-network-connection-structure-blue-background-artificial-intelligence-enginee_230610-1093.jpg

Source: www.freepik.com