# Chapter 3 Transport Layer

## 3.1 Transport layer services

- provide logical communication between app processes
- end system
  - send: breaks message into segments, pass to network layer
  - recv: reassemble segments from network layer, pass to application layer

### Transport layer vs. Netork layer

Network layer: logical communication between hosts
Transport layer: logical communication between processes

Transport layer protocols

- TCP ( reliable, in-order delivery )
  - congestion control
  - flow control
  - connection setup

- UDP ( unreliable, unordered delivery )
  - no-frills extension of “best-effort” IP

- services not available
  - delay guarantees
  - bandwidth guarantees

## 3.2 Multiplexing and Demultiplexing

- Multiplexing ( sender ): handle data for multiple sockets, add transport header for identification
- demultiplexing ( receiver ): use header info to deliver received segments to correct socket

### Demultiplexing

Host uses IP addresses & port numbers to direct segment to appropriate socket.

1. Connectionless, identify with ( UDP socket )

   - dest. IP, port

IP datagrams with same dest. port #, but different source IP addresses and/or source port numbers will be directed to same socket at dest.

2. Connection - oriented demux, identify with ( TCP socket )

   - src. IP, port
   - dest. IP, port

# 3.3 Connectionless transport: UDP

Usage: DNS, SNMP, streaming multi - media apps

- best effort, lost and out - of - order delivery may occur
- connectionless: no handshaking, segments handled independently

Reliable transfer over UDP:

- add reliability at application layer
- application-specific error recovery!

### UDP checksum

Goal: detect errors in transmitted segment.

1. Sender:
   - treat segment content as 16 - bit integer
   - checksum: addition of segment content
   - checksum value into UDP checksum field

2. Receiver:
   - compute checksum of received segment
   - check if computed checksum equals checksum field value