**Chapter One:**

**Packet Switching:**

**Circuit Switching:**

**Delays:**

**Dproc:**

**Dqueue:**

**Dtrans:**

**Dprop:**

**Chapter Two:**

**Internet protocol stack  
Application**

**Transport**

**Network**

**Link**

**Physical**

**Application Layer:**

**Client/server vs P2P**

**Socket**

**Addressing process**

**HTTP:**

**Client server model**

**Persistent vs non-persistent: response time**

**Response status codes**

**Caching: calculation**

**Conditional get**

**DNS: Distributed database in hierarchy**

**Application layer protocol**

**Local DNS name server (default name server): answer DNS query from host**

**DNS resolution: iterated vs recursive**

**DNS records: RR type**

**Chapter 3**

**Transport**

**TCP:**

**Reliable transfer**

**Flow control**

**Congestion control**

**Connection oriented**

**UDP:**

**Unreliable**

**Why bother? No connection establishment, simple, small header size, no congestion control: fast**

**Socket programming:**

**UDP: unreliable datagram**

**No connection between client and server**

**Lost/out of order**

**\*client server socket interaction in UDP**

**UDP segment format: source port, dest port, length, checksum, payload**

**Checksum: sender add 16 bits integers together, 1s complement, store in field**

**TCP: reliable, byte stream-oriented pipe**

**\*Socket programming: client contacts server etc.**

**\*client server socket interaction in TCP**

**Mux and demux: multiplex at sender & demultiplexing at receiver**

**RDT 1.0 – 3.0**

**Rdt3.0: stop and wait operation, equation**

**Pipeline protocols: increase sender utilization**

**GBN: up to N unpacked packets in pipeline, cumulative ack (don’t allow gap), sender timer for oldest unpacked packet, send all unacked upon timeout**

**Selective repeat: up to N unacked packets, individual ack, sender timer for each unacked packet, retransmit only that unacked packet upon timeout**

**TCP RTT, timeout: EstimatedRTT = (1- a)\*EstimatedRTT + a\*SampleRTT (a typicall = 0.125)**: influence of past sample decreases exponentially fast | timeout interval: EstimatedRTT plus safety margin, large variation in EstimatedRTT -> larger safety margin | **DevRTT = (1-b)\*DevRTT + b\*|SampleRTT-EstimatedRTT| b typicall = 0.25** | **TimeoutInterval = EstimatedRTT + 4\*DevRTT**

**TCP fast retransmit**: (timeout often long, detect loss through duplicate ACKs) if sender receives 3 ACKs for same data, resend unacked segment with smallest seq # right the way

**Principles of congestion control:** too many sources sending too much data too fast for network to handle -> lost package, long delay. Cost of congestion: retransmission, wasted transmit when package dropped

**TCP congestion control:** sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs. Additive increase (congestion avodiance): increase cwnd by 1 MSS every RTT until loss detected. Multiplicative decrease: cut cwnd in half after loss. | sender limits transmission: cwndLastByteSent- LastByteAcked < cwnd | TCP sending rate: send cwnd bytes, wait RTT for ACK, then inc or dec: **rate = cwnd/RTT (byte/sec)**