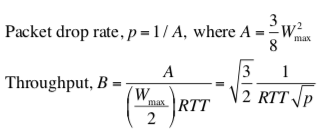
**Transport Layer (Congestion Control, TCP Fairness)**

* Connection Management
  + TCP 3-way handshake (agree on establish connection & connection parameters)
    - client → server, send TCP SYN message (init seq #, no data)
    - server → client, send TCP SYNACK message (init seq #), allocates buffers
    - client → server, received SYNACK, reply ACK (may contain data)
  + Why 3-way handshake?
    - The sequence number starts at the ISN, which is a randomly chosen value. And since TCP is a bi-directional communication, both parties can "speak", and therefore both must randomly generate an ISN as their starting Sequence Number. Which in turn means, both parties need to notify the other party of their starting ISN.
  + TCP: closing a connection (can be simultaneous)
    - clientSocket.close()
    - client → server: send TCP FINbit = 1
    - server → client: received FIN, close wait, reply ACK and FINbit = 1
    - client → server: received FIN, reply ACK
      * Timed wait: can retransmit ACK if last ACK lost
    - server received ACK, close
  + TCP SYN Attack (SYN flooding): solutions
    - Increase size of connection queue
    - Decrease timeout wait for the 3-way handshake
    - Firewalls: list of known bad source IP addresses
    - TCP SYN Cookies
* Principles of Congestion
  + Manifestations 表现
    - Lost packets (buffer overflow at routers)
    - Long delays (queueing in router buffers)
  + Without congestion control
    - Increases delivery latency, loss rate, retransmissions, congestion, cycle continues…
    - Wastes capacity of traffic
  + Cost of Congestion
    - Knee -- point after which
      * Throughput increases slowly
      * Delay increases fast
    - Cliff -- point after which
      * Throughput starts to drop to zero (congestion collapse)
      * Delay approaches infinity
* Congestion control
  + Congestion control vs Flow control
    - Flow control is a local, congestion control is global.
  + Approaches
    - End-end congestion control
      * No explicit feedback from network
      * Congestion inferred from end-system observed loss, delay
      * Approach taken by TCP
    - Network-assisted congestion control
      * Routers provide feedback to end systems
        + Single bit indicating congestion (SNA, DECbit, TCP/IP, ECN, ATM)
        + Explicit rate for sender to send at
* TCP’s Approach
  + Vary window size to control sending rate



* + Windows
    - CWND (Congestion window) consider it as MSS units, but as Bytes in reality.
    - RWND (Flow control window)
    - Sender-side window = minimum{CWND, RWND} (assume RWND >> CWND)
* 2 types of losses (“Congestion Event”)
  + Duplicate ACKs
  + Timeout
* Rate Adjustment
  + SS (Slow Start): initial rate is slow but ramps up exponentially fast until first loss event
    - Initially cwnd = 1 MSS
    - Double cwnd every RTT/incrementing cwnd for every ACK received
  + AIMD
    - Additive Increase: for each successful RTT, cwnd = cwnd + 1
    - Multiplicative decrease: cut cwnd in half after loss
  + ssthresh (slow start threshold)
    - Initialised to a large value
    - On timeout, ssthresh = CWND/2
    - When CWND = ssthresh, sender switches from slow-start to AIMD-style increase
  + State at sender
    - CWND initialised to a small constant
    - ssthresh initialised to a large constant
  + Loss Events
    - dupACK
      * ssthresh = CWND/2
      * CWND = CWND/2
    - Time out
      * ssthresh = CWND/2
      * CWND = 1
    - 为什么处理不一样？
      * 因为 time事件表明拥塞更严重
      * 3个重复的ack表示网络还能够传输一些segments
* TCP Fairness
  + Why AIMD? (4 alternatives)
    - AIAD: gentle increase, gentle decrease
      * Does not converge to fairness
    - AIMD: gentle increase, drastic decrease
      * Converges to fairness
    - MIAD: drastic increase, gentle decrease
    - MIMD: drastic increase and decrease
  + TCP Throughput
    - Timeout: CWND = W, throughput = W/RTT
    - After timeout: CWND = W/2, throughput = W/2RTT
    - Average throughput = 3W/4RTT



* + - Different RTTs
      * TCP unfair in the face of heterogeneous RTTs!
    - TCP will confuse corruption with congestion
    - short flows never leave slow start!
      * short flows never attain their fair share
    - too few packets to trigger dupACKs
      * Isolated loss may lead to timeouts
    - TCP fills up queues → long delays
      * Means that delays are large for *everyone*
    - Three easy ways to cheat
      * Increasing CWND faster than +1 MSS per RTT
      * Opening many connections
      * Using large initial CWND