CSCI 353 Acronym List

Bandwidth-Delay Product (BDP)

Carrier Sense Multiple Access (CSMA

CSMA/CD (Collision Detection)

CSMA/CA Collision Avoidance

Multiple Access Control (MAC)

Go-Back-N (GBN)

Selective Repeat (SR)

SNR (Signal/Noise)

Bit Error Rate (BER)

AS Autonomous Systems -- region of network under a single administrative entity)

Link State, e.g., Open Shortest Path First (OSPF)

Distance Vector, e.g., Routing Information Protocol (RIP)

Path Vector, e.g., Border Gateway Protocol (BGP)

LAN = Local Area Network

WAN = Wide Area Network

CIDR = Classless Interdomain Routing

eBGP: BGP sessions between border routers in different ASes

iBGP: BGP sessions between border routers and other  
routers within the same AS

IGP: “Interior Gateway Protocol” = Intradomain routing protocol

TTL – time to live

Maximum Transmission Unit” (MTU)

Differentiated Services Code Point (DSCP)

ToS type of service

Domain Name System (DNS)

**Address Translation Protocol (ARP)**

**Dynamic Host Configuration Protocol (DHCP)**

Top-level domain (TLD)

**Uniform Record Locator (URL)**

**Hyper Text Transfer Protocol (HTTP)**

Internet Protocol (IP)

Internet Control Message Protocol (ICMP)

Internet Group Management Protocol (IGMP)

Protocol-Independent Multicast (PIM)

User Datagram Protocol (UDP)

Transmission Control Protocol (**TCP**)

**Initial Sequence Number (ISN)**

**Timeout value (RTO) Recovery Time Objective**

**Congestion Window: CWND**

**Reciever Window (RWND)**

**MSS: Maximum Segment Size**

**Additive Increase Multiplicative Decrease (AIMD)**

**CDN (content delivery network)**

**BOB(bag of bits)**

QUIZ

Internet Protocol (IP) – datagrams transmitted over network layer, may be dropped, best effort, connectionless, end to end priciple

The Internet Control Message Protocol (ICMP) - provides information about the network layer to end hosts and routers. sits above IP and is therefore strictly a transport layer mechanism. tools “ping” and “traceroute” both rely on ICMP.

Unicast vs. Multicast – depends on topology

Multicast - send from one to many, cost less than many unicast, “similar reliability to unicast”, special case of broadcast, anonymous address: can send to computers without necessarily knowing who they are, more efficient: don’t send duplicates, instead let the network duplicate the packet

Internet Group Management Protocol (IGMP) - operates over broadcast LANs and point-to-point links

similar to ICMP in unicast IP, determine what IP mcast groups have receivers present on the LAN

Protocol-Independent Multicast (PIM) - Balance Dense and Sparse distribution

* Broadcast and prune keeps state off-tree and is suitable when members are densely distributed
* Explicit join/center-based approach keeps state on-tree and is suitable when members are sparsely distributed
* PIM attempts to combine the best of both worlds
* PIM-DM(dense mode): flood but use routing tables for optimizations
* PIM-SM(sparse mode): selectively forward packets based on membership spanning tree.

Why a transport layer: Demultiplexing, Need a way to decide which packets go to which applications

Role of the Transport Layer - Communication between processes, Provide common end-to-end services for app layer [optional], TCP and UDP are the common transport protocols, UDP is a minimalist, no-frills transport protocol, TCP is the whole-hog protocol

User Datagram Protocol (UDP)- connectionless, datagram, unrealiable - simpler, datagram delivery service between application processes

UDP socket: TYPE is SOCK\_DGRAM - OS stores (local port, local IP address)  socket

TCP socket: TYPE is SOCK\_STREAM - OS stores (local port, local IP, remote port, remote IP )  socket

port as a transport layer identifier, OS stores mapping between sockets and ports

IP

**4-bit**

**Version**

**4-bit**

**Header**

**Length**

**8-bit**

**Type of Service**

**(TOS)**

**16-bit Total Length (Bytes)**

16-bit Identification

**3-bit**

**Flags**

13-bit Fragment Offset

**8-bit Time to**

**Live (TTL)**

**8-bit Protocol**

16-bit Header Checksum

**32-bit Source IP Address**

**32-bit Destination IP Address**

Options (if any)

**Payload**

TCP

Source port

Destination port

Sequence number

Acknowledgment

Advertised window

HdrLen

Flags

0

Checksum

Urgent pointer

Options (variable)

Data

UDP

SRC port

DST port

checksum

length

DATA

Mechanisms for coping with bad events

Mechanisms for coping with bad events

Checksums: to detect corruption

ACKs: receiver tells sender that it received packet

NACK: receiver tells sender it did not receive packet

Sequence numbers: a way to identify packets

Retransmissions: sender resends packets

Timeouts: a way of deciding when to resend a packet

**“Stop and Wait” - world’s most inefficient, If TRANS << RTT then Throughput ~ DATA/RTT**

**Sliding Window - send up to *n* packets at a time, window slides on ACKs,Throughput ~ MIN[ nDATA/RTT, Link Bandwidth], cummalative or selective ACKs**

**Go-Back-N (GBN) – resends all since the last ACK**

**Selective Repeat (SR) – only resend packets without ACK**

**UDP: Datagram messaging service**

**-No-frills extension of “best-effort” IP**

**-Multiplexing/Demultiplexing among processes**

**-Discarding corrupted packets (optional)**

**TCP: Reliable, in-order delivery**

***-What UDP provides, plus:***

**-Retransmission of lost and corrupted packets**

**-Flow control (to not overflow receiver)**

**-Congestion control (to not overload network)**

**-“Connection” set-up & tear-down**

Receiver advertises to the sender where the receiver window currently ends (“righthand edge”) Sender agrees not to exceed this amount by setting its own window size to a value that can’t send beyond the receiver’s righthand edge

**Initial Sequence Number (ISN) - Sequence number for the very first byte, To establish a connection, hosts exchange ISNs**

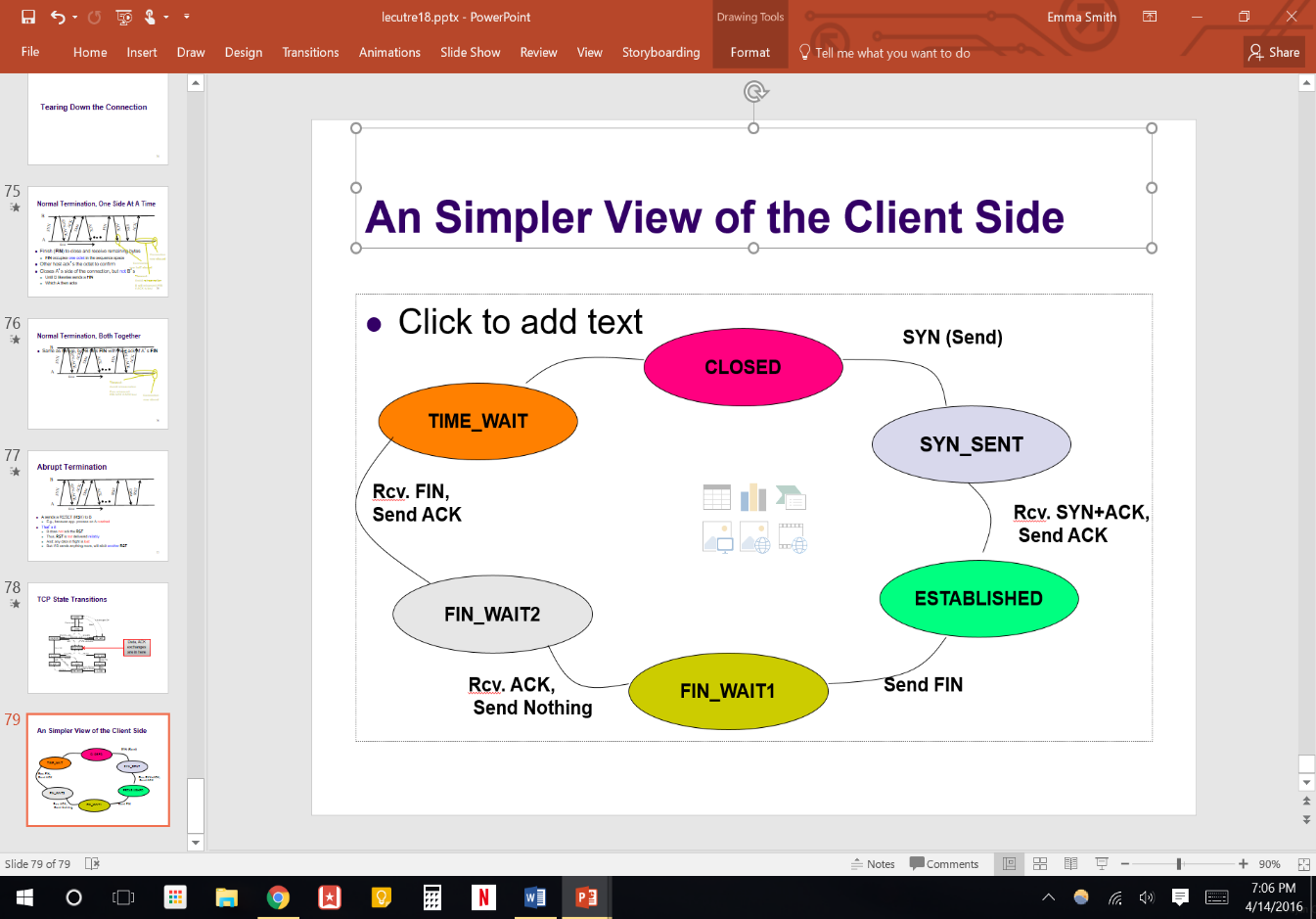
**Three-way handshake to establish connection**

**-Host A sends a SYN (open; “synchronize sequence numbers”) to host B**

**-Host B returns a SYN acknowledgment (SYN ACK)**

**-Host A sends an ACK to acknowledge the SYN ACK**

**TCP State Transitions**

**W:\Editorial\KARYN\Booksold\PD3e\final figures\Metafiles\05x07.WMF**

**Round Trip Time (RTT)  
**

**Karn/Partridge Algorithm**

**-Measure *SampleRTT* only for original transmissions**

**-Timeout value (RTO) = 2 × *EstimatedRTT***

**-Use exponential backoff for retransmissions, Every time RTO timer expires, set RTO ← 2·RTO**

**Jacobson/Karels Algorithm**

**-Deviation = | SampleRTT – EstimatedRTT |**

**-RTO = EstimatedRTT + 4 x EstimatedDeviation**

**Congestion Window: CWND - How many bytes can be sent without overflowing routers**

**Flow control window: AdvertisedWindow (RWND) - How many bytes can be sent without overflowing receiver’s buffers**

**Sender-side window = minimum{CWND, RWND}**

**MSS: Maximum Segment Size**

**“Slow Start” Phase - Sender starts at a slow rate but increases exponentially until first loss**

**-Initially, CWND = 1 So, initial sending rate is MSS/RTT**

**-Double the CWND for each RTT with no loss (Simpler: for each ACK, CWND += 1)**

**-Linear increase per ACK(CWND+1) 🡺 exponential increase per RTT (2xCWND)**

**“Additive Increase Multiplicative Decrease” (AIMD)**

**-Window grows by one MSS for every RTT with no loss**

**-For each successful RTT, CWND = CWND+1 (Simpler: for each ACK, CWND = CWND+ 1/CWND)**

**-On loss of packet, divide congestion window in half, CWND = CWND/2**

**“slow start threshold” (ssthresh) When does a sender stop Slow-Start and start Additive Increase?**

**-Initialized to a large value and On timeout, ssthresh = CWND/2**

**-When CWND = ssthresh, sender switches from slow-start to AIMD-style increase**

**If** CWND < ssthresh, CWND += 1(Slow start) **Else**  CWND = CWND + 1/CWND(add incr)

**On Timeout** ssthresh = CWND/2 and CWND = 1 (mult decr)

**If** dupACKcount = 3 /\* fast retransmit \*/ **then** ssthresh = CWND/2 and CWND = CWND/2

**Fast Recovery**

-If dupACKcount = 3, then ssthresh = cwnd/2 and cwnd = ssthresh + 3

TCP-Tahoe = CWND =1 on triple dupACK

TCP-Reno = CWND =1 on timeout, CWND = CWND/2 on triple dupack

TCP-newReno = TCP-Reno + improved fast recovery

TCP-SACK = incorporates selective acknowledgements

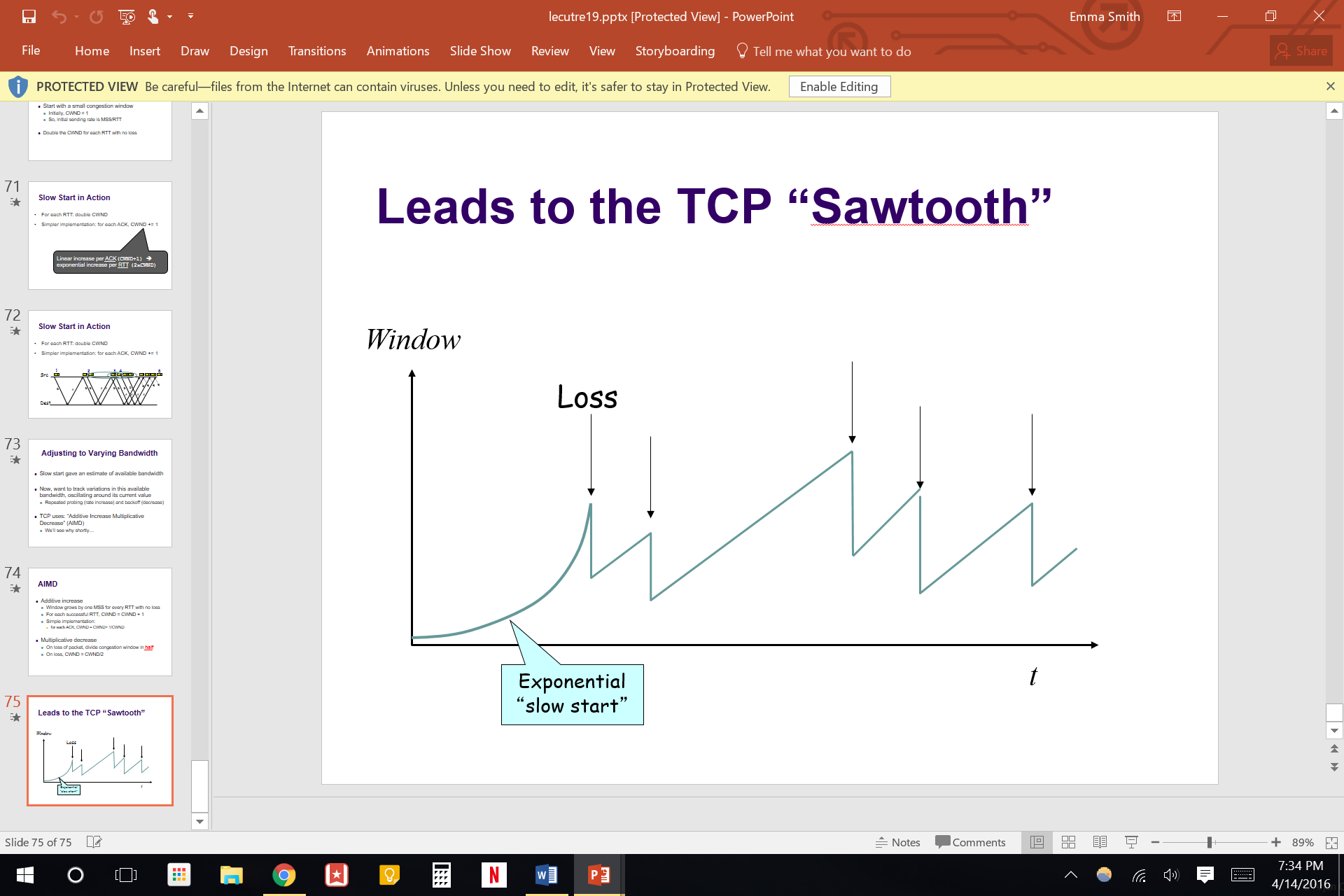
AIMD is the only one to converge on a fair and efficient use

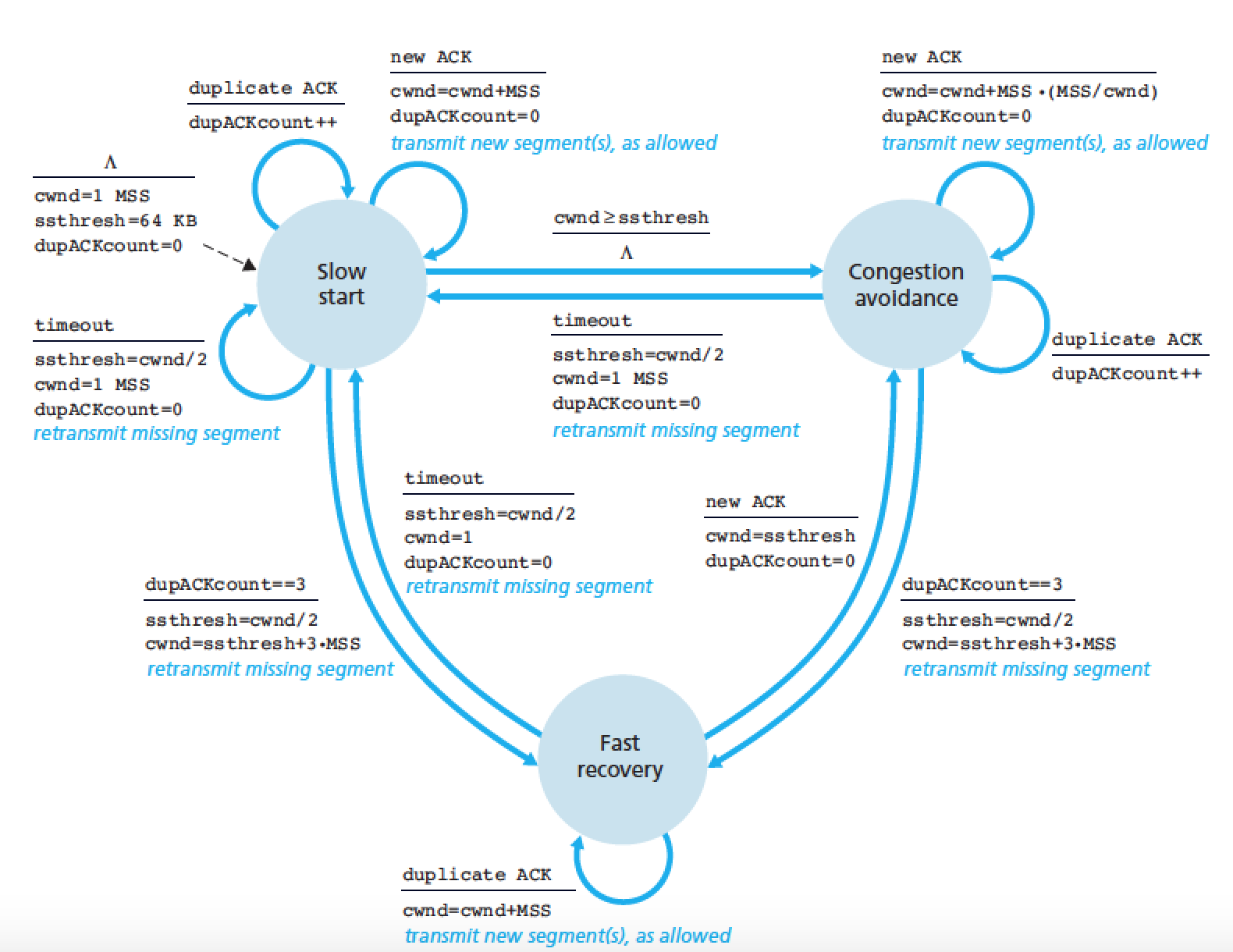
AIAD: gentle increase, gentle decrease

AIMD: gentle increase, drastic decrease

MIAD: drastic increase, gentle decrease

MIMD: drastic increase and decrease





* **CDN architecture (content delivery network)**