Midterm Review

EECS 489 Computer Networks

http://www.eecs.umich.edu/courses/eecs489/w07

Z. Morley Mao

Monday Feb 19, 2007

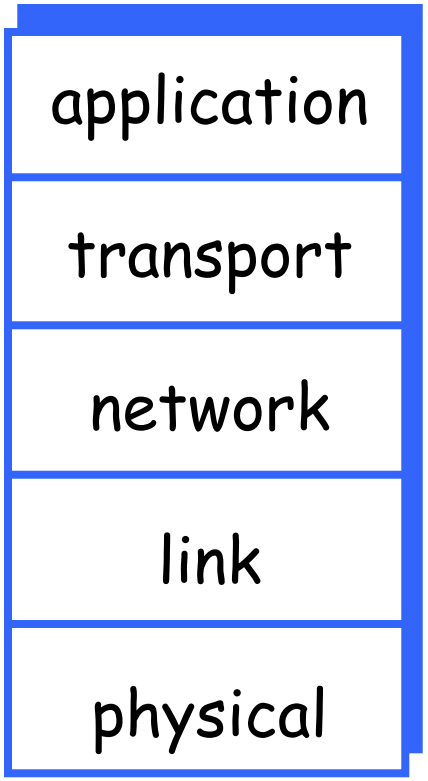
Acknowledgement: Some slides taken from Kurose&Ross and Katz&Stoica

Adminstrivia

- Homework 2
  - Problems from the book
  - You can either use Turnin program or turn in the homework on paper to my office.
  - Due date: tomorrow -- 2/20
- Midterm 1 is in class on Wednesday March 7th
  - Please let us know if you prefer to take it early
  - Material: Chapter 1-4
  - You can have one sheet of notes for the midterm.

Internet protocol stack

- application: supporting network applications
  - FTP, SMTP, HTTP
- transport: host-host data transfer
  - TCP, UDP
- network: routing of datagrams from source to destination
  - IP, routing protocols
- link: data transfer between neighboring network elements
  - PPP, Ethernet
- physical: bits “on the wire”



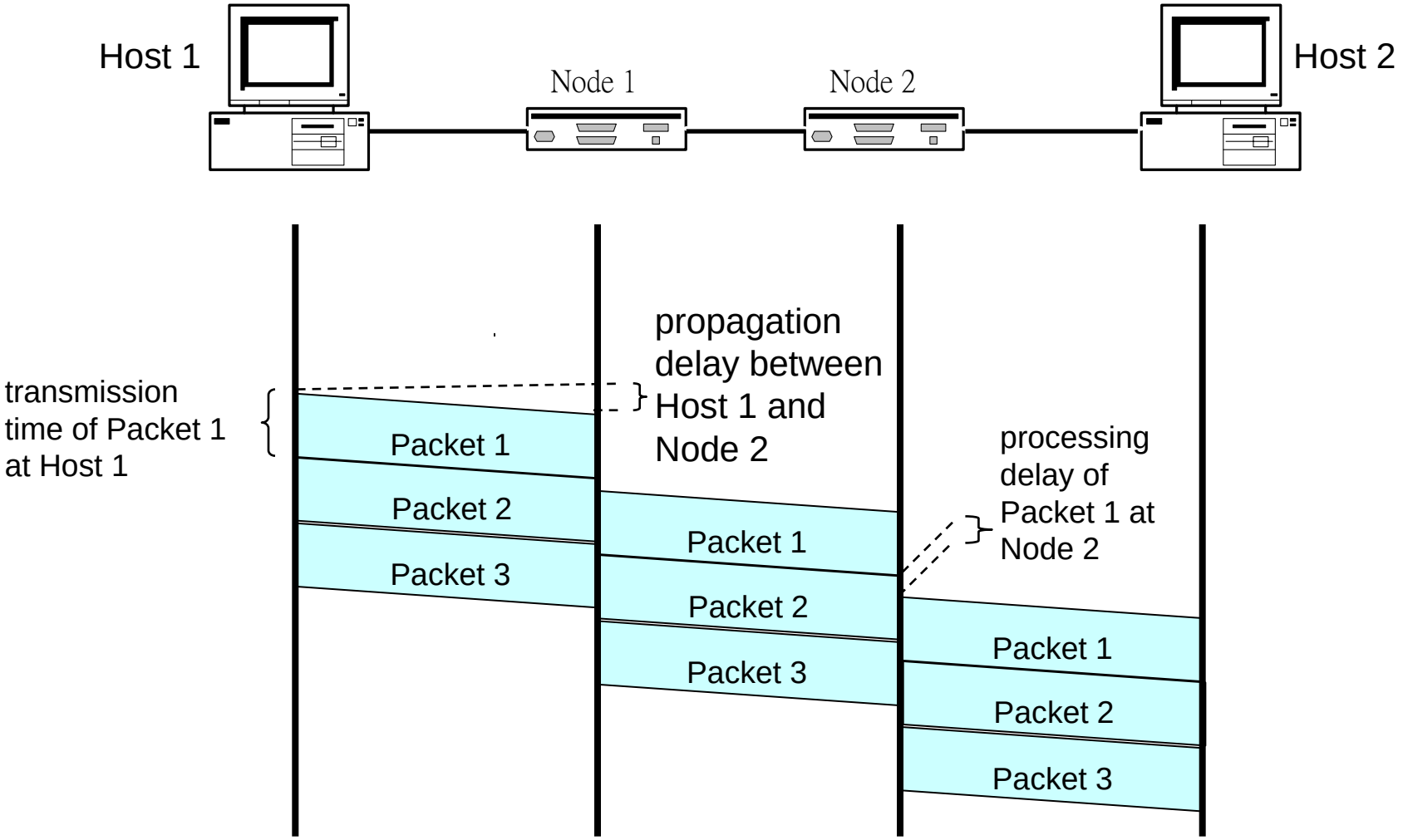
Think at two levels

- Protocols (Details of protocols may change!)
  - HTTP, SMTP, FTP, DNS, RTP, RTCP, RSVP, SNMP, SIP, H323, MobileIP
  - UDP, TCP, ICMP
  - BGP, RIP, OSPF, (link-state, distance-vector, path-vector)
  - IP, ARP
  - CSMA/CD (CA), MPLS, CDMA, FDMA
- Principles/concepts (**fundamental** to network design)
  - Packet switching, congestion control, flow control,
  - Caching/replication, layering (level of indirection), multiplexing
  - Hierarchical structure, signaling, pipelining, error coding
  - End to end principle, virtualization, randomization

Topics of importance

- Project assignments: PA1
  - Socket programming, blocking and non-blocking I/O
  - Server programming
- Packet switching vs. circuit switching
- Router architectures
  - Queue management
- TCP
  - Congestion control, flow control
- Routing protocols
  - Interdomain and intradomain routing

Timing of Datagram Packet Switching



TCP: Implementing AIMD

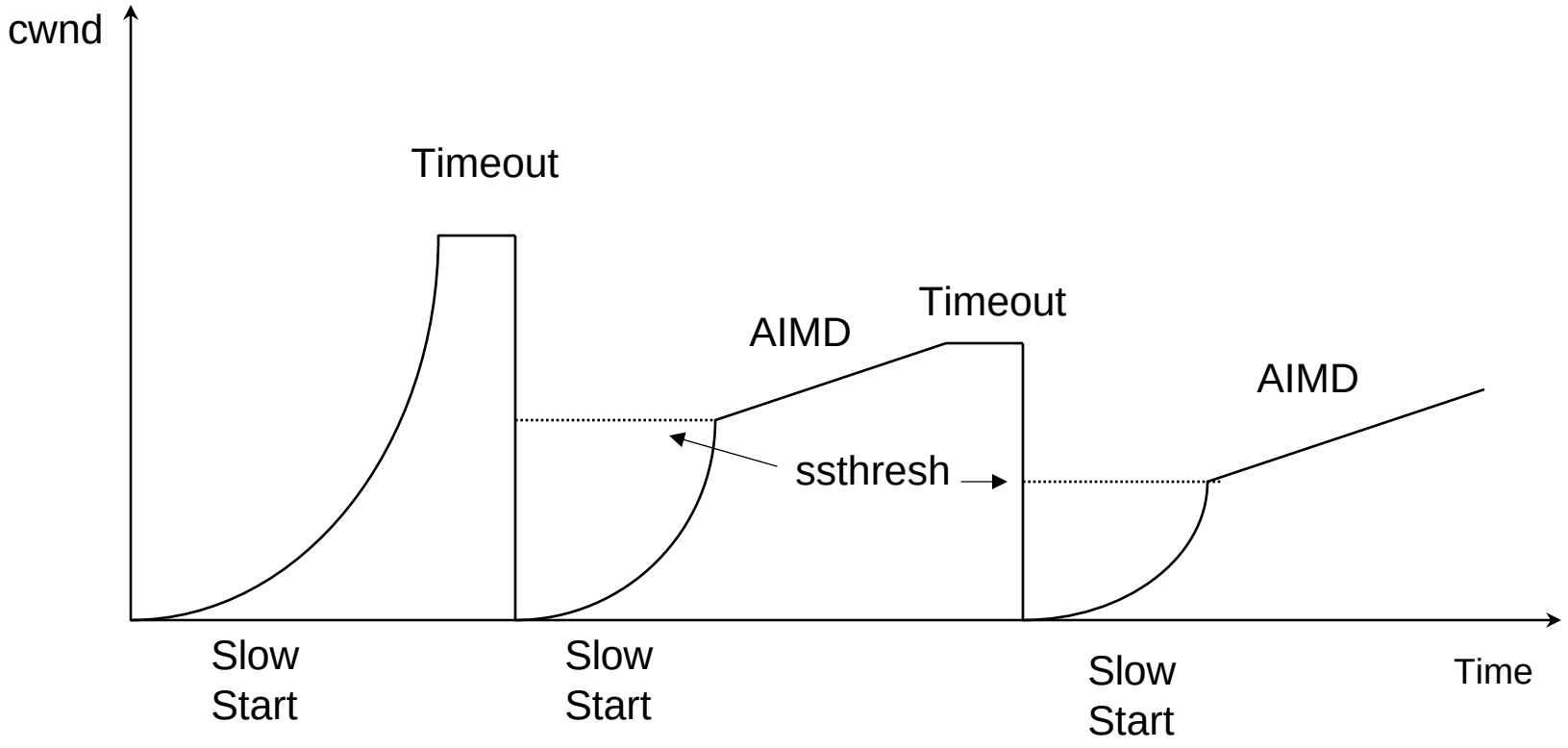
- After each ACK
  - increment *cwnd* by 1/*cwnd* (*cwnd* += 1/*cwnd*)
  - as a result, *cwnd* is increased by one only if all segments in a *cwnd* have been acknowledged

- But need to decide when to leave slow-start and enter AIMD
  - use ssthresh variable

Slow Start/AIMD Pseudocode

```
Initially:
    cwnd = 1;
    ssthresh = infinite;
New ack received:
    if (cwnd < ssthresh)
        /* Slow Start*/
        cwnd = cwnd + 1;
    else
        /* Congestion Avoidance */
        cwnd = cwnd + 1/cwnd;
Timeout:
    /* Multiplicative decrease */
    ssthresh = cwnd/2;
    cwnd = 1;
```

The big picture (with timeouts)

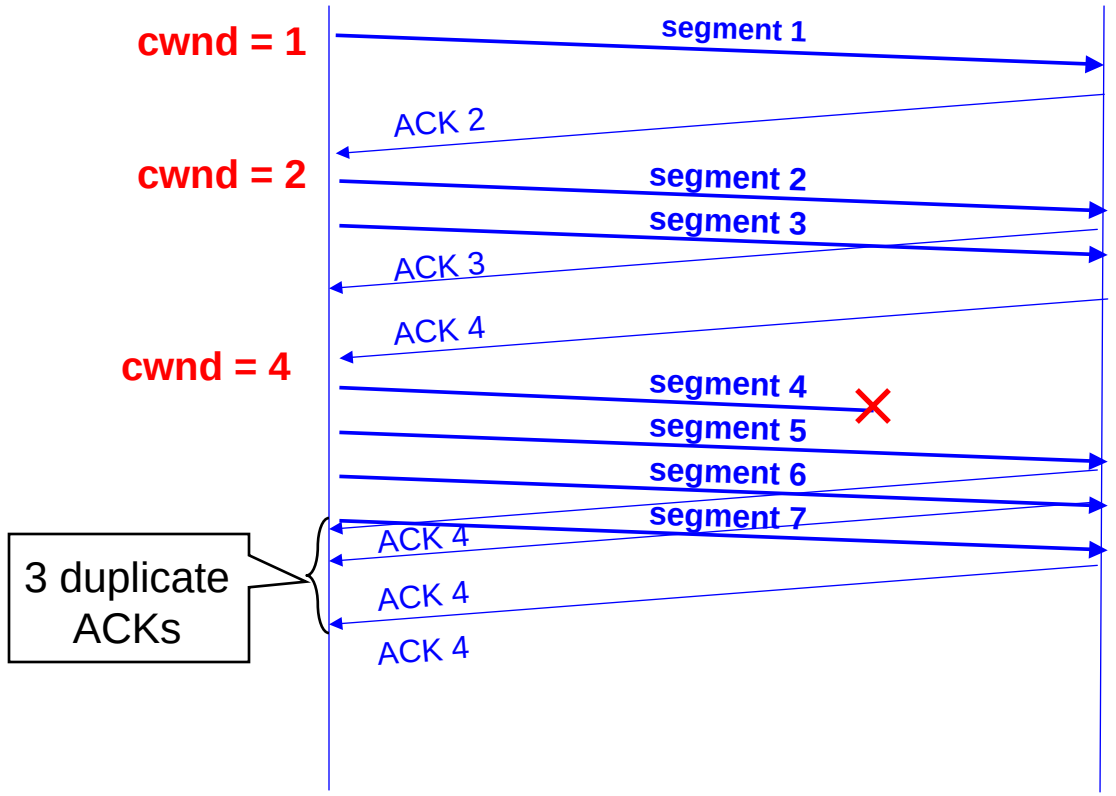


Congestion Detection Revisited

- Wait for Retransmission Time Out (RTO)
  - RTO kills throughput
- In BSD TCP implementations, RTO is usually more than 500ms
  - the granularity of RTT estimate is 500 ms
  - retransmission timeout is RTT + 4 * mean_deviation
- Solution: Don’t wait for RTO to expire

Fast Retransmits

- Resend a segment after 3 duplicate ACKs
  - a duplicate ACK means that an out-of sequence segment was received
- Notes:
  - ACKs are for next expected packet
  - packet reordering can cause duplicate ACKs
  - window may be too small to get enough duplicate ACKs



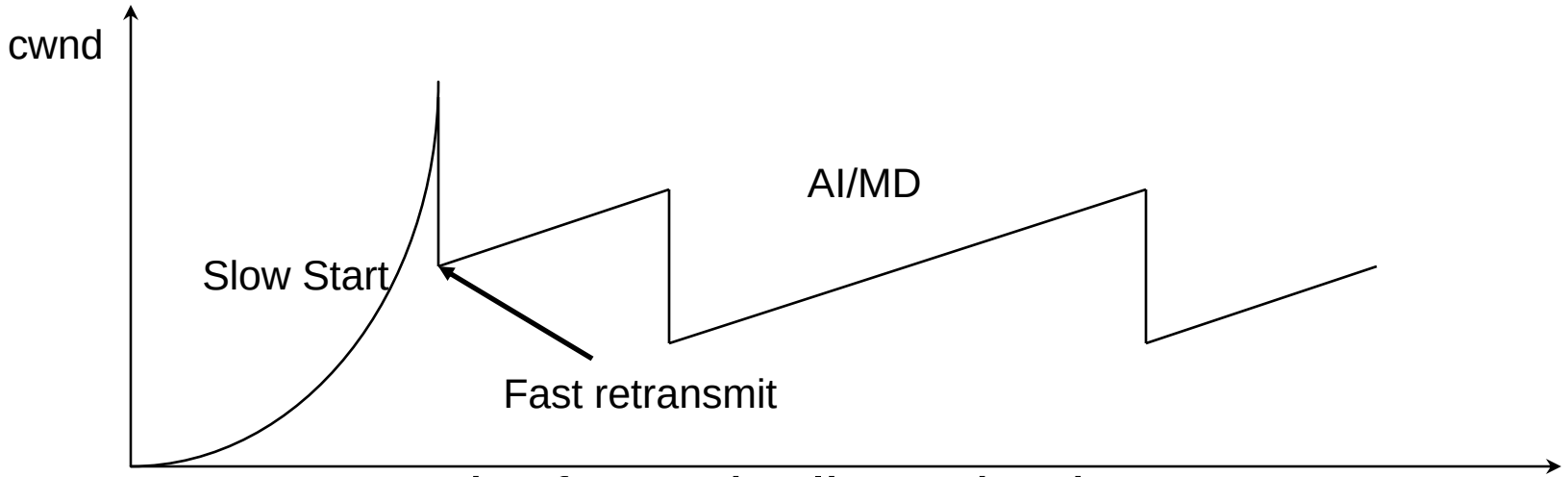
Fast Retransmit

- Time-out period often relatively long:
  - long delay before resending lost packet
- Detect lost segments via duplicate ACKs.
  - Sender often sends many segments back-to-back
  - If segment is lost, there will likely be many duplicate ACKs.
- If sender receives 3 ACKs for the same data, it supposes that segment after ACKed data was lost:
  - fast retransmit: resend segment before timer expires

Fast Recovery: After a Fast Retransmit

- *ssthresh* = *cwnd* / 2
- *cwnd* = *ssthresh*
  - instead of setting *cwnd* to 1, cut cwnd in half (multiplicative decrease)
- for each dup ack arrival
  - *dupack*++
  - MaxWindow = min(*cwnd* + *dupack*, AdvWin)
  - indicates packet left network, so we may be able to send more
- receive ack for new data (beyond initial dup ack)
  - *dupack* = 0
  - exit fast recovery
- But when RTO expires still do *cwnd* = 1

Fast Retransmit and Fast Recovery



- Retransmit after 3 duplicated acks
  - Prevent expensive timeouts
- Reduce slow starts
- At steady state, *cwnd* oscillates around the optimal window size

TCP Congestion Control Summary

- Measure available bandwidth
  - slow start: fast, hard on network
  - AIMD: slow, gentle on network
- Detecting congestion
  - timeout based on RTT
    - robust, causes low throughput
  - Fast Retransmit: avoids timeouts when few packets lost
    - can be fooled, maintains high throughput
- Recovering from loss
  - Fast recovery: don’t set cwnd=1 with fast retransmits