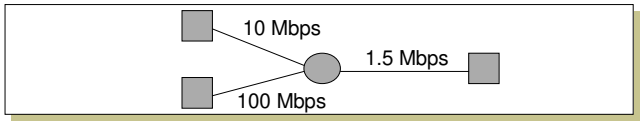
15-441 Computer Networking

Lecture 17 – TCP & Congestion Control

Outline

- Congestion sources and collapse
- Congestion control basics
- TCP connection setup/data transfer
- TCP flow control

Congestion

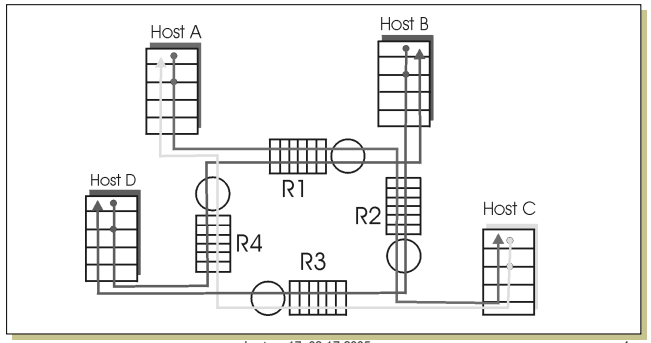


- Different sources compete for resources inside network
- Why is it a problem?
  - Sources are unaware of current state of resource
  - Sources are unaware of each other
- Manifestations:
  - Lost packets (buffer overflow at routers)
  - Long delays (queuing in router buffers)
  - Can result in throughput less than bottleneck link (1.5Mbps for the above topology) → a.k.a. congestion collapse

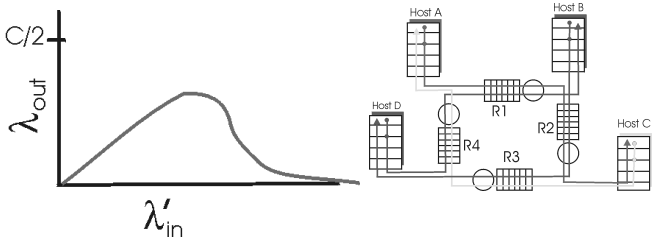
Causes & Costs of Congestion

- Four senders – multihop paths
- Timeout/retransmit

Q: What happens as rate increases?



Causes & Costs of Congestion



- When packet dropped, any "upstream transmission capacity used for that packet was wasted!

Congestion Collapse

- Definition: *Increase in network load results in decrease of useful work done*
- Many possible causes
  - Spurious retransmissions of packets still in flight
    - Classical congestion collapse
    - How can this happen with packet conservation
    - Solution: better timers and TCP congestion control
  - Undelivered packets
    - Packets consume resources and are dropped elsewhere in network
    - Solution: congestion control for ALL traffic

Congestion Control and Avoidance

- A mechanism which:
  - Uses network resources efficiently
  - Preserves fair network resource allocation
  - Prevents or avoids collapse
- Congestion collapse is not just a theory
  - Has been frequently observed in many networks

Approaches Towards Congestion Control

- Two broad approaches towards congestion control:

- 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 sender should send at
  - Problem: makes routers complicated

Example: TCP Congestion Control

- Very simple mechanisms in network
  - FIFO scheduling with shared buffer pool
  - Feedback through packet drops
- TCP interprets packet drops as signs of congestion and slows down
  - This is an assumption: packet drops are not a sign of congestion in all networks
    - E.g. wireless networks
- Periodically probes the network to check whether more bandwidth has become available.

Outline

- Congestion sources and collapse
- Congestion control basics
- TCP connection setup/data transfer
- TCP flow control

Objectives

- Simple router behavior
- Distributedness
- Efficiency: $X = \Sigma x_i(t)$
- Fairness: $(\Sigma x_i)^2/n(\Sigma x_i^2)$
  - What are the important properties of this function?
- Convergence: control system must be stable

Basic Control Model

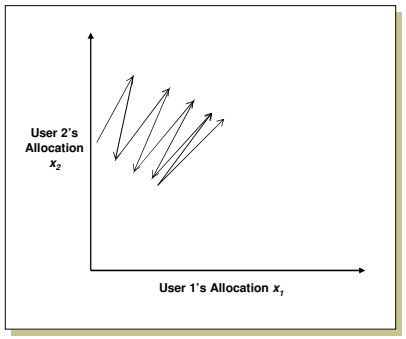
- Reduce speed when congestion is perceived
  - How is congestion signaled?
    - Either mark or drop packets
  - How much to reduce?
- Increase speed otherwise
  - Probe for available bandwidth – how?

Linear Control

- Many different possibilities for reaction to congestion and probing
  - Examine simple linear controls
    - Window(t + 1) = a + b Window(t)
    - Different a_i/b_i for increase and a_d/b_d for decrease
- Supports various reaction to signals
  - Increase/decrease additively
  - Increased/decrease multiplicatively
  - Which of the four combinations is optimal?

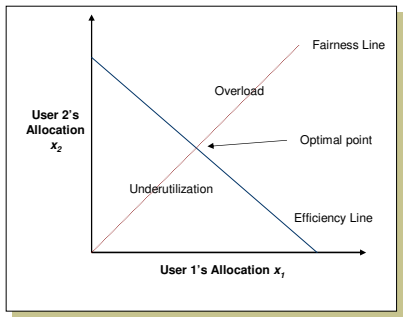
Phase Plots

- Simple way to visualize behavior of competing connections over time



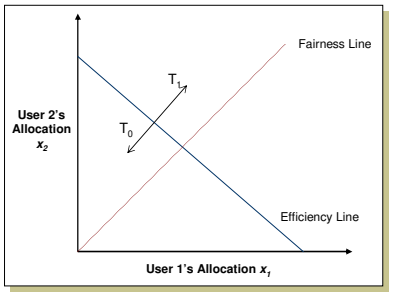
Phase Plots

- What are desirable properties?
- What if flows are not equal?



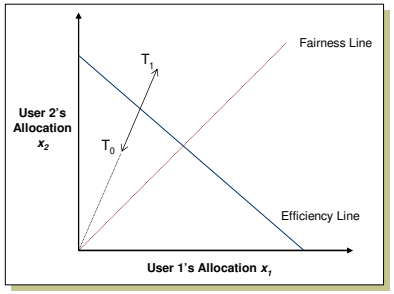
Additive Increase/Decrease

- Both X_1 and X_2 increase/ decrease by the same amount over time
  - Additive increase improves fairness and additive decrease reduces fairness

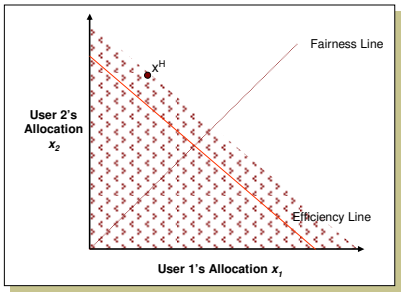


Muliplicative Increase/Decrease

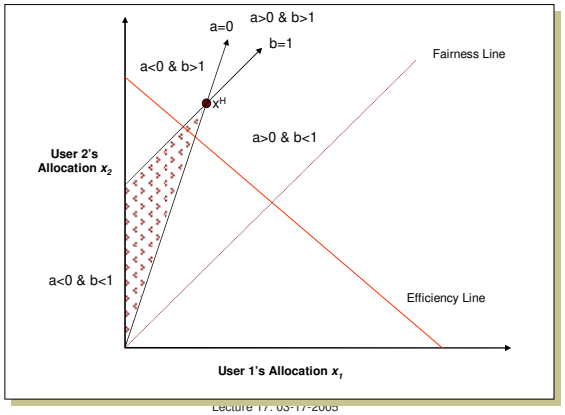
- Both X_1 and X_2 increase by the same factor over time
  - Extension from origin – constant fairness



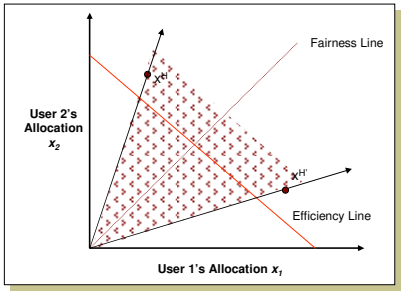
Convergence to Efficiency



Distributed Convergence to Efficiency

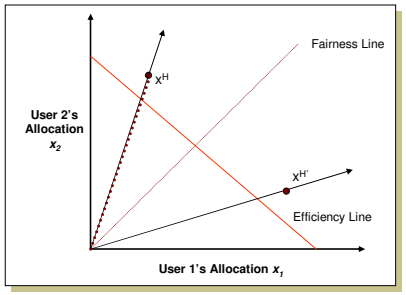


Convergence to Fairness



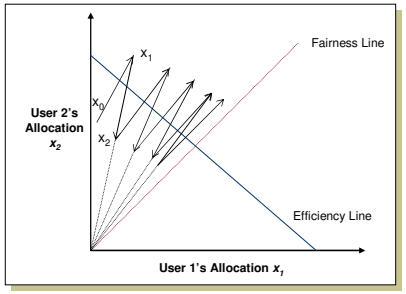
Convergence to Efficiency & Fairness

- Intersection of valid regions
- For decrease: a=0 & b < 1



What is the Right Choice?

- Constraints limit us to AIMD
  - Can have multiplicative term in increase (MAIMD)
  - AIMD moves towards optimal point



Good Ideas So Far…

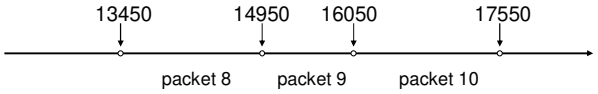
- Flow control
  - Stop & wait
  - Parallel stop & wait
  - Sliding window
- Loss recovery
  - Timeouts
  - Acknowledgement-driven recovery (selective repeat or cumulative acknowledgement)
- Congestion control
  - AIMD → fairness and efficiency
- How does TCP actually implement these?

Outline

- Congestion sources and collapse
- Congestion control basics
- TCP connection setup/data transfer
- TCP flow control

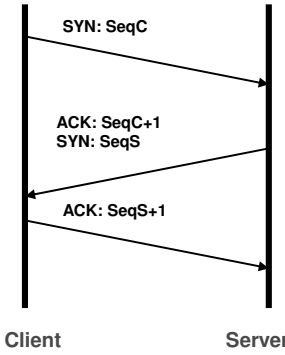
Sequence Number Space

- Each byte in byte stream is numbered.
  - 32 bit value
  - Wraps around
  - Initial values selected at start up time
- TCP breaks up the byte stream in packets.
  - Packet size is limited to the Maximum Segment Size
- Each packet has a sequence number.
  - Indicates where it fits in the byte stream



Establishing Connection: Three-Way handshake

- Each side notifies other of starting sequence number it will use for sending
  - Why not simply chose 0?
    - Must avoid overlap with earlier incarnation
    - Security issues
- Each side acknowledges other's sequence number
  - SYN-ACK: Acknowledge sequence number + 1
- Can combine second SYN with first ACK



TCP Connection Setup Example

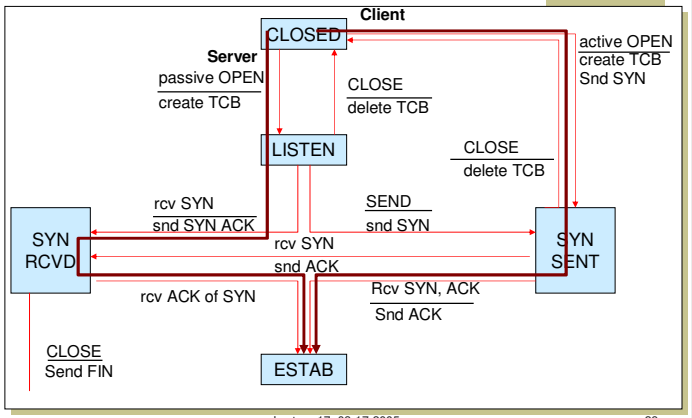
```
09:23:33.042318 IP 128.2.222.198.3123 > 192.216.219.96.80: S
  4019802004:4019802004(0) win 65535 <mss 1260,nop,nop,sackOK>
(DF)

09:23:33.118329 IP 192.216.219.96.80 > 128.2.222.198.3123: S
  3428951569:3428951569(0) ack 4019802005 win 5840 <mss
1460,nop,nop,sackOK> (DF)

09:23:33.118405 IP 128.2.222.198.3123 > 192.216.219.96.80: . ack
  3428951570 win 65535 (DF)
```

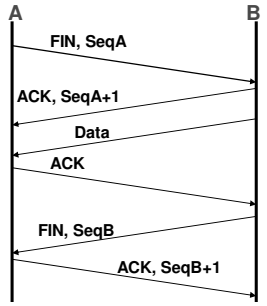
- Client SYN
  - SeqC: Seq. #4019802004, window 65535, max. seg. 1260
- Server SYN-ACK+SYN
  - Receive: #4019802005 (= SeqC+1)
  - SeqS: Seq. #3428951569, window 5840, max. seg. 1460
- Client SYN-ACK
  - Receive: #3428951570 (= SeqS+1)

TCP State Diagram: Connection Setup



Tearing Down Connection

- Either side can initiate tear down
  - Send FIN signal
  - "I'm not going to send any more data"
- Other side can continue sending data
  - Half open connection
  - Must continue to acknowledge
- Acknowledging FIN
  - Acknowledge last sequence number + 1



TCP Connection Teardown Example

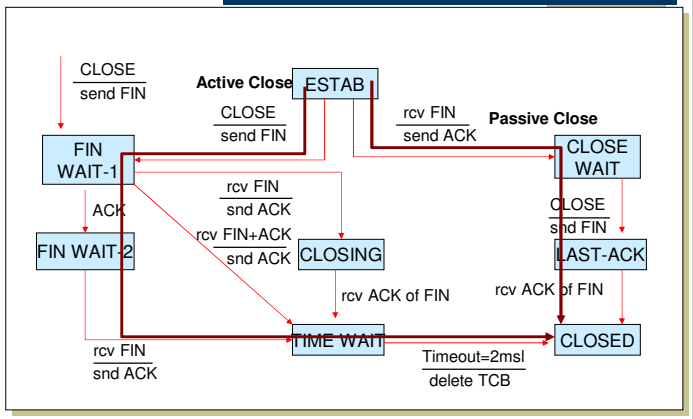
```
09:54:17.585396 IP 128.2.222.198.4474 > 128.2.210.194.6616: F
  1489294581:1489294581(0) ack 1909787689 win 65434 (DF)

09:54:17.585732 IP 128.2.210.194.6616 > 128.2.222.198.4474: F
  1909787689:1909787689(0) ack 1489294582 win 5840 (DF)

09:54:17.585764 IP 128.2.222.198.4474 > 128.2.210.194.6616: . ack
  1909787690 win 65434 (DF)
```

- Session
  - Echo client on 128.2.222.198, server on 128.2.210.194
- Client FIN
  - SeqC: 1489294581
- Server ACK + FIN
  - Ack: 1489294582 (= SeqC+1)
  - SeqS: 1909787689
- Client ACK
  - Ack: 1909787690 (= SeqS+1)

State Diagram: Connection Tear-down

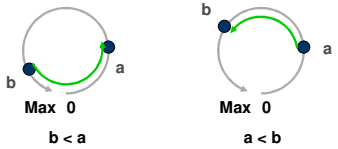


Outline

- Congestion sources and collapse
- Congestion control basics
- TCP connection setup/data transfer
- TCP flow control

Sequence Numbers

- 32 Bits, Unsigned
  - Circular Comparison

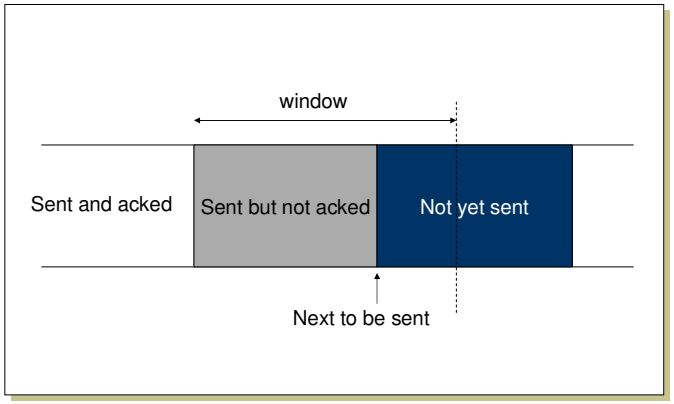


- Why So Big?
  - For sliding window, must have
    |Sequence Space| > |Sending Window| + |Receiving Window|
    - No problem
  - Also, want to guard against stray packets
    - With IP, packets have maximum lifetime of 120s
    - Sequence number would wrap around in this time at 286MB/s

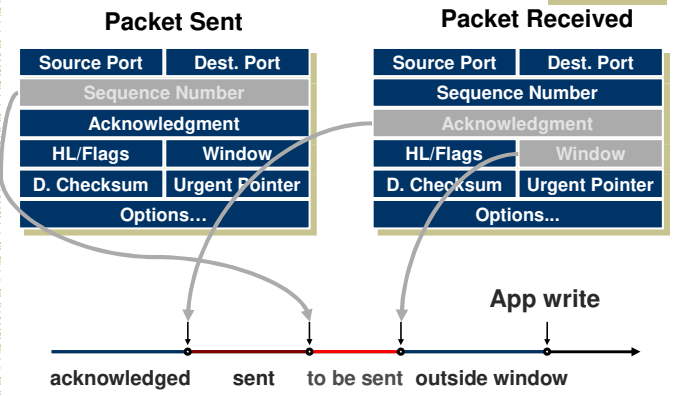
TCP Flow Control

- TCP is a sliding window protocol
  - For window size *n*, can send up to *n* bytes without receiving an acknowledgement
  - When the data is acknowledged then the window slides forward
- Each packet advertises a window size
  - Indicates number of bytes the receiver has space for
- Original TCP always sent entire window
  - Congestion control now limits this

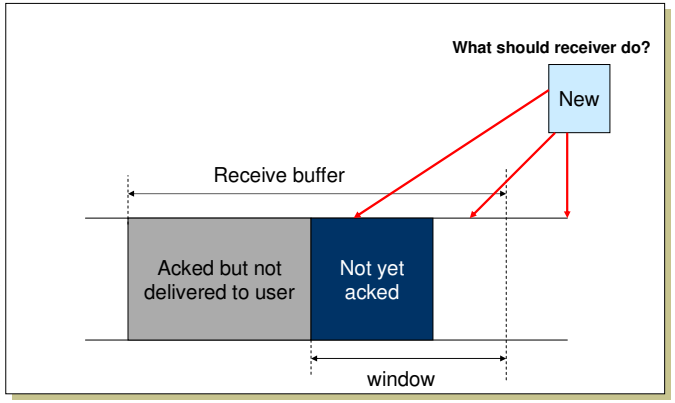
Window Flow Control: Send Side



Window Flow Control: Send Side



Window Flow Control: Receive Side



TCP Persist

- What happens if window is 0?
  - Receiver updates window when application reads data
  - What if this update is lost?
- TCP Persist state
  - Sender periodically sends 1 byte packets
  - Receiver responds with ACK even if it can't store the packet

Performance Considerations

- The window size can be controlled by receiving application
  - Can change the socket buffer size from a default (e.g. 8Kbytes) to a maximum value (e.g. 64 Kbytes)
- The window size field in the TCP header limits the window that the receiver can advertise
  - 16 bits → 64 KBytes
  - 10 msec RTT → 51 Mbit/second
  - 100 msec RTT → 5 Mbit/second
  - TCP options to get around 64KB limit

Important Lessons

- Why is congestion control needed?
- How to evaluate congestion control algorithms?
  - Why is AIMD the right choice for congestion control?
- TCP state diagram → setup/teardown
- TCP flow control
  - Sliding window → mapping to packet headers
  - 32bit sequence numbers (bytes)

EXTRA SLIDES

The rest of the slides are FYI

Detecting Half-open Connections

	TCP A		TCP B
1.	(CRASH)		(send 300, receive 100)
2.	CLOSED		ESTABLISHED
3.	SYN-SENT → <SEQ=400><CTL=SYN>	→	(??)
4.	(!!)	← <SEQ=300><ACK=100><CTL=ACK> ←	ESTABLISHED
5.	SYN-SENT → <SEQ=100><CTL=RST>	→	(Abort!!)
6.	SYN-SENT		CLOSED
7.	SYN-SENT → <SEQ=400><CTL=SYN>	→	

Other Congestion Collapse Causes

- Fragments
  - Mismatch of transmission and retransmission units
  - Solutions
    - Make network drop all fragments of a packet (early packet discard in ATM)
    - Do path MTU discovery
- Control traffic
  - Large percentage of traffic is for control
    - Headers, routing messages, DNS, etc.
- Stale or unwanted packets
  - Packets that are delayed on long queues
  - "Push" data that is never used