

15-744: Computer Networking

L-4 TCP



TCP Congestion Control



- Congestion Control
- RED
- Assigned Reading
 - [FJ93] Random Early Detection Gateways for Congestion Avoidance
 - [TFRC] Equation-Based Congestion Control for Unicast Applications (2 sections)

2

Introduction to TCP



- Communication abstraction:
 - Reliable
 - Ordered
 - Point-to-point
 - Byte-stream
 - Full duplex
 - Flow and congestion controlled
- Protocol implemented entirely at the ends
 - Fate sharing
- Sliding window with cumulative acks
 - Ack field contains last in-order packet received
 - Duplicate acks sent when out-of-order packet received

3

Key Things You Should Know Already



- Port numbers
- TCP/UDP checksum
- Sliding window flow control
 - Sequence numbers
- TCP connection setup
- TCP reliability
 - Timeout
 - Data-driven
- Chiu&Jain analysis of linear congestion control

4

Overview

- TCP congestion control
- TFRC
- TCP and queues
- Queuing disciplines
- RED

5

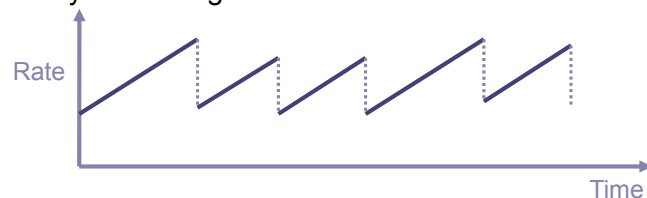
TCP Congestion Control

- Changes to TCP motivated by ARPANET congestion collapse
- Basic principles
 - AIMD
 - Packet conservation
 - Reaching steady state quickly
 - ACK clocking

6

AIMD

- Distributed, fair and efficient
- Packet loss is seen as sign of congestion and results in a multiplicative rate decrease
 - Factor of 2
- TCP periodically probes for available bandwidth by increasing its rate



7

Implementation Issue

- Operating system timers are very coarse – how to pace packets out smoothly?
- Implemented using a congestion window that limits how much data can be in the network.
 - TCP also keeps track of how much data is in transit
- Data can only be sent when the amount of outstanding data is less than the congestion window.
 - The amount of outstanding data is increased on a “send” and decreased on “ack”
 - $(\text{last sent} - \text{last acked}) < \text{congestion window}$
- Window limited by both congestion and buffering
 - Sender's maximum window = $\text{Min}(\text{advertised window}, \text{cwnd})$

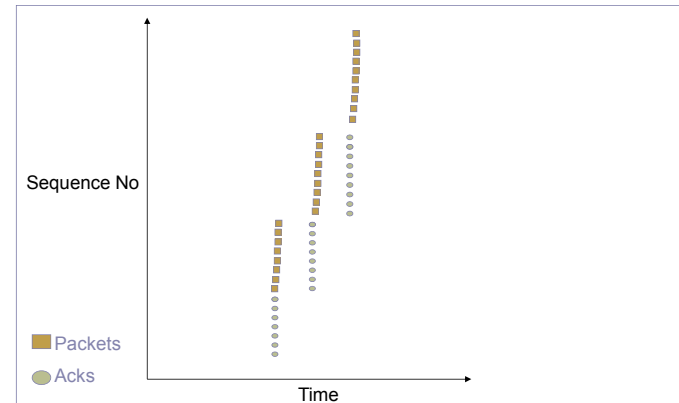
8

Congestion Avoidance

- If loss occurs when $cwnd = W$
 - Network can handle $0.5W \sim W$ segments
 - Set $cwnd$ to $0.5W$ (multiplicative decrease)
- Upon receiving ACK
 - Increase $cwnd$ by $(1 \text{ packet})/cwnd$
 - What is 1 packet? $\rightarrow$ 1 MSS worth of bytes
 - After $cwnd$ packets have passed by $\rightarrow$ approximately increase of 1 MSS
- Implements AIMD

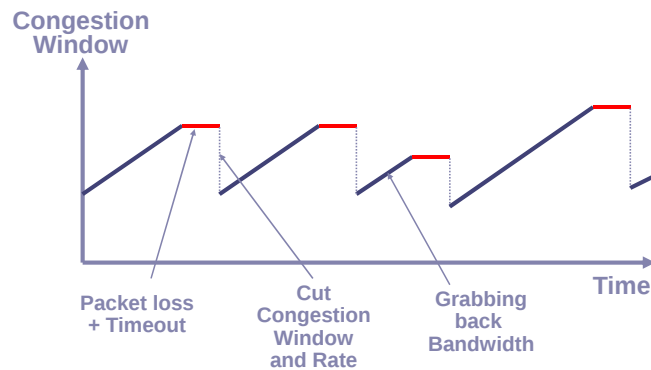
9

Congestion Avoidance Sequence Plot



10

Congestion Avoidance Behavior



11

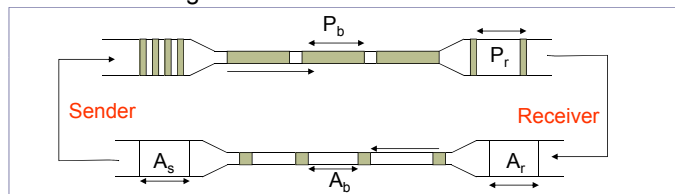
Packet Conservation

- At equilibrium, inject packet into network only when one is removed
 - Sliding window and not rate controlled
 - But still need to avoid sending burst of packets $\rightarrow$ would overflow links
 - Need to carefully pace out packets
 - Helps provide stability
- Need to eliminate spurious retransmissions
 - Accurate RTO estimation
 - Better loss recovery techniques (e.g. fast retransmit)

12

TCP Packet Pacing

- Congestion window helps to “pace” the transmission of data packets
- In steady state, a packet is sent when an ack is received
 - Data transmission remains smooth, once it is smooth
 - Self-clocking behavior



13

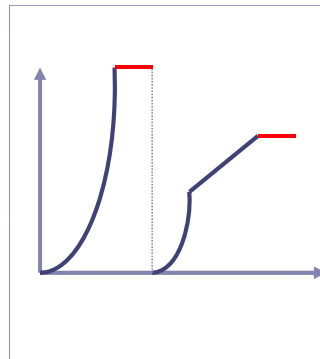
Reaching Steady State

- Doing AIMD is fine in steady state but slow...
- How does TCP know what is a good initial rate to start with?
 - Should work both for a CDPD (10s of Kbps or less) and for supercomputer links (10 Gbps and growing)
- Quick initial phase to help get up to speed (slow start)

14

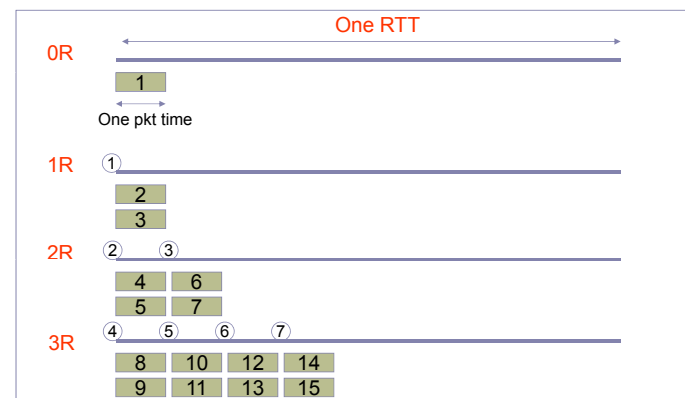
Slow Start Packet Pacing

- How do we get this clocking behavior to start?
 - Initialize $cwnd = 1$
 - Upon receipt of every ack, $cwnd = cwnd + 1$
- Implications
 - Window actually increases to W in $RTT * \log_2(W)$
 - Can overshoot window and cause packet loss



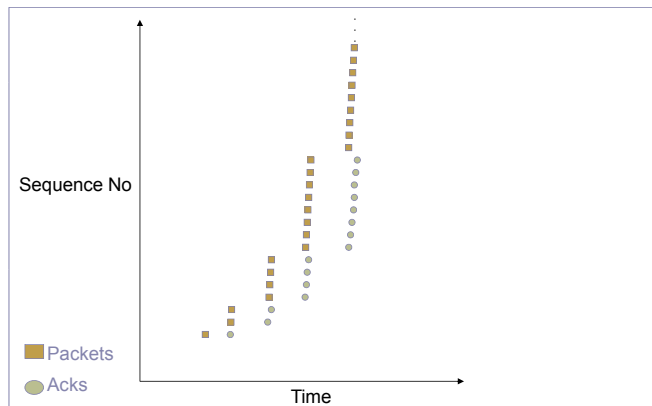
15

Slow Start Example



16

Slow Start Sequence Plot



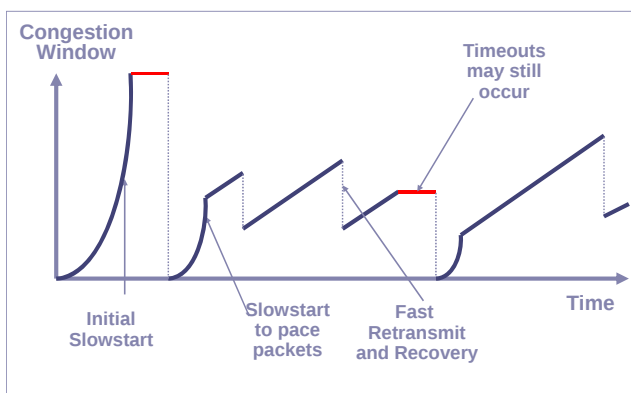
17

Return to Slow Start

- If packet is lost we lose our self clocking as well
 - Need to implement slow-start and congestion avoidance together
- When timeout occurs set ssthresh to $0.5w$
 - If $cwnd < ssthresh$, use slow start
 - Else use congestion avoidance

18

TCP Saw Tooth Behavior



19

Questions

- Current loss rates – 10% in paper
- Uniform reaction to congestion – can different nodes do different things?
 - TCP friendliness, GAIMD, etc.
- Can we use queuing delay as an indicator?
 - TCP Vegas
- What about non-linear controls?
 - Binomial congestion control

20

Overview



- TCP congestion control
- TFRC
 - See Matt Mukerjee's slides
- TCP and queues
- Queuing disciplines
- RED

21

Changing Workloads



- New applications are changing the way TCP is used
- 1980's Internet
 - Telnet & FTP → long lived flows
 - Well behaved end hosts
 - Homogenous end host capabilities
 - Simple symmetric routing
- 2000's Internet
 - Web & more Web → large number of short xfers
 - Wild west – everyone is playing games to get bandwidth
 - Cell phones and toasters on the Internet
 - Policy routing
- How to accommodate new applications?

22

TCP Friendliness



- What does it mean to be TCP friendly?
 - TCP is not going away
 - Any new congestion control must compete with TCP flows
 - Should not clobber TCP flows and grab bulk of link
 - Should also be able to hold its own, i.e. grab its fair share, or it will never become popular
- How is this quantified/shown?
 - Has evolved into evaluating loss/throughput behavior
 - If it shows $1/\sqrt{p}$ behavior it is ok
 - But is this really true?

23

TCP Friendly Rate Control (TFRC)



- Equation 1 – real TCP response

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1 + 32p^2)}$$

- 1st term corresponds to simple derivation
- 2nd term corresponds to more complicated timeout behavior
 - Is critical in situations with > 5% loss rates → where timeouts occur frequently
- Key parameters
 - RTO
 - RTT
 - Loss rate

24

RTO/RTT Estimation



- RTO not used to perform retransmissions
 - Used to model TCP's extremely slow transmission rate in this mode
 - Only important when loss rate is high
 - Accuracy is not as critical
- Different TCP's have different RTO calculation
 - Clock granularity critical → 500ms typical, 100ms, 200ms, 1s also common
 - $RTO = 4 * RTT$ is close enough for reasonable operation
- EWMA RTT
 - $RTT_{n+1} = (1-\alpha)RTT_n + \alpha RTTSAMP$

25

Loss Estimation



- Loss event rate vs. loss rate
- Characteristics
 - Should work well in steady loss rate
 - Should weight recent samples more
 - Should increase only with a new loss
 - Should decrease only with long period without loss
- Possible choices
 - Dynamic window – loss rate over last X packets
 - EWMA of interval between losses
 - Weighted average of last n intervals
 - Last n/2 have equal weight

26

Loss Estimation



- Dynamic windows has many flaws
- Difficult to choose weight for EWMA
- Solution WMA
 - Choose simple linear decrease in weight for last n/2 samples in weighted average
 - What about the last interval?
 - Include it when it actually increases WMA value
 - What if there is a long period of no losses?
 - Special case (history discounting) when current interval > 2 * avg

27

Slow Start



- Used in TCP to get rough estimate of network and establish ack clock
 - Don't need it for ack clock
 - TCP ensures that overshoot is not > 2x
 - Rate based protocols have no such limitation – why?
- TFRC slow start
 - New rate set to $\min(2 * sent, 2 * rcvd)$
 - Ends with first loss report → rate set to $\frac{1}{2}$ current rate

28

Congestion Avoidance

- Loss interval increases in order to increase rate
 - Primarily due to the transmission of new packets in current interval
 - History discounting increases interval by removing old intervals
 - .14 packets per RTT without history discounting
 - .22 packets per RTT with discounting
- Much slower increase than TCP
- Decrease is also slower
 - 4 – 8 RTTs to halve speed

29

Overview

- TCP congestion control
- TFRC
- TCP and queues
- Queuing disciplines
- RED

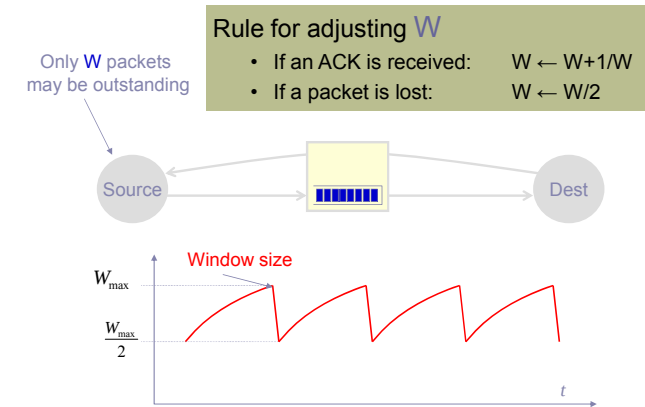
38

TCP Performance

- Can TCP saturate a link?
- Congestion control
 - Increase utilization until... link becomes congested
 - React by decreasing window by 50%
 - Window is proportional to rate * RTT
- Doesn't this mean that the network oscillates between 50 and 100% utilization?
 - Average utilization = 75%??
 - **No...this is *not* right!**

31

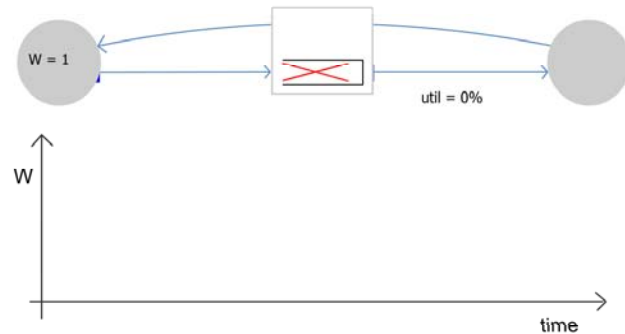
TCP Congestion Control



32

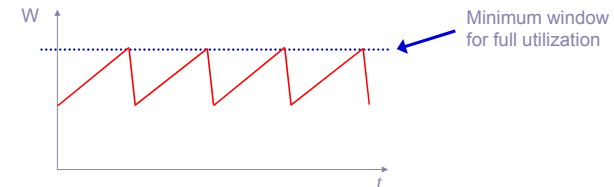
Single TCP Flow

Router *without* buffers



33

Summary Unbuffered Link



- The router can't fully utilize the link
 - If the window is too small, link is not full
 - If the link is full, next window increase causes drop
 - With no buffer it still achieves 75% utilization

34

TCP Performance

- In the real world, router queues play important role
 - Window is proportional to rate * RTT
 - But, RTT changes as well the window
 - Window to fill links = propagation RTT * bottleneck bandwidth
 - If window is larger, packets sit in queue on bottleneck link

35

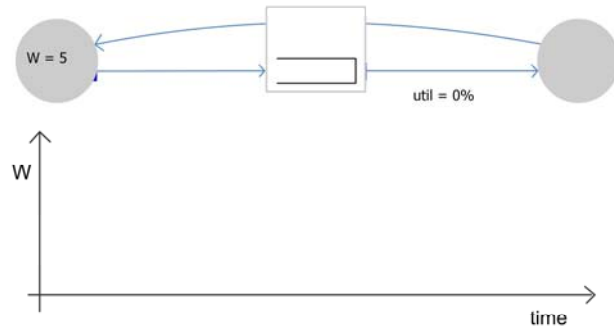
TCP Performance

- If we have a large router queue → can get 100% utilization
 - But, router queues can cause large delays
- How big does the queue need to be?
 - Windows vary from $W \rightarrow W/2$
 - Must make sure that link is always full
 - $W/2 > RTT * BW$
 - $W = RTT * BW + Qsize$
 - Therefore, $Qsize > RTT * BW$
 - Ensures 100% utilization
 - Delay?
 - Varies between RTT and $2 * RTT$

36

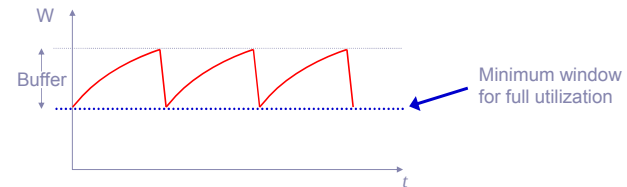
Single TCP Flow

Router with large enough buffers for full link utilization



37

Summary Buffered Link



- With sufficient buffering we achieve full link utilization
 - The window is always above the critical threshold
 - Buffer absorbs changes in window size
 - Buffer Size = Height of TCP Sawtooth
 - Minimum buffer size needed is $2T \cdot C$
 - This is the origin of the rule-of-thumb

38

Overview

- TCP congestion control
- TFRC
- TCP and queues
- Queuing disciplines
- RED

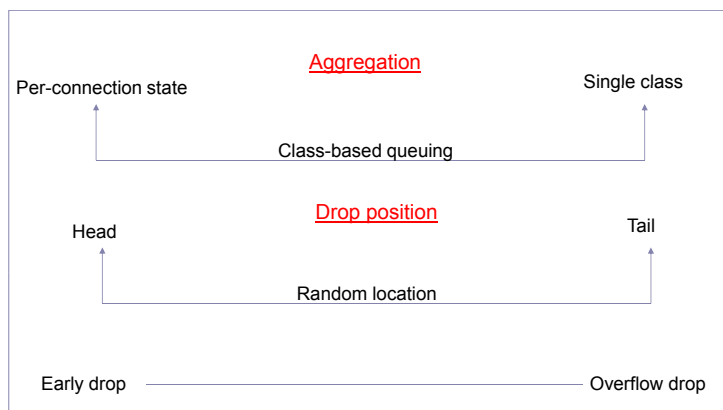
39

Queuing Disciplines

- Each router **must** implement some queuing discipline
- Queuing allocates both bandwidth and buffer space:
 - Bandwidth: which packet to serve (transmit) next
 - Buffer space: which packet to drop next (when required)
- Queuing also affects latency

40

Packet Drop Dimensions



41

Typical Internet Queuing

- FIFO + drop-tail
 - Simplest choice
 - Used widely in the Internet
- FIFO (first-in-first-out)
 - Implies single class of traffic
- Drop-tail
 - Arriving packets get dropped when queue is full regardless of flow or importance
- Important distinction:
 - FIFO: scheduling discipline
 - Drop-tail: drop policy

42

FIFO + Drop-tail Problems

- Leaves responsibility of congestion control to edges (e.g., TCP)
- Does not separate between different flows
- No policing: send more packets → get more service
- Synchronization: end hosts react to same events

43

Active Queue Management

- Design active router queue management to aid congestion control
- Why?
 - Routers can distinguish between propagation and persistent queuing delays
 - Routers can decide on transient congestion, based on workload

44

Active Queue Designs



- Modify both router and hosts
 - DECbit – congestion bit in packet header
- Modify router, hosts use TCP
 - Fair queuing
 - Per-connection buffer allocation
 - RED (Random Early Detection)
 - Drop packet or set bit in packet header as soon as congestion is starting

45

Overview



- TCP congestion control
- TFRC
- TCP and queues
- Queuing disciplines
- RED

46

Internet Problems



- Full queues
 - Routers are forced to have large queues to maintain high utilizations
 - TCP detects congestion from loss
 - Forces network to have long standing queues in steady-state
- Lock-out problem
 - Drop-tail routers treat bursty traffic poorly
 - Traffic gets synchronized easily → allows a few flows to monopolize the queue space

47

Design Objectives



- Keep throughput high and delay low
- Accommodate bursts
- Queue size should reflect ability to accept bursts rather than steady-state queuing
- Improve TCP performance with minimal hardware changes

48

Lock-out Problem



- Random drop
 - Packet arriving when queue is full causes some random packet to be dropped
- Drop front
 - On full queue, drop packet at head of queue
- Random drop and drop front solve the lock-out problem but not the full-queues problem

49

Full Queues Problem



- Drop packets before queue becomes full (early drop)
- Intuition: notify senders of incipient congestion
 - Example: early random drop (ERD):
 - If $qlen > \text{drop level}$, drop each new packet with fixed probability p
 - Does not control misbehaving users

50

Random Early Detection (RED)



- Detect incipient congestion, allow bursts
- Keep power (throughput/delay) high
 - Keep average queue size low
 - Assume hosts respond to lost packets
- Avoid window synchronization
 - Randomly mark packets
- Avoid bias against bursty traffic
- Some protection against ill-behaved users

51

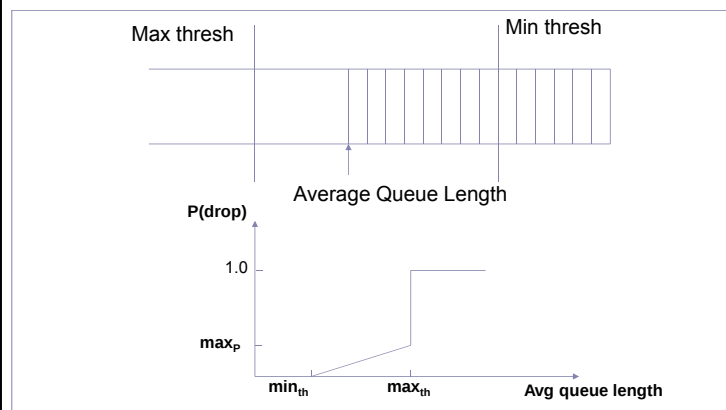
RED Algorithm



- Maintain running average of queue length
- If $avgq < min_{th}$ do nothing
 - Low queuing, send packets through
- If $avgq > max_{th}$, drop packet
 - Protection from misbehaving sources
- Else mark packet in a manner proportional to queue length
 - Notify sources of incipient congestion

52

RED Operation



53

RED Algorithm

- Maintain running average of queue length
 - Byte mode vs. packet mode – why?
- For each packet arrival
 - Calculate average queue size (avg)
 - If $\text{min}_{th} \leq \text{avgq} < \text{max}_{th}$
 - Calculate probability P_a
 - With probability P_a
 - Mark the arriving packet
 - Else if $\text{max}_{th} \leq \text{avg}$
 - Mark the arriving packet

54

Queue Estimation

- Standard EWMA: $\text{avgq} = (1-w_q) \text{avgq} + w_q \text{qlen}$
 - Special fix for idle periods – why?
- Upper bound on w_q depends on min_{th}
 - Want to ignore transient congestion
 - Can calculate the queue average if a burst arrives
 - Set w_q such that certain burst size does not exceed min_{th}
- Lower bound on w_q to detect congestion relatively quickly
- Typical $w_q = 0.002$

55

Thresholds

- min_{th} determined by the utilization requirement
 - Tradeoff between queuing delay and utilization
- Relationship between max_{th} and min_{th}
 - Want to ensure that feedback has enough time to make difference in load
 - Depends on average queue increase in one RTT
 - Paper suggest ratio of 2
 - Current rule of thumb is factor of 3

56

Packet Marking



- $\max_p$ is reflective of typical loss rates
- Paper uses 0.02
 - 0.1 is more realistic value
- If network needs marking of 20-30% then need to buy a better link!
- Gentle variant of RED (recommended)
 - Vary drop rate from $\max_p$ to 1 as the avgq varies from $\max_{th}$ to $2 * \max_{th}$
 - More robust to setting of $\max_{th}$ and $\max_p$

57

Coming Up



- Wednesday lecture: Fair Queuing
 - Read WFQ paper
 - First two sections of XCP paper
 - Presentation by Nicolas Feltman
- Friday lecture: Multimedia
 - Will fix readings

58