S10: Transmission Control Protocol

TCP

Instructor:
Behnam Momeni

Fall 2018
Congestion Control

• Congestion Control

• Assigned Reading
  • [Jacobson and Karels] Congestion Avoidance and Control
  • [TFRC] Equation-Based Congestion Control for Unicast Applications (sections 1-3)
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
  • Etc..
Where to Prevent Collapse?

• Can end hosts prevent problem?
  • Yes, but must trust end hosts to do right thing
  • E.g., sending host must adjust amount of data it puts in the network based on detected congestion

• Can routers prevent collapse?
  • No, not all forms of collapse
  • Doesn’t mean they can’t help
    • Sending accurate congestion signals
    • Isolating well-behaved from ill-behaved sources
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
Two broad approaches towards congestion control:

End-to-end

- No explicit feedback from network
- Congestion inferred from end-system observed loss, delay
- Approach taken by TCP

Network-assisted

- Routers provide feedback to end systems
  - Explicit rate sender should send at
  - Single bit indicating congestion (SNA, DEC bit, TCP/IP ECN, ATM)
- 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 control basics
• TCP congestion control
• TFRC
• TCP and queues
• Other TCPs?
Objectives

- Simple router behavior
- Distributedness
- Efficiency: $X_{knee} = \sum x_i(t)$
- Fairness: $(\sum x_i)^2/n(\sum x_i^2)$
- Power: $(\text{throughput}^\alpha/\text{delay})$
- Convergence: control system must be stable
Basic Control Model

• Let’s assume window-based control
• Reduce window when congestion is perceived
  • How is congestion signaled?
    • Either mark or drop packets
  • When is a router congested?
    • Drop tail queues – when queue is full
    • Average queue length – at some threshold
• Increase window 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?

![Diagram showing Phase plots with axes for User 1's Allocation $x_1$ and User 2's Allocation $x_2$, lines for Efficiency and Fairness, and points for Optimal, Underutilization, and Overload.]
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
Multiplicative Increase/Decrease

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

User 1's Allocation $x_1$

User 2's Allocation $x_2$

Efficiency Line

Fairness Line

$X^H$
Distributed Convergence to Efficiency

\[ x^H \]

User 1’s Allocation \( x_1 \)

User 2’s Allocation \( x_2 \)

Efficiency Line

Fairness Line

\( a=0 \)

\( b=1 \)
Convergence to Fairness

\[ \text{User 1's Allocation } x_1 \]

\[ \text{User 2's Allocation } x_2 \]

\[ x^H \]

\[ x^{H'} \]

Fairness Line

Efficiency Line
Convergence to Efficiency & Fairness

User 1's Allocation $x_1$

Efficiency Line

User 2's Allocation $x_2$

Fairness Line

$x^H$

$x^{H'}$
Increase Efficiency Line

User 1’s Allocation $x_1$

User 2’s Allocation $x_2$

Efficiency Line

Fairness Line

$x^L$
Constraints

• Distributed efficiency
  • I.e., $\Sigma \text{Window}(t+1) > \Sigma \text{Window}(t)$ during increase
    • $a_i > 0$ & $b_i \geq 1$
    • Similarly, $a_d < 0$ & $b_d \leq 1$

• Must never decrease fairness
  • $a$ & $b$’s must be $\geq 0$
  • $a_i/b_i > 0$ and $a_d/b_d \geq 0$

• Full constraints
  • $a_d = 0$, $0 \leq b_d < 1$, $a_i > 0$ and $b_i \geq 1$
What is the Right Choice?

- Constraints limit us to AIMD
  - Can have multiplicative term in increase (MAIMD)
  - AIMD
Questions

• Fairness – why not support skew → AIMD/GAIMD analysis
• More bits of feedback → DECbit, XCP
• Other types of feedback (e.g. delay)
• Guess # of users → hard in async system, look at loss rate?
• Stateless vs. stateful design
• Wired vs. wireless
• Non-linear controls → Bionomial
Outline

- Congestion control basics
- TCP congestion control
- TFRC
- TCP and queues
- Other TCPs?
TCP Congestion Control

- Motivated by ARPANET congestion collapse
- Underlying design principle: packet conservation
  - At equilibrium, inject packet into network only when one is removed
  - Basis for stability of physical systems
- Why was this not working?
  - Connection doesn’t reach equilibrium
  - Spurious retransmissions
  - Resource limitations prevent equilibrium
TCP Congestion Control - Solutions

• Reaching equilibrium
  • Slow start
• Eliminates spurious retransmissions
  • Accurate RTO estimation
  • Fast retransmit
• Adapting to resource availability
  • Congestion avoidance
TCP Congestion Control

• Changes to TCP motivated by ARPANET congestion collapse

• Basic principles
  • AIMD
  • Packet conservation
  • Reaching steady state quickly
  • ACK clocking
• 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
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”
    • (last sent – last acked) < congestion window

• Window limited by both congestion and buffering
  • Sender’s maximum window = Min (advertised window, cwnd)
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 $(1 \text{ packet})/\text{cwnd}$
    • What is 1 packet? $\rightarrow$ 1 MSS worth of bytes
    • After cwnd packets have passed by $\rightarrow$ approximately increase of 1 MSS

• Implements AIMD
Congestion Avoidance Sequence Plot
Congestion Avoidance Behavior

Packet loss + Timeout
Cut Congestion Window and Rate
Grabbing back Bandwidth

Congestion Window

Time
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 → 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)
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
Aside: Packet Pair

- What would happen if a source transmitted a pair of packets back-to-back?

- FIFO scheduling
  - Unlikely that another flows packet will get inserted in-between
  - Packets sent back-to-back are likely to be queued/forwarded back-to-back
  - Spacing will reflect link bandwidth

- Fair queuing
  - Router alternates between different flows
  - Bottleneck router will separate packet pair at exactly fair share rate

- Basis for many measurement techniques
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 cellular (100s of Kbps or less) and for datacenter links (40 Gbps and growing)

• Quick initial phase to help get up to speed (slow start)
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
Slow Start Example

One RTT

0R

1

One pkt time

1R

1

2

3

2R

2

3

4

6

5

7

3R

4

5

6

7

8

10

12

14

9

11

13

15
Slow Start Sequence Plot

Sequence No

Packets

Acks

Time
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
TCP Saw Tooth Behavior

- **Time**
- **Congestion**
- **Window**

**Initial Slowstart**

**Slowstart to pace packets**

**Fast Retransmit and Recovery**

Timeouts may still occur
Questions

• Current loss rates vs. 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 → BBR

• What about non-linear controls?
  • Binomial congestion control
Outline

- Congestion control basics
- TCP congestion control
- TFRC
- TCP and queues
- Other TCPs?
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
• 2010’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?
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?
TCP Friendly Rate Control (TFRC)

• Equation 1 – real TCP response

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

• 1\textsuperscript{st} term corresponds to simple derivation
• 2\textsuperscript{nd} term corresponds to more complicated timeout behavior
  • Is critical in situations with > 5% loss rates \( \rightarrow \) where timeouts occur frequently

• Key parameters
  • RTO
  • RTT
  • Loss rate
Outline

• Congestion control basics
• TCP congestion control
• TFRC
• TCP and queues
• Other TCPs?
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!
TCP Congestion Control

Only $W$ packets may be outstanding

**Rule for adjusting $W$**
- If an ACK is received: $W \leftarrow W + 1/W$
- If a packet is lost: $W \leftarrow W/2$

![Diagram showing source, destination, and window size](image)

Window size

$W_{\text{max}}$

$W_{\text{max}}/2$

$t$
Single TCP Flow

Router without buffers

$W = 1$

util = 0%
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
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
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
    • $W/2 > RTT \times BW$
    • $W = RTT \times BW + Qsize$
    • Therefore, $Qsize > RTT \times BW$
  • Ensures 100% utilization
• Delay?
  • Varies between RTT and $2 \times RTT$
Single TCP Flow
Router with large enough buffers for full link utilization
• 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*C
  • This is the origin of the rule-of-thumb
Outline

- Congestion control basics
- TCP congestion control
- TFRC
- TCP and queues
- Other TCPs?
Issues/Solutions

• Long delay-BW product
  • TCP slow to ramp up
  • Low loss rates cause underutilization
  • BIC/CUBIC – now default in Linux

• Can we train to operating conditions
  • RemyCC – simulation-based “trained” congestion control

• Bufferbloat

• What about router help…
CUBIC

- Multiplicative decrease after loss
- $W_{\text{max}}$ is window size before last loss

$$W_{cubic} = C(T - K)^3 + W_{\text{max}}$$

$C$ is a scaling constant, and $K = \sqrt[3]{\frac{W_{\text{max}}\beta}{C}}$
Next Lecture

• Router-Based Resource Allocation

• Reading:
  • Analysis and Simulation of a Fair Queueing Algorithm (read fully)
  • Congestion Control for High Delay-Bandwidth Product Networks (read intro)