SCOM/ SRSI
Congestion Control in Best-Effort Networks

Ana Aguiar

DEEC, FEUP

2018-19

Contents

- TCP Congestion Control
  - Congestion avoidance
  - Slow Start
  - Fast Retransmit
- Active queue management
  - DECbit
  - RED
  - ECN

Best Effort Service Model

- IP networks typically offer same treatment of all packets and no service guarantees
  - Packets are forwarded if and when possible
  - Packets are treated independently
    - No concept of flow
- Matters pertaining to flows are handled by hosts at higher layers
  - TCP, RTP/ RTCP, …

- Queueing and scheduling disciplines do not avoid congestion
- In a best-effort network congestion is dealt with end-to-end
- Congestion avoidance is not the same as routing around congestion

Dealing with Network Congestion

- TCP Congestion control
  - Host based
  - Avoiding, detecting and reacting to congestion

- Congestion avoidance = preventing congestion
  - Host based: hosts predict congestion based on RTT observation
  - Active Queue Management is host and router based: routers actively signal expected congestion

TCP Congestion Control

- End-to-end mechanism that enables each source to adapt send rate according to available bandwidth
  - Deals with congestion and variable bandwidth
  - Window-based and feedback-based

- Different from flow control!

TCP Congestion Control

- Goal: Use the available bandwidth as much as possible without causing buffer overflow at the routers and providing fair allocation in a distributed manner
  – Fair means equal in best effort networks

- Three mechanisms control the congestion window
  – Congestion Avoidance (cwin >= ssthresh)
  – Slow start (cwin < ssthresh)
  – Fast retransmit/ fast recovery

TCP Congestion Control

- End-to-end mechanism that enables each source to adapt send rate according to available bandwidth
  - Deals with congestion and variable bandwidth
  - Window- and feedback-based
  - Provides fair resource allocation across flows
- Different from flow control
  - Flow control => avoid overloading receiver
  - Congestion control => avoid overloading the network

TCP Congestion Control

- Change calculation of effective window size to accommodate congestion control window

MaxWindow = Min(cwindow, AdvertisedWindow)

cwindow: Congestion window

AdvertisedWindow: flow control window advertised by receiver

EffectiveWindow =

MaxWindow – (LastByteSent – LastByteACKed)

TCP Congestion Window

- How to set the size of the congestion window?
  - According to congestion perceived by endpoint

- cwindow is adapted during a connection
  - Increased when congestion level decreases
  - Decreased when congestion level increases

- Congestion level estimated based on packets that are not delivered
  - Assumes that packet losses are caused by congestion

TCP Congestion Control (Recall)

- Goal: Use the available bandwidth as much as possible without causing buffer overflow at the routers and providing fair allocation in a distributed manner

- Three mechanisms control the congestion window
  - Congestion Avoidance (cwin >= ssthresh)
  - Slow start (cwin < ssthresh)
  - Fast retransmit/ fast recovery

TCP Congestion Avoidance

- Distributed mechanism to adapt congestion window to available bandwidth
  - Steady-state behaviour
  - Additive increase/ multiplicative decrease (AIMD)
- Successful packet => no congestion: increase congestion window
  - cwindow += MSS x (MSS/cwindow)
    - MSS : Maximum Segment Size
  - Increment should not exceed 1 MSS per RTT
- Lost packet => congestion: decrease congestion window

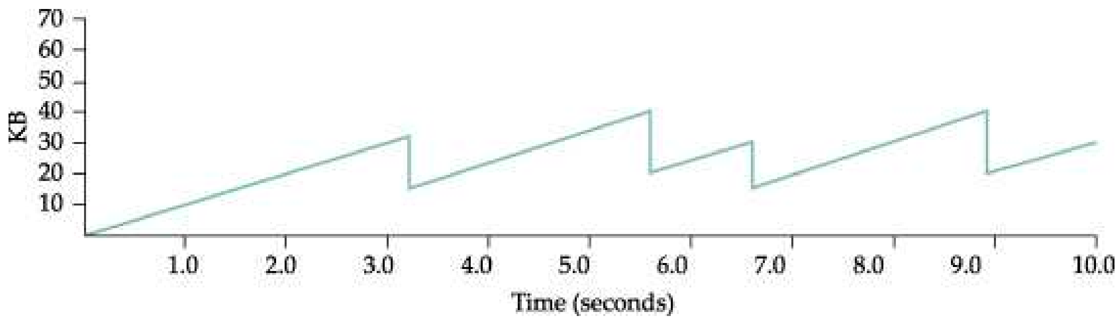
TCP Additive Increase/ Multiplicative Decrease

- Why decrease faster than increase?
  - Consequences of congestion are more severe than consequences of not fully utilising link
  - Multiplicative decrease guarantees stability, i.e. guarantees that queues will reduce size in acceptable time
  - Converges to fairness

D.M. Chiu and R. Jain. 1989. Computer Networks and ISDN Systems. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. Vol. 17, Nr. 1 (June 1989), 1-14. DOI=10.1016/0169-7552(89)90019-6

TCP AIMD

- Evolution of CongestionWindow with time for a connection in steady-state follows a sawtooth pattern
- And so does the instantaneous throughput

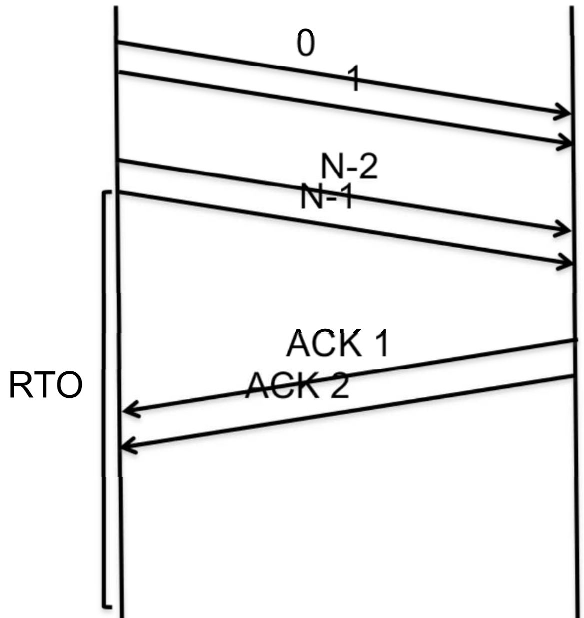


- Implicit fairness among TCP flows

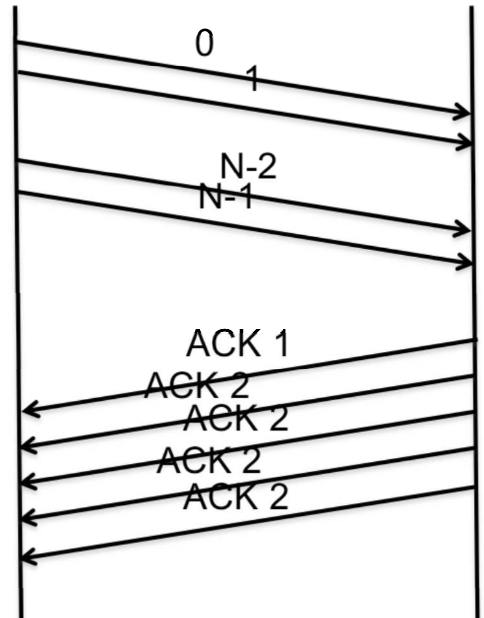
Image in “Computer Networks: a Systems Approach”, L. Peterson, B. Davie

TCP: Inferring Packet Losses

- Retransmission timeout
  - No communication indicates heavy congestion



- Duplicate acknowledgements
  - Ongoing communication indicates light congestion



TCP Congestion Avoidance

- Upon a packet loss detected by retransmission timeout (RTO)
  - cwindow = 1*MSS
  - ssthreshold = max(FlightSize/2, 2*MSS)
    - FlightSize: Data in transit in the network
    - FlightSize = LastByteSent – LastByteACKed
  - Use Slow Start algorithm until cwindow >= ssthreshold

TCP Fast Retransmit/ Fast Recovery

- Round Trip Time (RTT) calculation is coarse
- When a packet is lost, waiting for timeout before retransmission can be very ineffective

- When a single packet is lost, it should not be interpreted as congestion, but as a loss

TCP Fast Retransmit/ Fast Recovery

- Duplicate Acknowledgements
  - Packet loss followed by successful receptions is seen at the sender as repeated ACK
- Fast Retransmit
  - Trigger retransmission after three duplicate ACKs
  - Earlier reaction than retransmission timeout
  - Use Fast Recovery algorithm

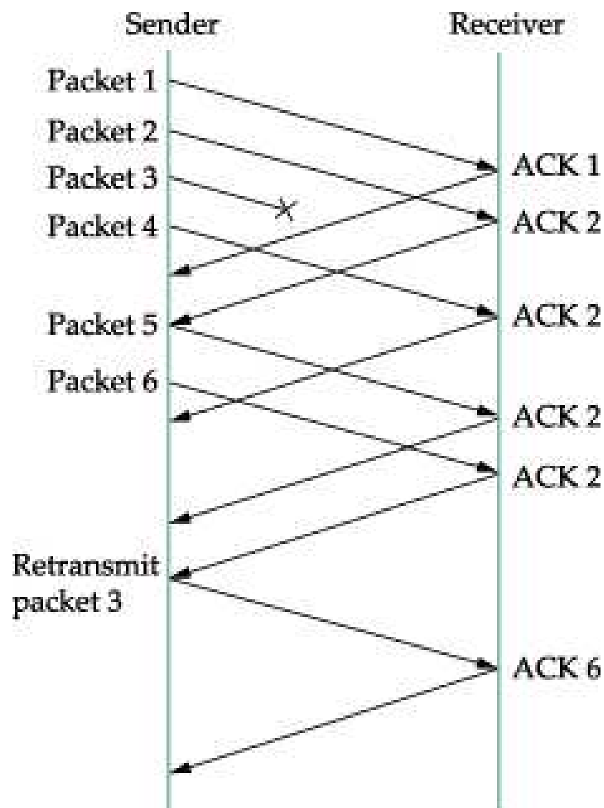


Image in “Computer Networks: a Systems Approach”, L. Peterson, B. Davie

TCP Fast Retransmit/ Fast Recovery

- Fast Recovery
  - ssthresh = cwindow/2
  - Retransmit unacknowledged data
  - Inflate cwindow: cwindow = ssthresh + 3*MSS
  - For each duplicate ack cwindow += MSS
  - Transmit data if EffectiveWindow allows
  - Deflate window to cwindow = ssthresh when missing data is acknowledged (first non-duplicate ack arrives)

TCP Slow Start

- Mechanism to discover available bandwidth during initial phase of a connection
  - Goal is not to flood the network with a burst of packets equivalent to a full transmission window
  - Q: Does this enable efficient network utilisation?
    - Think about short flows, e.g. for a simple web page

- Also used upon a retransmission timeout
    - ssthresh = (cwindow before loss)/2
    - See slide 12

TCP Slow Start

- cwindow = 1*MSS
- For every ACK received

  cwindow *= 2

- Congestion window grows exponentially until

  cwindow >= ssthresh

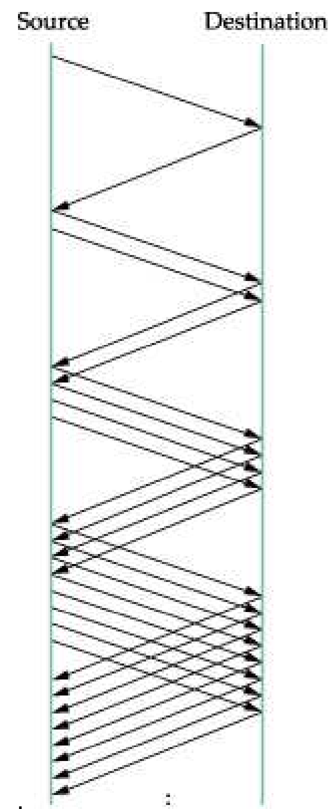


Image in “Computer Networks: a Systems Approach”, L. Peterson, B. Davie

Dealing with Congestion

- TCP is the widespread on the Internet, but alternative solutions exist

- Congestion control: reacting to congestion
  - Host based or router based
  - Window or rate based
  - After detecting congestion
- Congestion avoidance: preventing congestion
  - Only host based: hosts predict congestion based on RTT observation
  - Active Queue Management is host and router based: routers actively signal expected congestion

Active Queue Management

- Mechanisms that detect congestion in advance and inform hosts to reduce send rate
- Congestion avoidance mechanisms that involve routers as well as hosts

- DECbit
- Random Early Detection (RED)
- Explicit Congestion Notification (ECN)

DECbit

- Bit in packet header can be set by routers with queues larger than threshold
- Receiving host will signal it back to sender
- Sender has a congestion window which is managed according to additive increase/ multiplicative decrease
  - Increase window by 1 when less than 50% packets within the window were dropped
  - Decrease window to 87.5% if more than 50% packets within the window are dropped

DECbit: Motivations

- Processing at routers must be simple
  - To not impact forwarding performance
  - To be feasible in hardware
  - To be error/bug resilient
  - Changing a bit is easily done in hardware
- Routers can only talk to receiver
  - That is where the packet being forwarded is going
  - Receiver must signal congestion back to sender

Random Early Detection (RED)

- Designed to be used with TCP
- Routers drop packets before congestion occurs to signal congestion to the source
- Source will react by activating congestion control mechanisms, reducing their send rate

Random Early Detection (RED)

- Router drops packets when average queue length exceeds a drop threshold
  - Queue length monitored using a moving average

  AvgQLen = (1-α) x AvgQLen + α x SampleQLen

  0 < α < 1
- Senders react to drops not faster than one RTT (≈ 100ms)

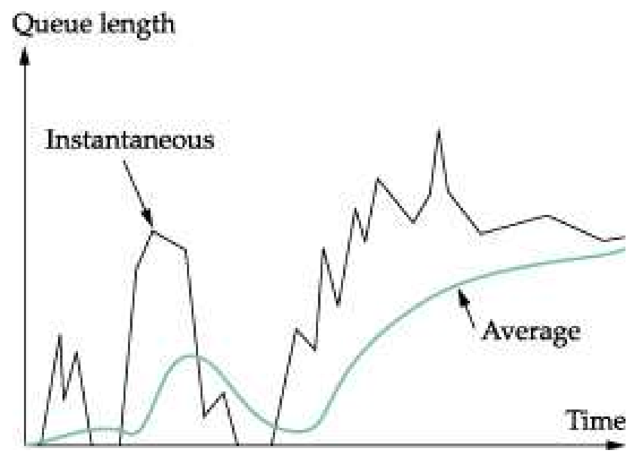
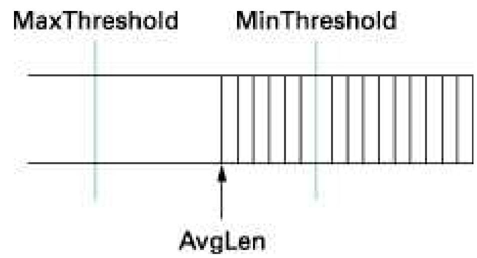
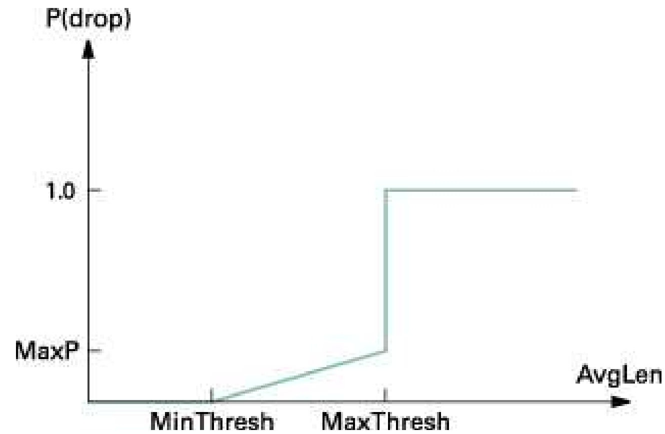


Image in “Computer Networks: a Systems Approach”, L. Peterson, B. Davie

Random Early Detection (RED)

- Router drops packets with probability P if AvgQLen is between MinThreshold and MaxThreshold
  - Algorithm avoids burst drops by calculating P as an increasing function of enqueued packets since the last drop
- Router drops all arriving packets if AvgQLen is larger than MaxThreshold





Images in “Computer Networks: a Systems Approach”, L. Peterson, B. Davie

Random Early Detection (RED)

- By randomly dropping packets, flows that use a larger bandwidth share have larger probability of seeing their packets dropped
  - Simply because they send more packets
- So, RED is inherently fair in this sense

- Still, it is a best-effort approach, which treats all flows equally

Explicit Congestion Notification (ECN)

- Alternative to RED which uses IP header bits to signal congestion instead of dropping packets
  - Router sets congestion bit
  - Congestion is signalled to receiver, but data arrives
  - Receiver sets congestion bit in ACK
  - Sender behaves as if the packet was dropped
- Does not cause data loss
- Uses 2 bits in TOS field of IP header
  - 1 for the sender to signal that it supports ECN
  - 1 for a router to signal congestion

Host-Based Congestion Avoidance

- Senders use observable parameters to detect that congestion will happen soon
  - Increasing RTT of successive packets
  - Combination of RTT and CongestionWindow variations
  - Flattened sending rate or throughput rate, e. g. estimated as bytes in the network / RTT

- Senders adapt send rate to detected congestion

Reading

- These topics are covered in
  - chapter 6 of the book “Computer Networks, A Systems Approach”, L. Peterson and B. Davie
  - 3.6 and 3.7 of the book “Computer Networking, A Top Down Approach”, J. Kurose and K. Ross

QUALITY OF SERVICE MECHANISMS