Van Jacobson’s Congestion Avoidance and Control

presented by Leon Poutievski
Problem: Congestion

- Congestion
  - Load exceeds the capacity of the network
  - Overflow at router queues
- ATM: channel is reserved during the connection setup
- IP: if queue is full, packets are dropped
Problem: Congestion

• Problem
  – Wasted bandwidth (retransmission required)
  – Unpredictable delay
• October 1986: first “congestion collapse”
  – Caused by TCP retransmissions
• Scenario:
  TCP sends a window's worth of data
  Some of it gets lost; the rest sits in queue
  Sender times out, retransmits everything
  Result: Multiple copies of same data in queue!
(REPEAT)
Problem: Congestion

• **Crux of the problem:**
  – When the router is congested (i.e. its queue is full), **retransmitting data all at once is exactly the wrong thing to do**
  – But this is what TCP spec said to do
    • Go-back-N protocol
  – **Send window is the only constraint on sending rate**

  \[
  \text{sending rate} = \frac{\text{send window}}{\text{RTT}}
  \]

Extremely important concept!
Problem: Congestion

• Thus: send window controls how fast sender x-mits
• Original TCP (flow control):
  \[\text{send window} := \text{rcv window}\]
• Conclusion: sender should also consider the network capacity
• Idea: reduce send window to reduce rate under congestion
• Solution: add congestion window
  \[\text{send window} := \min (\text{rcv window}, \text{congestion window})\]
"Conservation of packets" principle

• Equilibrium = “stable, full window of data in transit”

• “Conservative” packet flow
  – New is not put into network until an old leaves

• Possible problems with packet conservation
  1. Didn’t get to equilibrium
  2. Sender ejects new packets before old has exited
  3. Equilibrium can’t be reached
Getting to Equilibrium: Slow-start

- **Self-clocking system**
  - Automatically adjusts to BW & delay variations
  - Sender uses ACK as a ‘clock’
  - Hard to start

- Packet size (area) = BW x time
Slow-start algorithm

- Add "congestion window" to algorithm
  - send window := min (rcv window, cwnd);
- When starting (restarting)
  cwnd := 1 packet;
- On ACK for new data
  cwnd += 1 packet;
Slow-start, analysis

- “Slow” is misnomer:
  - rate grows exponentially
  - On each ack 2 pkts are sent
    - one for the packet acked (left n/w)
    - one to increase the window
  - So SWS doubles every RTT

- It takes $R \log_2 W$
  - $R$: round trip time
  - $W$: window size in packets
Slow-start, analysis

Without Slow-start

With Slow-start
Conservation at Equilibrium (round-trip timing)

• Need good estimate of
  – RTT mean estimate
  – RTT variation estimate
    • It increases quickly with load

  for retransmit timeout interval (rto)
  (described later)

• Exponential backoff after retransmit
  – Reasoning: linear systems
Congestion avoidance
(adapting to the path)

• Assumption:
  – Losses are due to congestion
    • Losses due damage are rare (<< 1%)

• Strategy
  – If congestion, network must signal to endpoints so that endpoints decrease utilization
  – If no congestion and thus no signals, endpoints increase network utilization
Congestion avoidance
Multiplicative decrease

• $L_i = N + \lambda L_{i-1}$,
  $L_i$ – load at interval $i$, average queue length
  $N$ – constant

• $\lambda \approx 0$, no congestion

• $\lambda > 1$, congestion
  – $L_n = \lambda^n L_0$, grows exponentially

• System stabilizes only if traffic is reduced as quickly as queues are growing
Congestion avoidance: Multiplicative decrease

• On congestion: \( W_i = d W_{i-1} \) \((d < 1)\)
• Take \( d = 0.5 \), \( W_i = W_{i-1} / 2 \)
  – Motivation: give up \( \frac{1}{2} \) BW for a new connection, everybody adapts to new situation

• Congestion detection
  – Did not receive any ack (timeout)
  – Received 3 duplicate acks
Congestion avoidance
Additive increase

• Also known as "Additive Increase, Multiplicative Decrease" (AIMD)
  – Refers to change in cwnd per RTT

• If no congestion detected:
  \[ W_i = W_{i-1} + u \quad (u << W_{max}) \]
  – Take \( u = 1 \), so \( W_i = W_{i-1} + 1 \)
Congestion avoidance Algorithm

• On timeout:
  \[ cwnd = \frac{\text{send window}}{2}; \] //multiplicative decrease

• On ack:
  \[ cwnd += \frac{1}{cwnd}; \] //additive increase

• send window := min (rcv window, cwnd);
Congestion avoidance, Analysis (sequence numbers)

No congestion avoidance

With congestion avoidance
Congestion avoidance, Analysis (relative BW)

- No congestion avoidance
- With congestion avoidance

Total BW

Delivered BW
Combined algorithm  
(slow start + congestion avoidance)

- \texttt{cwnd} = congestion window
- \texttt{ssthresh} = threshold, to switch between algs
- \texttt{send window} = \texttt{min (rcv window, cwnd)};
- **On timeout:**
  \begin{verbatim}
  ssthresh = \texttt{send window} / 2; //mult. decrease, cong. avoid.
  \texttt{cwnd} = 1;                  //start slow start
  \end{verbatim}
- **On ack on new data:**
  \begin{verbatim}
  if (\texttt{cwnd} < \texttt{ssthresh})
    \texttt{cwnd} += 1 //slow start
  else
    \texttt{cwnd} += 1/\texttt{cwnd}; //congestion avoidance
  \end{verbatim}
Conservation at Equilibrium

• Need good
  – RTT mean estimate (R)
  – RTT variation estimate (b)
for retransmit timeout interval (rto)

• RFC793: use low pass filter
  \[ R = aR + (1-a)M, \text{ } M \text{ – new measurement} \]

• RFC813 suggests: \( rto = bR = 2R \)
RTT estimation, theory

- \( A := (1-g)A + gM, \quad 0 < g < 1 \)
  after rearranging:
- \( A := A + g(M-A) \)
  \( M-A = Er + Ee \), where
  - \( Er \) (random error): due to noise in measurement
    random kick, they will cancel out
  - \( Ee \) (estimation error): due to bad choice of \( A \)
    kick in the right direction
- \( A := A + gEr + gEe \), we want large \( g \) to get most of \( Ee \), but small \( g \) to reduce \( Er \)
- Usually take \( 0.1 \leq g \leq 0.2 \)
RTT estimation, practice

• Goal: estimate variance of \( M \)
• \( \sigma^2 = \sum |M-A|^2 \)
  – Squaring can cause overflow
  – Use mean deviation instead
  – \( \text{mdev} = \sum |M-A| \)
• \( \text{mdev}^2 = (\sum |M-A|)^2 \leq \sum |M-A|^2 = \sigma^2 \)
• For normal distribution:
  \( \text{mdev} = \sqrt{\pi/2} \) \( sdev \approx 1.25 \) \( sdev \)
RTT estimation, practice

- \( \text{Err} = M - A \)
- \( A := A + g \text{Err} \)
- \( D := D + g(|\text{Err}| - D) \)

- \( M -= (SA >> 3) \)
- \( SA += M \)
- \( \text{If } (M < 0) \)
  - \( M = -M \)
- \( M -= (SD >> 3) \)
- \( SD += M \)

- \( g = \frac{1}{2^n} \)
- \( SA = 2^nA \)
- \( SD = 2^nD \)
RTT estimation, final

- Err = M – A
- A := A + gErr
- D := D + g(|Err| - D)
- rto := A + 2D

- M -= (SA >> 3)
- SA += M;
- If (M < 0)
  - M = -M;
  - M -= (SD >> 2);
  - SD += M;
  - rto = ((SA >> 2) + SD) >> 1

- \( g_A = \frac{1}{2^3} \quad SA = 2^3A \)
- \( g_D = \frac{1}{2^2} \quad SD = 2^2D \)
Analysis

y – time from send till reception of ack by sender

Old TCP

New mean + variance