15-744: Computer Networking

L-9 Transport Alternatives

- TCP Vegas
- Alternative Congestion Control
- Header Compression
- Assigned reading
  - [BP95] TCP Vegas: End to End Congestion Avoidance on a Global Internet
  - [FHPW00] Equation-Based Congestion Control for Unicast Applications

Overview

- TCP Vegas
- TCP Modeling
- TFRC and Other Congestion Control

TCP Vegas Slow Start

- ssthresh estimation via packet pair
- Only increase every other RTT
  - Tests new window size before increasing
Packet Pair

• What would happen if a source transmitted a pair of packets back-to-back?
• Spacing of these packets would be determined by bottleneck link
  • Basis for ack clocking in TCP
• What type of bottleneck router behavior would affect this spacing
  • Queuing scheduling

Packet Pair

• 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

Packet Pair in Practice

• Most Internet routers are FIFO/Drop-Tail
• Easy to measure link bandwidths
  • Bprobe, pathchar, pchar, nettimer, etc.
• How can this be used?
  • NewReno and Vegas use it to initialize ssthresh
  • Prevents large overshoot of available bandwidth
  • Want a high estimate – otherwise will take a long time in linear growth to reach desired bandwidth

TCP Vegas Congestion Avoidance

• Only reduce cwnd if packet sent after last such action
  • Reaction per congestion episode not per loss
• Congestion avoidance vs. control
• Use change in observed end-to-end delay to detect onset of congestion
  • Compare expected to actual throughput
  • Expected = window size / round trip time
  • Actual = acks / round trip time
TCP Vegas

- If actual < expected < actual + α
  - Queues decreasing → increase rate
- If actual + α < expected < actual + β
  - Don’t do anything
- If expected > actual + β
  - Queues increasing → decrease rate before packet drop
- Thresholds of α and β correspond to how many packets Vegas is willing to have in queues

TCP Vegas

- Fine grain timers
  - Check RTO every time a dupack is received or for “partial ack”
  - If RTO expired, then re-xmit packet
  - Standard Reno only checks at 500ms
- Allows packets to be retransmitted earlier
  - Not the real source of performance gain
- Allows retransmission of packet that would have timed-out
  - Small windows/loss of most of window
  - Real source of performance gain
  - Shouldn’t comparison be against NewReno/SACK

TCP Vegas

- Flaws
  - Sensitivity to delay variation
  - Paper did not do great job of explaining where performance gains came from
  - Some ideas have been incorporated into more recent implementations
- Overall
  - Some very intriguing ideas
  - Controversies killed it

Overview

- TCP Vegas
- TCP Modeling
- TFRC and Other Congestion Control
TCP Modeling

• Given the congestion behavior of TCP can we predict what type of performance we should get?
• What are the important factors
  • Loss rate
    • Affects how often window is reduced
  • RTT
    • Affects increase rate and relates BW to window
  • RTO
    • Affects performance during loss recovery
  • MSS
    • Affects increase rate

Overall TCP Behavior

• Let’s concentrate on steady state behavior with no timeouts and perfect loss recovery
• Packets transferred = area under curve

Transmission Rate

• What is area under curve?
  • \( W = \text{pkts/RTT}, T = \text{RTTs} \)
  • \( A = \text{avg window} \times \text{time} = \frac{3}{4} W \times T \)
• What was bandwidth?
  • \( BW = \frac{A}{T} = \frac{3}{4} W \)
  • In packets per RTT
  • Need to convert to bytes per second
  • \( BW = \frac{3}{4} W \times \text{MSS} / \text{RTT} \)

Simple TCP Model

• Some additional assumptions
  • Fixed RTT
  • No delayed ACKs
• In steady state, TCP losses packet each time window reaches W packets
  • Window drops to W/2 packets
  • Each RTT window increases by 1 packet \( \rightarrow W/2 \)
  • RTT before next loss
Simple Loss Model

- What was the loss rate?
  - Packets transferred = \( \left( \frac{3}{4} \frac{W}{RTT} \right) \times \left( \frac{W}{2} \times \frac{RTT}{2} \right) = \frac{3W^2}{8} \)
  - 1 packet lost \( \rightarrow \) loss rate = \( p = \frac{8}{3W^2} \)

- \( W = \frac{8}{3p} \)

- \( BW = \frac{3\sqrt{8W^2} \cdot MSS}{\sqrt{3p} \cdot 2p} \)
  - \( BW = \frac{MSS \cdot RTT}{\sqrt{8} \cdot \sqrt{3p} \cdot \sqrt{2p}} \)

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?

Overview

- TCP Vegas
- TCP Modeling
- TFRC and Other Congestion Control

TCP Friendly Rate Control (TFRC)

- Equation 1 – real TCP response
  - 1st term corresponds to simple derivation
  - 2nd 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
**RTO Estimation**
- Not used to actually determine 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 \( \rightarrow \) 500ms typical, 100ms, 200ms, 1s also common
  - \( RTO = 4 \times RTT \) is close enough for reasonable operation

**RTT Estimation**
- \( \text{EWMA} \ (RTT_{n+1} = (1-\alpha)RTT_n + \alpha RTTSAMP) \)
- \( \alpha = ? \)
  - Small (.1) \( \rightarrow \) long oscillations due to overshooting link rate
  - Large (.5) \( \rightarrow \) short oscillations due to delay in feedback (1 RTT) and strong dependence on RTT
- Solution: use large \( \alpha \) in T rate calculation but use ratio of \( RTTSAMP^{1/2}/RTT^{1/2} \) for inter-packet spacing

**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

**Loss Estimation**
- Dynamic windows has many flaws
- Difficult to chose 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
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 * recvd)
  - Ends with first loss report → rate set to ½ current rate

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

Binomial Congestion Control
- In AIMD
  - Increase: $W_{n+1} = W_n + \alpha$
  - Decrease: $W_{n+1} = (1 - \beta) W_n$
- In Binomial
  - Increase: $W_{n+1} = W_n + \alpha/W_n^k$
  - Decrease: $W_{n+1} = W_n - \beta W_n^l$
  - $k=0$ & $l=1$ → AIMD
  - $l < 1$ results in less than multiplicative decrease
    - Good for multimedia applications

Binomial Congestion Control
- Rate ~ $1/\text{(loss rate)}^{1/(k+l+1)}$
  - If $k+l=1$ → rate ~ $1/p^{0.5}$
    - TCP friendly if $l = 1$
  - AIMD ($k=0$, $l=1$) is the most aggressive of this class
    - Good for applications that want to probe quickly and can use any available bandwidth
Overview

• TCP Vegas

• TCP Modeling

• TFRC and Other Congestion Control

• Changing Workloads

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

Short Transfers

• Fast retransmission needs at least a window of 4 packets
  • To detect reordering

• Should not be necessary if small outstanding number of packets
  • Adjust threshold to \( \min(3, \text{cwnd/outstanding}) \)

• Some paths have much more reordering than others
  • Adapt threshold to past reordering

• Allow new packets to be transmitted for first few dupacks
  • Will create new dupacks and force retransmission
  • Will not reduce goodput in situations of reordering
  • Follows packet conservation

Enhanced TCP Loss Recovery
## Enhanced TCP Loss Recovery

![Enhanced TCP Loss Recovery Diagram]

### Short Transfers

- Short transfer performance is limited by slow start → RTT
  - Start with a larger initial window
  - What is a safe value?
    - TCP already burst 3 packets into network during slow start
    - Large initial window = min (4*MSS, max (2*MSS, 4380 bytes)) [rfc2414]
  - Enables fast retransmission
  - Only used in initial slow start not in any subsequent slow start

## Well Behaved vs. Wild West

- How to ensure hosts/applications do proper congestion control?
- Who can we trust?
  - Only routers that we control
  - Can we ask routers to keep track of each flow
    - No, we must avoid introducing per flow state into routers
  - Active router mechanisms for control in later lectures

## Next Lecture: AQM

- Active Queue Management
- Assigned reading
  - [FJ93] Random Early Detection for Congestion Avoidance