Advanced Computer Networks

Congestion Control (2)

Jianping Pan
Summer 2007
Feedback on reading & presentation

• Be aware of deadlines

• Ideas, strengths and weaknesses
  – of the paper and related research work!
  – design paper vs performance analysis/improvement paper

• Paper presentation
  – present main ideas: e.g., schemes, analysis approaches
  – interact with the audience
  – adjust the presentation adaptively

• People from the same group
  – lead the discussion!
Review: TCP congestion control

• Design principle
  – packet conservation with ack self-clocking

• Congestion control algorithms
  – slow-start
  – congestion avoidance
  – timeout retransmission
  – fast retransmit
  – fast recovery
Discussion

• Critics on TCP congestion control
Network congestion

• What can endpoint observe?
  – longer round-trip time
    • extra queuing delay at routers
  – higher packet loss ratio
    • buffer overflow at routers
  – lower throughput
TCP Vegas [BOP94]

• More aggressive retransmission with fine-grained timer
  – Reno: 500 ms coarse-grained timer
    • also usually one timer for a window of packets
• More conservative congestion avoidance
  – Reno: cwnd increased by one MSS every RTT
    • or 0.5 MSS if delayed acknowledgment is used
• Slower than “slow-start”
  – Reno: cwnd doubled every RTT
• Changes only at TCP sender
Fine-grained timer

- Fine-grained timer for each packet
  - for a more accurate RTT calculation
  - sender: read and record system clock
    - e.g., utilize TCP timestamps option

- Retransmission triggers
  - check fine-grained timeout when receiving
    - duplicate acknowledgment
    - first or second acknowledgment after retransmission
  - also “fall back” to Reno timeout

- Only reduce cwnd for loss event at current rate

Q: delayed acknowledgment?
Proactive congestion control

• Related work
  – Wang and Crowcroft's DUAL
    • cwnd increases as Reno
    • every 2 RTTs, reduce cwnd by 1/8 if RTT>(RTTmin+RTTmax)/2
  – Jain's CARD
    • every two RTT, if Diff(win)*Diff(rtt) > 0, decease win by 1/8
    • otherwise, increase win by 1 MSS
    • oscillate between WINmin and WINmax
  – Wang and Crowcroft's Tri-S
    • increase 1 MSS every RTT
    • if the throughput improvement is less than a half of the initial segment, reduce cwnd by 1 MSS
Conservative congestion avoidance

• Vegas calculates
  – expected throughput: cwnd / baseRTT
  – actual throughput: cwnd / currentRTT
  – Diff = Expected – Actual > 0

• Window adjustment algorithm
  – two thresholds: a < b
  – if Diff < a, increase cwnd linearly
  – if Diff > b, decrease cwnd linearly
  – goal: a < Diff < b
    • try to probe for extra capacity

Q: how to determine baseRTT?
Q: how to determine a and b?
Slower than “slow-start”

- Vegas in “slow-start”
  - exponential cwnd increase in every other RTT
  - to allow the comparison
    - expected throughput
    - actual throughput
    - Diff = Expected – Actual > 0
  - c: “slow-start” threshold
  - if Diff > c, do congestion avoidance

Q: how to determine c?
Performance evaluation

• Simulation and experimentation
  – one-on-one with Reno
    • Reno is not adversely affected
      - with background traffic
        • considerable performance improvement over Reno
  • Implementation in x-kernel: extra features
    – e.g., reduction by 1/4, large initial win, burst limit, ...
Further discussion

- Critics on TCP Vegas
This lecture

• Delay-based congestion avoidance
  – TCP Vegas

• Explore further
  – http://netlab.caltech.edu/FAST/references.html#vegas
Next lectures

• TCP-friendly congestion control

• Explicit congestion control