TCP (Part II)

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

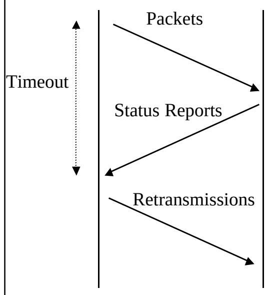
- TCP interactive data flow
- TCP bulk data flow
- TCP congestion control
- TCP timers
- TCP futures and performance

Ref: Chap 19-24; RFC 793, 1323, 2001, papers by Jacobson, Chiu/Jain, Karn/Partridge

Reliability *Models*

- Reliability => requires *redundancy* to recover from uncertain loss or other failure modes.
- Two types of redundancy:
  - Spatial redundancy: independent backup copies
    - Forward error correction (FEC) codes
    - Problem: requires huge *overhead*, since the FEC is also part of the packet(s) it cannot recover from erasure of all packets
  - Temporal redundancy: retransmit if packets lost/error
    - Lazy: trades off *response time* for reliability
    - Design of status reports and retransmission optimization (see next slide) important

Temporal Redundancy Model



- Sequence Numbers
- CRC or Checksum

- ACKs
- NAKs,
- SACKs
- Bitmaps

- Packets
- FEC information

Status Report Design

- *Cumulative* acks:
  - Robust to losses on the reverse channel
  - Can work with go-back-N retransmission
  - Cannot pinpoint *blocks* of data which are lost
    - The first lost packet can be pinpointed because the receiver would generate duplicate acks

Status Report Design (Continued)

- *Selective* acks: (SACKs)
  - For a byte-stream model like TCP, need to specify ranges of bytes received (requires large overhead)
  - SACK is a TCP option over-and-above the cumulative acks
- *Bitmaps: identify received and lost information*
  - Not efficient for TCP: a bit is needed for every byte!
- *NAKs* have same problems like SACKs and bitmaps, but also are not robust to reverse channel losses

Retransmission Optimization

- Default retransmission:
  - *Go-back-N:* I.e. retransmit the entire window.
  - Triggered by timeout or persistent loss in TCP
  - Not efficient if windows are large: high speed n/ws

Retransmission Optimization (Continued)

- *Selective* retransmission:
  - Retransmit *one* packet based upon *duplicate acks*
    - Recovers quickly from isolated loss, but not from burst loss
  - *TCP-SACK* is an enhancement which identifies a *block* of packets to be retransmitted.
  - Such retransmitted packets must finally be confirmed by acks since SACK is only an option and not reliable

TCP *Interactive* Data Flow

- Problems:
  - Overhead: 40 bytes header + 1 byte data
  - *Packets*: To *batch* or not to batch: response time important
- *Batching acknowledgements:*
  - Delay-ack timer: piggyback ack on reverse traffic if available
  - 200 ms timer (fig 19.3) if no reverse traffic

TCP Interactive Data Flow

- Batching data:
  - Nagle's algo: Don't send packet until next ack is received.
  - Developed because of congestion in WANs

TCP *Bulk* Data Flow

- *Sliding window:*
  - Send multiple packets while waiting for acks (fig 20.1) upto a limit (W)
  - Receiver need not ack every packet
  - Acks are cumulative.
  - Ack # = Largest consecutive sequence number received + 1
  - Two transfers of the data can have different dynamics (eg: fig 20.1 vs fig 20.2)

TCP Bulk Data Flow (Continued)

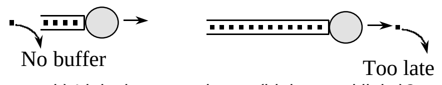
- *Receiver window* field:
  - Reduced if TCP receiver short on buffers
  - End-to-end flow control
  - Window update acks: receiver ready
  - Default buffer sizes: 4096 to 16384 bytes.
  - Ideal: window and receiver buffer = bandwidth-delay product

TCP Bulk Data Flow (Continued)

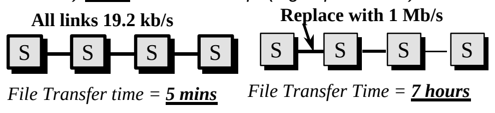
- TCP window terminology: figs 20.4, 20.5, 20.6
  - Right edge, Left edge, usable window
  - "closes" => left edge (snd_una) advances
  - "opens" => right edge advances (receiver buffer freed => receiver window increases)
  - "shrinks" => right edge moves to left (rare)

The Congestion Problem

- *Problem: demand outstrips available capacity …*
- Q: Will the "congestion" problem be solved when:
  - *a) Memory becomes cheap (infinite memory)?*

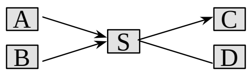


  - *b) Links become cheap (high speed links)?*



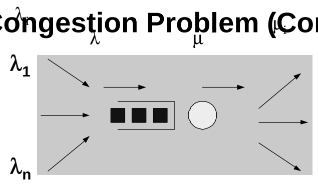
The Congestion Problem (Continued)

- c) *Processors* become cheap (fast routers switches)?



Scenario: All links 1 Gb/s. A & B send to C.

The Congestion Problem (Continued)



- If information about λ_i, λ and μ is known in a central location where control of λ_i can be effected with zero time delays,
  - the congestion problem is solved!

The Congestion Problem (Continued)

- Problems:
  - Incomplete information (eg: loss indications)
  - Distributed solution required
  - Congestion and control/measurement locations different
  - Time-varying, heterogeneous time-delay

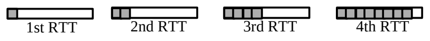
TCP Congestion Control

- *Window* flow control: avoid receiver overrun
- *Dynamic window congestion control:* avoid/control network overrun
  - Observation: Not a good idea to start with a large window and dump packets into network
  - Treat network like a black box and start from a window of 1 segment *("slow start")*

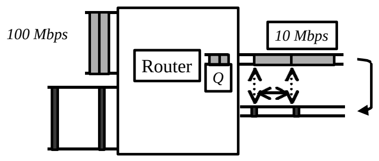
TCP Congestion Control (Continued)

- *Dynamic window congestion control:* avoid/control network overrun (Continued).
  - Increase window size exponentially *("exponential increase")* over successive RTTs => quickly grow to claim available capacity.
  - Technique: Every ack: increase *cwnd* (new window variable) by 1 segment.
  - Effective window = Min(*cwnd, Wrcvr*)

Dynamics

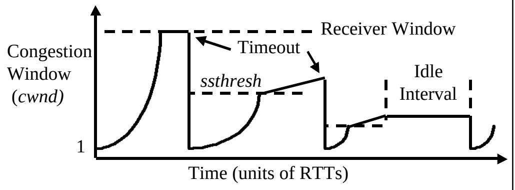


- Rate of acks = rate of packets at the bottleneck: *"Self-clocking"* property.



Congestion Detection

- *Packet loss as an indicator of congestion.*
  - Set slow start threshold *(ssthresh)* to min(*cwnd, Wrcvr)/2*
  - Retransmit pkt, set *cwnd* to 1 (reenter slow start)

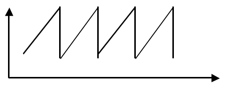


Congestion Avoidance

- Increment *cwnd* by 1 per ack until *ssthresh*
- Increment by 1/*cwnd* per ack afterwards *("Congestion avoidance" or "linear increase")*
- Idea: *ssthresh* estimates the bandwidth-delay product for the connection.
- Initialization: *ssthresh* = Receiver window or default 65535 bytes. Larger values thru options.
- If source is idle for a long time, cwnd is reset to one MSS.

Congestion Avoidance (Continued)

- Implications of using packet loss as congestion indicator
  - Late congestion detection if the buffer sizes larger
  - Higher speed links or large buffers => larger windows => higher probability of burst loss
  - Interactions with retransmission algorithm and timeouts



Congestion Avoidance (Continued)

- Implications of ack-clocking
  - More batching of acks => bursty traffic (harder to manage)
  - Less batching leads to a large fraction of Internet traffic being just acks (huge overhead)
- Additive Increase/Multiplicative Decrease Dynamics:
  - TCP approximates these dynamics

Timeout and RTT Estimation

- Timeout: for robust detection of packet loss
- Problem: How long should timeout be ?
  - Too long => underutilization; too short => wasteful retransmissions
  - Solution: <u>*adaptive timeout:*</u> *based on RTT*

Timeout and RTT Estimation (Continued)

- RTT estimation:
  - Early method: exponential averaging:
    - R ← α*R + (1 - α)*M { M =measured RTT}
    - **RTO = β*R** {β = delay variance factor}
    - Suggested values: α = 0.9, β = 2
    - Jacobson [1988]: this method has problems w/ large RTT fluctuations

RTT Estimation

- New method: Use mean & deviation of RTT
  - A = smoothed average RTT
  - D = smoothed mean deviation
  - Err = M - A { M = measured RTT}
  - A ← A + g*Err {g = gain = 0.125}
  - D ← D + h*(|Err| - D) {h = gain = 0.25}
  - **RTO = A + 4D**
  - Integer arithmetic used throughout. Complex initialization process ...

Timer Backoff/Karn's Algorithm

- Timer backoff: If timeout, RTO = 2*RTO {exponential backoff}
- Retransmission *ambiguity* problem:
  - During retransmission, it is unclear whether an ack refers to a packet or its retransmission. Problem for RTT estimation
  - Karn/Partridge: *don't update RTT estimators during retransmission.*
    - Restart RTO only after an ack received for a segment that is not retransmitted

TCP Performance Optimization

- SACK: selective acknowledgments: specifies blocks of packets received at destination.
- *Random early drop (RED)* scheme spreads the dropping of packets more uniformly and reduces average queue length and packet loss rate.
- *Scheduling* mechanisms protect well-behaved flows from rogue flows.
- *Explicit Congestion Notification (ECN):* routers use a explicit bit-indication for congestion instead of loss indications.

Congestion Control Summary

- *Sliding* window limited by receiver window.
- Dynamic <u>***windows***</u>: slow start (exponential rise), congestion avoidance (linear rise), multiplicative decrease.
- Adaptive <u>***timeout***</u>: need mean RTT & deviation
- Timer backoff and Karn's algo during retransmission

Congestion Control Summary (Continued)

- Go-back-N or Selective ***retransmission***
- Cumulative and Selective ***acknowledgements***
- Advanced topics:
  - *Timeout avoidance:* Fast Retransmit
  - ***Drop*** policies
  - ***Scheduling***
  - ***ECN: Explicit congestion notification***

Rensselaer Polytechnic Institute Shivkumar Kalyanaraman

31

Gigabit Networks

- "Higher *Bandwidth* Networks"
- Propagation *latency unchanged*.
  - Increasing bandwidth from 1.5Mb/s to 45 Mb/s (factor of 29) decreases file transfer time of 1MB by a factor of 25.
  - But, increasing from 1 Gb/s to 2 Gb/s gives an improvement of only 10% !
  - Transfer time = propagation time + transmission time + queueing/processing.
- Design networks to minimize delay (queueing, processing, reduce retransmission latency)

Rensselaer Polytechnic Institute Shivkumar Kalyanaraman

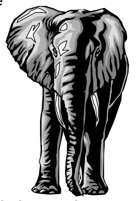
32

Window Scaling Option

- Long Fat Pipe Networks (LFN): Satellite links
- Need very large window sizes.
- Normally, Max window = 2^{16} = 64 KBytes
- Window scale: Window = W × 2^{Scale}

Kind = 3	Length = 3	Scale

- Max window = $2^{16} \times 2^{255}$
- Option sent only in SYN and SYN + Ack segments.
- RFC 1323



Rensselaer Polytechnic Institute Shivkumar Kalyanaraman

33

Timestamp Option

- For LFNs, need accurate and more frequent RTT estimates.
- Timestamp option:
  - Place a timestamp value in any segment.
  - Receiver echoes timestamp value in ack
  - If acks are delayed, the timestamp value returned corresponds to the ***earliest*** segment being acked.
- Segments lost/retransmitted => RTT overestimated

Rensselaer Polytechnic Institute Shivkumar Kalyanaraman

34

PAWS: Protection against wrapped sequence numbers

- Largest receiver window = 2^30 = 1 GB
- "Lost" segment may reappear before MSL, and the sequence numbers may have wrapped around

Rensselaer Polytechnic Institute Shivkumar Kalyanaraman

35

PAWS: Protection against wrapped sequence numbers (Continued)

- The receiver considers the timestamp as an extension of the sequence number => discard out-of-sequence segment based on both seq # and timestamp.
- Reqt: timestamp values need to be monotonically increasing, and need to increase by at least one per window

Rensselaer Polytechnic Institute Shivkumar Kalyanaraman

36

Summary

- Interactive and bulk TCP flow
- TCP congestion control
- ***Informal exercises:*** Perform some of the experiments described in chaps 19-21 to see various facets of TCP in action