Lecture 21a
Other Transport Protocols

Peter Steenkiste
Department of Computer Science and
Electrical and Computer Engineering
Carnegie Mellon University

15-441 Networking, Spring 2006
http://www.cs.cmu.edu/~prs/15-441

Outline of the "Transport Lectures"

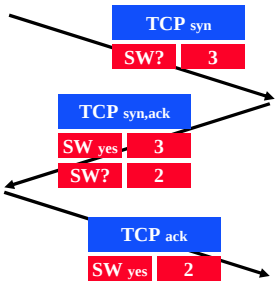
- Transport protocols introduction.
  - » Functions, UDP
- Flow and error control.
  - » Stop and go, go back N, sliding window, ...
- TCP.
  - » Connections, flow control, error handling, extensions, ...
- Congestion control.
  - » Congestion definition, congestion control strategies
- Congestion control in TCP.
  - » More TCP, applied congestion control
- Other transports.
  - » RPC, TCP conformance, multimedia

TCP Performance Considerations

- The window size can be controlled by the receiving application.
  - » Can change the socket buffer size from a default (e.g. 16Kbytes) to a maximum value (e.g. 64 Kbytes)
- The window size field in the TCP header limits the window that the receiver can advertise.
  - » 16 bits -> 64 KBytes
  - » 10 msec RTT -> 51 Mbit/second
  - » 100 msec RTT -> 5 Mbit/second
- "Large window" option (RFC 1323).
  - » Negotiated by the hosts during connection establishment
  - » Option 3 specifies the number of bits by which to shift the value in the 16 bit window field
  - » Independently set for the two transmit directions

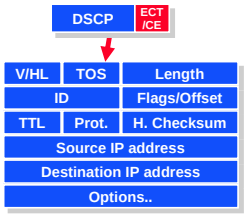
Window Scaling: Example Use of Options

- The scaling factor specifies bit shift of the window field in the TCP header.
  - » Scaling value of 2 translates into a factor of 4
- Old TCP implementations will simply ignore the option.
  - » Definition of an option
- Scaling results in a loss of accuracy.
  - » Not a big deal
- Alternatives?



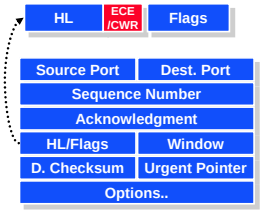
Explicit Congestion Notification (ECN)

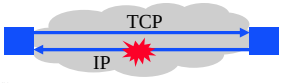
- The goal is to provide explicit congestion notification to senders.
  - » Complements the implicit feedback through packet drops
- Bits 6-7 of the TOS bit form the ECN field.
  - » The ECN-Capable Transport (ECT) bit is set by the sender to indicate that the end-points are ECN-capable
  - » The Congestion Experience (CE) bit is set by the router to signal congestion
- The ECN is received by the receiver, who is responsible for forwarding the information to the sender.



ECN in TCP

- Receiver signals congestion to the sender by setting the ECN-Echo flag in the TCP header.
  - » Bit 9 in the reserved field of the TCP header
  - » Handles asymmetric routes
  - » ECN-Echo flag also used to negotiate ECN use





Use of ECN with TCP

- The TCP sender should respond to ECN feedback as if a single packet loss occurred.
  - » Reduce the congestion window size
  - » Send "Congestion Window Reduced" flag (Bit 8) to ack
    - – So receiver knows to stop ECE bit
- ECN and RED can leverage each other.
  - » The router should set the CE bit if it would otherwise have dropped the packet (for a non-ECN enabled flow)
  - » When RED is used, this happens before the queues fill up so ECN and RED combined can result in congestion notification without packet loss
- Deployment seems quite practical.
  - » Can be introduced one router at a time
  - » But not a lot of deployment so far

Outline of the "Transport Lectures"

- Transport protocols introduction.
  - » Functions, UDP
- Flow and error control.
  - » Stop and go, go back N, sliding window, ...
- TCP.
  - » Connections, flow control, error handling, extensions, ...
- Congestion control.
  - » Congestion definition, congestion control strategies
- Congestion control in TCP.
  - » More TCP, applied congestion control
- Other transports.
  - » RPC, TCP conformance, multimedia

Other Transport Protocols

- Some applications need a form of reliability, congestion avoidance, or flow control different from that of TCP.
  - » May not need a 100% reliable transport
  - » Timeouts or slow down for congestion avoidance may introduce unacceptable delays
- Other transport protocols are often implemented in user space on top of UDP.
  - » Use the addressing provided by UDP
- Examples:
  - » Generic UDP-based protocols – TCP-conformance
  - » Remote procedure calls
  - » Multimedia

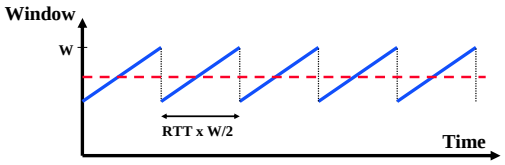
Transport Protocol Functions

	Mux/ Demux	Reliable	Flow Control	Congestion Control	Optimize
TCP	Yes	Yes	Yes	Yes	Yes
UDP	Yes	No	No	No	No
Appl		?	?	Yes	?

TCP Conformance

- Application-level congestion control comes in many forms.
  - » No congestion control
  - » More aggressive than a typical TCP implementation
  - » Similar behavior to that of TCP implementation, i.e. under congestion it behaves like TCP
- TCP conformant (or friendly) means that the flow coexists gracefully with TCP flows.
  - » Gets average bandwidth similar to what TCP would get
- How do we implement a "conformant" congestion control mechanism?
  - » Use "additive increase multiplicative decrease" algorithm
  - » Estimate and use equivalent TCP rate

TCP Throughput Calculation



- What is the average bandwidth?
  - » Average congestion window size is ¾ W
  - » That is how many packets we send per RTT
  - » Multiply with MSS to get average throughput
- Throughput = (¾ W / RTT) * MSS
- But what is W?

Bandwidth Based on Simple Loss Model

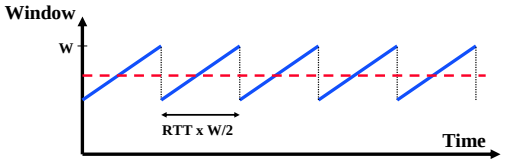
- W depends on loss rate: we lose one packet per "tooth".
  - » Packets transferred = (¾ W) * (W/2) = 3W²/8
  - » 1 packet lost → loss rate = p = 8/3W²

$$W = \sqrt{\frac{8}{3p}}$$

- BW = ¾ * W * MSS / RTT

$$W = \sqrt{\frac{8}{3p}} = \frac{4}{3} \times \sqrt{\frac{3}{2p}} \qquad BW = \frac{MSS}{RTT \times \sqrt{2p/3}}$$

TCP Throughput



$$B = 1.22 \times \frac{MSS}{RTT \times \sqrt{loss}}$$

Is this fair? Of course! "RTT fair".

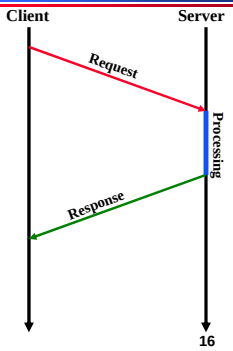
* Assumes no timeouts!

TCP Conformant Transport Protocols

- Reimplement TCP in the application.
  - » I.e. AIMD – a lot of work!
- Use a TCP-friendly congestion control protocol.
  - » "Conforms" to TCP, e.g. based on formula
  - » Can be different, e.g. smoother rate adjustments
  - » Example: TCP-Friendly Rate Control (TFRC, RFC3448)
- Active area of research – examples:
  - » Datagram Congestion Control Protocol (DCCP, RFC 4340): unreliable streaming but with congestion control
    - – E.g. TFRC (RFC4342)
  - » Stream Control Transmission Protocol (SCTP, RFC 3286):
    - – Reliable message streaming, support for multiple streams, uses TCP algorithms for flow and congestion control

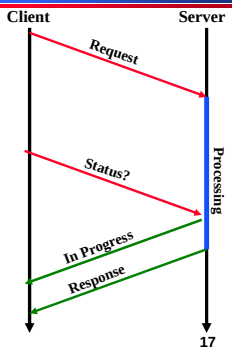
Remote Procedure Call

- Procedure call, but procedure is executed remotely.
- Client sends a request to the server.
- Server responds after processing the request.
- Issues:
  - » Reliability
  - » Congestion control
  - » Multiplexing/demultiplexing
  - » Semantics



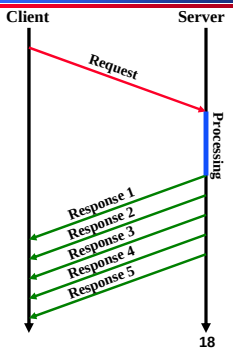
RPC Reliability

- Simple case: response serves as an acknowledgement for the request.
- Client starts a timer when it sends response and requests status updates periodically.
  - » Server responds with "in progress" message
  - » Alternatively: retransmit request
- Server needs to keep result in case response gets lost.
  - » Retransmit on next status
  - » Discard after ack or next request



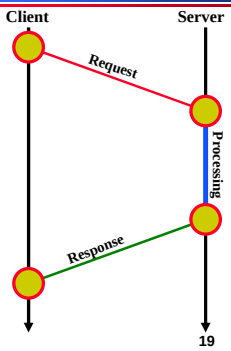
RPC Flow and Congestion Control

- Not an issue with small requests and responses.
  - » Effectively stop-and-wait
- With large transfers, RPC must implement packetization and reassembly.
  - » Must handle retransmissions, reordering, duplication, ...
  - » Should also implement congestion and flow control but often not done
- Alternative: run RPC over TCP.
  - » TCP provides reliability and congestion control



RPC Multiplexing

- Server must know which "procedure" the client wants to invoke.
  - » Done using a procedure identifier
- Goal of RPC is transparency.
  - » RPC should look like a local call
  - » Hide: use of procedure identifier, parameter handling (marshalling), RPC protocol, data format representation
- Solution: stub compiler that generates "stubs"
  - » Compiler takes procedure interface definition and generates the code for the client and server



RPC Semantics

- RPC looks like regular procedure call
  - » At least syntactically
  - » Typical behavior is that client blocks while call executes
- But RPC can have different performance and failure modes.
  - » Can be much slower for simple operations
  - » Procedure call can fail without the client failing, e.g. due to server crashes or network failures
    - – Did the call execute or not?
- Variants that optimize performance tend to break RPC model:
  - » Asynchronous RPC: client executes in parallel with call and resynchronizes later
  - » Batch RPC: single server request includes multiple calls

RPC Examples

- Sun RPC.
  - » Very widely used
  - » Retransmits requests – may not be "at most once"
  - » Basis for NFS
- "Distributed Computing Environment" (DCE) remote procedure call.
  - » Open Software Foundation "standard"
  - » Uses NDR for data format conversion
  - » Used as the basis for Corba
- Integrated environments.
  - » Also provide support for service discovery, …
  - » Jini, Java RMI, Corba, ..

Streaming Audio and Video Requirements

- Multimedia applications have real time requirements: samples must arrive in time at the destination to be useful.
  - » Frames in video stream
  - » Samples for audio
  - » Also applies to other data (slides, ..)
- Applies both to stored and live scenarios.
  - » Stored data is somewhat easier to deal with since it can more easily be "sent ahead"
- Traditionally audio and video have used dedicated networks that provide guaranteed bandwidth.
  - » Telephone and cable network
- Real Time Protocol: see Multimedia Lecture.