High-Performance Networking
The University of Kansas EECS 881
End-to-End Transport

James P.G. Sterbenz
Department of Electrical Engineering & Computer Science
Information Technology & Telecommunications Research Center
The University of Kansas

jpgs@eecs.ku.edu

http://www.ittc.ku.edu/~jpgs/courses/hsnets

18 November 2010 rev. 10.0 © 2004–2010 James P.G. Sterbenz

End-to-End Transport

Outline

TL.1 Functions and mechanisms
TL.2 State management
TL.3 Framing and multiplexing
TL.4 Error control
TL.5 Transmission control
TL.6 TCP optimisations
### End-to-End Transport

#### Outline

- 7.1. Functions and mechanisms
- 7.2. State management
- 7.3. Framing and multiplexing
- 7.4. Error control
- 7.5. Flow & congestion control
- 7.6. Security & info assurance

#### Transport Layer

**Service and Interfaces**

- Transport layer (L4) service to application layer (L7)
- Transfer PDU E2E (end-to-end)
  - sender: encapsulate ADU into TPDU and transmit
  - receiver: receive TPDU and decapsulate into ADU
- Multiplexing to application on end system
- Optional reliability:
  - error checking/correction/retransmission
  - need depends on application
  - may be done application-to-application (A2A)
- Flow control between end systems
  - assistance for network congestion control
Transport Layer
Service and Interfaces

- Transport layer *uses* service of network layer (L3)
  - network layer establishes a path through the network
- Transport mechanisms depend on L3 service
  - datagrams vs. connections
  - best effort vs. QoS
  - error characteristics
  - strength and symmetry of connectivity

What are the Internet assumptions & service model?

Transport Layer
Types of Transport Protocols

- Spectrum of transport protocol specialisation
  - general purpose
  - functionally partitioned
  - application-oriented
  - special purpose
- Software engineering and implementation choice
Transport Layer
Service and Interfaces

- **Transport PDU** encapsulates *application data unit*
  - ADU – application data unit
  - TPDU – transport protocol data unit
    - TPDU = header + ADU + optional trailer (protocol dependent)

---

Transport Layer
Functional Placement

- Transport layer functionality only in end system
  - *host software* or
  - *network interface* (NIC)
Transport Layer
End-to-End vs. Hop-by-Hop

- Transport layer is E2E analog of HBH link layer
  - link layer (L2) transfers packets HBH
  - network layer (L3) determines path of concatenated links

Ideal End-to-End Network Model
Bandwidth and Delay

- Zero end-to-end delay
- Unlimited end-to-end bandwidth
### TL.1 Functions and Mechanisms

**TL.1.1 End-to-end semantics**

**TL.1.2 End-to-end mechanisms**

**TL.2 State management**

**TL.3 Framing and multiplexing**

**TL.4 Error control**

**TL.5 Transmission control**

**TL.6 TCP Optimisations**

---

### E2E Functions and Mechanisms

**End-to-End Semantics**

- **Hop-by-hop functions**
  - link layer protocols
  - link compression and FEC
  - network forwarding
  - link / subnet error control
  - embedded protocols (e.g., protocol boosters)

- **End-to-end functions**
  - transport protocols
  - source routing
  - end-to-end encryption
  - session protocols
  - application protocols
E2E Functions and Mechanisms

**End-to-End Argument**

- Hop-by-hop function composition ≠ end-to-end
- Examples
  - HBH encryption has data in clear inside network nodes
  - HBH link error control doesn’t cover network layer errors

**End-to-End Argument T-3**

*Functions required by communicating applications can be correctly and completely implemented only with the knowledge and help of the applications themselves. Providing these functions as features within the network itself is not possible.*

---

E2E Functions and Mechanisms

**Hop-by-Hop Performance Enhancement**

- E2E functions replicated HBH if performance benefit
- Example
  - per link error control in high bandwidth×delay networks reduce E2E control loop when error

**Hop-by-Hop Performance Enhancement Corollary T-3A**

*It is beneficial to duplicate an end-to-end function hop-by-hop if the result is an overall (end-to-end) improvement in performance.*
End-to-End vs. Hop-by-Hop Performance Enhancement Example

- E2E vs. HBH error control for reliable communication
  - E2E argument says error control must be done E2E
    - e.g. E2E ARQ (error check code and retransmit if necessary)
    - but should HBH error control also be done?

  ![Network Diagram]

- Effect of high loss rate on wireless link
  - ~250 ms RTT retransmission for every corrupted packet
  - Error control on wireless link reduces to ~1 \( \mu \)s
    - shorter control loop results in dramatically lower E2E delay
E2E Functions and Mechanisms

E2E Argument Misinterpretations

- **E2E-only**
  - do not replicate E2E services or features HBH
  - violated HBH performance enhancement corollary
- **Everything E2E**
  - implement as many services or feature E2E as possible
  - misstatement of Internet design philosophy:
    simple stateless network for resilience and survivability

---

E2E Functions and Mechanisms

Meaning of “In the Network”

- **Functional** (general use in this tutorial)
  - layers 1 – 3
    - physical
    - MAC
    - link
    - network
- **Topological**
  - colocated with network nodes
- **Administrative**
  - owned by network service provider
End-to-End Protocols

**TL.1.2 End-to-End Mechanisms**

**TL.1 Functions and mechanisms**
- **TL.1.1** End-to-end semantics
- **TL.1.2** End-to-end mechanisms

**TL.2 State management**
**TL.3 Framing and multiplexing**
**TL.4 Error control**
**TL.5 Transmission control**
**TL.6 TCP Optimisations**

---

**E2E Functions and Mechanisms**

**End-to-End Mechanisms**

- Framing
- Multiplexing
- End-to-end state and connection management
- Error control
- Flow control
- Congestion control
E2E Functions and Mechanisms

Transport Protocol Service Categories

- Transfer mode
  - connectionless datagram
  - connection-oriented
  - transaction
  - continuous media streaming
- Reliability
  - loss tolerance
- Delivery order
  - ordered
  - unordered
- Traffic characteristics

E2E Functions and Mechanisms

Types of Transport Protocols

- Spectrum of transport protocol specialisation
  - general purpose
  - functionally partitioned
  - application-oriented
  - special purpose
- Software engineering and implementation choice

Transport protocols must be organised to deliver the set of end-to-end high-bandwidth, low latency services needed by applications. Options and service models should be modularly accessible, without unintended performance degradation and feature interactions.
E2E Functions and Mechanisms

Example 7.2  End-to-End Internet Protocols

- Transport protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Name</th>
<th>Function</th>
<th>Status</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>transmission control protocol</td>
<td>reliable data transfer with congestion control</td>
<td>standard</td>
<td>RFC 793 STD 007</td>
</tr>
<tr>
<td>UDP</td>
<td>user datagram protocol</td>
<td>socket access to unreliable IP datagrams</td>
<td>standard</td>
<td>RFC 768 STD 006</td>
</tr>
<tr>
<td>RTP</td>
<td>real-time protocol</td>
<td>file and document transfer (typically over UDP)</td>
<td>standards track</td>
<td>RFC 1889</td>
</tr>
<tr>
<td>T/TCP</td>
<td>TCP for transactions</td>
<td>remote login</td>
<td>experimental</td>
<td>RFC 1644</td>
</tr>
<tr>
<td>RDP</td>
<td>reliable data protocol</td>
<td>reliable data transfer with no congestion control</td>
<td>experimental</td>
<td>RFC 908</td>
</tr>
<tr>
<td>SCTP</td>
<td>stream control transmission protocol</td>
<td>signalling proposed for wireless</td>
<td>proposed standard</td>
<td>RFC 2960</td>
</tr>
</tbody>
</table>

- “Application layer” protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Name</th>
<th>Function/use</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>hypertext transfer protocol</td>
<td>Web browsing</td>
</tr>
<tr>
<td>SMTP</td>
<td>simple mail transfer protocol</td>
<td>email relay and delivery</td>
</tr>
<tr>
<td>FTP</td>
<td>file transfer protocol</td>
<td>file and document transfer</td>
</tr>
<tr>
<td>Telnet</td>
<td>telnet</td>
<td>remote login</td>
</tr>
<tr>
<td>NFS</td>
<td>network file system</td>
<td>remote access to files</td>
</tr>
<tr>
<td>RTSP</td>
<td>real-time streaming protocol</td>
<td>control of multimedia streaming</td>
</tr>
</tbody>
</table>

*“application layer” only because they run over TCP or UDP*
E2E Functions and Mechanisms

Control of State

- Open loop
  - control parameters
  - no feedback
- Closed loop
  - initial parameters
  - feedback
    - dynamic adaptation
- Closed loop with HBH feedback
- Nested control loops

Open-loop control
  - use when possible to eliminate control loop latency
    - particularly for high bandwidth-delay product networks

Closed-loop feedback control
  - use when necessary, e.g. reliability

Open- vs. Closed-loop Control

Use open-loop control based on knowledge of the network path to reduce the delay in closed loop convergence. Use closed-loop control to react to dynamic network and application behaviour; intermediate per hop feedback can sometimes reduce the time to converge.
E2E Functions and Mechanisms

Control of State

- Aggressiveness of closed-loop control
  - rapid enough to system converges
  - not too aggressive avoid instability and oscillations

Aggressiveness of Closed-Loop Control

The response to closed-loop feedback must be rapid enough so the system converges quickly to a new stable state – but not so aggressive that oscillations occur due to overreaction to transients.

End-to-End Transport

TL.2 State Management

TL.1 Functions and mechanisms
TL.2 State management
TL.3 Framing and multiplexing
TL.4 Error control
TL.5 Transmission control
TL.6 TCP optimisations
End-to-End Protocols

State Management

7.1 Functions and mechanisms
7.2 State management
  7.2.1 Impact of high speed
  7.2.2 Transfer modes
  7.2.3 State establishment and maintenance
  7.2.4 Assumed initial conditions
7.3 Framing and multiplexing
7.4 Error control
7.5 Flow and congestion control
7.6 Security and information assurance

State Management

Transport and Network Connections

- L4 connections
  - required over
    - connectionless L3
    - path with any connectionless subnetwork
  - may exploit
    - connection-oriented L3
State Management
Impact of High Speed

- Connection shortening
  - transmission delay decreases
  - signalling delay constant
- E2E signalling
  - significant fraction of time

State Management
Impact of Long Latency

- Connection lengthening
  - transmission delay increases
  - signalling delay increases
- E2E signalling
  - open state longer
  - denial of resource to others
Transfer Modes

Datagram

- Independent packets
  - each has fully self-describing header
  - no network state needed to forward

Connection

- Explicit connection setup
  - 3-way handshake
    SETUP / CONNECT / ACK
- Data flow
  - packets need connection or flow identifier
    - used by nodes to look up state
- Release of resources and state
  - explicit RELEASE
  - time out state
Transfer Modes

Connection

- Connection State Machine
  - idle
  - establishing
    - install state
  - initialising
    - state convergence
  - steady state
  - closing
    - release state

Transfer Modes

Stream

- Various mechanisms to start stream
  - explicit client request
  - server push
  - may or may not establish connection state

- Data flow
  - synchronisation and control
    - embedded or
    - out-of-band
Transfer Modes

Stream

- Playback buffer to reorder and absorb jitter
  - adds delay

Continuous Media Streaming

Timeliness of delivery should not be traded for reliable transfer in real-time continuous media streams. Use a playback buffer to reorder and absorb jitter.

Transaction

- Transaction request
  - may or may not establish connection state
  - explicit release of connection state optional
- Data returned
- Overlap to reduce latency
  - request/response with control

Minimise Transaction Latency

Combine connection establishment with request and state transfer to reduce end-to-end latency for transactions.
E2E State Management

**Hard vs. Soft**

- **Hard state**
  - explicitly established and released
  - deterministic
  - delay to establish and memory to maintain

- **Soft state**
  - established and removed as needed and memory allows
  - robust to failures
  - if not explicitly established, delay to accumulate

**Hard vs. Soft State**

Balance the determinism and stability of hard state against the adaptability and robustness to failure of soft state.

---

E2E State Management

**State Aggregation and Sharing**

- **Sharing of state**
  - reduces granularity of individual entities
  - state shared is fate share

**Per State**

- rate/window
- ACKs/NAKs
- path MTU
- RTT estimate
- (sub)network characteristics
- max link rate
- link characteristics

**Per application transfer**

- per application transfer
- app id
- app id

**Per application**

- per application

**Per connection/stream**

- per connection/stream

**Per E2E path**

- per E2E path

**Per network**

- per network

**Global (per interface)**

- global (per interface)
E2E State Management

State Aggregation and Sharing

- Sharing of state
  - reduces granularity of individual entities
  - state shared is fate shared state

State Aggregation

Spatially aggregate state to reduce complexity, but balance against loss in granularity. Temporally aggregate to reduce or eliminate establishment and initialisation phase. State shared is fate shared.

E2E State Management

Assumed Initial Conditions

- Rapid convergence to steady state
  - assume initial conditions
    - normal or common case
    - past behaviour
    - current network state

Use Initial Assumptions to Converge Quickly

The time to converge to steady state depends on the accuracy of the initial conditions, which depends on the currency of state information and the dynamicity of the network. Use past information to make assumptions that will converge quickly.
E2E Framing and Multiplexing
Packet Size and Control Overhead

- Small packets
  - large overhead
  - short interarrival
- Grouped
- Blocked / bursts
- Large
  - increased delay
- Jumbogram

Balance Packet Size
*Trade between control overhead and fine enough grained control. Choose size and aggregation methods that make sense end-to-end.*
**E2E Framing and Multiplexing**

**Packet Size and Control Overhead**

- Large packets
  - impose jitter and delay on one another

---

**E2E State Management**

**MTU and per Hop Fragmentation**

- Subnetworks have limits on packet size
  - MTU: maximum transfer unit
- If $|\text{TPDU}| > \min(\text{MTU})$
  - fragmentation occurs in the network if permitted
    - significant impact on packet processing rate and delay
  - most modern networks drop
    - fragmentation disabled to prevent overwhelming slow path

---

**Avoid Hop-by-Hop Fragmentation**

*The transport layer should be aware of or explicitly negotiate the maximum transfer unit of the path to avoid the overhead of fragmentation and reassembly.*
E2E Framing and Multiplexing
Packet Size and Control Overhead

To the degree possible, match ADU and TPDU structure. In cases where the application can better determine structure and react to lost and misordered ADUs, ALF should be employed.

Application Layer Framing

Transport layer fragmentation

Application Layer Framing

Layered Multiplexing

- Multiplexing
  - application
    - interapplication coordination
  - transport
    - applications sharing network interface
  - network
    - end systems sharing switch
  - MAC and link
    - interfaces sharing medium
E2E Framing and Multiplexing

Integrated Multiplexing

- Multiplexing
  - significant overhead
  - processing limits
  - delay
  - undesirable fate sharing
  - QoS
  - perform all layers at once
    - network layer for wired
    - MAC layer for wireless

Limit Layered Multiplexing

Layered multiplexing should be minimised and performed in a single integrated manner for all layers at the same time.

End-to-End Transport

TL.4 Error Control

TL.1 Functions and mechanisms
TL.2 State management
TL.3 Framing and multiplexing
TL.4 Error control
  - TL.4.1 Types and causes of errors
  - TL.4.2 Impact of high speed
  - TL.4.3 Closed-loop retransmission
  - TL.4.4 Open-loop error control
TL.5 Transmission control
TL.6 TCP optimisations
E2E Error Control

Types of Errors₀

What are the types of errors?

Types of Errors₁

- Bit errors
- Packet errors

types?
Types of Errors

- Bit errors
- Packet errors
  - packet loss
  - fragment loss
  - burst loss
  - packet misordering
  - packet insertion
  - packet duplication

Causes of Bit Errors

- Bit errors
  
  causes?
E2E Error Control

Causes of Bit Errors

- Bit errors
  - flaky hardware
  - wireless channels
    - noise and interference
  - optical channels
    - long-haul fiber links engineered on margin with FEC
    - control of amplifier and regenerator cost

E2E Error Control

Causes of Packet Loss

- Packet loss
  
  \textit{causes?}
E2E Error Control

Causes of Packet Loss

- Packet loss
  - network buffer overflow or intentional drop
  - transport protocol packet discard when bit error

- Packet fragment loss
  causes?
  impact?
E2E Error Control

Causes of Packet Loss

- Packet loss
  - network buffer overflow or intentional drop
  - transport protocol packet discard when bit error
- Packet fragment loss
  - loss of a piece of a fragmented packet
  - effect: error multiplication
    - loss of a fragment in net generally results in entire packet loss

E2E Error Control

Causes of Packet Burst Loss

- Packet burst loss
E2E Error Control

Causes of Packet Burst Loss

- Packet burst loss
  - network buffer overflow during congestion
  - wireless channel fades

Causes of Packet Misordering

- Packet misordering causes?
### E2E Error Control

#### Causes of Packet Misordering

- Packet misordering
  - multiple paths through switch or network
  - non-deterministic paths through switch or network
  - path reconfiguration
    - after failure
    - due to mobility
    - due to load balancing

---

#### E2E Error Control

#### Causes of Packet Insertion

- Packet *insertion*
  - meaning?
  - causes?
E2E Error Control

Causes of Packet Insertion

- Packet duplication
  - undetectable header error
    - packet appears that was destined elsewhere
  - long packet life
    - packet with same destination and sequence number still in net

Causes of Packet Duplication

- Packet duplication causes?
E2E Error Control

Causes of Packet Duplication

- Packet duplication
  - retransmission but original still arrives

E2E Error Control

Impact of High Speed

- Bandwidth-\times\text{-delay} product increases
  - fixed duration error event larger relative impact
  - closed-loop reaction occurs later in terms of \# bits sent

- Sequence numbers
  - sequence numbers wrap if sequence space too small
    - causes packet insertion
End-to-End Transport

**TL.4.3 Closed-Loop Retransmission**

- **TL.1** Functions and mechanisms
- **TL.2** State management
- **TL.3** Framing and multiplexing
- **TL.4** Error control
  - **TL.4.1** Types and causes of errors
  - **TL.4.2** Impact of high speed
  - **TL.4.3** Closed-loop retransmission
  - **TL.4.4** Open-loop error control
- **TL.5** Transmission control
- **TL.6** TCP optimisations

---

E2E Error Control

**ARQ Closed Loop Retransmission**

- Automatic repeat request (ARQ)
  - PDUs are retransmitted for reliable transfer
- Acknowledgments are used to request retransmission
  - ACK: positive acknowledgement
  - NAK (or NACK): negative acknowledgement

*which is better?*
E2E Error Control
ARQ Closed Loop Retransmission

- Automatic repeat request (ARQ)
  - PDUs are retransmitted for reliable transfer
- Acknowledgments are used to request retransmission
  - ACK: positive acknowledgement
    - required for full reliability by E2E arguments
  - NAK (or NACK): negative acknowledgement
  - tradeoff between error rate and predictability
    - hybrid: NAKs supplemented by ACKs when necessary for E2E

Simplest ARQ mechanism?

E2E Error Control
Closed Loop Retransmission: Stop-and-Wait

- Each packet is acknowledged
E2E Error Control
Closed Loop Retransmission: Stop-and-Wait

• Each packet is acknowledged

- subsequent packet waits on previous ACK

disadvantages?
E2E Error Control
Closed Loop Retransmission: Stop-and-Wait

- Each packet is acknowledged
  - subsequent packet waits on previous ACK
  - significant delay and underutilisation
  - 1 RTT per packet
  - serious penalty in long-delay environment

- sender timer $t_{ack}$ fires if ACK not received

issues?
E2E Error Control
Closed Loop Retransmission: Stop-and-Wait

- Each packet is acknowledged
  - subsequent packet waits on previous ACK
    - significant delay and underutilisation
    - 1 RTT per packet
    - serious penalty in long-delay environment
  - sender timer $t_{\text{ack}}$ fires if ACK not received
    - too conservative: issues?
    - too aggressive: issues?
E2E Error Control
Closed Loop Retransmission: Stop-and-Wait

- Each packet is acknowledged
  - subsequent packet waits on previous ACK
    - significant delay and underutilisation
    - 1 RTT per packet
    - serious penalty in long-delay environment
  - sender timer $t_{ack}$ fires if ACK not received
    - too conservative: unnecessary delay
    - too aggressive: spurious retransmissions

*How can we do better?*

---

E2E Error Control
Closed Loop Retransmission: Go-Back-\(n\)

- Go-back-\(n\): pipeline transmissions
E2E Error Control
Closed Loop Retransmission: Go-Back-$n$

- Go-back-$n$: *pipeline* transmissions
  + multiple packets simultaneously in flight

- Packets are *sequentially* acknowledged
E2E Error Control
Closed Loop Retransmission: Go-Back-\( n \)

- Go-back-\( n \): *pipeline* transmissions
  + multiple packets simultaneously in flight
- Packets are *sequentially* acknowledged
  
  how is packet loss detected?

- previous ACK retransmitted for
  - subsequent packets after loss
  - out of sequence packet
E2E Error Control
Closed Loop Retransmission: Go-Back-\( n \)

- Go-back-\( n \): pipeline transmissions
  + multiple packets simultaneously in flight
- Packets are sequentially acknowledged
  o previous ACK retransmitted for
    - subsequent packets after loss
    - out of sequence packet
  o sender timer fires if ACK not received
    implication?
E2E Error Control
Closed Loop Retransmission: Go-Back-\( n \)

- Go-back-\( n \): pipeline transmissions
  - multiple packets simultaneously in flight
- Packets are sequentially acknowledged
  - previous ACK retransmitted for
    - subsequent packets after loss
    - out of sequence packet
  - sender timer fires if ACK not received
    - reset transmission beginning at lost packet
E2E Error Control
Closed Loop Retransmission: Go-Back-$n$

- Go-back-$n$: pipeline transmissions
  - multiple packets simultaneously in flight
- Packets are sequentially acknowledged
  - previous ACK retransmitted for
    - subsequent packets after loss
    - out of sequence packet
  - sender timer fires if ACK not received
    - reset transmission beginning at lost packet
E2E Error Control
Closed Loop Retransmission: Go-Back-n

- **Go-back-n**: pipeline transmissions
  - multiple packets simultaneously in flight
- Packets are sequentially acknowledged
  - previous ACK retransmitted for
    - subsequent packets after loss
    - out of sequence packet
  - sender timer fires if ACK not received
    - reset transmission beginning at lost packet

**Disadvantages?**

- significant loss penalty for high bw×delay
  - go back and retransmit all since loss
  - many unneeded retransmissions
  - significant additional delay
E2E Error Control
Closed Loop Retransmission: Go-Back-\( n \)

- Optimisation possible for go-back-\( n \)?

- Optimisation go-back-\( n \)
  
  timer the only way to detect loss?
E2E Error Control
Closed Loop Retransmission: Fast Retransmit

- Fast retransmit
  - optimisation for go-back-n
E2E Error Control

Closed Loop Retransmission: Fast Retransmit

- Fast retransmit
  - optimisation for go-back-\( n \)
E2E Error Control
Closed Loop Retransmission: Fast Retransmit

- Fast retransmit
  - optimisation for go-back-\( n \)
- Assume several duplicate ACKs are loss
  - even if timer hasn’t yet fired
E2E Error Control

Closed Loop Retransmission: Fast Retransmit

- Fast retransmit
  - optimisation for go-back-\( n \)
- Assume several duplicate ACKs are loss
  - even if timer hasn’t yet fired
  - recovers from loss more quickly
E2E Error Control
Closed Loop Retransmission: Fast Retransmit

- Fast retransmit
  - optimisation for go-back-$n$
- Assume several duplicate ACKs are loss
  - even if timer hasn’t yet fired
  + recovers from loss more quickly
E2E Error Control

Closed Loop Retransmission: Fast Retransmit

- Fast retransmit
  - optimisation for go-back-\( n \)
- Assume several duplicate ACKs are loss
  - even if timer hasn’t yet fired
  + recovers from loss more quickly

- still suffers from go-back-\( n \) inefficiencies
E2E Error Control

Closed Loop Retransmission: Delayed ACK

- **Delayed ACK**
  - optimisation for go-back-
- Aggregate several ACKs
  - reduces ACK traffic
  - allows some receiver resequencing
    - within ACK aggregation quantum

E2E Error Control

Closed Loop Retransmission: Go-Back-

- Go-back-
  - fast retransmit and delayed ACK help slightly
  - but still significant delay penalty for losses

*Alternative?*
E2E Error Control
Closed Loop Retransmission: Selective Repeat

- Go-back-$n$
  - fast retransmit and delayed ACK help slightly
  - significant delay penalty for losses
- Alternative
  - don’t go back: *selective repeat*

*Selective repeat*
- all packets acknowledged
E2E Error Control

Closed Loop Retransmission: Selective Repeat

- Selective repeat
  - all packets acknowledged
E2E Error Control
Closed Loop Retransmission: Selective Repeat

- Selective repeat
  - all packets acknowledged

- Missequenced packets
  - acknowledged and held at receiver
E2E Error Control

Closed Loop Retransmission: Selective Repeat

- Selective repeat
  - all packets acknowledged
- Missequenced packets
  - packets held and reordered at receiver
    - increases receiver complexity
- Lost packets
  - selectively retransmitted
E2E Error Control

Closed Loop Retransmission: Selective Repeat

- Selective repeat
  - all packets acknowledged
- Missequenced packets
  - packets held and reordered at receiver
    - increases receiver complexity
- Lost packets
  - selectively retransmitted
    + no go-back-$n$ latency penalty
    - requires more receiver buffer space
E2E Error Control

Closed Loop Retransmission: Selective Repeat

- Selective repeat
  - all packets acknowledged
- Missetsequenced packets
  - packets held and reordered at receiver
    - increases receiver complexity
- Lost packets
  - selectively retransmitted
    + no go-back-$n$ latency penalty
    - requires more receiver buffer space
- Latency and Bandwidth reduced
E2E Error Control
Closed Loop Retransmission: Periodic State

- Selective ACK blocks
  - bit vectors can aggregate ACKs
    - significant reduction in control packets
    - simple receiver implementation possible
E2E Error Control

Impact of High Speed

• Bandwidth-×-delay product increases
  – fixed duration error event larger relative impact
  – reaction to errors decreases in terms of number of bits sent
• Sequence numbers
  – sequence numbers wrap if sequence space too small
    • causes packet misinsertion

Packet Control Field Values

Optimise header control field values to trade efficiency against expected future requirements. Fields that are likely to be insufficient for future bandwidth-×-delay products should contain a scale factor.

E2E Error Control

Decoupling of Control Mechanisms

• Error control mechanisms
  – difficult to optimise for high performance
    when entangled with flow/congestion control
    (more later)

Decouple Error, Flow, and Congestion Control

Exploit selective acknowledgements and open-loop rate control to decouple error, flow, and congestion control mechanisms.
**E2E Error Control**

**Closed Loop Retransmission: Multicast**

- ACK implosion: ACK suppression needed
  - NAKs (negative ACKs) with liveness polling
  - Localised hop-by-hop buffering and retransmission
    - Significant reduction of traffic on relatively reliable links

---

**Error Control**

**Example: SNA Error Control**

- SNA (IBM Systems Network Architecture)

<table>
<thead>
<tr>
<th>Scope</th>
<th>Layer</th>
<th>Name</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>hop-by-hop</td>
<td>link</td>
<td>data link control</td>
<td>yes</td>
</tr>
<tr>
<td>end-to-end</td>
<td>transport</td>
<td>path control</td>
<td>no</td>
</tr>
</tbody>
</table>

- Not in agreement with end-to-end arguments, but...
  - Engineered reliability of store-and-forward hosts
    - No part of data path not covered by ECC: processor + memory
End-to-End Transport

**TL.4.4 Open-Loop Error Control**

TL.1 Functions and mechanisms
TL.2 State management
TL.3 Framing and multiplexing
TL.4 Error control
   TL.4.1 Types and causes of errors
   TL.4.2 Impact of high speed
   TL.4.3 Closed-loop retransmission
   TL.4.4 Open-loop error control
TL.5 Transmission control
TL.6 TCP optimisations

---

E2E Error Control

**Open Loop**

- **Open loop error control**
  - some applications do not need guaranteed reliability
  - applications that are tolerant to some loss
  - real-time or interactive delay requirements
    - eliminates control loop latency for recovery
    - requires additional end-systems bandwidth & processing
    - requires additional network bandwidth
- **Forward error correction**
  - block codes: per block recovery
  - convolutional codes: continuous over window
E2E Error Control

Hybrid Open/Closed-Loop Control

• Hybrid open loop with closed loop when necessary
  – statistical error bound with FEC
  – retransmission only when FEC doesn’t allow correction
  – appropriate for
    • high latency and bandwidth-delay paths
    • asymmetric bandwidth paths

Forward Error Correction

Use FEC for low-latency, open-loop flow control when bandwidth is available and statistical loss tolerance bounds are acceptable to the applications.

End-to-End Transport

TL.5 Transmission Control

TL.1 Functions and mechanisms
TL.2 State management
TL.3 Framing and multiplexing
TL.4 Error control
TL.5 Transmission control
  TL.5.1 Impact of high speed
  TL.5.2 Open-loop rate control
  TL.5.3 Closed-loop flow control
  TL.5.4 Closed-loop congestion control
  TL.5.5 Hybrid flow and congestion control
TL.6 TCP optimisations
E2E Transmission Control

Definitions

- Flow control
  - control transmission to avoid overwhelming receiver
  - end-to-end control

- Congestion control
  - control transmission to avoid overwhelming network paths
E2E Transmission Control

Definitions

- Flow control
  - control transmission to avoid overwhelming receiver
  - end-to-end control
- Congestion control
  - control transmission to avoid overwhelming network paths
  - network control with end-to-end assistance (throttling)
- Types
  - explicit relies on congestion information from nodes
    - allows decoupling of error, flow, and congestion mechanisms
  - implicit assumes loss is congestion
    - bad assumption in wireless networks (see [Krishnan 2004])

E2E Transmission Control

Impact of High Speed

- Bandwidth × delay product increases
  - response to an event happens after more bits transferred
  - it may be too late to react
    - all bits causing problem may already be transmitted
    - other cause of problem may have gone away
E2E Transmission Control

Open-Loop Rate Control

- Initial negotiation
  - admission control limits traffic to available resources
  - receiver negotiates what it can accept (flow control)
E2E Transmission Control

Open-Loop Rate Control

- Initial negotiation
  - admission control limits traffic to available resources
  - receiver negotiates what it can accept (flow control)
- Steady state
  - policing enforces traffic contract from transmitter

- excess traffic may be marked and passed if capacity available
E2E Transmission Control

Open-Loop Rate Control

- Requires connection establishment
  - overhead and delay for connection setup
    - optimistic establishment or fast reservations ameliorate
  + QOS guarantees possible in steady state
  + feedback control loops not needed during transmission
  + network stability less dependent on congestion control

Use Knowledge of Network Paths for Open-Loop Control

Exploit open-loop rate and congestion control based on a priori knowledge to the degree possible to reduce the need for feedback control.

E2E Transmission Control

Open-Loop Rate Control: Parameters

- Main parameters
  - peak rate $r_p$
  - average rate $r_a$
  - burstiness (peak to average ratio or max burst size)

- Derived parameters
  - jitter
Use Closed-Loop Congestion Control to Adjust to Network Conditions

Closed-loop feedback is needed to adjust to network conditions when there are no hard reservations.

Closed-Loop Flow Control

- Feedback from receiver to limit rate
- Window to limit amount of unacknowledged data
  - static window
  - dynamic window
  - combined with congestion control
**Congestion Control**

**Ideal Network**

- Ideal network:
  - throughput: carried load = offered load (45° line)
  - zero delay (flat line)

  *Why isn’t this possible?*

**Real Network**

- Delay can’t be zero: speed-of-light
  - delay through network channels
  - delay along paths in network nodes

- Throughput can’t be infinite
  - channel capacity limits bandwidth
  - switching rate of components limits bit rate
**Congestion Control**

**Desired Real Network Behaviour**

- Desired network:
  - carried load = offered load up to share of capacity
  - what happens to delay?

**Causes of Congestion**

- Congestion when offered load → link capacity
- Best effort
  - occurs whenever load is high
- Probabilistic guarantees
  - occurs when load exceeds resource reservations
- Absolute guarantees
  - can't happen (in theory)
Congestion Control

Infinite Buffers / No Retransmission

- Congestion when offered load $\rightarrow$ link capacity
- Infinite buffers
  - queue length builds to infinity
  - $d \rightarrow \infty$

Finite Buffers with Retransmission

- Finite buffers cause packet loss
  - retransmissions contribute to offered load
    - assuming reliable protocol
  - throughput curve levels off
Congestion Control

**Multihop Network**

- Multiple hops
  - downstream packet loss
  - upstream transmission wasted

---

**Consequences of Congestion**

- **Delay** increases
  - due to packet queuing in network nodes
  - due to retransmissions when packets overflow buffers
    - finite buffers must drop packets when full
- **Throughput**
  - levels off gradually (with real traffic)
  - then decreases
    - due to retransmissions when packets dropped
    - particularly over multiple hops
  - congestion collapse
    - "cliff" of the throughput curve
Congestion Control

Local Action

- Local action at point of congestion
- Drops packets: discard policy
  - tail drop simplest: drop packets that overflow buffers
  - more intelligent policies possible
    - need flow or connection state in switches
    - discriminate *which* flows are causing congestion and penalise

Congestion Control

End-to-End Action

- End-to-end action
- Throttle source
  - reduce transmission rates
  - prevent unnecessary retransmissions
- Types
  - explicit
  - implicit
**E2E Transmission Control**

**Congestion Avoidance and Control**

- Congestion should be avoided before it happens. Keep queues from building and operate just to the left of the knee to maximise throughput and minimise latency.

**Closed-Loop Congestion Control**

- Feedback from *network* to limit rate
- Window to limit amount of unacknowledged data
  - dynamic window
  - conservation of packets in the network
  - self clocking
- Required unless open loop with hard reservations

*Use Closed-Loop Congestion Control to Adjust to Network Conditions*  
Closed-loop feedback is needed to adjust to network conditions when there are no hard reservations.
Congestion Control

Window Size

- Window size critical
  - big enough to fill pipe

Slow-Start Initialisation

- Slow start initialisation
  - increase window until path loaded
- Critical parameters
  - initial window size
  - rate of increase
  - Tradeoffs?
E2E Flow and Congestion Control

**Slow-Start Initialisation**

- Slow start initialisation
  - increase window until path loaded
- Critical parameters
  - initial window size
  - rate of increase
- Tradeoffs
  - conservative on high bw-x-delay:
    - multiple round trips
    - never get to full rate for transactions
  - aggressive:
    - induced congestions

**Congestion Control**

**AIMD Steady State**

- AIMD steady state
  - additive-increase slowly increases rate
    - increment window
  - multiplicative-decrease quickly throttles with congestion
    - divide window
  - RED attempts to reduce when congestion impending
Congestion Control

**AIMD Fairness and Stability**

- **AIMD**
  - $R =$ available bandwidth
  - initial bandwidth share $I$
  - **AI:** both increase
    - until loss beyond $R$
  - **MD:** both decrease
    - half way to origin
    - halve window size
  - trajectory toward ideal
    - intersection of $R$ and equal

([based on Kurose fig. 3.53])

---

**Combination Control**

**Slow Start + AIMD**

- Initialisation phase: slow start
- Steady-state phase: AIMD
**Congestion Control**  
**Optimisation for High-Speed**

- **Initialisation**
  - start with bigger window
  - increase more rapidly than “slow start” doubling
  - danger: too aggressive can cause congestion collapse

- **Steady state**
  - better estimate capacity to avoid MD halving
    - TCP BIC and CUBIC binary search for faster convergence
Congestion Control

Implicit Control

- Implicit control
  - missing ACK assumed to be congestion
    good assumption?
  - reasonable when all losses are due to congestion
    - fiber optic channels connected by reliable switches
Congestion Control

Implicit Control

- Implicit control
  - missing ACK assumed to be congestion
  - reasonable when all losses are due to congestion
    - fiber optic channels connected by reliable switches
  - performs poorly when significant losses in channel
    why?
Congestion Control

Implicit Control

- Implicit control
  - missing ACK assumed to be congestion
  - reasonable when all losses are due to congestion
    - fiber optic channels connected by reliable switches
  - performs poorly when significant losses in channel
    - mobile wireless links
    - under-provisioned CDMA (including optical CDMA)

Alternatives?

Explicit Control

- Explicit control
  - explicit congestion notification (ECN)
    - throttle with ECN message
  - some decoupling of error and congestion control
    - throttle before packet loss
    - not sufficient for lossy wireless link
Congestion Control

Explicit Control

- Explicit control
  - explicit congestion notification (ECN)
    - throttle with ECN message
  - some decoupling of error and congestion control
    - throttle before packet loss
    - not sufficient for lossy wireless link
  - not sufficient for discrimination of corrupted packets
    - ELN (explicit loss notification) necessary

Congestion Control

Steady State Comparison

- Implicit control
  - standard
  - with fast retransmit
- Explicit control
  - ECN
End-to-End Transport

TL.6 TCP Optimisations

TL.1 Functions and mechanisms
TL.2 State management
TL.3 Framing and multiplexing
TL.4 Error control
TL.5 Transmission control

TL.6 TCP optimisations
  TL.6.1 TCP overview
  TL.6.2 TCP optimisations for high performance

Transmission Control Protocol

Overview

- TCP: transmission control protocol
  RFC 0793 / STD 0007
  + RFC 1122 implementation requirements
  + RFC 1323 high performance extensions
  + RFC 2018 SACK (selective acknowledgements)
  + RFC 5681 congestion control
  + RFC 3465 appropriate byte counting
  + RFC 3168 ECN (explicit congestion notification)
TCP Overview

Transfer Mode and Characteristics

- Connection-oriented transfer mode
  - latency of connection setup
  - connection state must be maintained on each end system
- Point-to-point full-duplex
  - reliable byte stream: no message boundaries
  - pipelined transfer (not just stop-and-wait)
- Flow control
  - will not overwhelm receiver
- Congestion control
  - attempt to avoid congesting network
  - implicit and optional ECN

TCP Overview

Segment Format: Overview

- Relatively large header
  - fixed length & position fields
  - variable length options
  - header length in Bytes
    - $= 40 + \text{|options|} \text{ B}$
- Variable length payload
- Options
  - may not be present
- Checksum on entire segment
  - same algorithm as UDP

checksum
vel data pointer
...
TCP Overview

Segment Format: Header Flags

- Control flags (1 bit each)
  - CWR cong. win. reduced
  - ECE ECN echo
  - URG urgent data
  - ACK acknowledgement
  - PSH push data
  - RST reset connection
  - SYN connection setup
  - FIN connection teardown

Connection Management: Segment Format

- Flags define packet type
  - ACK segment
    - ACK for SYN or FIN
    - SYNACK may be piggybacked
  - RST (reset) segment
    - e.g. bad port received
  - SYN (synchronise) segment
    - connection setup
  - FIN (finish) segment
    - connection release
**TCP Overview**

**Connection Management: Setup**

- Three way handshake
  - “client” and “server”
  1. SYN (init seq#) →
     - randomised for security
  2. ← SYNACK (init seq#)
     - randomised for security
     - after buffer allocation
     - SYN flood vulnerability
  3. ACK →
     - buffer allocation
     - segment may include data
     - client may send more
  4. server may return data

**Connection Management: Teardown**

- Two two-way handshakes
  - each is a half-close
  - may occur simultaneously
    - full close
  - ACK can’t be piggybacked
- Client closes socket
  1. FIN →
  2. ← ACK
     - timed wait before close
- Server closes socket
  1. ← FIN
  2. ACK →
TCP Overview
RFC0793 State Machine

[Diagram showing TCP connection states]

Actual Connection State Machine

[Sewell's diagram of TCP states]

[Sewell] http://www.cl.cam.ac.uk/users/pes20/Netsem
TCP Overview

Segment Format: Sequence and Window

- **Sequence #**
  - forward data transfer
  - 1st byte # in segment

- **ACK number**
  - reverse acknowledgements
  - seq # of next byte expected
  - if ACK flag set
  - data segment may piggyback ACK

- **Receive window**
  - # unACKed bytes

TCP Overview

Unidirectional Data Transfer

- **Data is sequence of bytes**
  - number by byte #
  - (not segment #)
TCP Overview

Unidirectional Data Transfer

- Data is sequence of bytes
  - number by byte # (not segment #)
  - each segment ≤ MSS B
- Sequence numbers
  - byte # in byte stream of 1st byte in segment
- ACK numbers
  - seq # of next byte expected
• Data is sequence of bytes
  – number by byte # (not segment #)
  – each segment ≤ MSS B
• Sequence numbers
  – byte # in byte stream of 1st byte in segment
• ACK numbers
  – seq # of next byte expected
• Data is sequence of bytes
  – number by byte # (not segment #)
  – each segment ≤ MSS B
• Sequence numbers
  – byte # in byte stream of 1st byte in segment
• ACK numbers
  – seq # of next byte expected
  – cumulative ACKs
TCP Overview

Unidirectional Data Transfer

- Data is sequence of bytes
  - number by byte # (not segment #)
  - each segment ≤ MSS B
- Sequence numbers
  - byte # in byte stream of 1st byte in segment
- ACK numbers
  - seq # of next byte expected
  - cumulative ACKs
- Out of order arrival
  - implementation specific

TCP Overview

Bidirectional Data Transfer

- Data both directions
  - ACKs returned both ways
TCP Overview

Bidirectional Data Transfer

- Data both directions
  - ACKs returned both ways
  - may be *piggybacked*

what next?

- Data both directions
  - ACKs returned both ways
  - may be *piggybacked*
  - *delayed ACK* waits briefly
**TCP Overview**

**Bidirectional Data Transfer**

- Data both directions
  - ACKs returned both ways
  - may be *piggybacked*
  - *delayed ACK* waits briefly
- Bidirectional stream
  - data segments
  - piggybacked ACKs

---

**End-to-End Transport**

**TL.6.2 TCP Optimisations for High Performance**

- TL.1 Functions and mechanisms
- TL.2 State management
- TL.3 Framing and multiplexing
- TL.4 Error control
- TL.5 Transmission control
- TL.6 TCP optimisations
  - TL.6.1 TCP overview
  - TL.6.2 TCP optimisations for high performance
TCP Optimisations

Example 7.8 Congestion Control

- Fast recovery
  - 1/2 congestion window after 3dupACK rather than slow start
- Partial acknowledgement response [Hoe 1996]
  - 1/2 window reduction only once with partial retransmit ACK
  - rather than per packet
- ABC: appropriate byte counting for cwnd [Allman 2003]
- RED: random early detection (discard) [Floyd 1993]
  - discard packets when router queue threshold exceeded
  - throttle TCP source earlier before congestion occurs
- ECN: explicit congestion notification [Floyd 1994]
  - use IP ECN to trigger multiplicative decrease

TCP Optimisations

Example 7.8 Explicit Congestion Notification

- Network nodes set congestion indication
  - hist. ICMP source quench
  - ECN bits in IP header
- TCP reacts
  - uses flags for signalling
TCP Optimisations

Example Congestion Control

• Research optimisations [jury still out]
  – Pacing [Visweswaraiah 1997]
    • spread segment transmissions
    • rather than burst
  – Rate control [Padhye 1998]
    • rate based on equations that describe TCP behaviour
    • can also make UDP transmission “TCP friendly”
  – ELN: explicit loss notification [Samaraweera 1997]
  – ETEN: explicit transport error notification [Krishnan 2004]
    • signal loss due to channel bit errors
    • loss of ACK is not misconstrued as congestion

Example Congestion Control Research

• Research optimisations [jury still out]
  – Vegas [Brakmo 1995]
    • RTT estimate rather than AIMD
  – HSTCP (high-speed TCP) [RFC 3649 Floyd 2003]
    • adaptive congestion window in response to LFP
  – STCP (scalable TCP) [Kelly 2003]
  – XCP (explicit control protocol) [Katabi 2003]
    • congestion header with cwnd, RTT estimate, feedback field
  – and many others...
TCP Optimisations

Example 7.8 TCP Implementations

- **Tahoe**
  - slow start and congestion avoidance algorithms
  - fast retransmit after triple duplicate ACKs
- **Reno** – widely implemented
  - Tahoe + fast recovery
- **NewReno** [Floyd 1999]
  - Reno + partial ACK recovery
- **BIC (binary-increase TCP)** [Xu 2004]
  - binary search of window size during steady state
  - CUBIC variant is now default in Linux kernel > 2.6.19
- **CTCP (compound TCP)** [Tan 2005]
  - maintains dual AIMD and Vegas-style delay windows
  - default in MS-Windows since Vista

E2E Transmission Control

Hybrid Control

- **Dynamic rate control**
  - open-loop rate control modified by network feedback
    - example: ATM ABR
- **Pacing to reduce burstiness**
  - sender base rate control augments closed-loop control
  - window transmission spread over RTT
  - options
    - pace initial window, allow ACK self clocking to take over
    - periodic repacing to compensate for ACK compression
    - continuous pacing
TCP Optimisations

Example 7.9 Header Fields

- TCP/IP headers
- bw-×-delay
  - fields that limit
    - sequence space
    - timer related
    - window size
- Field predictability
  - use template for
    - constant
    - predictable
  - must compute
    - highly variable

High-bandwidth-×-delay paths (long fat pipes)

Problem?
TCP Optimisations

Example 7.9 High-Bandwidth-×-Delay

• High bandwidth-×-delay paths (long fat pipes)
  – problem: large number of bits in flight

Implications to TCP?

• Implications to TCP
  – max window size is 64KB problem?
  – packet loss has significant impact on goodput
  – only one RTT measurement per window
  – sequence numbers limited to 32 bit problem?
TCP Optimisations

Example 7.9 High-Bandwidth-x-Delay

- Common optimisations [RFC 1323]
  - Window scale option [RFC 1323]
    - SYN option power-of-2 multiplier to initial window
  - RTTM (round-trip time measurement) [RFC 1323]
    - timestamp to allow RTT measurement
  - PAWS (protection against wrapped seq. nos.) [RFC 1323]
    - 32-bit timestamp augments 32-bit sequence number
  - SACK (selective acknowledgement) [RFC 2018]
    - byte range header options for 3 – 4 selective ACK ranges
  - Fast retransmit [RFC 2581]
    - retransmit on triple duplicate ACKs without waiting for timer

TCP Optimisations

Example 7.9 SACK

- TCP provides reliable byte stream
  - cumulative ACKs perform poorly if loss rate not very low
  - cumulative ACKs limit ability to reorder
- Selective ACKs [RFC2018]
  - selective repeat ARQ mechanism
    - used in conjunction with cumulative ACK mechanisms
  - SACK byte range in TCP header option
- Retransmissions triggered by:
  - holes in SACK ranges (SACK is positive ACK)
**TCP Optimisations**

**Example 7.9 SACK Header**

- SACK permitted negotiation
  - kind=4 in SYN
- SACK ranges
  - kind = 05
    - after kind=01 NOP pads
  - length = 8n+2 B for n SACK blocks
    - max of 3 SACK blocks
      - assuming another option
    - each block:
      - seq num of 1st byte
      - seq num of last byte +1
    - ack semantics unchanged

---

**TCP Optimisations**

**Example 7.9 SACK Operation**

- Capability announced with SYN
  - SACK permitted option 5
- Cumulative ACKs as before
  - ACK byte in last segment
  - with no missing prior segments
- Selective ACKs
  - sent only when non-contiguous segments received
  - indicate byte range received
  - holes between SACK blocks indicate missing byte ranges
TCP Optimisations

Example 7.9  TCP High-Speed Optimisations

- Common optimisations [RFC 1323]
  - Window scale option [RFC 1323]
    - SYN option power-of-2 multiplier to initial window
  - RTTM (round-trip time measurement) [RFC 1323]
    - timestamp to allow RTT measurement
  - PAWS (protection against wrapped seq. nos.) [RFC 1323]
    - 32-bit timestamp augments 32-bit sequence number
  - SACK (selective acknowledgement) [RFC 2018]
    - byte range header options for 3 – 4 selective ACK ranges
  - Fast retransmit [RFC 2581]
    - retransmit on triple duplicate ACKs without waiting for timer

- Experimental optimisations
  - larger initial window ≈ 4KB [RFC 3390]
  - discussion of initial cwnd of 10 segments on TCPM list
  - concern about aggressiveness

- Protocol Research
  - trailer checksum [Bridges 1994]
    - hotly debated ID
    - dismissed by IETF community
    - question of performance gain vs. implementation pain
  - pacing [Partridge 2001]
    - spreads window transmission within RTT
    - bandwidth estimation and RTT measurement
TCP Optimisations

Example 7.9 TCP High-Speed Optimisations

- Implementation optimisations
  - header prediction [Jacobson 1990, Pink 1994]
- Implementation research
  - TCP/IP ILP [Clark 1990]
    - complex interactions with cache

End-to-End Transport

Additional Sources

- Michael Welzl
  *Network Congestion Control: Managing Internet Traffic*
  Wiley 2005
  http://www.welzl.at/congestion
End-to-End Transport

Acknowledgements

Some material in these foils comes from the textbook supplementary materials:

- Kurose & Ross
  *Computer Networking: A Top-Down Approach Featuring the Internet*
  http://wps.aw.com/aw_kurose_network_3

- Sterbenz & Touch
  *High-Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication*
  http://hsn-book.sterbenz.org