Protocol Stack Revisited

Application
Presentation
Session
Transport
Network
Data Link
Physical

UDP and TCP/IP

So far…
# Application vs. Network

<table>
<thead>
<tr>
<th>Application Needs</th>
<th>Network Char.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliable, Ordered, Single-Copy Message Delivery</td>
<td>Drops, Duplicates and Reorders Messages</td>
</tr>
<tr>
<td>Arbitrarily large messages</td>
<td>Finite message size</td>
</tr>
<tr>
<td>Allows Flow Control by Receiver</td>
<td>Arbitrary Delay</td>
</tr>
<tr>
<td>Supports multiple applications per-host</td>
<td>…</td>
</tr>
</tbody>
</table>
User Datagram Protocol (UDP)

- Simplest transport-layer protocol
- Just exposes IP packet functionality to application level
- *Ports* identify sending/receiving process
  - Demultiplexing information
  - (port, host) pair identifies a network process
UDP End-to-End Model

- Multiplexing/Demultiplexing with Port number
Using Ports

• Client contacts Server at a *well-known port*
  – DNS: port 53
  – POP3: port 110
  – Unix talk : port 517
  – In unix, ports are listed in `/etc/services`

• Sometimes Client and Server agree on a different port for subsequent communication

• Ports are an abstraction
  – Implemented differently on different OS’s
  – Typically a message queue
Transmission Control Protocol (TCP)

• Most widely used protocol for reliable byte streams
  – Reliable, in-order delivery of a stream of bytes
  – Full duplex: pair of streams, one in each direction
  – Flow and congestion control mechanisms
  – Like UDP, supports ports

• Built on top of IP (hence TCP/IP)
TCP End-to-End Model

• Buffering corrects errors but may introduce delays
Packet Format

- **Flags**
  - SYN
  - FIN
  - RESET
  - PUSH
  - URG
  - ACK

- **Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SequenceNum</td>
<td>16</td>
</tr>
<tr>
<td>SrcPort</td>
<td>16</td>
</tr>
<tr>
<td>DstPort</td>
<td>16</td>
</tr>
<tr>
<td>Acknowledgment</td>
<td>31</td>
</tr>
<tr>
<td>UrgPtr</td>
<td>31</td>
</tr>
<tr>
<td>Advert.Wind.</td>
<td>31</td>
</tr>
<tr>
<td>Checksum</td>
<td>31</td>
</tr>
<tr>
<td>Flags</td>
<td>31</td>
</tr>
<tr>
<td>Options (variable)</td>
<td>31</td>
</tr>
</tbody>
</table>

DATA
Three-Way Handshake

Active participant (client)  Passive participant (server)

SYN, $\text{SequenceNum} = x$

SYN + ACK, $\text{SequenceNum} = y$,
$\text{Acknowledgment} = x + 1$

ACK, $\text{Acknowledgment} = y + 1$
TCP State Transitions

1. **CLOSED**
   - Passive open
   - Close

2. **LISTEN**
   - Active open/SYN
   - Send/SYN

3. **SYN_RCVD**
   - SYN/SYN + ACK
   - SYN/ACK/ACK
   - Close/FIN

4. **ESTABLISHED**
   - FIN/ACK
   - CLOSE_WAIT
   - LAST_ACK
   - CLOSED

5. **FIN_WAIT_1**
   - ACK
   - FIN/ACK

6. **FIN_WAIT_2**
   - FIN/ACK

7. **CLOSING**
   - ACK
   - Timeout after two segment lifetimes

8. **TIME_WAIT**
   - CLOSED
TCP Receiver

• Maintains a buffer from which application reads
• Advertises < buffer size as the window for sliding window
• Responds with Acknowledge and AdvertisedWindow on each send; updates byte counts when data O.K.
• Application blocked until read() O.K.
TCP Sender

- Maintains a buffer; sending application is blocked until room in the buffer for its write
- Holds data until acknowledged by receiver as successfully received
- Implement window expansion and contraction; note difference between flow and congestion control
TCP Flow & Congestion Control

• Flow vs. Congestion Control
  – Flow control protects the recipient from being overwhelmed.
  – Congestion control protects the network from being overwhelmed.

• TCP Congestion Control
  – Additive Increase / Multiplicative Decrease
  – Slow Start
  – Fast Retransmit and Fast Recovery
Increase and Decrease

- A value CongestionWindow is used to control the number of unacknowledged transmissions.
- This value is increased linearly until timeouts for ACKs are missed.
- When timeouts occur, CongestionWindow is decreased by half to reduce the pressure on the network quickly.
- The strategy is called “additive increase / multiplicative decrease”.
Additive Increase
TCP Sawtooth Pattern
Slow Start

• Sending the entire window immediately could cause a traffic jam in the network.

• Begin “slowly” by setting the congestion window to one packet.

• When acknowledgements arrive, double the congestion window.

• Continue until ACKs do not arrive or flow control dominates.
Slow Start
Protocol Stack Revisited

Application
- SMTP, HTTP, SNMP, FTP, ...

Presentation

Session

Transport

Network

Data Link

Physical

So far...
Common Features

- SMTP, HTTP, SNMP, …
  - Request/Reply protocols built on TCP or UDP
  - Designed to handle a fixed set of messages
  - Companion *data format*
  - Many applications

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Data Format</th>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP</td>
<td>RFC 822 and MIME</td>
<td>Pine, NSMail, Eudora, Outlook,...</td>
</tr>
<tr>
<td>HTTP</td>
<td>HTML</td>
<td>Explorer, Netscape, Opera,...</td>
</tr>
<tr>
<td>SNMP</td>
<td>MIB</td>
<td>snmpget, snmpset,...</td>
</tr>
</tbody>
</table>
SMTP: Simple Mail Transfer Protocol

- **Data format RFC822**
  - [http://www.faqs.org/rfcs/rfc822.html](http://www.faqs.org/rfcs/rfc822.html)
  - ASCII text
  - Header and Body

- **MIME: Multipurpose Internet Mail Extensions**
  - Mail systems assume ASCII
    - Only 64 valid characters A-Z, a-z, 0-9, +, /
  - Some datatypes include arbitrary binary data (e.g. JPEG)
  - Base64 encoding
    - 3 bytes of data map to 4 ASCII Characters
    - A=0,B=1,…
RFC822 Headers

- <CRLF>-terminated lines containing pairs of form **type: value**
- Many valid Header types
- Some headers filled out by client
  - To: stevez@cis.upenn.edu
  - Subject: CSE331
- Others filled out by mail delivery system
  - Date:
  - Received:
  - From:
This is a multi-part message in MIME format.

This is the body.

Hello

-----------020307000708030506070607
Content-Type: image/jpeg; name="doc.jpg"
Content-Transfer-Encoding: base64
Content-Disposition: inline; filename="doc.jpg"

/9j/4AAQSkZJRgABAQEASABIAAD//gAXQ3JIYXRlZCB3aXRoIFRoZSBHSU1Q/9sAQwAIBwUIBwcHCQkICgwUDQwLCwwZElMPFB0aHx4dGhwcICQuJyAiLCMcHCg3KSwwMTQ0NB8nOT04...
SMTP

• Mail Reader
  – User edits/reads/search e-mail

• Mail Daemon
  – Process running on each host (port 27)
  – Uses SMTP/TCP to transmit mail to daemons on other machines
  – Most daemons based on Berkley’s sendmail

• Mail Gateways
  – Store and forward e-mail (much like IP router)
  – Buffers on disk
  – Attempts to resend
SMTP Mail Gateways

- No need for explicit host in e-mail address
  - User can receive mail at different machines
- Recipient’s machine may not be up
  - Mail gateway can hold message for later
SMTP Dialogs

• Client posts commands
  – HELO, MAIL, RCPT, DATA, QUIT

• Server responds with code and human-readable explanation
Example SMTP Dialog

HELO seas.upenn.edu
250 Hello daemon@smtpauth.seas.upenn.edu [158.130.12.180]

MAIL FROM:<stevez@seas.upenn.edu>
250 OK

RCPT TO:<billg@microsoft.com>
250 OK

RCPT TO:<ted@microsoft.com>
550 No such user here

DATA
354 Start mail input; end with <CRLF>.<CRLF>
Blah blah blah...
<CRLF>.<CRLF>
250 OK

QUIT
221 Closing Connection