Announcements

• Project 3 will be available on the web today.
  – Due Date: April 21st (Last day of classes)
  – Group project: you must work in groups of 2 or 3 people.
    • Mail groups to cis551staff@seas.upenn.edu
    • If you have trouble finding a group, post on the class news group

• Final Exam has been Scheduled:
  – Friday, May 5th
  – 9-11 a.m.
  – Moore 216
Internet Protocol Interoperability

Overlays (running at hosts)

Virtual Network Infrastructure (runs globally)

Networks (run locally)
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>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

<table>
<thead>
<tr>
<th>SrcPort</th>
<th>DestPort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>Checksum</td>
</tr>
<tr>
<td>IP Packet Data</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
UDP End-to-End Model

- Multiplexing/Demultiplexing with Port number
Using Ports

• Client contacts Server at a *well-known port*
  – SMPT: port 25
  – 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

```
0  15  31
SrcPort   DstPort
SequenceNum
Acknowledgment
HL  0  Flags  Advert.Wind.
Checksum   UrgPtr
Options (variable)
DATA
```
Three-Way Handshake

Active participant (client)  Passive participant (server)

SYN, SequenceNum = x

SYN + ACK, SequenceNum = y, Acknowledgment = x + 1

ACK, Acknowledgment = y + 1
TCP State Transitions

- **CLOSED**
  - Passive open
  - Active open/SYN

- **LISTEN**
  - Close

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

- **SYN_SENT**
  - Send/SYN
  - SYN/SYN + ACK

- **ESTABLISHED**
  - FIN/ACK
  - Close/FIN

- **FIN_WAIT_1**
  - ACK

- **FIN_WAIT_2**
  - FIN/ACK

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

- **TIME_WAIT**
  - FIN/ACK

- **CLOSE_WAIT**
  - Close/FIN

- **LAST_ACK**
  - ACK

- **CLOSE**
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
Network Vulnerabilities

- **Anonymity**
  - Attacker is remote, origin can be disguised
  - Authentication

- **Many points of attack**
  - Attacker only needs to find weakest link
  - Attacker can mount attacks from many machines

- **Sharing**
  - Many, many users sharing resources

- **Complexity**
  - Distributed systems are large and heterogeneous

- **Unknown perimeter**
- **Unknown attack paths**
Syn Flood Attack

• Recall TCP’s 3-way handshake:
  – SYN --- SYN+ACK --- ACK

• Receiver must maintain a queue of partially open TCP connections
  – Called SYN_RECV connections
  – Finite resource (often small: e.g. 20 entries)
  – Timeouts for queue entries are about 1 minute.

• Attacker
  – Floods a machine with SYN requests
  – Never ACKs them
  – Spoofs the sending address (Why? Two reasons!)
Reflected denial of service

• Broadcast a ping request
  – For sender’s address put target’s address
  – All hosts reply to ping, flooding the target with responses

• Hard to trace

• Hard to prevent
  – Turn off ping? (Makes legitimate use impossible)
  – Limit with network configuration by restricting scope of broadcast messages
(Distributed) Denial of Service

- Coordinate multiple subverted machines to attack
- Flood a server with bogus requests
  - TCP SYN packet flood
  - > 600,000 packets per second
- Detection & Assessment?
  - 12,800 attacks at 5000 hosts! (in 3 week period during 2001)
  - IP Spoofing (forged source IP address)
    - [http://www.cs.ucsd.edu/users/savage/papers/UsenixSec01.pdf](http://www.cs.ucsd.edu/users/savage/papers/UsenixSec01.pdf)
- Prevention?
  - Filtering?
  - Decentralized file storage?
Protocol Stack Revisited

Application
Presentation
Session
Transport
Network
Data Link
Physical

SMTP, HTTP, SNMP, FTP, …

{So far…}
Common Features

• SMTP, HTTP, SNMP, FTP…
  – 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:
From: Steve Zdancewic <stevez@cis.upenn.edu>
MIME-Version: 1.0
To: stevez@cis.upenn.edu
Subject: Example Mail
Content-Type: multipart/mixed; boundary="----------020307000708030506070607"

This is a multi-part message in MIME format.

----------020307000708030506070607
Content-Type: text/plain; charset=us-ascii; format=flowed
Content-Transfer-Encoding: 7bit

This is the body.

----------020307000708030506070607
Content-Type: text/plain; name="example.txt"
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="example.txt"

Hello

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

/9j/4AAQSkZJRgABAQEASABIAAD//gAXQ3JlYXRlZCB3aXRoIFRoZSBHSU1Q/9sAQwAIBgYHBgUIBwcHCQkICgwUDQwLCwwZEhMPFB0aHx4dGhwcICQuJyAiLCMcHCg3KSwwMTQ0NB8nOT04...
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