Today’s lecture

- Protocol stack
- Application layer
  - Client-server architecture
  - Example protocol: HTTP
  - Demo
  - Application requirements
  - If time: email
Protocol layering

- What happens when data traverses a protocol stack?

Layered protocol stack

- Processing of layer depends on location
Application layer

- Network applications

- Client-server architecture
  - Server
    - Provides service (e.g., access to data)
    - Is always on and waits for requests
    - Has well known address
  - Client
    - Initiates communication with server
    - Often determines what is communicated
    - Often unknown to server
  - Communication via “socket”

- Peer-to-peer architecture
  - End-systems act as clients and servers
  - Indexing system figures out which end-system to connect to
Application layer

- Functionality provided layer below (transport layer):
  - UNIX socket (="bit pipe")

Application layer example: HTTP

- Hypertext Transfer Protocol (HTTP)
  - Protocol to access to web pages
- Remember: purpose of protocol
  - Definition of message format
    » Types of messages
    » Syntax of messages (i.e., fields and delineation)
    » Semantics of fields
  - Definition of message exchange
    » When and how to send messages
    » When and how to respond
- What does HTTP look like?
Web documents

- Typical web pages use HTML format
  ```html
  <html>
    <head>
      <!-- meta content= "text/html; charset=utf-8" -->
      <meta name="GENERATOR" content="Microsoft FrontPage 6.0">
      <meta name="ProgID" content="FrontPage.Editor.Document">
      <title>Computer Networks</title>
    </head>
    <body>
      <!-- more HTML content -->
    </body>
  </html>
  ```

- Multiple objects for each page
  - Base HTML document
  - Images
  - Etc.

- Browser
  - Fetches HTML page and objects
  - Renders HTML on screen

Hypertext Transfer Protocol

- Sequence of messages
  1. Establish connection (transport layer and below)
  2. Send request for document
  3. Receive document
  4. Repeat 2.-3. or close connection

- Format of messages
  - Plain text
  - Request format:
Hypertext Transfer Protocol

- Example:

Request line:
```
GET /ece/wolf/courses/ECE697AA/ HTTP/1.1
Host: www.ecs.umass.edu
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; ...)
Accept: text/xml,application/xml,application/xhtml+xml,...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: SignOnDefault=ATWOLF
```

Response line:
```
HTTP/1.1 200 OK
Date: Wed, 30 Jan 2008 20:22:02 GMT
Server: Apache/1.3.26 (Unix) mod_ssl/2.8.10 OpenSSL/0.9.6e
Last-Modified: Tue, 29 Jan 2008 15:14:47 GMT
ETag: "373857-14ae-41f90567"
Accept-Ranges: bytes
Content-Range: bytes 0-5293/5294
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

ECE697AA – 09/04/08 UMass Amherst – Tilman Wolf

Performance considerations

- What is the performance bottleneck of web surfing?
Performance considerations

- Performance improvements
  - Persistent connections
  - Caching of documents

User state in HTTP

- How can a shopping site remember what I have in my shopping cart?
User state in HTTP

- Cookies in HTTP

Application requirements

- Not all applications have same needs
- What are examples of different applications?
  - Requirements: reliability, bandwidth, time-sensitivity
Application requirements

<table>
<thead>
<tr>
<th>Application</th>
<th>Data Loss</th>
<th>Bandwidth</th>
<th>Time-Sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>File transfer</td>
<td>No loss</td>
<td>Elastic</td>
<td>No</td>
</tr>
<tr>
<td>E-mail</td>
<td>No loss</td>
<td>Elastic</td>
<td>No</td>
</tr>
<tr>
<td>Web documents</td>
<td>No loss</td>
<td>Elastic (few kbps)</td>
<td>No</td>
</tr>
<tr>
<td>Internet telephone/Video</td>
<td>Loss-tolerant</td>
<td>Audio: few kbps—1Mbps</td>
<td>Yes: 100s of msec</td>
</tr>
<tr>
<td>conferencing</td>
<td>Vedio: 10 kbps—5Mbps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stored audio/video</td>
<td>Loss-tolerant</td>
<td>Same as above</td>
<td>Yes: few seconds</td>
</tr>
<tr>
<td>Interactive games</td>
<td>Loss-tolerant</td>
<td>Few kbps—10 kbps</td>
<td>Yes: 100s of msec</td>
</tr>
<tr>
<td>Instant messaging</td>
<td>No loss</td>
<td>Elastic</td>
<td>Yes and no</td>
</tr>
</tbody>
</table>

- Reliability
  - Different sockets: TCP is reliable, UDP is unreliable

Application protocol examples

- Many different application layer protocols

<table>
<thead>
<tr>
<th>Application</th>
<th>Application-Layer Protocol</th>
<th>Underlying Transport Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronic mail</td>
<td>SMTP [RFC 2821]</td>
<td>TCP</td>
</tr>
<tr>
<td>Remote terminal access</td>
<td>Telnet [RFC 854]</td>
<td>TCP</td>
</tr>
<tr>
<td>Web</td>
<td>HTTP [RFC 2616]</td>
<td>TCP</td>
</tr>
<tr>
<td>File transfer</td>
<td>FTP [RFC 959]</td>
<td>TCP</td>
</tr>
<tr>
<td>Streaming multimedia</td>
<td>HTTP (e.g., YouTube), RTP</td>
<td>TCP or UDP</td>
</tr>
<tr>
<td>Internet telephony</td>
<td>SIP, RTP, or proprietary (e.g., Skype)</td>
<td>Typically UDP</td>
</tr>
</tbody>
</table>
Electronic mail

- **Mail servers**
  - Transfer mail
  - Store mail in
    - mailboxes
- **User agents**
  - Access mail from server
  - Transmit new mail
- **Protocols**
  - Simple Mail Transfer Protocol (SMTP)
  - Post Office Protocol (POP)
  - Internet Mail Access Protocol (IMAP)
  - Web-based email access (HTTP)

Electronic mail

- **Sending of email:**
  - SMTP "pushes" email to destination mail server
  - POP/IMAP/HTTP "pulls" email from mail server
  - SMTP messages are in cleartext

SMTP session:

```plaintext
Trying 128.119.91.124... Connected to franklin.ecs.umass.edu.
Escape character is '^]'.
HELO ece697aa.umass.edu
250 franklin.ecs.umass.edu Hello nog.ecs.umass.edu [128.119.91.192], pleased to meet you
MAIL FROM: <ece697aa@ecs.umass.edu>
250 2.1.0 <ece697aa@ecs.umass.edu>... Sender ok
RCPT TO: <tw@ieee.org>
250 2.1.5 <tw@ieee.org>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
You don't have to do your homework.
Your ECE697AA instructor
250 2.0.0 \000\004\014\027 Message accepted for delivery
QUIT
221 2.0.0 franklin.ecs.umass.edu closing connection
connection closed by foreign host.
```
Assignments

- Read
  - Kurose & Ross: Chapter 3
- SPARK
  - Assessment quiz